Computer Communications and Networks

Florin Pop
Joanna Kołodziej
Beniamino Di Martino *Editors*

Resource Management for Big Data Platforms

Algorithms, Modelling, and High-Performance Computing Techniques

Springer

Computer Communications and Networks

Series editor

A.J. Sammes
Centre for Forensic Computing
Cranfield University, Shrivenham Campus
Swindon, UK

The **Computer Communications and Networks** series is a range of textbooks, monographs and handbooks. It sets out to provide students, researchers, and non-specialists alike with a sure grounding in current knowledge, together with comprehensible access to the latest developments in computer communications and networking.

Emphasis is placed on clear and explanatory styles that support a tutorial approach, so that even the most complex of topics is presented in a lucid and intelligible manner.

More information about this series at http://www.springer.com/series/4198

Florin Pop · Joanna Kołodziej
Beniamino Di Martino
Editors

Resource Management for Big Data Platforms

Algorithms, Modelling, and High-Performance Computing Techniques

Springer

Editors
Florin Pop
University Politehnica of Bucharest
Bucharest
Romania

Beniamino Di Martino
Second University of Naples
Naples, Caserta
Italy

Joanna Kołodziej
Cracow University of Technology
Cracow
Poland

ISSN 1617-7975 ISSN 2197-8433 (electronic)
Computer Communications and Networks
ISBN 978-3-319-44880-0 ISBN 978-3-319-44881-7 (eBook)
DOI 10.1007/978-3-319-44881-7

Library of Congress Control Number: 2016948811

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To our Families and Friends with Love and Gratitude

Preface

Many applications generate Big Data, like social networking and social influence programs, Cloud applications, public web sites, scientific experiments and simulations, data warehouse, monitoring platforms, and e-government services. Data grow rapidly since applications produce continuously increasing volumes of both unstructured and structured data. Large-scale interconnected systems aim to aggregate and efficiently exploit the power of widely distributed resources. In this context, major solutions for scalability, mobility, reliability, fault tolerance, and security are required to achieve high performance. The impact on data processing, transfer and storage is the need to re-evaluate the approaches and solutions to better answer the user needs.

Extracting valuable information from raw data is especially difficult considering the velocity of growing data from year to year and the fact that 80 % of data is unstructured. In addition, data sources are heterogeneous (various sensors, users with different profiles, etc.) and are located in different situations or contexts. This is why the Smart City infrastructure runs reliably and permanently to provide the context as a public utility to different services. Context-aware applications exploit the context to adapt accordingly the timing, quality and functionality of their services. The value of these applications and their supporting infrastructure lies in the fact that end users always operate in a context: their role, intentions, locations, and working environment constantly change.

Since the introduction of the Internet, we have witnessed an explosive growth in the volume, velocity, and variety of the data created on a daily basis. This data is originated from numerous sources including mobile devices, sensors, individual archives, the Internet of Things, government data holdings, software logs, public profiles on social networks, commercial datasets, etc. The so-called Big Data problem requires the continuous increase of the processing speeds of the servers and of the whole network infrastructure. In this context, new models for resource management are required. This poses a critically difficult challenge and striking development opportunities to Data-Intensive (DI) and High-Performance Computing (HPC): how to efficiently turn massively large data into valuable

information and meaningful knowledge. Computationally-effective DI and HPC are required in a rapidly increasing number of data-intensive domains.

Successful contributions may range from advanced technologies, applications, and innovative solutions to global optimization problems in scalable large-scale computing systems to development of methods, conceptual and theoretical models related to Big Data applications and massive data storage and processing. Therefore, it is imperative to gather the consent of researchers to muster their efforts in proposing unifying solutions that are practical and applicable in the domain of high-performance computing systems.

The Big Data era poses a critically difficult challenge and striking development opportunities to High-Performance Computing (HPC). The major problem is an efficient transformation of the massive data of various types into valuable information and meaningful knowledge. Computationally effective HPC is required in a rapidly increasing number of data-intensive domains. With its special features of self-service and pay-as-you-use, Cloud computing offers suitable abstractions to manage the complexity of the analysis of large data in various scientific and engineering domains. This book surveys briefly the most recent developments on Cloud computing support for solving the Big Data problems. It presents a comprehensive critical analysis of the existing solutions and shows further possible directions of the research in this domain including new generation multi-datacenter cloud architectures for the storage and management of the huge Big Data streams.

The large volume of data coming from a variety of sources and in various formats, with different storage, transformation, delivery or archiving requirements, complicates the task of context data management. At the same time, fast responses are needed for real-time applications. Despite the potential improvements of the Smart City infrastructure, the number of concurrent applications that need quick data access will remain very high. With the emergence of the recent cloud infrastructures, achieving highly scalable data management in such contexts is a critical challenge, as the overall application performance is highly dependent on the properties of the data management service. The book provides, in this sense, a platform for the dissemination of advanced topics of theory, research efforts and analysis and implementation for Big Data platforms and applications being oriented on Methods, Techniques and Performance Evaluation. The book constitutes a flagship driver toward presenting and supporting advanced research in the area of Big Data platforms and applications.

This book herewith presents novel concepts in the analysis, implementation, and evaluation of the next generation of intelligent techniques for the formulation and solution of complex processing problems in Big Data platforms. Its 23 chapters are structured into four main parts:

1. *Architecture of Big Data Platforms and Applications*: Chapters 1–7 introduce the general concepts of modeling of Big Data oriented architectures, and discusses several important aspects in the design process of Big Data platforms and applications: workflow scheduling and execution, energy efficiency, load balancing methods, and optimization techniques.

2. *Big Data Analysis*: An important aspect of Big Data analysis is how to extract valuable information from large-scale datasets and how to use these data in applications. Chapters 8–12 discuss analysis concepts and techniques for scientific application, information fusion and decision making, scalable and reliable analytics, fault tolerance and security.
3. *Biological and Medical Big Data Applications*: Collectively known as computational resources or simply infrastructure, computing elements, storage, and services represent a crucial component in the formulation of intelligent decisions in large systems. Consequently, Chaps. 13–16 showcase techniques and concepts for big biological data management, DNA sequence analysis, mammographic report classification and life science problems.
4. *Social Media Applications*: Chapters 17–23 address several processing models and use cases for social media applications. This last part of the book presents parallelization techniques for Big Data applications, scalability of multimedia content delivery, large-scale social network graph analysis, predictions for Twitter, crowd-sensing applications and IoT ecosystem, and smart cities.

These subjects represent the main objectives of ICT COST Action IC1406 High-Performance Modelling and Simulation for Big Data Applications (cHiPSet) and the research results presented in these chapters were performed by joint collaboration of members from this action.

Acknowledgments

We are grateful to all the contributors of this book, for their willingness to work on this complex book project. We thank the authors for their interesting proposals of the book chapters, their time, efforts and their research results, which makes this volume an interesting complete monograph of the latest research advances and technology development on Big Data Platforms and Applications. We also would like to express our sincere thanks to the reviewers, who have helped us to ensure the quality of this volume. We gratefully acknowledge their time and valuable remarks and comments.

Our special thanks go to Prof. Anthony Sammes, editor-in-chief of the Springer "Computer Communications and Networks" Series, and to Wayne Wheeler and Simon Rees, series managers and editors in Springer, for their editorial assistance and excellent cooperative collaboration in this book project.

Finally, we would like to send our warmest gratitude message to our friends and families for their patience, love, and support in the preparation of this volume.

We strongly believe that this book ought to serve as a reference for students, researchers, and industry practitioners interested or currently working in Big Data domain.

Bucharest, Romania — Florin Pop
Cracow, Poland — Joanna Kołodziej
Naples, Italy — Beniamino Di Martino
July 2016

Contents

Part I
Architecture of Big Data Platforms and Applications

Chapter 1
Performance Modeling of Big Data-Oriented Architectures

Marco Gribaudo, Mauro Iacono and Francesco Palmieri

1.1 Introduction

Big Data-oriented platforms provide enormous, cost- efficient computing power and unparalleled effectiveness in both massive batch and timely computing applications, without the need of special architectures or supercomputers. This is obtained by means of a very targeted use of resources and a successful abstraction layer founded onto a proper programming paradigm. A key factor for the success in Big Data is the management of resources: these platforms use a significant and flexible amount of virtualized hardware resources to try and optimize the trade off between costs and results. The management of such a quantity of resources is definitely a challenge.

Modeling Big Data-oriented platforms presents new challenges, due to a number of factors: complexity, scale, heterogeneity, hard predictability. Complexity is inner in their architecture: computing nodes, storage subsystem, networking infrastructure, data management layer, scheduling, power issues, dependability issues, virtualization all concur in interactions and mutual influences. Scale is a need posed by the nature of the target problems: data dimensions largely exceed conventional storage units, the level of parallelism needed to perform computation within useful deadlines is high, obtaining final results require the aggregation of large numbers of partial results. Heterogeneity is a technological need: evolvability, extensibility, and maintainability of the hardware layer imply that the system will be partially integrated, replaced or

M. Gribaudo
DEIB, Politecnico di Milano, via Ponzio 34/5, 20133 Milan, Italy
e-mail: marco.gribaudo@polimi.it

M. Iacono (✉)
DMF, Seconda Università Degli Studi di Napoli, viale Lincoln 5, 81100 Caserta, Italy
e-mail: mauro.iacono@unina2.it

F. Palmieri
DI, Università Degli Studi di Salerno, via Giovanni Paolo II, 132, 84084 Fisciano, Italy
e-mail: francesco.palmieri@unisa.it

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_1

extended by means of new parts, according to the availability on the market and the evolution of technology. Hard predictability results from the previous three factors, the nature of computation and the overall behavior and resilience of the system when running the target application and all the rest of the workload, and from the fact that both simulation, if accurate, and analytical models are pushed to the limits by the combined effect of complexity, scale, and heterogeneity.

The most of the approaches that literature offers for the support of resource management are based on the benchmarking of existing systems. This approach is a posteriori, in the meaning that it is specially suitable and applicable to existing systems, and for tuning or applying relatively small modifications of the system with respect to its current state. Model-based approaches are more general and less bound to the current state, and allow the exploration of a wider range of possibilities and alternatives without a direct impact on the normal operations of a live system. Proper modeling techniques and approaches are of paramount importance to cope with the hard predictability problem and to support maintenance, design and management of Big Data-oriented platforms. The goal of modeling is to allow, with a reasonable approximation, a reasonable effort and in a reasonable time, the prediction of performances, dependability, maintainability and scalability, both for existing, evolving, and new systems. Both simulative and analytical approaches are suitable for the purpose, but a proper methodology is needed to dominate complexity, scale, and heterogeneity at the different levels of the system. In this chapter, we analyze the main issues related to Big Data Systems, together with a methodological proposal for a modeling and performance analysis approach that is able to scale up sufficiently while providing an efficient analysis process.

1.2 Big Data Applications

In order to understand the complexity of Big Data architectures, a brief analysis of their characteristics is helpful. A first level of complexity comes from their performance requirements: typical Big Data applications need massively parallel computing resources because of the amount of data involved in a computation and/or because of the fact that results are needed within a given time frame, or they may lose their value over time. Although Big Data applications are rarely timely critical, timeliness is often an important parameter to be considered: a good example is given by social network data stream analysis, in which sentiment analysis may be more valuable if it provides a fast characterization of a community, but, in general, whenever data are continuously generated at a given rate at high scale longer computations may result in more need for storage and eventually a different organization of the computing process itself. The main point is in the costs, which may scale up quickly and cannot be worth the value of the results because of different kinds of overheads.

Big Data applications may be seen as the evolution of parallel computing, but with the important difference of the scale. The scale effect, in this case, does not only have the same consequences that it has in ordinary parallel computing, but

pushes to a dimension in which an automated management of the resources and of their exploitation is needed, instead of a manual configuration of them or a theory-driven resource crafting and allocation approach. As management may become an expensive and time- consuming activity, human intervention is more dedicated to handle macroscopic parameters of the system rather than on fine grain ones, and automated parallelization is massively applied, e.g. by means of the Map-Reduce approach, which can be in some sense considered as an analogous of OpenMP or other similar tools.

In some sense, Big Data applications may recall analogous in the Data Warehousing field. In both cases, actually, huge amounts of data are supposed to be used to extract synthetic indications on a phenomenon: an example can be given by Data Mining applications. In this case, the difference is mainly in a minor and two main factors: as first, typical Data Warehousing applications are off line, and use historical data spanning over long time frames; as second, the scale of Big Data data bases is higher; as third, the nature of the data bases in Data Warehousing and Big Data are very different. In the first case, data is generally extracted from structured sources, and is filtered by a strict and expensive import process; this results into a high value, easily computable data source. Big Data data sources are instead often noisy, practically unfilterable, poorly or not structured, with a very low a priori value per data unit[1]: this means that, considering the low value per and the high number of data units, in the most of the cases the unitary computing cost must be kept very low, to avoid making the process unsustainable.

Finally, even if Cloud Computing can be a means to implement Big Data architectures, common Cloud Computing applications are rather different from Big Data applications. While in both cases the overall workload of the system is comparably high, as the amount of resources to be managed and the scale of the system, and virtualization can be usefully exploited in both cases, the similarities are in the underlying architectures: typically, Cloud Computing architectures serve fine grain, loosely coordinated (if so) applications, run on behalf of big numbers of users that operate independently, from different locations, possibly on own, private, non shared data, with a significant amount of interactions, rather than being mainly batch oriented, and generally fit to be relocated or with highly dynamic resource needs. Anyway, notwithstanding such significant differences, Cloud Computing and Big Data architectures share a number of common needs, such as automated (or autonomic) fine grain resource management and scaling related issues.

Given this basic profile of Big Data applications, it is possible to better understand the needs and the problems of Big Data architectures.

[1] A significant exception is given by high energy physics data, which are generated at very high costs: this does not exclude the fact that, mutatis mutandis, their experimental nature make them valuable per se and not because of the costs, and that their value is high if the overall results of the experiment are satisfying; this kind of applications is obviously out of the market, so radically different metrics for costs and results are applied.

1.3 Big Data Architectures

As storage, computing and communication technologies evolve towards a converged model, we are experiencing a paradigm shift in modern data processing architectures from the classical *application-driven* approach to new *data-driven* ones. In this scenario, huge data collections (hence the name "Big Data"), generated by Internet-scale applications, such as social networks, international scientific corporations, business intelligence, and situation aware systems as well as remote control and monitoring solutions, are constantly migrated back and forth on wide area network connections in order to be processed in a timely and effective way on the hosts and data centers that provide enough available resources. In a continuously evolving scenario where the involved data volumes are estimated to double or more, every year, Big Data processing systems are actually considered as a silver bullet in the computing arena, due to their significant potential of enabling new distributed processing architectures that leverage the virtually unlimited amount of computing and storage resources available on the Internet to manage extremely complex problems with previously inconceivable performances. Accordingly, the best recipe for success becomes efficiently retrieving the right data from the right location, at the right time, in order to process it where the best resource mix is available [1]. Such approach results in a dramatic shift from the old *application-centric* model, where the needed data, often distributed throughout the network, are transferred to the applications when necessary, to a new *data-centric* scheme, where applications are moved through the network in order to run them in the most convenient location, where adequate communication capacities and processing power are available. As a further complication it should be considered that the location of data sources and their access patterns may change frequently, according to the well known spatial and temporal locality criteria. Of course, as the amount of involved data and their degree of distribution across the network grow, the role of the communication architecture supporting the data migration among the involved sites become most critical, in order to avoid to be origin of performance bottlenecks in data transfer activities adversely affecting the execution latency in the whole big data processing framework.

1.3.1 Computing

The significant advantages of Big Data systems have a cost: as they need high investments, their sustainability and profitability are critical and strongly depend on a correct design and management. Research is still very active in exploring the best solutions to provide scalability of computing and data storage architecture and algorithms, proper querying and processing technologies, efficient data organization, planning, system management and dependability oriented techniques. The main issues related to this topic have been analyzed in [2–7], which we suggest to the interested readers.

To be able to dominate the problems behind Big Data systems, a thorough explorations of the factors that generate their complexity is needed. The first important aspect to be considered is the fact that the computing power is provided by a very high number of computing nodes, each of which has its resources that have to be shared on a high scale. This is a direct consequence of the dimensions workloads: a characterization of typical workloads for systems dealing with large datasets is provided in [8], which surveys the problem, also from the important point of view of energy efficiency, comparing Big Data environments, HPC systems, and Cloud systems. The scale exacerbates known management and dimensioning problems, both with relation to architecture and resource allocation and coordination, with respect to classical scientific computing or data base systems. In fact, efficiency is the key to sustainability: while classical data warehouse applications operate on quality assured data, thus justify an high investment per data unit, in the most of the cases Big Data applications operate on massive quantities of raw, low quality data, and do not ensure the production of value. As a consequence, the cost of processing has to be kept low to justify investments and allow sustainability of huge architectures, and the computing nodes are common COTS machines, which are cheap and are easily replaceable in case of problems, differently from what traditionally has been done in GRID architectures. Of course, sustainability also includes the need for controlling energy consumption. The interested reader will find in [9] some guidelines for design choices, and in [10] a survey about energy-saving solutions.

The combination between low cost and high scale allows to go beyond the limits of traditional data warehouse applications, which would not be able to scale enough. This passes through new computing paradigms, based on special parallelization patterns and divide-and-conquer approaches that can be not strictly optimal but suitable to scale up very flexibly. An example is given by the introduction of the Map-Reduce paradigm, which allows a better exploitation of resources without sophisticated and expensive software optimizations. Similarly, scheduling is simplified within a single application, and the overall scheduling management of the system is obtained by introducing virtualization and exploiting the implicitly batch nature of Map-Reduce applications. Moving data between thousands of nodes is also a challenge, so a proper organization of the data/storage layer is needed.

Some proposed middleware solutions are Hadoop [11, 12] (that seems to be the market leader), Dryad [13] (a general-purpose distributed execution engine based on computational vertices and communication channels organized in a custom graph execution infrastructure), Oozie [14], based on a flow oriented Hadoop Map-Reduce execution engine. As data are very variable in size and nature and data transfer are not negligible, one of the main characteristics of the frameworks is the support for a continuous reconfiguration. This is a general need of Big Data applications, which are naturally implemented on Cloud facilities. Cloud empowered Big Data environments benefit of the flexibility of virtualization techniques and enhance their advantages, providing the so-called elasticity feature to the platform. Commercial high-performance solutions are represented by Amazon EC2 [15] and Rackspace [16].

1.3.2 Storage

The design of a performing storage subsystem is a key factor for Big Data systems. Storage is a challenge both at the low level (the file system and its management) and at the logical level (database design and management of information to support applications). File systems are logically and physically distributed along the architecture, in order to provide a sufficient performance level, which is influenced by large data transfers over the network when tasks are spawn along the system. In this case as well, the lesson learned in the field of Cloud Computing is very useful to solve part of the issues. The management of the file system has to be carefully organized and heavily relies on redundancy to keep a sufficient level of performances and dependability. According to the needs, the workloads and the state of the system, data reconfigurations are needed, thus the file system is a dynamic entity in the architecture, often capable of autonomic behaviors. An example of exploitation of Cloud infrastructure to support Big Data analytics applications is presented in [17], while a good introduction to the problems of data duplication and deduplication can be found in [5]. More sophisticated solutions are based on distributed file systems using erasure coding or peer to peer protocols to minimize the impact of duplications while keeping a high level of dependability: in this case, data are preprocessed to obtain a scattering on distribution schemata that, with low overheads, allow a faster reconstruction of lost data blocks, by further abstracting physical and block-level data management. Some significant references are [18–21]; a performance oriented point of view is taken in [22–29].

On the logical level, traditional relational databases do not scale enough to efficiently and economically support Big Data applications. The most common structured solutions are generally based on NoSQL databases, which speed up operations by omitting the heavy features of RDBMS (such as integrity, query optimization, locking, and transactional support) focusing on fast management of unstructured or semi-structured data. Such solutions are offered by many platforms, such as Cassandra [30], MongoDB [31] and HBase [32], which have been benchmarked and compared in [33].

1.3.3 Networking

High-performance networking is the most critical prerequisite for modern distributed environments, where the deployment of data-intensive applications often requires moving many gigabytes of data between geographically distant locations in very short time lapses, in order to meet I/O bandwidth requirements between computing and storage systems. Indeed, the bandwidth necessary for such huge data transfers, exceeds of multiple orders of magnitude the network capacity available in state-of-the-art networks. In particular, despite the Internet has been identified as the fundamental driver for modern data-intensive distributed applications, it does not seem

able to guarantee enough performance in moving very large quantities of data in acceptable times neither at the present nor even in the foreseeable near future. This is essentially due to the well-known scalability limits of the traditional packet forwarding paradigm based on statistical multiplexing, as well as to the best-effort delivery paradigm, imposing unacceptable constraints on the migration of large amounts of data, on a wide area scale, by adversely affecting the development of Big Data applications. In fact, the traditional shared network paradigm, characterizing the Internet is based on a best-effort packet-forwarding service that is a proven efficient technology for transmitting in sequence multiple bursts of short data packets, e.g., for consumer oriented email and web applications. Unfortunately this is not enough to meet the challenge of the large-scale data transfer and connectivity requirement of the modern network-based applications. More precisely, the traditional packet forwarding paradigm, does not scale in its ability of rapidly moving very large data quantities between distant sites. Making forwarding decisions every 1500 bytes is sufficient for emails or 10–100k web pages. This is not the optimal mechanism if we have to cope with data size of ten orders (or more) larger in magnitude. For example, copying 1.5 TB of data using the traditional IP routing scheme requires adding a lot of protocol overhead and making the same forwarding decision about 1 billion times, over many routers/switches along the path, with the obvious consequence in terms of introduced latency and bandwidth waste [34].

Massive data aggregation and partitioning activities, very common in Big Data processing architectures structured according to the Map-Reduce paradigm, require huge bandwidth capacities in order to effectively support the transmission of massive data between a potentially very high number of sites, as the result of multiple data aggregation patterns between mappers and reducers [1]. For example, the intermediate computation results coming from a large number of mappers distributed throughout the Internet, each one managing data volumes up to tens of gigabytes, can be aggregated on a single site in order to manage more efficiently the `reduce` task. Thus the current aggregated data transfer dimension for Map-Reduce-based data-intensive applications can be expressed in the order of petabytes and the estimated growth rate for the involved data sets currently follows an exponential trend. Clearly, moving these volumes of data across the Internet may require hours or, worse, days. Indeed, it has been estimated [35] that up to 50 % of the overall task completion time in Map-Reduce-based systems may be associated to data transfers performed within the data shuffling and spreading tasks. This significantly limits the ability of creating massive data processing architectures that are geographically distributed on multiple sites over the Internet [1]. Several available solutions for efficient data transfer based on novel converged protocols have been explored in [36] whereas a comprehensive survey of map-reduce-related issues associated to adaptive routing practices has been presented in [37].

1.4 Evaluation of Big Data Architectures

A key factor for the success in Big Data is the management of resources: these platforms use a significant and flexible amount of virtualized hardware resources to try and optimize the trade off between costs and results. The management of such a quantity of resources is definitely a challenge.

Modeling Big Data-oriented platforms presents new challenges, due to a number of factors: complexity, scale, heterogeneity, hard predictability. Complexity is inner in their architecture: computing nodes, storage subsystem, networking infrastructure, data management layer, scheduling, power issues, dependability issues, virtualization all concur in interactions and mutual influences. Scale is a need posed by the nature of the target problems: data dimensions largely exceed conventional storage units, the level of parallelism needed to perform computation within useful deadlines is high, obtaining final results requires the aggregation of large numbers of partial results. Heterogeneity is a technological need: evolvability, extensibility and maintainability of the hardware layer imply that the system will be partially integrated, replaced or extended by means of new parts, according to the availability on the market and the evolution of technology. Hard predictability results from the previous three factors, the nature of computation and the overall behavior and resilience of the system when running the target application and all the rest of the workload, and from the fact that both simulation, if accurate, and analytical models are pushed to the limits by the combined effect of complexity, scale and heterogeneity.

The value of performance modeling is in its power to enable developers and administrators to take informed decisions. The possibility of predicting the performances of the system helps in better managing it, and allows to reach and keep a significant level of efficiency. This is viable if proper models are available, which benefit of information about the system and its behaviors and reduce the time and effort required for an empirical approach to management and administration of a complex, dynamic set of resources that are behind Big Data architectures.

The inherent complexity of such architectures and of their dynamics translates into the non triviality of choices and decisions in the modeling process: the same complexity characterizes models as well, and this impacts on the number of suitable formalisms, techniques, and even tools, if the goal is to obtain a sound, comprehensive modeling approach, encompassing all the (coupled) aspects of the system. Specialized approaches are needed to face the challenge, with respect to common computer systems, in particular because of the scale. Even if Big Data computing is characterized by regular, quite structured workloads, the interactions of the underlying hardware-software layers and the concurrency of different workloads have to be taken into account. In fact, applications potentially spawn hundreds (or even more) cooperating processes across a set of virtual machines, hosted on hundreds of shared physical computing nodes providing locally and less locally [38, 39] distributed resources, with different functional and non functional requirements: the same abstractions that simplify and enable the execution of Big Data applications complicate and modeling problem. The traditional system logging practices are potentially

themselves, on such a scale, Big Data problems, which in turn require significant effort for an analysis. The system as a whole has to be considered, as in a massively parallel environment many interactions may affect the dynamics, and some computations may lose value if not completed in a timely manner.

Performance data and models may also affect the costs of the infrastructure. A precise knowledge of the dynamics of the system may enable the management and planning of maintenance and power distribution, as the wear and the required power of the components is affected by their usage profile.

Some introductory discussions to the issues related to performance and dependability modeling of big computing infrastructures can be found in [40–46]. More specifically, several approaches are documented in the literature for performance evaluation, with contributions by studies on large-scale cloud- or grid-based Big Data processing systems. They can loosely be classified into monitoring focused and modeling focused, and may be used in combination for the definition of a comprehensive modeling strategy to support planning, management, decisions, and administration. There is a wide spectrum of different methodological points of view to the problem, which include classical simulations, diagnostic campaigns, use and demand profiling or characterization for different kinds of resources, predictive methods for system behavioral patterns.

1.4.1 Monitoring-Focused Approaches

In this category some works are reported that are mainly based on an extension, or redesign, or evolution of classical monitoring or benchmarking techniques, which are used on existing systems to investigate their current behavior and the actual workloads and management problems. This can be viewed as an empirical approach, which builds predictions onto similarity and regularity assumptions, and basically postulates models by means of perturbative methods over historical data, or by assuming that knowledge about real or synthetic applications can be used, by means of a generalization process, to predict the behaviors of higher scale applications or of composed applications, and of the architecture that supports them. In general, the regularity of workloads may support in principle the likelihood of such hypotheses, specially in application fields in which the algorithms and the characterization of data are well known and runs tend to be regular and similar to each other. The main limits of this approach, which is widely and successfully adopted in conventional systems and architectures, is in the fact that for more variable applications and concurrent heterogeneous workloads the needed scale for experiments and the test scenarios are very difficult to manage, and the cost itself of running experiments or tests can be very high, as it requires an expensive system to be diverted from real use, practically resulting in a non-negligible downtime from the point of view of productivity. Moreover, additional costs are caused by the need for an accurate design and planning of the tests, which are not easily repeatable for cost matters: the scale is of the order of

thousands of computing nodes and petabytes of data exchanged between the nodes by means of high speed networks with articulated access patterns.

Two significant examples of system performances prediction approaches that represent this category are presented in [47, 48]. In both cases, the prediction technique is based on the definition of test campaigns that aim at obtaining some well chosen performance measurements. As I/O is a very critical issue, specialized approaches have been developed to predict the effects of I/O over general application performances: an example is provided in [49], which assumes the realistic case of an implementation of Big Data applications in a Cloud. In this case, the benchmarking strategy is implemented in the form of a training phase that collects information about applications and system scale to tune a prediction system. Another approach that presents interesting results and privileges storage performance analysis is given in [50], which offers a benchmarking solution for cloud-based data management in the most popular environments.

Log mining is also an important resource, which extracts value from an already existing asset. The value obviously depends on the goals of the mining process and on the skills available to enact a proper management and abstraction of an extended, possibly heterogeneous harvest of fine grain measures or events tracking. Some examples of log mining-based approaches are given in Chukwa [51], Kahuna [52], and Artemis [53]. Being this category of solutions founded onto technical details, these approaches are bound to specific technological solutions (or different layers of a same technological stack), such as Hadoop or Dryad: for instance, [54] presents an analysis of real logs from a Hadoop-based system that is composed of 400 computing nodes, while [55, 56] offers data from Google cloud backend infrastructures.

1.4.2 Simulation-Focused Approaches

Simulation-based approaches and analytical approaches are based on previous knowledge or on reasonable hypotheses about the nature and the inner behaviors of a system, instead of inductive reasoning or generalization from measurements. Targeted measurements (on the system, if existing, or on similar systems, if not existing yet) are anyway used to tune the parameters and to verify the goodness of the models.

While simulation (e.g., event-based simulation) offers in general the advantage of allowing great flexibility, with a sufficient number of simulation runs to include stochastic effects and reach a sufficient confidence level, and eventually by means of parallel simulation or simplifications, the scale of Big Data architectures is still a main challenge. The number of components to be modeled and simulated is huge, consequently the design and the setup of a comprehensive simulation in a Big Data scenario are very complex and expensive, and become a software engineering problem. Moreover, being the number of interactions and possible variations huge as well, the simulation time that is needed to get satisfactory results can be unacceptable and not fit to support timely decision-making. This is generally bypassed by a trade off

between the degree of realism, or the generality, or the coverage of the model and simulation time. Simulation is anyway considered a more viable alternative to very complex experiments, because it has more economic experimental setup costs and a faster implementation.

Literature is rich of simulation proposals, specially borrowed from the Cloud Computing field. In the following, only Big Data specific literature is sampled.

Some simulators focus on specific infrastructures or paradigms: Map-Reduce performances simulators are presented in [57], focusing on scheduling algorithms on given Map-Reduce workloads, or provided by non workload-aware simulators such as SimMapReduce [58], MRSim [59], HSim [60], or Starfish [61, 62] what-if engine. These simulators do not consider the effects of concurrent applications on the system. MRPerf [63] is a simulator specialized in scenarios with Map-Reduce on Hadoop; X-Trace [64] is also tailored on Hadoop and improves its fitness by instrumenting it to gather specific information. Another interesting proposal is Chukwa [51]. An example of simulation experience specific for Microsoft based Big Data applications is in [65], in which a real case study based on real logs collected on large scale Microsoft platforms.

To understand the importance of the workload interference effects, specially in cloud architectures, for a proper performance evaluation, the reader can refer to [66], which proposes a synthetic workload generator for Map-Reduce applications.

1.4.2.1 Simulating the Communication Stratum

Network simulation can be very useful in the analysis of Big Data architectures, since it provides the ability to perform proof-of-concept evaluations, by modeling the interactions between multiple networked entities when exchanging massive data volumes, before the real development of new Big Data architectures and applications as well as selecting the right hardware components/technologies enabling data transfers between the involved geographical sites. This also allows testing or studying the effects of introducing modifications to existing applications, protocols or architectures in a controlled and reproducible way.

A significant advantage is the possibility of almost completely abstracting from details which are unnecessary for a specific evaluation task, and focus only on the topics that are really significant, by achieving, however, maximum consistency between the simulated model and the problem to be studied. A satisfactory simulation platform must provide a significant number of network devices and protocols as its basic building blocks, organized into extensible packages and modules that allow us to simply and flexibly introduce new features or technologies in our model.

Modern network simulators usually adopt ad-hoc communication models and operate on a logical event-driven basis, by running on large dedicated systems or in virtualized runtime environments distributed on multiple sites [67]. Indeed, complex simulation experiments may be also handled in a fully parallel and distributed way significantly improving simulation performance by running on huge multi-processors system, computing clusters or network-based distributed computing organization

such as grids or clouds. Another important feature that can be considered simultaneously as a strength and a drawback of network simulation, is that it does not operate in real-time. This implies the possibility of arbitrarily compressing or stretching the time scale on a specific granularity basis, by compressing a very long simulated period (e.g., a day or a week) into few real-time seconds, or conversely requiring a long time (maybe days or months) for simulating a few seconds lapse in a complex experiment. Of course this inhibits natively any kind of man-in-the-loop involvement within the simulation framework.

There are plenty of network simulation tools available, with widely varying targets and able to manage from the simplest to the more complex scenarios. Some of them are focused on studying a specific networking area or behavior (i.e., a particular network type or protocol), whereas other one are extremely flexible and adaptive and able to target a wider range of protocols and mechanisms.

Basically, a network simulation environment should enable users to model any kind of network topology, as well as creating the proper scenarios to be simulated, with the involved network devices, the communication links between them and the different kind of traffic flowing on the network. More complex solution allow users to configure in a very detailed way the protocols used to manage the network traffic and provide a simulation language with network protocol libraries or Graphical user interfaces that are extremely used to visualize and analyze at a glance the results of the simulation experiments.

A very simplified list of the most used network simulation environments include OPNET [68], NS-2 [69], NS-3 [70], OMNeT++ [71], REAL [72], SSFNet [73], J-Sim [74], and QualNet [75].

OPNET is a commercial system providing powerful visual or graphical support in a discrete event simulation environment that can be flexibly used to study communication networks, devices, protocols, and applications.

NS2, originally based on REAL network simulator, is an open source object-oriented, discrete event-driven network simulator which was originally developed at University of California, Berkeley and supporting C++ and OTcl (Object-oriented Tcl) as its simulation languages

Analogously, NS3, originally designed to replace NS2, is another discrete-event solution, flexibly programmable in C++ and Python, released under the GNU GPLv2 license and targeting modern networking research applications. NS3 is not an NS2 upgrade since its simulation engine has been rewritten from the scratch without preserving the backward-compatibility with NS2.

Like NS2 and NS3, OMNeT++ is an open-source, component-based network simulation environment, mainly targeted on communication networks, providing a rich GUI support. It is based on a quite general and flexible architecture ensuring its applicability also in other sectors such as IT systems, queuing networks, hardware systems, business processing, and so on.

SSFNet is a clearinghouse for information about the latest tools for scalable high-performance network modeling, simulation, and analysis, providing open-source Java models of protocols (IP, TCP, UDP, BGP4, OSPF, and others), network elements, and assorted support classes for realistic multi-protocol, multi-domain Internet mod-

eling and simulation. It also supports an Integrated Development Environment (IDE) combining the open-source modeling components with simulation kernels, DML database implementations, and assorted development tools.

REAL is an old network simulation environment, written in C and running on almost any Unix flavor, originally intended for studying the dynamic behavior of flow and congestion control schemes in packet-switched networks.

J-Sim (formerly known as JavaSim) is a platform-neutral, extensible, and reusable simulation environment, developed entirely in Java and providing a script interface to allow integration with different scripting languages such as Perl, Tcl, or Python. It has been built upon the notion of the autonomous component programming model and structured according to a component-based, compositional approach. The behavior of J-Sim components are defined in terms of contracts and can be individually designed, implemented, tested, and incrementally deployed in a software system

The QualNet communications simulation platform is a commercial planning, testing and training tool that mimics the behavior of a real communications network, providing a comprehensive environment for designing protocols, creating and animating network scenarios, and analyzing their performance. It can support real-time speed to enable software-in-the-loop, network emulation, and human-in-the-loop modeling.

1.4.2.2 Beyond Simulation: Network Emulation Practices

Unfortunately, simulation is not generally able to completely substitute sophisticated evaluation practices involving complex network architectures, in particular in the different testing activities that characterize real-life Big Data applications scenarios.

In this situation, we can leverage network emulation, which can be seen as a hybrid practice combining virtualization, simulation and field test. In detail, in emulated network environments the end systems (e.g., computing, storage, or special-purpose equipment), as well as the intermediate ones (e.g., networking devices), eventually virtualized to run on dedicated VMs, communicate over a partially abstract network communication stratum, where part of the communication architecture (typically the physical links) is simulated in real time. This allows us to explore the effects of distributing Big Data sources on huge geographical networks, made of real network equipment whose firmware runs on dedicated VMs, without the need of obtaining a real laboratory/testbed with plenty of wide area network links scattered over the Internet.

In other words, using enhanced virtualization and simulation technologies, a fully functional and extremely realistic networking environment can be reproduced, in which all the involved entities behave exactly as they were connected through a real network. This allows the observation of the behavior of the network entities under study on any kind of physical transport infrastructure (e.g., wired, wireless, etc.), by also introducing specific QoS features (e.g., end-to-end latency, available bandwidth) or physical impairments (faults, packet losses, transmission errors etc.) on the virtualized communication lines. Thus, any large scale Big Data architecture, relaying on

any kind of network topology can be emulated, involving a large number of remote sites connected withh each other in many ways (dedicated point-to-point links, rings, heterogeneous meshes) with the goal of assessing in real time the performance or the correct functionality of complex network-centric or data-centric Big Data applications and analyzing or predicting the effect of modifications, or re-optimizations in architectures, protocols, or changes traffic loads. Clearly, in order to ensure a realistic emulation experience, leading to accurate and reliable results, the simulated communication layer must enforce the correct timing and QoS constraints, as well as consider and reproduce the right network conditions when delivering packets between the emulated network entities. This can be achieved through the careful implementation of artificial delays and bandwidth filters, as well as mimicking congestion phenomena, transmission errors or generic impairments, to reflect the specific features of the involved communication lines [67].

Complex network emulation architectures can be structured according to a centralized or a fully distributed model. Centralized solutions use a single monolithic machine for running all the virtualized network entities together with the simulated physical communication layer, and consequently despite the obvious advantages in terms of implicit synchronization, the scalability of the resulting architecture is conditioned by the computing power characterizing the hosting machine.

To cope with such a limitation, fully distributed emulation architectures can rely on a virtually unlimited number of machines hosting the VMs associated to the involved network entities, by using complex communication protocols to implement the simulated links in a distributed way, and by also ensuring synchronization between the different components running on multiple remote machines locates on different and distant sites. While introducing significant benefits in terms of scalability and efficiency, such infrastructures are much harder to implement and manage, since an additional "real" transport layer is introduced under the simulated one, and this should be considered when simulating all the physical links' transmission features (capacity, delay, etc.). Strict coordination is also needed between the involved nodes (and the associated hypervisors), usually implemented by local communication managers running on each participating machine. Usually, to ensure the consistency of the whole emulation environment in presence of experiments characterized by real-time communication constraints, distributed architectures run on multiple systems located on the same local area network or on different sites connected by dedicated high-performance physical links, providing plenty of bandwidth, limited delay, and extreme transmission reliability [67].

In addition, distributed emulation environments can reach a degree of scalability that cannot be practically reached in traditional architectures. Virtualization of all the involved equipment (both proof-of-concept/prototype architectures under test and production components making the communication infrastructure), becomes a fundamental prerequisite for effectively implementing complex architectures, emulating plenty of different devices and operating systems, by disassociating their execution from the hardware on which they run and hence allowing the seamless integration/interfacing of many heterogeneous devices and mechanisms into a fully manageable emulation platform [67].

Early experiences in network emulation, essentially focused on TCP/IP performance tests, were based on the usage of properly crafted hosts with the role of gateways specialized for packet inspection and management. More recent approaches leverage special-purpose stand-alone emulation frameworks supporting granular packet control functions.

NS2, despite more popular in simulation arena, can also be used as a limited-functionality emulator. In contrast, a typical network emulator such as WANsim [76] s a simple bridged WAN emulator that utilizes several specialized Linux kernel-layer functionalities.

On the other hand, the open source GNS3 environment [77], developed in Python and supporting distributed multi-host deployment of its hypervisor engines, namely: Dynamips, Qemu, and VirtuaBox allows real physical machines to be integrated and mixed with the virtualized ones within the simulation environment. These specialized hypervisors can be used to integrate real network equipment's images from several vendors (e.g., Cisco and Juniper) together with Unix/Linux or MS-Windows hosts, each running on a dedicated VM. Such VMs can be hosted by a single server or run on different networked machines as well as within a public or private cloud, according to a fully distributed emulation schema.

1.4.3 Analytical Models-Focused Approaches

The definition of proper comprehensive analytical models for Big Data systems suffers as well the scale. Classical state space-based techniques (such as Petri nets-based approaches) generate huge state spaces, which are nontreatable in the solution phase, if not exploiting (or forcing) symmetries, reductions, strong assumption, or narrow aspects of the problem. In general, a faithful modeling requires an enormous number of variables (and equations), which is hardly manageable if not with analogous reductions or with the support of tools, or by having a hierarchical modeling method, based on overall simplified models that use the results of small, partial models to compensate approximations.

Literature proposes different analytical techniques, sometimes focused on part of the architecture.

As the network is a limiting factor in modern massively distributed systems, data transfers have been targeted in order to get traffic profiles over interconnection networks. Some realistic Big Data applications have been studied in [78], which points out communication modeling as foundation on which more complete performance models can be developed. Similarly [79] found the analysis on communication patterns, which are shaped by means of hardware support to obtain sound parameters over time.

A classical mathematical analytical description is chosen in [80] and in [81, 82], in which "Resource Usage Equations" are developed to take into account the influence on performances of large datasets in different scenarios. Similarly, [83] presents a rich analytical framework suitable for performance prediction in scientific

applications. Other sound examples of predictive analytical model dedicated to large-scale applications is in [84], which presents the SAGE case study, and [85], which focus on load performance prediction.

An interesting approximate approach, suitable for the generation of analytical stochastic models for systems with a very high number of components, is presented, in various applications related to Big Data, in [40–43, 46, 86, 87]. The authors deal with different aspects of Big Data architectures by applying Mean Field Analysis and Markovian agents, exploiting the property of these methods to exploit symmetry to obtain a better approximation as much as the number of components grows. This can be also seen as a compositional approach, i.e. an approach in which complex analytical models can be obtained by proper compositions of simpler model according to certain given rules. An example is in [88] that deals with performance scaling analysis of distributed data-intensive web applications. Multiformalism approaches, such as [41, 86, 87], can also fall in this category.

Within the category of analytical techniques we finally include two diverse approaches, which are not based on classical dynamical equations or variations. In [89] workload performances is derived by means of a black box approach, which observes a system to obtain, by means of regression trees, suitable model parameters from samples of its actual dynamics, updating them at major changes. In [90] resource bottlenecks are used to understand and optimize data movements and execution time with a shortest needed time logic, with the aim of obtaining optimistic performance models for MapReduce applications that have been proven effective in assessing the Google and Hadoop MapReduce implementations.

1.5 An Integrated Modeling Methodology

At the best of our knowledge, a critical review of the available literature leads us to conclude that there is no silver bullet, nor it is likely to pop up in the future, which can comprehensively and consistently become the unique reference to support performance design in Big Data systems, due to the trade off between the goals of users and administrators, which proposes on a bigger picture the latency versus throughput balance.

In fact, the analysis of the literature confirms that the issues behind Big Data architectures have to be considered not only at different levels, but with a multiplicity of points of view. The authors agree generally on the main lines of the principles behind an effective approach to modeling and analysis, but their actual detail focuses spread on different aspects of the problem, scattering the effort as a complex mosaic of particulars, in which the different proposals are articulated.

As seen, besides the obvious classification presented in Sect. 1.4, a main, essential bifurcation between rough classes of approaches can be connected to the prevalent stakeholder. Users are obviously interested in binding the analysis to a single application, or a single application class, thus considering it in isolation or as it were the main reference of the system, which is supposed to be optimized around it. While

such a position is clearly not justifiable if a cloud-based use of an extended architecture, this cannot be intended as obviously restrictive when a cloud-based architecture is dedicated to Big Data use, as the scale of the application and the scheduling of the platform play a very relevant role in evaluating this assumption. In principle, if the data to be processed are enough and independent enough to be successfully organized so that the computation can effectively span over all, or the most, of the available nodes, and the application can scale up sufficiently and needs a non negligible execution time during this massively parallel phase, there is at least a very significant period of usage of the architecture that sees an optimal exploitation of the system if the system is optimized for that application. If the runs of such an applications are recurring, it makes absolutely sense to consider the lifespan of the architecture as organized in phases, to be analyzed, thus modeled, differently one from the other (at the cost of raising some question about the optimal modeling of transitions between phases and their cost). Conversely, if the span of the application, in terms of execution time or span of needed resources, is a fraction of the workload, the point of view of a single user (that is, a single application) is still important, but seems not sufficiently prevalent to influence the assessment of the whole system, so the modeling and evaluation process of the architecture.

If many applications coexist during a same phase of the life of the system, which can be assumed as the general case, the user point of view should leave the place of honor to the administrator point of view. The administrator here considered is of course an abstract figure including all the team that is responsible for managing with all the aspects of the care and efficiency of the architecture, be it a dedicated system, a data center, a federation of data centers, or a multicloud, including those aspect that are not bound to technical administration, maintenance, evolution and management but are rather related to profitability, budgeting, and commercial strategies in general. Analogously, also the throughput concept should be considered in a generalized, even if with informal meaning and with a macroscopic abuse of notation, abstract way, which also encompasses the commercial part of the administrator concerns. The focus is thus on the system as a whole, and on related metrics, but anyway the goal can be classified as multi-objective and the performance specifications must be reconducted to the factors that allow to keep all applications within their tolerable range of requirements while maximizing the overall, generalized throughput of the system.

It is though necessary to model microscopic and macroscopic aspects of the systems, including all its components: hardware, operating systems, network infrastructure, communication protocols, middleware, resource scheduling policies, applications, usage patterns, workloads. This is possible in principle on existing systems, or can be designed as a set of sets of specifications for non existing systems. In order to keep realism, the most of the modeling process must rely on analogies: with other existing systems, with well known, even if coarsely understood, macroscopic characteristics of the dynamics of the system, the users and the workload, with available information about parts of the system that are already available or anyway are specified with a higher level of detail. This pushes somehow back the problem into the domain of analysis.

Anyway, the heaviness of the scale of the problem may be relieved by exploiting an expectable degree of symmetry, due to the fact that, for practical reasons, the structure of huge architectures is generally modular: it is quite unlikely that all computing nodes are different, that there is a high lack of homogeneity in the operating systems that govern them, that the network architecture is not regularly structured and organized, that parts of a same Big Data application are executed on completely different environments. This inclination towards homogeneity is a reasonable hypothesis, as it stems from several factors.

A first factor is rooted into commercial and administrative causes. The actual diversity of equivalent products in catalogs (excluding minor configuration variants, or marketing oriented choices) is quite low, also because of the reduced number of important component producers for memories, processors and storage devices that are suitable for heavy duty use. A similar argument can be asserted for operating systems, even if configurations may vary in lots of parameters, and for middleware, which yet must offer a homogeneous abstraction to the application layer, and is probably to be rather considered an unification factor. Additionally, system management and maintenance policies benefit from homogeneity and regularity of configurations, so it is safe to hypothesize that the need for keeping the system manageable pushes towards behaviors that tend to reduce the heterogeneity of system components and allows a class based approach to the enumeration of the elements of the system that need to be modeled.

Our working assumption is thus that we can always leverage the existence of a given number of classes of similar components in a Big Data system, including hardware, software, and users, which allows to dominate the scale problem, at least in a given time frame that we may label as epoch, and obtain a significant model of the system in an epoch.

It is sufficiently evident that, in the practical exercise of a real Big Data system, classes representing hardware components (and, to some extent, operating system and middleware) will be kept through the epochs for a long period of time, as physical reconfigurations are rather infrequent with respect to the rate of variability of the application bouquet and workload, while classes representing applications may significantly vary between epochs.

A modeling technique that exhibits a compositional feature may exploit this class oriented organization, allowing the design of easily scalable models by a simple proper assembly of classes, eventually defined by specialists of the various aspects of the system. A compositional class oriented organization offers thus a double advantage, which is a good start in the quest for a sound modeling methodology: a simplification of the organizational complexity model and a flexible working method.

In fact, the resulting working method is flexible both with respect to the efficiency of the management of the modeling construction process and the possibility of using a design strategy based on prototypes and evolution. In other words, such an approach enables a team to work in parallel on different specialized parts of the model, to speed up the design process and to let every specialized expert free of an independent contribution under the supervision of the modeling specialist; and allows the model

to be obtained as a growing set of refinable and extendable modeling classes[2] that may be checked and verified and reused before the availability of the whole model.

A class-based modeling approach with these characteristics is then suitable to become the core of a structured modeling and analysis methodology that must necessarily include some ancillary prodromic and conclusive complementary steps, to feed the model with proper parameters and to produce the means to support the decision phase in the system development process: anyway, the approach needs a solid and consistent foundation in a numerical, analytical, or simulative support for the actual evaluation of the behaviors of the system. It is here that the scale of the system dramatically manifests its overwhelming influence, because, as seen in Sect. 1.4, analytical (and generally numerical as well) tools are likely to easily meet their practical or asymptotic limitations, and simulative tools need enormous time and a complex management to produce significant results. In our opinion, a significant solution is the adoption of Markovian Agents as backing tool for the modeling phase, as they exhibit all the features here postulated as successful for the goals, while other traditional monitoring tools, complemented in case with traditional simulation or analytic tools, are needed to support the prodromic steps and/or the conclusive steps.

1.5.1 Markovian Agents for Big Data Architectures

Markovian Agents are a modeling formalism tailored to describe systems composed by a large number of interacting agents. Each one is characterized by a set of *states*, and it behaves in a way similar to Stochastic Automata, and in particular to Continuous Time Markov Chains (CTMCs). The state transitions of the models can be partitioned into two different types: *local transitions* and *induced transitions*. The former represent the local behavior of the objects: they are characterized by an infinitesimal generator that is independent of the interaction with the other agents. Differently from CTMCs, the local behavior of MAs also includes self-loop transitions: a specific notation is thus required since this type of transition cannot be included in conventional infinitesimal generators [91]. Self-loop transitions can be used to influence the behavior of other agents. Induced transitions are caused by the interaction with the other MAs. In this case, the complete state of the model *induces* agents to change their state

Formally, a Markovian Agent Model (MAM) is a collection of *Markovian Agents* (MAs) distributed across a set of locations $\mathcal{V}$. Agents can belong to different classes $c \in \boldsymbol{C}$, each one representing a different agent behavior. In Big Data-oriented applications, ante classes are used to model different types of application requirements or different steps of map-reduce jobs and so-on. In general space $\mathcal{V}$ can be either discrete

[2]The term "class" is here intended to define a self contained model element that captures the relevant features of a set (a class, as in the discussion in the first past of this section) of similar parts of the system, and should not be confused with software class as defined in object oriented software development methodologies, although in principle there may be similarities.

or continuous: when modeling Big Data-oriented applications, $\mathcal{V} = \{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots \boldsymbol{v_N}\}$ is a set of locations $\boldsymbol{v_i}$. Usually locations represents component of a cloud infrastructure: they can range from nodes to racks, corridors, availability zones and even regions. MAM can be analyzed studying the evolution of $p_j^{\{c\}}(t, \boldsymbol{v})$: the probability that a class c agent is in state $1 \leq j \leq n^{\{c\}}$ at time t, at location $\boldsymbol{v} \in \mathcal{V}$. In order to tackle the complexity of the considered systems, we use *counting process* and we exploit the mean field approximation [92, 93], which states that, if the evolution of the agents depends only on the count of agents in a given state, then $p_j^{\{c\}}(t, \boldsymbol{v})$ tends to be deterministic and to depend only on the mean count of the number of agents. In particular, let as call $\rho^{\{c\}}(t, \boldsymbol{v})$ the total number of class c agents in a location $\boldsymbol{v}$ at time t. Let us also call $\pi_j^{\{c\}}(t, \boldsymbol{v}) = p_j^{\{c\}}(t, \boldsymbol{v}) \cdot \rho^{\{c\}}(t, \boldsymbol{v})$ the density of class c agents in state j at location $\boldsymbol{v}$ and time t. Note that if each location has exactly one agent, we have $\pi_j^{\{c\}}(t, \boldsymbol{v}) = p_j^{\{c\}}(t, \boldsymbol{v})$. We call *static* a MAM in which $\rho(t, \boldsymbol{v})$ does not depend on time, and *dynamic* otherwise.

The state distribution of a class c MA in position $\boldsymbol{v}$ at time t is thus described by row vector $\boldsymbol{\pi}^{\{c\}}(t, \boldsymbol{v}) = |\pi_j^{\{c\}}(t, \boldsymbol{v})|$. We also call $\Pi_{\mathcal{V}}(t) = \{(c, \boldsymbol{v}, \boldsymbol{\pi}^{\{c\}}(t, \boldsymbol{v})) : 1 \leq c \leq C, \boldsymbol{v} \in \mathcal{V}\}$ the ensemble of the probability distribution of all the agents of all the classes at time t. We can use the following equation to described the evolution of the agents:

$$\frac{d\boldsymbol{\pi}^{\{c\}}(t, \boldsymbol{v})}{dt} = \boldsymbol{\nu}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}}) + \boldsymbol{\pi}^{\{c\}}(t, \boldsymbol{v}) \cdot \boldsymbol{K}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}}). \tag{1.1}$$

Term $\boldsymbol{\nu}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ is the *increase* kernel and $\boldsymbol{K}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ is the *transition* kernel. They can both either depend on the class c, on the position $\boldsymbol{v}$, and on the time t. Moreover to allow induction, they can also depend on the ensemble probability $\Pi_{\mathcal{V}}$. The increase kernel $\boldsymbol{\nu}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ can be further subdivided into two terms:

$$\boldsymbol{\nu}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}}) = \boldsymbol{b}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}}) + \boldsymbol{m}_{[in]}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}}). \tag{1.2}$$

Kernel $\boldsymbol{\nu}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ model the increase of the number of agents in a point in space. It component $\boldsymbol{b}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ is addressed as the *birth* term, and it is used to model the generation of agents. It is measured in agents per time unit, and expresses the rate at which class c agents are created in location $\boldsymbol{v}$ at time t. In Big Data models where agents represents virtual machines or map-reduce tasks, the birth term can be used to describe the launch of new instances or the submissions of new jobs to the system. Term $\boldsymbol{m}_{[in]}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ is the *input* term, and accounts for class c agents that moves into location $\boldsymbol{v}$ at time t from other points in space. In the considered Big Data scenario, it can be used to model the start of new virtual machines due to a migration process.

The transition kernel $\boldsymbol{K}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ can be subdivided into four terms:

$$\boldsymbol{K}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}}) = \boldsymbol{Q}^{\{c\}}(t, \boldsymbol{v}) + \mathcal{I}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}}) + -\boldsymbol{D}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}}) - \boldsymbol{M}^{\{c\}}_{[out]}(t, \boldsymbol{v}, \Pi_{\mathcal{V}}). \tag{1.3}$$

It is used to model both the state transitions of the agents, and the effects that reduces the number of agents in one location $\boldsymbol{v}$. *Local transitions* are defined by matrix $\boldsymbol{Q}^{\{c\}}(t, \boldsymbol{v}) = |q^{\{c\}}_{ij}(t, \boldsymbol{v})|$, where $q^{\{c\}}_{ij}(t, \boldsymbol{v})$ defines the rate at which a class c agents jumps from state i to state j for an agent position $\boldsymbol{v}$ at time t. In Big Data application, it is used to model the internal actions of the agents: for example, it can model the failure-repair cycle of a storage unit, or the acquisition or release of resources such as RAM in a computation node. The *influence matrix* $\mathcal{I}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ expresses the rate of *induced transitions*. Its elements $\mathcal{I}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ can depend on the state probabilities of the other agents in the model, and must be defined in a way that preserves the infinitesimal generator matrix property for $\boldsymbol{Q}^{\{c\}}(t, \boldsymbol{v}) + \mathcal{I}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$. In Big Data applications, they can model advanced scheduling policies that stop or start nodes in a given section of a data center to reduce the cooling costs, or the reconstruction of broken storage blocks from the surviving ones using erasure conding. The *death* of agents is described by diagonal matrix $\boldsymbol{D}^{\{c\}}(t, \boldsymbol{v})$. Its elements $d^{\{c\}}_{ii}(t, \boldsymbol{v})$ represent the rate at which class c agents in state i on location $\boldsymbol{v}$ at time t leaves the model. In Big Data models they can be used to describe the termination of virtual machines, the completion of map-reduce tasks or jobs, and the loss of storage blocks due to the lack of enough surviving data and parity blocks to make the erasure code effective. Finally, Matrix $\boldsymbol{M}^{\{c\}}_{[out]}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ is the output counterpart of vector $\boldsymbol{m}^{\{c\}}_{[in]}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ previously introduced. It is a matrix whose terms $m^{out:\{c\}}_{ij}(t, \boldsymbol{v})$ consider the output for a class c agent from a location $\boldsymbol{v}$ at time t. If $i = j$, the change of location does not causes a change of state. Otherwise the state of the agent changes from i to j during its motion. To maintain constant the number of agents, $\boldsymbol{M}^{\{c\}}_{[out]}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ and $\boldsymbol{m}^{\{c\}}_{[in]}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ are related by specific conservation laws. For instance, the two terms could be related such that $\boldsymbol{m}^{\{c\}}_{[in]}(t, \boldsymbol{u}, \Pi_{\mathcal{V}}) = |\lambda, \ldots|\boldsymbol{\pi}^{\{c\}}(t, \boldsymbol{v})$ and $\boldsymbol{M}^{\{c\}}_{[out]}(t, \boldsymbol{v}, \Pi_{\mathcal{V}}) = diag(\lambda, \ldots)$.

MAMs are also characterized by the initial state of the system. In particular, $\rho^{\{c\}}(0, \boldsymbol{v})$, represents the initial density of class c agents in location $\boldsymbol{v}$, and $p^{\{c\}}_j(0, \boldsymbol{v})$, the corresponding initial state probability. The initial condition of Eq. (1.1) can then be expressed as:

$$\pi^{\{c\}}_j(0, \boldsymbol{v}) = p^{\{c\}}_j(0, \boldsymbol{v}) \cdot \rho^{\{c\}}(0, \boldsymbol{v}). \tag{1.4}$$

1.5.2 A Case Study

In the case study proposed in [40] locations are used to model different data centers of a geographically distributed cloud infrastructure. Locations $\mathcal{V} = \{dc_1, dc_2, \ldots\}$ are used to model regions and availability zones of the data centers composing the

infrastructure. Agents are used to model computational nodes that are able to run Virtual Machines (VMs), and storage units capable of saving data blocks (SBs). Different classes $1 \leq c \leq C$ are used to represent the applications running in the system, where the states of the agents characterize the resource usage of each type of application. In particular, the agent density function $\rho^{\{c\}}(t, dc_j)$ determines the number of class c applications running in data center dc_j.

The transition kernel $\tilde{\boldsymbol{K}}^{\{c\}}(\Pi_{\mathcal{V}})$ models the computational and storage speed of each application class as function of the resources used. In particular, the local transition kernel $\boldsymbol{Q}^{\{c\}}(t, \boldsymbol{v}) = 0$ since the speed at which application acquires and releases resources depends on the entire state of the data center, and $\tilde{\boldsymbol{K}}^{\{c\}}(\Pi_{\mathcal{V}}) = \mathcal{I}^{\{c\}}(t, \boldsymbol{v})$.

If we consider batch processing, where a fixed number of applications is continuously run, the birth term and death term are set to $\boldsymbol{b}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}}) = 0$ and $\boldsymbol{D}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}}) = 0$. If we consider applications that can be started and stopped, such as web or application servers in an auto-scaling framework, $\boldsymbol{b}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ defines the rate at which new VMs are activated, and the terms $1/d_{ii}^{\{c\}}(t, \boldsymbol{v})$ of $\boldsymbol{D}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ defines the average running time of a VM. As introduced, application migration can be modeled using terms $\boldsymbol{M}_{[out]}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$ and $\boldsymbol{m}_{[in]}^{\{c\}}(t, \boldsymbol{v}, \Pi_{\mathcal{V}})$. In particular they can describe the rate at which applications are moved from one data center to another to support load-balancing applications that works at the geographical infrastructure level.

1.5.3 Designing New Systems

We already presented in Sect. 1.4 some references showing the value and the effectiveness of Markovian Agents for big scale applications. To illustrate the applicability of a Markovian Agents model based approach, we propose here a structured methodology, based on the analysis and the considerations previously presented in this Section, suitable for supporting the design of a new Big Data-oriented system from scratch.

The methodology is organized into 8 steps, on which iterations may happen until a satisfactory result is reached in the final step. Figure 1.1 shows an ideal, linear path along the steps.

In this case, as there is no existing part of the system, everything has to be designed, thus a fundamental role is played by the definition of the target. This is done in the first step.

The first step is composed of 3 analogous activities, aiming at structuring hypotheses on the workload, the computing architecture and the network architecture of the target system. The 3 activities are not independent, but loosely coupled, and may be under the responsibility of 3 different experts, which may be identified in the overall responsible of the facility, the system architect, or administrator and the network architect, or administrator. The first task is probably the most sensitive, as it needs, besides the technical skills, awareness about the business perspectives and the plans

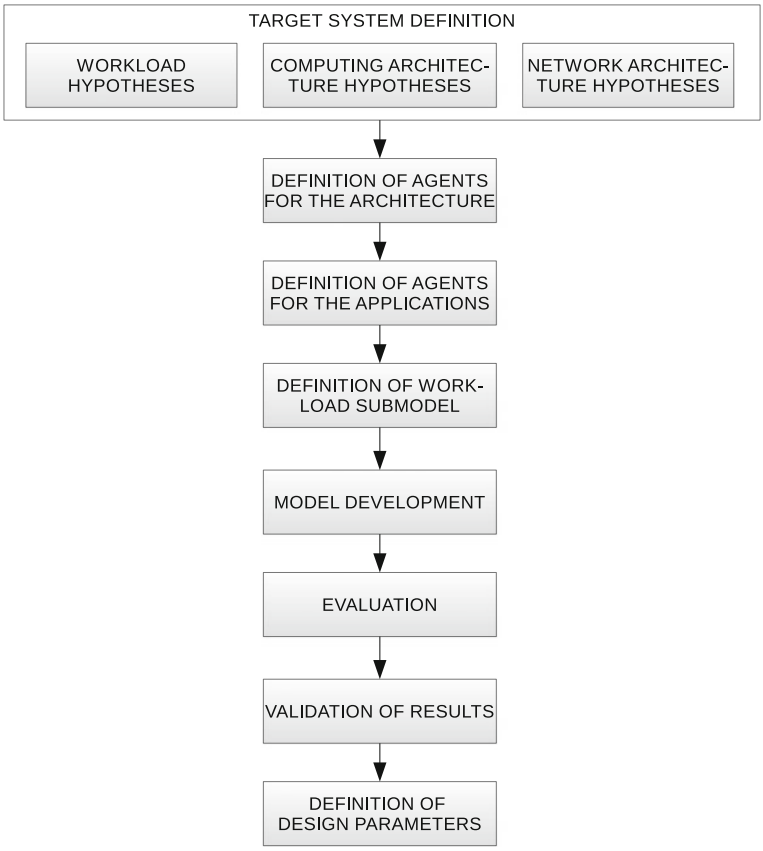


Fig. 1.1 Design steps for a new system

related to the short and medium term of the facility, including the sustainability constraints. The second task is not less critical, but is partially shielded by the first about the most relevant responsibilities, and it is essentially technical. While hypothesizing the computing infrastructure, including operating systems and middleware, the most important management issues have to be kept into account, e.g., maintenance needs and procedures. The third is analogous to the second, even if possible choices about the network architecture are generally less free than the ones related to the computing architecture. An important factor related to network hypotheses is bound to storage management, as network bandwidth and resource allocation can heavily impact and be influenced by the choices about storage organization and implementation. The hypotheses can be performed by using existing knowledge about similar systems or applications, and may be supported by small scale or coarse grain analytical, numerical or simulation solutions (such as the ones presented in Sect. 1.4). The outcomes of this step consist of a first order qualitative model of the 3 components, with quantita-

tive hypotheses on the macroscopic parameters of the system, sketching the classes of the components.

The second step consists of the development of the agents needed to simulate the architecture. In this phase the outcomes of the first step are detailed into Markovian Agents submodels, by defining their internal structure, the overall macrostructure of the architecture and the embedded communication patterns, and by converting the quantitative hypotheses from the first step into local model parameters. When a satisfactory set of agents is available, classes are mapped onto the agent set. The outcome is the set of agents that is sufficient to fully represent the architecture and its behaviors, together with the needed documentation.

The third step is an analogous of the second one, with the difference that the agents should now include the variability of the applications within and between epochs, defining all reference application classes and the set of architectural agents that are potentially involved by their execution. The outcome is the set of agents that is sufficient to fully represent the various classes of applications that will run on the system, together with the needed documentation.

The fourth step consists of the definition of agents representing the activation patterns of the application agents. Here are included users, data generated by the environment, external factors that may impact onto the activation patterns (including, in case, what needed to evaluate the availability and the dependability of the system). The outcome is the set of agents that is sufficient to fully represent the activation patterns of all other agents representing the system, together with the needed documentation.

In the fifth step a model per epoch is defined, by instantiating agents with the needed multiplicity and setting up the start up parameters. Every model (a single one in the following, for the sake of simplicity) is checked to ensure that it actually represents the desired scenario. The outcome is a model that is ready for the evaluation.

In the sixth step the model is evaluated, and proper campaigns are run to obtain significant test cases that are sufficient to verify the model and to define suitable parameters sets that support the final decision. The outcomes consist of the results of the evaluation, in terms of values for all the target metrics.

The seventh step is the evaluation of results with the help of domain experts, to check their trustability and accept the model as correct and ready to be used as a decision support tool. The outcome is an acceptance, or otherwise a revision plan that properly leads back to the previous steps according to the problem found in it.

The last step is the definition of the final design parameters, which allow to correctly instantiate the design.

1.5.4 Evolving Existing Systems

The same ideas may be applied to a structured methodology for supporting the enhancement and reengineering process or an existing architecture. In this case,

the system is already available for a thorough analysis, and traces and historical information about its behaviors provide a valuable resource to be harvested to produce a solid base on which a good model can be structured, with the significant advantage that the available information is obtained on the same real system. In this case, a precious tool is provided by monitoring techniques like the ones presented in Sect. 1.4.

The methodology is organized into 12 steps, on which iterations may happen until a satisfactory result is reached in the final step. Figure 1.1 shows an ideal, linear path along the steps, similarly to what presented in the previous case.

The first step is dedicated to understanding the actual workload of the system. This is of paramount importance, as the need for evolving the system stems from the inadequacy of the system to successfully performing what required by the existing workload, or because of additional workload that may be needed to be integrated to the existing one, which in turn is probably dominant, as it is likely to be composed by an aggregate of applications. The outcomes of this step is a complete characterization of the workload (Fig. 1.2).

In the second step all components are analyzed, exploiting existing data about the system and the influence of the workload, in order to obtain a set of parameters for each component that characterizes it and allows a classification. The outcomes are this set of characterizations and the classification.

The third step is analogous to the second step of the previous case, with the advantage of using actual data in place of estimations, obtained in the previous step. The outcomes are constituted by the agents that describe the components of the system.

The fourth step is analogous to the third step of the previous case, with the same advantages resulting from a complete knowledge of the existing situation. As in the previous step, the agents describing the applications are the outcomes.

In the fifth step the outcomes from the first step are used to define the agents that describe the workload, similarly as what seen for the fourth step of the previous case. Also in this case, agents are the outcomes.

In the sixth step the model is defined, with the significant advantage that it is supposed to represent the existing system and is thus relatively easy to perform tunings with comparisons with the reality. The outcome consists of the model itself.

The seventh step is dedicated to the validation of the model, which benefits from the availability of real traces and historical data. This avoids the need for experts, as everything can be checked by internal professionals, and raises the quality of the process. The outcome is a validated model.

The eight step is dedicated to the definition of the agents that describe the desired extensions to the system. This can be done by reusing existing agents with different parameters or designing new agents from scratch. The outcome is an additional set of agents, designed to be coherent with the existing model, which describe the new components that are supposed to be added or replaced in the system.

The ninth step is devoted with the extension of the model with a proper instantiation of the new agents, and the needed modifications. The outcome is the extended model.

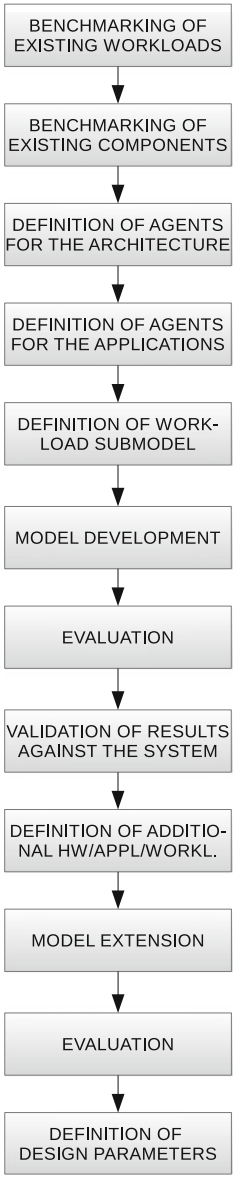


Fig. 1.2 Design steps for evolving an existing system

In the tenth step the new model is used to evaluate the new behavior of the extended system, to support the decision process. The model is used to explore the best parameters with the hypothesized architecture and organization. The outcome is the decision, which implies a sort of validation of the results, of a rebuttal of the new model, with consequent redefinition of the extensions and partial replay of the process.

The last step is, again, the definition of the final design parameters, which allow to correctly instantiate the design.

1.6 Conclusions

The emerging interest on big data architectures is the straightforward consequence of a modern fully digitalized society, where more and more people, but also independent and unattended applications and machines produce, utilize and exchange large amount of data all over the world. The origins of such a global "digitalization" process may be found in several trends and phenomena, affecting almost any sector of our society and everyday life, ranging from the proliferation of the data sources available on the Internet to the ever growing adoption of digital data representation and storage formats as well as from the new available ways for sharing and obtaining information through flexible and powerful semantic/social networking organizations, to the rapid evolution and widespread deployment of the networking devices with their ubiquitous and pervasive interconnection through high speed communication infrastructures. In such scenario, the "always connected life" paradigm, fostered by the diffusion of mobile personal communication devices has further propelled the need of new architectures specially crafted for capturing, combining, correlating and analyzing massive amount of heterogeneous and unstructured data coming from everywhere on the Internet.

While the exploitation of the potential of these big data architectures can bring unique and unforeseen opportunities for improving the social conditions as well as the quality of life (e.g., in surveillance, finance, healthcare, etc.), resulting in a real "quantum leap" in the efficiency of any kind of IT process or workflow, their performance is still a partially uncharted territory, due to the extension of the problem perimeter and extreme complexity of all the involved factors and technology. Accordingly, we faced this challenge by analyzing the main issues related to Big Data Systems, together with a methodological proposal for a modeling and performance analysis approach that is able to scale up sufficiently while providing an efficient analysis process.

Our effort resulted in a glance over the main aspects of performance evaluation for Big Data architectures, by providing examples of model based evaluation, in order to show how it is possible to characterize these large scale architectures in order to support their correct management and capacity planning practices.

Acknowledgments We would like to thank Dr E. Barbierato for his precious comments, that helped us to improve the quality of this chapter.

References

1. Fiore, U., Palmieri, F., Castiglione, A., De Santis, A.: A cluster-based data-centric model for network-aware task scheduling in distributed systems. Int. J. Parallel Prog. **42**(5), 755–775 (2014)
2. Wu, Y., Li, G., Wang, L., Ma, Y., Kolodziej, J., Khan, S.U.: A review of data intensive computing. In: The 12th IEEE International Conference on Scalable Computing and Communications (ScalCom 2012), IEEE (Dec 2012)
3. Madden, S.: From databases to Big Data. IEEE Int. Comput. **16**(3), 4–6 (2012)
4. Bertino, E., Bernstein, P., Agrawal, D., Davidson, S., Dayal, U., Franklin, M., Gehrke, J., Haas, L., Halevy, A., Han, J., et al.: Challenges and Opportunities with Big Data. (2011)
5. Fu, Y., Jiang, H., Xiao, N.: A scalable inline cluster deduplication framework for Big Data protection. In: Proceedings of the 13th International Middleware Conference. Middleware '12, pp. 354–373. Springer, New York (2012)
6. Bryant, R.E., Katz, R.H., Lazowska, E.D.: Big-data computing: Creating revolutionary breakthroughs in commerce, science, and society. In: Computing Research Initiatives for the 21st Century. Computing Research Association (2008)
7. deRoos, D., Eaton, C., Lapis, G., Zikopoulos, P., Deutsch, T.: Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. 1st edn. McGraw-Hill Osborne Media (2011)
8. Inacio, E.C., Dantas, M.A.R.: A survey into performance and energy efficiency in hpc, cloud and big data environments. IJNVO **14**(4), 299–318 (2014)
9. Regola, N., Cieslak, D.A., Chawla, N.V.: The need to consider hardware selection when designing big data applications supported by metadata. In: Hu, W.C., Kaabouch, N. (eds.) Big Data Management, Technologies, and Applications. IGI Global pp. 381–396. (2014)
10. Majeed, A., Shah, M.A.: Energy efficiency in big data complex systems: a comprehensive survey of modern energy saving techniques. Complex Adapt. Syst. Model. **3**(1), 1–29 (2015)
11. Apache Hadoop: Apache Hadoop web site
12. White, T.: Hadoop: The Definitive Guide. 1st edn. O'Reilly Media, Inc. (2009)
13. Isard, M., Budiu, M., Yu, Y., Birrell, A., Fetterly, D.: Dryad: distributed data-parallel programs from sequential building blocks. In: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007. EuroSys '07, pp. 59–72. ACM, New York, NY, USA (2007)
14. Oozie: Oozie web site (2011)
15. Amazon Inc.: Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/#pricing (2008)
16. Rackspace, US Inc.: The Rackspace Cloud. http://www.rackspace.com/cloud/ (2010)
17. Jung, G., Gnanasambandam, N., Mukherjee, T.: Synchronous parallel processing of big-data analytics services to optimize performance in federated clouds. In: Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing. CLOUD '12, 811–818. Washington, DC, USA, IEEE Computer Society (2012)
18. Weatherspoon, H., Kubiatowicz, J.: Erasure coding vs. replication: A quantitative comparison. In: Revised Papers from the First International Workshop on Peer-to-Peer Systems. IPTPS '01, pp. 328–338. Springer, London, UK, (2002)
19. Kameyama, H., Sato, Y.: Erasure codes with small overhead factor and their distributed storage applications. In: 41st Annual Conference on Information Sciences and Systems, 2007. CISS '07, pp. 80–85 (March 2007)
20. Dandoush, A., Alouf, S., Nain, P.: Simulation analysis of download and recovery processes in P2P storage systems. In: 21st International Teletraffic Congress, 2009. ITC 21 2009, pp. 1–8 (Sept 2009)
21. Aguilera, M., Janakiraman, R., Xu, L.: Using erasure codes efficiently for storage in a distributed system. In: Proceedings of the International Conference on Dependable Systems and Networks, 2005. DSN 2005, pp. 336–345 (June 2005)

22. Wu, F., Qiu, T., Chen, Y., Chen, G.: Redundancy schemes for high availability in dhts. In: Pan, Y., Chen, D., Guo, M., Cao, J., Dongarra, J. (eds.) ISPA. Volume 3758 of Lecture Notes in Computer Science., pp. 990–1000. Springer (2005)
23. Rodrigues, R., Liskov, B.: High availability in dhts: Erasure coding vs. replication. In: Peer-to-Peer Systems IV 4th International Workshop IPTPS 2005, Ithaca, New York (Feb 2005)
24. Xiang, Y., Lan, T., Aggarwal, V., Chen, Y.F.R.: Joint latency and cost optimization for erasure-coded data center storage. SIGMETRICS Perform. Eval. Rev. **42**(2), 3–14 (2014)
25. Sathiamoorthy, M., Asteris, M., Papailiopoulos, D., Dimakis, A.G., Vadali, R., Chen, S., Borthakur, D.: Xoring elephants: novel erasure codes for big data. In: Proceedings of the 39th International Conference on Very Large Data Bases. PVLDB'13, VLDB Endowment pp. 325–336 (2013)
26. Lian, Q., Chen, W., Zhang, Z.: On the impact of replica placement to the reliability of distributed brick storage systems. In: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, 2005. ICDCS 2005, pp. 187–196 (June 2005)
27. Simon, V., Monnet, S., Feuillet, M., Robert, P., Sens, P.: SPLAD: scattering and placing data replicas to enhance long-term durability. Rapport de recherche RR-8533, INRIA (May 2014)
28. Gribaudo, M., Iacono, M., Manini, D.: Modeling replication and erasure coding in large scale distributed storage systems based on CEPH. In: ITAIS2015: Proceedings of XII Conference of the Italian Chapter of AIS. Volume to Appear of Lecture Notes in Information Systems and Organisation. Springer, Berlin, Heidelberg (2016)
29. Gribaudo, M., Iacono, M., Manini, D.: Improving reliability and performances in large scale distributed applications with erasure codes and replication. Future Gener. Comput. Syst. **56**, 773–782 (2016)
30. Apache Cassandra: Apache Cassandra web site (2009)
31. MongoDB: MongoDB web site (2011)
32. Apache HBase: Apache HBase web site
33. Gandini, A., Gribaudo, M., Knottenbelt, W.J., Osman, R., Piazzolla, P.: Performance Evaluation of NoSQL Databases. In: Proceedings of the Computer Performance Engineering: 11th European Workshop, EPEW 2014, Florence, Italy, September 11-12, 2014, pp. 16–29. Springer International Publishing, Cham (2014)
34. Palmieri, F., Pardi, S. In: Enhanced Network Support for Scalable Computing Clouds. Volume 0 of Computer Communications and Networks. pp. 127–144. Springer, London (2010)
35. Chowdhury, M., Zaharia, M., Ma, J., Jordan, M., Stoica, I.: Managing data transfers in computer clusters with orchestra. SIGCOMM-Comput. Commun. Rev. **41**(4), 98–109 (2011)
36. Tierney, B., Kissel, E., Swany, D.M., Pouyoul, E.: Efficient data transfer protocols for Big Data. In: eScience, IEEE Computer Society, pp. 1–9 (2012)
37. Zahavi, E., Keslassy, I., Kolodny, A.: Distributed adaptive routing for big-data applications running on data center networks. In: Proceedings of the Eighth ACM/IEEE Symposium on Architectures for Networking and Communications Systems. ANCS '12, pp. 99–110. ACM, New York, NY, USA (2012)
38. Palmieri, F., Pardi, S.: Towards a federated Metropolitan Area Grid environment: The SCoPE network-aware infrastructure. Future Gener. Comput. Syst. **26**(8), 1241–1256 (2010)
39. Esposito, C., Ficco, M., Palmieri, F., Castiglione, A.: Interconnecting federated clouds by using publish-subscribe service. Cluster Comput. **16**(4), 887–903 (2013)
40. Castiglione, A., Gribaudo, M., Iacono, M., Palmieri, F.: Modeling performances of concurrent big data applications. Softw.: Pract. Experience **45**(8), 1127–1144 (2015)
41. Barbierato, E., Gribaudo, M., Iacono, M.: Performance evaluation of NoSQL Big Data applications using multi-formalism models. Future Gener. Comput. Syst. **37**, 345–353 (2014)
42. Castiglione, A., Gribaudo, M., Iacono, M., Palmieri, F.: Exploiting mean field analysis to model performances of big data architectures. Future Gener. Comput. Syst. **37**, 203–211 (2014)
43. Cerotti, D., Gribaudo, M., Iacono, M., Piazzolla, P.: Modeling and analysis of performances for concurrent multithread applications on multicore and graphics processing unit systems. Concurrency Comput.: Pract. Experience **28**(2), 438–452 cpe.3504 (2016)

44. Xu, L., Cipar, J., Krevat, E., Tumanov, A., Gupta, N., Kozuch, M.A., Ganger, G.R.: Agility and performance in elastic distributed storage. Trans. Storage **10**(4), 16:1–16:27 (2016)
45. Yan, F., Riska, A., Smirni, E.: Fast eventual consistency with performance guarantees for distributed storage. In: 32nd International Conference on Distributed Computing Systems Workshops (ICDCSW), 2012, pp. 23–28 (June 2012)
46. Barbierato, E., Gribaudo, M., Iacono, M.: Modeling and evaluating the effects of Big Data storage resource allocation in global scale cloud architectures. Int. J. Data Warehous. Min. **12**(2), 1–20 (2016)
47. Duan, S., Thummala, V., Babu, S.: Tuning database configuration parameters with iTuned. Proc. VLDB Endow. **2**(1), 1246–1257 (2009)
48. Zheng, W., Bianchini, R., Janakiraman, G.J., Santos, J.R., Turner, Y.: JustRunIt: Experiment-based management of virtualized data centers. In: Proceedings of the 2009 Conference on USENIX Annual Technical Conference. USENIX'09, pp. 18–18 USENIX Association, Berkeley, CA, USA (2009)
49. Mytilinis, I., Tsoumakos, D., Kantere, V., Nanos, A., Koziris, N.: I/O performance modeling for big data applications over cloud infrastructures. In: 2015 IEEE International Conference on Cloud Engineering, IC2E 2015, Tempe, AZ, USA, March 9–13, 2015, IEEE, pp. 201–206 (2015)
50. Shi, Y., Meng, X., Zhao, J., Hu, X., Liu, B., Wang, H.: Benchmarking cloud-based data management systems. In: Proceedings of the Second International Workshop on Cloud Data Management. CloudDB '10, pp. 47–54. ACM, New York, NY, USA (2010)
51. Boulon, J., Konwinski, A., Qi, R., Rabkin, A., Yang, E., Yang, M.: Chukwa, a large-scale monitoring system. In: Proceedings of CCA, vol. 8 (2008)
52. Tan, J., Pan, X., Marinelli, E., Kavulya, S., Gandhi, R., Narasimhan, P.: Kahuna: Problem diagnosis for mapreduce-based cloud computing environments. In: Network Operations and Management Symposium (NOMS), 2010 IEEE, IEEE, pp. 112–119 (2010)
53. Creţu-Ciocârlie, G.F., Budiu, M., Goldszmidt, M.: Hunting for problems with artemis. In: Proceedings of the First USENIX Conference on Analysis of system logs, pp. 2–2. USENIX Association (2008)
54. Kavulya, S., Tan, J., Gandhi, R., Narasimhan, P.: An analysis of traces from a production mapreduce cluster. In: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), 2010, IEEE, pp. 94–103 (2010)
55. Hellerstein, J.: Google cluster data (2010)
56. Wilkes, J.: More google cluster data (2011)
57. Cardona, K., Secretan, J., Georgiopoulos, M., Anagnostopoulos, G.: A grid based system for data mining using MapReduce. Technical report, Technical Report TR-2007-02, AMALTHEA (2007)
58. Teng, F., Yu, L., Magoulès, F.: SimMapReduce: a simulator for modeling mapreduce framework. In: 5th FTRA International Conference on Multimedia and Ubiquitous Engineering (MUE), 2011, IEEE, pp. 277–282 (2011)
59. Hammoud, S., Li, M., Liu, Y., Alham, N.K., Liu, Z.: Mrsim: A discrete event based mapreduce simulator. In: Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2010, volume 6., IEEE, pp. 2993–2997 (2010)
60. Liu, Y., Li, M., Alham, N.K., Hammoud, S.: HSim: a mapreduce simulator in enabling cloud computing. Future Gener. Comput. Syst. **29**(1), 300–308 (2013)
61. Herodotou, H., Babu, S.: Profiling, what-if analysis, and cost-based optimization of MapReduce programs. Proc. VLDB Endowment **4**(11), 1111–1122 (2011)
62. Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F.B., Babu, S.: Starfish: A self-tuning system for big data analytics. In: Proceedings of the Fifth CIDR Conference (2011)
63. Wang, G., Butt, A.R., Pandey, P., Gupta, K.: A simulation approach to evaluating design decisions in MapReduce setups. In: IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems, 2009. MASCOTS'09. IEEE, pp. 1–11 (2009)

64. Fonseca, R., Porter, G., Katz, R.H., Shenker, S., Stoica, I.: X-trace: A Pervasive Network Tracing Framework. In: Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation. NSDI'07, pp. 20–20. USENIX Association, Berkeley, CA, USA (2007)
65. Ananthanarayanan, G., Kandula, S., Greenberg, A., Stoica, I., Lu, Y., Saha, B., Harris, E.: Reining in the outliers in map-reduce clusters using mantri. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, pp. 1–16. USENIX Association (2010)
66. Chen, Y., Ganapathi, A.S., Griffith, R., Katz, R.H.: Towards Understanding Cloud Performance Tradeoffs Using Statistical Workload Analysis and Replay. Technical Report UCB/EECS-2010-81, EECS Department, University of California, Berkeley (May 2010)
67. Ficco, M., Avolio, G., Palmieri, F., Castiglione, A.: An hla-based framework for simulation of large-scale critical systems. Concurrency Comput. **28**(2), 400–419 (2016)
68. OPNET: Opnet modeler. http://www.opnet.com/. Accessed 30 April 2016
69. NS2: Ns2 official website. http://www.isi.edu/nsnam/ns/. Accessed 30 April 2016
70. NS3: Ns3 official website. http://www.nsnam.org/documents.html. Accessed30 April 2016
71. OMNeT: Omnet++ official website. http://www.omnetpp.org/. 30 April 2016
72. REAL: Real 5.0 simulator overview. http://www.cs.cornell.edu/skeshav/real/overview.html. 30 April 2016
73. SSFNet: Scalable simulation framework (ssf), ssfnet homepage. http://www.ssfnet.org/homePage.html. Accessed 30 April 2016
74. J-Sim: J-sim homepage. https://sites.google.com/site/jsimofficial/. Accessed 30 April 2016
75. QualNet: Qualnet official site. http://web.scalable-networks.com/content/qualnet. Accessed 30 April 2016
76. Wan simulators and emulators. http://www.wan-sim.net/. Accessed 30 April 2016
77. Welsh, C.: GNS3 network simulation guide. Packt Publ. (2013)
78. Sivasubramaniam, A., Singla, A., Ramachandran, U., Venkateswaran, H.: On characterizing bandwidth requirements of parallel applications. In: ACM SIGMETRICS Performance Evaluation Review, vol 23, pp. 198–207. ACM (1995)
79. Papaefstathiou, E., Kerbyson, D.J., Nudd, G.R.: A layered approach to parallel software performance prediction: A case study. (1994) Technical Report CS-RR-262
80. Schopf, J.M., Berman, F.: Performance prediction in production environments. In: Parallel Processing Symposium, 1998. IPPS/SPDP 1998. Proceedings of the First Merged International... and Symposium on Parallel and Distributed Processing 1998, IEEE, pp. 647–653 (1998)
81. Armstrong, B., Eigenmann, R.: Performance forecasting: Characterization of applications on current and future architectures. Purdue Univ. School of ECE, High-Performance Computing Lab. Technical report ECE-HPCLab-97202 (1997)
82. Armstrong, B., Eigenmann, R.: Performance forecasting: Towards a methodology for characterizing large computational applications. In: Proceedings of the 1998 International Conference on Parallel Processing, 1998, IEEE, pp. 518–525. (1998)
83. Carrington, L., Snavely, A., Gao, X., Wolter, N.: A performance prediction framework for scientific applications. Computat. Sci. ICCS 2003, pp. 701–701 (2003)
84. Kerbyson, D.J., Alme, H.J., Hoisie, A., Petrini, F., Wasserman, H.J., Gittings, M.: Predictive performance and scalability modeling of a large-scale application. In: Proceedings of the 2001 ACM/IEEE Conference on Supercomputing (CDROM), pp. 37–37. ACM (2001)
85. Dinda, P.A., O'Hallaron, D.R.: An evaluation of linear models for host load prediction. In: Proceedings of the The Eighth International Symposium on High Performance Distributed Computing, 1999. IEEE, pp. 87–96 (1999)
86. Barbierato, E., Gribaudo, M., Iacono, M.: A performance modeling language for big data architectures. In: Rekdalsbakken, W., Bye, R.T., Zhang, H. (eds.) ECMS, European Council for Modeling and Simulation, pp. 511–517 (2013)
87. Barbierato, E., Gribaudo, M., Iacono, M.: Modeling apache hive based applications in big data architectures. In: 7th International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2013. (Dec 2013)

88. Andresen, D., Yang, T., Ibarra, O.H., Eğecioğlu, Ö.: Adaptive partitioning and scheduling for enhancing www application performance. J. Parallel Distrib. Comput. **49**(1), 57–85 (1998)
89. Bodík, P., Griffith, R., Sutton, C., Fox, A., Jordan, M., Patterson, D.: Statistical machine learning makes automatic control practical for internet datacenters. In: Proceedings of the 2009 Conference on Hot topics in Cloud Computing. HotCloud'09, USENIX Association Berkeley, CA, USA (2009)
90. Anderson, E., Ganger, G.R., Wylie, J.J., Krevat, E., Shiran, T., Tucek, J.: Applying Performance Models to Understand Data-Intensive Computing Efficiency (2010)
91. Trivedi, K.S.: Probability and Statistics with Reliability, Queuing and Computer Science Applications. John Wiley and Sons Ltd., Chichester, UK (2002)
92. Kurtz, T.: Approximation of Population Processes. Society for Industrial and Applied Mathematics (1981)
93. Bobbio, A., Gribaudo, M., Telek, M.: Analysis of large scale interacting systems by mean field method. In: 5th International Conference on Quantitative Evaluation of Systems—QEST2008, St. Malo (2008)

Chapter 2
Workflow Scheduling Techniques for Big Data Platforms

Mihaela-Catalina Nita, Mihaela Vasile, Florin Pop and Valentin Cristea

2.1 Introduction

Today, almost everyone is connected to the Internet and uses different Cloud solutions to store, deliver, and process data. Cloud computing assembles large networks of virtualized services, such as hardware and software resources [1]. The use of cloud resources by end users is made in an asynchronous way and in many cases using mobile devices over different types of networks. Interoperability for such type of systems with the main aim to ensure dependability and resilience is one of the major challenges for heterogeneous distributed systems.

While cloud computing optimizes the use of resources, it does not (yet) provide an effective solution for processing complex applications described by workflows. Some example of such applications is hosting multimedia content-driven, and process tsunami (often in real-time) of content from heterogeneous sources, such as surveillance cameras, medical imaging devices, etc. The current need is an optimal and validated middleware framework and that can support end-to-end life-cycle operations of different multimedia content-driven applications on more standard cloud infrastructures [2].

Many scientific applications are defined as a set of ordered tasks that are linked by data dependencies. A workflow management system is used to define, manage, and execute these workflow applications on cluster, grid, cloud environments. In this context, a workflow scheduling strategy is used to map the task on the different resources [3, 4].

We live in the data age, and a key metric of present times is the amount of data that is generated anywhere around us. The largest scientific institution of present times, CERN near Geneva, Switzerland, produces in the Large Hadron Collider project over 30 PB of data per year (as of 2013) [5]. Thus the notion of Big Data,

M.-C. Nita · M. Vasile · F. Pop (✉) · V. Cristea
Computer Science Department, University Politehnica of Bucharest,
Bucharest, Romania
e-mail: florin.pop@cs.pub.ro

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_2

a commonplace in all business discussions involving technology. Yet, its definition is not that clear. Big Data represents data sets of sizes larger than the common ability of traditional technologies to process given a certain service level agreement. And this last part brings us to a more meaningful definition: Big Data is the process of delivering decision-making insights. But, rather than focusing on people, this process uses a much more powerful and evolved technology, given the latest breakthroughs in this field, to quickly analyze huge streams of data, from a variety of sources, and to produce one single stream of human-level knowledge [6]. Nevertheless, in 2001, META Group (now Gartner), proposed a three-dimensional view regarding data growth challenges and opportunities, taking into consideration the increasing volume (the amount of data), the velocity (the speed of data in and out) and variety (the range of data types and sources) [7]. In 2012, Gartner updated this report as follows: Big Data is high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision-making, insight discovery and process optimization.[1] Lately, another key point—veracity is added by some organizations to make a strong case for the high need of accuracy of Big Data. We can extend this model to 8-V dimensions of Big Data: volume, velocity, variety, veracity, variability, visualization, volatile, and value [8]. We consider Grids and Clouds as suitable systems for Big Data platforms.

In this context, we face with the following assumptions for workflow scheduling. The challenges came from dynamic systems affected by faults. In this context of variety, the stimulating relationship between users, who require better computing services, and providers, who discover new ways to satisfy them, is the motivation to introduce future trends oriented on self-* capabilities [9]. For multimedia applications, real-time scheduling and processing are done on reliable and unreliable resources (an example is based on mixed processing based on satellite images [10, 11], live data streaming and sensor data for video surveillance). The main challenges are to ensure deadlines (very important in real-time interaction), budget (pay-per-use Cloud model), energy consumption (battery saving for mobile devices), and QoS (to guarantee SLA) for complex workflow applications.

When addressing the problem of adaptive workflow scheduling we first need to investigate the current solutions for managing such workflow applications. With specific focus on scientific workflows we can find some tools designed by the research community to help the scientists address/investigate/simulate issues from every area like: astronomy, bioinformatics, earth science, chemistry. It is important to have a global picture of the current tools in order to properly choose the one that suits perfectly in our research area [12, 13].

The chapter is organized as follow. First, we discuss the workflow scheduling algorithms and techniques in grid and cloud. We present the strengths and weaknesses of several existing scheduling strategies and we make a critical analysis of workflow management systems.

[1] http://www.gartner.com/it-glossary/big-data/.

2.2 Workflow Scheduling in Distributed Systems

We are facing with a strong development of various technologies leading to complex applications, which are able to process Big Data sets and execute different experiments/tasks on distributed systems. These applications are covering important aspects of everyday life: health, education, astronomy, research engineering, etc. and are described by a number of interdependent tasks called workflows. Scientific workflows represent the automation of a scientific process in which tasks are organized based on their control and data dependency.

In this context, distributed systems are offering several advantages, such as: utilizing geographically distributed resources, increasing throughput, reducing costs by not investing in proprietary resources and using the shared ones, engaging various scientific teams with different expertise.

A workflow is described by connecting multiple tasks according to their dependency (execution dependency or data dependency). This pool of interconnected tasks may take two shapes: that of a directed acyclic graph (DAG) or a non-DAG, the difference consisting in the existence of repeating tasks.

The DAG structure can be categorized as sequential, parallel and choice, while the non-DAG structure is adding one more structure type: iteration. In a sequential structure, the tasks are ordered in a serial manner and one task starts only after the previous one ended. In a parallel structure, the tasks may be computed simultaneous and in a choice structure, the decision of following a certain path in the graph is done at runtime. In the iteration structure, several tasks of the workflow may be repeated. A workflow may be described by all the presented structure types simultaneously [14].

2.2.1 Workflow Scheduling Algorithms and Techniques in Grid

There are two main classes of workflow scheduling: best effort and QoS constraints-based scheduling [15]. The best-method class wants to minimize the makespan and ignores other constraints, like budget, energy, etc. On the other hand the QoS constrained class, as the name suggests, attempts to minimize cost/time under some QoS metrics.

2.2.1.1 Best-Effort-Based Workflow Scheduling

Best-effort-based workflow scheduling algorithms are more specifics for community-based environments, like grids. In this type of environments, there are factors like elasticity or cost that does not count and the main focus is on the execution time. This class of scheduling has as target to complete execution at the earliest time. Best-effort-based scheduling algorithms have two different approaches: heuristics

based or meta-heuristics based. The heuristic approach is to develop a scheduling algorithm for a particular type of problem, while the meta-heuristic-based approach is to develop an algorithm based on a meta-heuristic method which provides a general solution for a specific class of problems.

Individual task scheduling method, as the name suggests it makes scheduling choices based on a single task at a time. It is not aware of the general context. The Myopic algorithm [16] implements this method and it schedules an unmapped ready task to the resource that is expected to complete the task earliest, until all tasks have been scheduled.

List scheduling prioritizes workflow tasks and schedules the tasks based on their priorities. It includes two phases: task priority and resource allocation. The list scheduling models can be categorized in three groups:batch mode scheduling, dependency mode scheduling, and a hybrid version: Batch-Dependency Mode scheduling.

Batch mode scheduling algorithms intent to schedule parallel independent tasks on a pool of resources. Since the number of resources is much less than the number of tasks, the tasks need to be scheduled on the resources in a certain order [17]. This priority is made through the following algorithms: Min-Min, Max-Min, and Suffer.

Min-Min heuristic schedules sets of independent tasks iteratively. In each iterative step, it computes the ECT (Early Completion Time) of each task on its every available resource and obtains the MCT (Minimum Estimated Completion Time). The task with minimum MCT is chosen to be scheduled first. The task is assigned on the resource which is expected to finish it at first.

Max-Min heuristic is similar to the Min-Min heuristic, but it sets high scheduling priority to tasks which have long execution time.

Sufferage sets high scheduling priority to tasks whose completion time by the second best resource is far from the first which can complete the task at earliest time. This method may have optimal results in heterogeneous environments.

Dependency mode intends to provide strategies to map workflow tasks on heterogeneous resources based on analyzing the dependencies of the entire task DAG. Unlike batch mode algorithms, it ranks the priorities of all tasks based on the whole application context.

The Heterogeneous Earliest Finish Time (HEFT) algorithm proposed in [18] by Topcuoglu et al. has been applied by the ASKALON project and it first calculates average execution time for each task and average communication time between resources of two dependent tasks. Then, each task receives a rank value which is computed in a recursive manner based on the rank value of the following dependent tasks. So, the exit task in the graph will have the smallest rank value, as being the average execution time. The tasks previous the exit task will have their average execution time + the maximum ([communication time from a resource to another resource] + [the rank value of the successor]). The task with the highest priority will be scheduled first.

Sakellariou and Zhao [19] proposed a *hybrid heuristic for scheduling DAGs* on heterogeneous systems. The heuristic combines dependency mode and batch mode. It first computes rank values of each task and ranks all tasks in the decreasing order of their rank values. Then it creates groups of independent tasks. Each group will

have a group number based on the rank values of the group tasks. Then it schedules tasks group by group and uses a batch mode algorithm to reprioritize the tasks in the group.

Cluster-based scheduling and duplication-based scheduling aim to avoid the communication time of the data for interdependent tasks, such that it is able to reduce the overall execution time. The cluster-based scheduling clusters tasks and assign tasks of the same cluster to the same resource. The duplication-based scheduling use the idling time of a resource to duplicate some parent tasks and it schedules them on other resources. Bajai and Agrawal [20] proposed a task duplication-based scheduling algorithm for network of heterogeneous systems (TANH). This algorithm combines both cluster-based scheduling and duplication-based scheduling. It first traverses the DAG to compute parameters of each node including earliest start and completion time, latest start and completion time, critical immediate parent task, best resource, and the level of the task. Afterwards, it clusters tasks based on these parameters and it scales down the number of clusters until it is less or equal than the number of resources. In case of number of clusters being less than the number of resources, it utilizes the idle times of resources to duplicate tasks and rearrange tasks in order to decrease the overall execution time.

Genetic Algorithms (GAs) [21] provide robust search techniques that allow an optimal solution to be derived from a large search space by applying the principle of evolution. It first creates an initial population consisting of randomly generated solutions and then it applies genetic operators (selection, crossover and mutation). Then the individuals are selected based on their fitness values and included in the next selection steps. These steps are repeated until an optimal solution is found. The art of scheduling consists in finding the proper fitness function. As an example of such function designed for scheduling [22] is $f(x) = C_{max} - FT(I)$, where C_{max} is the maximum completion time observed so far and $FT(I)$ is the completion time of the individual I. While trying to minimize the completion time, the individuals with a larger fitness value will be selected for the next steps.

Simulated Annealing (SA) [23] is inspired by the Monte Carlo method for statistically searching the global optimal between several local optimal. The concept is taken from the annealing process, which repeats the heating and slowly cooling of a structure. The input of the algorithm is an initial solution which is constructed by assigning a resource to each task randomly. Then, based on an acceptance rate, the solutions are selected for the next step. At each iterative step the acceptance is decreased.

2.2.1.2 QoS-Constraint-Based Workflow Scheduling

When talking about QoS Constraints algorithms, in general, we are talking about two different perspectives: user perspective and scheduler perspective. If the user perspective refers to QoS of the entire workflow, the scheduler perspective refers to the QoS of the task. The scheduler may assure a QoS for the entire workflow only if the QoS of each individual task is assured.

Many workflow applications have a tight deadline so the deadline constrained scheduling algorithms are designed mainly for these types of applications. However being in pay-per-use environment such as cloud, it is also important to minimize the cost. In this context, this class of scheduling classes pays attention to the time framework but without ignoring the cost.

The backtracking heuristic developed by Menasce and Casalicchio [24] assigns available tasks to least expensive computing resources (in case of many available tasks, it assigns the most CPU intensive task to the fastest resource). The procedure is repeated until all tasks are mapped. After each iterative step, the execution time of current assignment is computed and in case of the execution time exceeding the time constraint, the heuristic backtracks the previous step and remove the least expensive resource from its resource list and reassigns tasks with the reduced resource set.

Another deadline constraint heuristic is the *deadline distribution* [25] heuristic which partitions a workflow and distributes the overall deadline into each task based on their workload and dependencies. After deadline distribution, the entire workflow scheduling problem has been divided into several subtask scheduling problems. A sub-deadline can be also assigned to each task based on the deadline of its task partition.

The Budget Constrained scheduling puts accent on the cost constraints while minimizing the execution time. LOSS and GAIN scheduling approach [26] adjusts a schedule which is generated by a time optimized heuristic and a cost optimized heuristic to respect budget constraints. There are two situations:

1. Total execution cost generated by time optimized schedule is grater than the budget; the LOSS approach is applied: gain a minimum loss in execution time for the maximum money savings by amending the schedule to satisfy the budget.
2. Total execution cost generated by a cost optimized scheduler is less than the budget, the GAIN approach is applied in order to use the surplus for decreasing the execution time: gain the maximum benefit in execution time for the minimum monetary cost, while amending the schedule.

In [27] a *scheduling data-intensive workflows onto storage-constrained distributed* method is proposed. Important improvement in storage use for workflow data is to add a cleanup job algorithm to erase data files when they are not longer in use or required by current task or any other task in the workflow process. If the compute tasks are mapped to many resources then the data file is also replicated on all resources thus the cleanup jobs/tasks are added per resource basis. In some situations the data file required by a task comes from another resource and must not be deleted by the algorithm at the source before its transferred to the child resource or task. Cleaning up files on the workflow when having multiple file dependencies is challenging as the algorithm inserts cleanup jobs along the executable workflow and cleans up along the executions. This can generate for complex workflows greater number of cleanup jobs than the compute tasks itself. More efficient is to have an algorithm for storage aware when workflow is mapped considering the overall storage requirements, minimizing also the requirements. Thinking at an efficient execution of the tasks, these

must be mapped around the workflow taking in consideration the execution time and using a resource with ample disk space. Main idea is to consider space requirement and then performance of the resource when allocating the tasks. The algorithm can be split in a 3-phase algorithm:

1. identification of the resources capable to accommodate the data files;
2. task allocation based on the shortest run time for that task; and
3. cleanup any unnecessary files indicated by the cleanup jobs.

In [28] the problem of *scheduling multiple workflows* is addressed: how to better plan multiple workflows instead of merging them as previous solutions proposed. The proposed solution has three components:

(a) DAG Planner: assigns each job local priority (HEFT-based priority), manages the job interdependence and submits the jobs to the Job Pool;
(b) Job Pool an unsorted list with all jobs waiting to be scheduled;
(c) Executor reprioritizes the job into the Job Pool.

Each workflow has its own DAG Planner. Each DAG planner sends only independent jobs at a time. Once it has finished the execution, the DAG planner is notified and it will send the successor of that job. At the Executor level there are two types of priority for two different cases:

(a) If jobs are from the same DAG planner, the Highest Rank first rule will be applied (HEFT);
(b) If jobs are from the different DAG planers, the Lowest Rank first rule will be applied in order to avoid the starvation of the exit jobs of some DAG planner.

2.2.2 *Workflow Scheduling Algorithms and Techniques in Cloud*

One of the main advantages of moving to the cloud is application scalability, which allows real-time provisioning of resources to respect service level agreements (SLAs)/application requirements. This enables workflow management systems (WMS) to support real-time provisioning instead of advanced reservations. In this context, workflow scheduling algorithms will need to adapt and assure the agreed level of QoS.

As concluded in [29] the main requirements for Cloud workflow scheduling are: satisfaction of QoS requirements for individual workflow instances, minimization of the running cost, ability of assigning fine-grained QoS to facilitate SLA management and good scalability. These conclusions lead to a two level workflow scheduling: service-level scheduling and task-level scheduling. The service level is responsible for the first and third objective while the task-level is responsible for the task VM optimization process (second and forth objective). The service level is a global scheduler that identifies the resources needs, makes provisioning and the task-level is a local

scheduler which implements the optimal scheduling plan for the given workflow on the resources obtained.

In [30] Meng Xu et al. propose a multiple QoS constrained scheduling strategy of multiple workflows for cloud computing (MQMW) which has a similar architecture with the one proposed in [28], but different names: *Preprocessor*, *Scheduler* and *Executor*.

The *Preprocessor* will computed the following attributes for the received tasks:

- the available service number;
- the covariance for time and cost;
- the time quota: the time limit when the task is executed;
- the cost quota: the cost limit when the task is executed;
- the time surplus of the workflow: the difference between the time attribute of QoS and the finish time of the workflow if all the resources are available;
- the cost surplus of the workflow: the difference between the cost attribute of QoS and the cost of the workflow if all the resources are available.

Afterwards, the Preprocessor inserts the ready tasks into the queue and for the first time only entry tasks will be submitted. After the notification given by the Executor that a certain job has finished, the dependent tasks will be also submitted. The Scheduler will reorder the jobs in the list and it will allocate a job to the optimal resource. When a task will be finished the Executor will notify the Preprocessor about the task completion status.

The *Scheduling Strategy* is the following:

1. the task with minimum available service number should be scheduled first (that the task would have not available services, if other tasks are scheduled first);
2. the tasks which belong to the workflow with minimum time surplus and cost surplus should be scheduled first;
3. the tasks with minimum covariance should be scheduled first. The covariance describes the strength of the correlation between time and cost. The minimum covariance means when the time decreasing a definite value, the cost will increase mostly. So the task should be scheduled first. Otherwise, we should pay more or the time would increase more.

This algorithm was evaluated in parallel with the one described in [28] and the results shows that the previous one improves the execution time of the workflow no matter of the costs. However MQMW respects all the QoS constraints.

In [31] a multi-objective heterogeneous earliest finish time algorithm. The method called MOHEFT is an extension of the Heterogeneous Earliest Finish Time (HEFT) algorithm and intents to provide an optimal solution to the problem of makespan and energy reduction. In this context, it is not always possible to find a solution that minimizes both makespan and energy consumption so they introduced the concept of dominance. A solution $\times 1$ dominates a solution $\times 2$ if the makespan and energy consumption of $\times 1$ are smaller than those of $\times 2$.

In this context, two solutions are said to be nondominated whenever none of them dominates the other (i.e.one is better in makespan and the other in energy

Scheduling Method	Algorithm
Heuristics Based Approach	
Individual Task Scheduling	Myopic
List Scheduling	
▲ Batch Mode	Min-Min Max-Min Sufferage
▲ Dependency Mode	HEFT
▲ Batch – Dependency Mode	Hybrid
Cluster Based Scheduling	
Duplication Based Scheduling	THAN
Meta-Heuristics Approach	
Genetic Algorithms	
Simulated Annealing	

Fig. 2.1 Overview of the best-effort workflow scheduling models

consumption). The set of optimal nondetermined solutions are called Pareto Front. Similar to HEFT, MOHEFT ranks first the tasks and then instead of creating an empty solution as HEFT does, it creates a set S of K empty solutions. Afterwards, the mapping phase begins in which MOHEFT iterates first over the list of ranked tasks. The idea is to extend every solution by mapping the tasks onto all possible resources. This strategy results in an exhaustive search if there is any restriction taken into consideration. Only the best K trade-off solutions from the temporary set are kept. A solution belongs to the best trade-off if it is not dominated by any other solution and if it contributes to the diversity of the set. The diversity of the set is described as the highest crowding distance (the area surrounding a solution where no other trade-off solution is placed nearby).

An overview of the best-effort workflow scheduling models is presented in Fig. 2.1.

2.2.3 Scheduling Methods

In the following table, we present a critical analysis of existing methods for workflow scheduling, by highlighting the strengths and weaknesses and a short description for each presented method.

A Planner-Guided Scheduling Strategy for Multiple Workflow Applications and Dynamic scheduling multiple DAGs [28]	
Description	Dynamic scheduling multiple DAGs; Poisson distribution of the arrival jobs; Custom ranking system (DAG path for example); Highest rank first between jobs from the same DAG; Lowest rank first between jobs from different DAGs, resulting that, applications that are closing to finish are not starving for resources (the last jobs will have lower priorities)
Strengths	Multiple DAGs scheduling; hybrid priority (improvement of highest rank first in case of multiple DAGs)
Weaknesses	If lower rank workflows are coming continuously, the higher rank task scheduling will be postponed; not focused on deadline and budget constraints; not SLA awareness at the Executor level
Immediate Mode: Individual Task Scheduling [16]	
Description	Myopic algorithm; The best-effort scheduling strategy (more suitable for grids); Individual task scheduling; Scheduling decision is made only for one task at a moment; Task is mapped at the resource that is expected to finish the task first
Strengths	Works fine for short tasks and a small number of requests/second
Weaknesses	Not suitable for cloud environments; It does not apply any logic in tasks priority (FIFO rule); which will generate starvation in case of longer tasks
Batch Mode Scheduling : Min-Min, Max-Min, and Sufferage strategies [17]	
Description	Batch mode scheduling strategy (best effort/list scheduling); Provides a strategy to map a number of tasks (T) on a number of resources R, where $T >> R$; MIN-MIN a task having a min MECT (Minimum Estimated Completion Time) will be scheduled first; MAX-MIN task with max MECT will be executed first; SUFFERAGE priority based on the suffrage value (the difference between 1st ECT (earliest completion time) and 2nd ECT
Strengths	Experimental results shows that generally MIN-MIN outperforms MAX-MIN; MAX-MIN may be better in cases of having much more short tasks than longer tasks (it will not generate starvation for longer tasks); Suffrage performs better in heterogeneity environments where there is a remarkable difference between resources performance
Weaknesses	Application type-dependent strategy; best effort; suitable only for grid
Dependency Mode Scheduling HEFT [18]	
Description	Graph dependency is analyzed before scheduling; It ranks the priorities of all tasks at a time 2 metrics: average execution time and average communication time; All the tasks are ordered based on a rank (computed recursively based on previous ranks)
Strengths	Based on heterogeneous resources
Weaknesses	Mean value is the only best practice; Best effort; Suitable only for grid
Batch-Dependency Mode [19]	
Description	Hybrid method between batch and dependency methods; Independent tasks are added into separate groups; Group rank ordering
Strengths	Parallel execution of multiple different jobs
Weaknesses	Not SLA aware; Suitable for grid environments

Cluster-based scheduling [20]	
Description	Avoid transfer communication cost; tasks are grouped in clusters; one cluster is scheduled on the same resource
Strengths	Avoids data transfer cost
Weaknesses	Difficult to respect deadline constraints
Duplication-based scheduling [20]	
Description	Avoid transfer communication cost; Use the idle time of an resource to schedule parent tasks (task duplication)
Strengths	Good in a heterogeneous environment where the machine performance is unknown
Weaknesses	Parallel job scheduling.
Hybrid method of cluster and duplication scheduling [20]	
Description	Cluster tasks based on earliest start and completion time, latest start and completion time, critical parent task and best resource; If the number of clusters $< R$ (resources): duplication; If nr. of clusters $> R$, scaling down by merging some clusters
Strengths	Avoids data transfer cost; Good in a heterogeneous environment where the machine performance is unknown
Weaknesses	Difficult to respect deadline constraints; Merging metrics
Genetic Algorithms meta-heuristics [21]	
Description	Generates new solutions by modifying currently known good solutions
Strengths	Optimized solution
Weaknesses	Its limitations depends on the fitness valuation function
Simulated Annealing SA meta-heuristics [23]	
Description	Generates new solutions by modifying currently known good solutions
Strengths	Optimizes the solution
Weaknesses	Its limitation depends on the acceptance function reduction value
Backtracking [24]	
Description	Assign available tasks to least expensive resources; Largest computational demand to fastest resource; After each step, the execution time of the current assignment is computed; if its exceed the time constraint, the task will be reassigned
Strengths	Deadline assurance
Weaknesses	SLA awareness; Multiple constraints; Dynamic environments

Deadline distributions [25]	
Description	Synchronization task (multiple parents) versus simple task; Grouped in sub-workflows with different deadlines; Distributed deadline
Strengths	Assures the deadline commitment at a granular level (the smallest unit task)
Weaknesses	SLA awareness; Multiple constraints; Dynamic environments
LOSS and Gain [26]	
Description	Gain a minimum loss in the execution time while maximizing the cost savings; First is applied a time optimization heuristics
Strengths	Budget constraint; assurance
Weaknesses	SLA awareness; Multiple constraints; Dynamic environments
MQMW [30]	
Description	Scheduling based on the following metrics: service available number, time surplus, workflow surplus, the covariance for time and cost
Strengths	Multiple workflow; Multiple QoS successfully respected
Weaknesses	It is not very clear what is the algorithm behavior in case of having a number of tasks respecting the conditions for prioritization greater than the number of available resources; Speed; Scalability
MOHEFT—Multi-objective energy-efficient workflow scheduling using list-based heuristics [31]	
Description	Extends HEFT by finding the optimal solution and respecting multiple objectives
Strengths	Multiple objectives: energy and time; Interesting to scale for multiple objectives: cost, energy and time
Weaknesses	Multiple workflows problem

2.3 Workflow Modeling and Existing Platforms

This section presents a detailed picture of the major tools used for workflow management.

Pegasus [29] is a Workflow system that can take a workflow description, transform it in an executable sequence of jobs and map it on a local machine, cluster, Condor Pool, or a cloud (Amazon EC2, Google Cloud Storage). It is developed since 2001 and the last release was in May 2015. Pegasus has been used in several scientific areas including bioinformatics (DNA sequencing, Epigenomics, etc.), astronomy (Galactic Plane—NASA Collaboration; Montage–Caltech, etc.), earthquake science, gravitational wave physics, and ocean science. Pegasus major components (Fig. 2.2):

1. *Mapper*—transforms the abstract workflow definition into an executable set of dependent jobs. The final will find the appropriate software, data, and computational resources required for workflow execution.
2. *Execution Engine*—executes in the specific order the tasks defined in the workflow. This component relies on the compute, storage, and network resources defined in the executable workflow to perform the necessary activities.

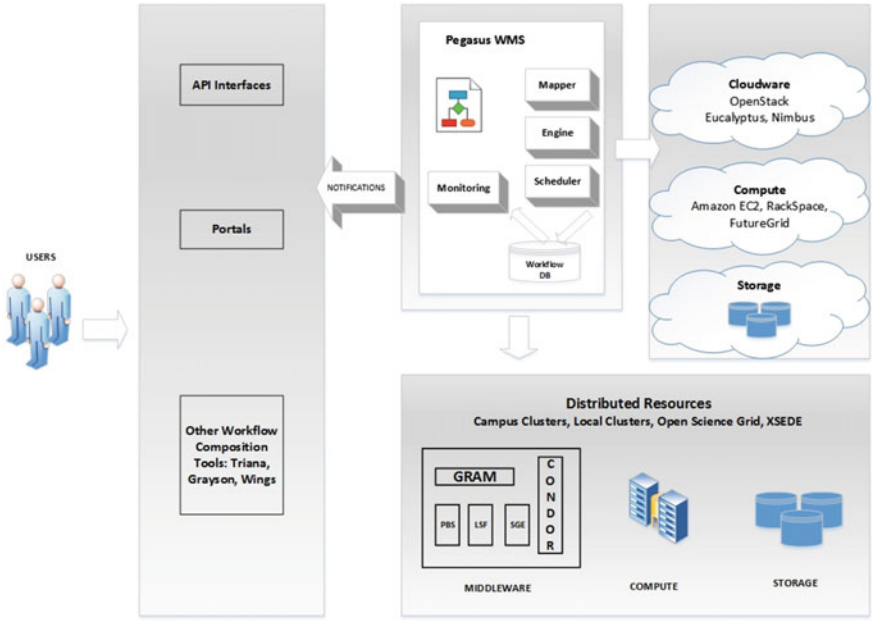


Fig. 2.2 Pegasus components

3. *Task Manager*—manages single workflow tasks, by supervising their execution on local or remote resources.

Workflows successfully execution is based on the information gathered from the following components:

1. Replica Catalog: looks for input and output data locations;
2. Transformation Catalog: looks for executables locations: binary files;
3. Site Catalog: looks for the environment infrastructure.

Pegasus is also able to make decisions in order to improve the overall performance: cluster small jobs together, data reuse (identical jobs, different workflows). Regarding the subject that is treated in this report, the scheduling problem, Pegasus by default implements the following policies:

1. Random—the sites are randomly chosen; default option.
2. Round Robin—jobs will be assigned in a round robin manner amongst the sites that can execute them. A site cannot execute all types of jobs so the round robin scheduling will be applied on a sorted list of sites. The sorting is done based on the number of jobs a particular site has been assigned in that job class so far. If a

job cannot be run on the first site in the queue (due to no matching entry in the transformation catalog), it goes to the next one and so on.
3. Group—jobs are grouped and a specific group will be assigned to the site that can execute them. A job will be put into a specific group based on a profile key group from the DAG. The jobs that do not have the profile key associated with them, will be put in the default group. The jobs in the default group are handed over to the "Random" Site Selector for scheduling.
4. HEFT—HEFT processor scheduling algorithm is used to schedule jobs in the workflow to multiple grid sites. The implementation assumes default data communication costs when jobs are not scheduled on to the same site. The runtime for the jobs is specified in the transformation catalog by associating the Pegasus profile key runtime with the entries. The number of processors in a site is picked up from the attribute idle-nodes associated with the job manager of the site in the site catalog.
5. NonJavaCallout—it will call out to an external site selector. A temporary file is prepared containing the job information that is passed to the site selector as an argument while invoking it. The path to the site selector is specified by setting the property `pegasus.site.selector.path`. The environment variables that need to be set to run the site selector can be specified using the properties with a `pegasus.site.selector.env.prefix`. The target sites used in planning are specified on the command line using the sites option to pegasus-plan. If not specified, then it will pick up all the sites in the Site Catalog as candidate sites and it will map a job on a specific site only if it finds an installed executable on that site.

Taverna [32] is an open-source Java-based workflow management system developed at the University of Manchester with the main target in supporting the life sciences community (biology, chemistry, and medicine) to design and execute scientific workflows and support research experiments. However, it can be applied to a wide range of fields since it can invoke any web service by simply providing the URL of its WSDL document. In addition to web services, Taverna supports the invocation of local Java services (Beanshell scripts), local Java API (API Consumer), R scripts on an R server (Rshell scripts), and imports data from a CVS or Excel spreadsheet. Taverna main components are:

- Taverna Engine—enacting workflows.
- Taverna Workbench—client application; users graphically create, edit, and run workflows on a desktop computer.
- Taverna Server—users set up a dedicated server for executing workflows remotely.
- A Command Line Tool—quick execution of workflows from a command prompt.

Triana [33] is a Java-based scientific workflow system, developed at the Cardiff University, which combines a visual interface with data analysis tools. It can con-

nect heterogeneous tools (e.g., web services, Java units, and JXTA services) in one workflow. Triana uses its own custom workflow language, although it can use other external workflow language representations, such as Business Process Execution Language (BPEL), available through pluggable language readers and writers.

One of the most powerful aspects of Triana on the other hand is its graphical user interface. It has evolved in its Java form for over 10 years and contains a number of powerful editing capabilities, wizards for on-the-fly creation of tools and GUI builders for creating user interfaces. Trianas editing capabilities include: multilevel grouping for simplifying workflows, cut/copy/paste/undo, the ability to edit input/output nodes (to make copies of data and add parameter dependencies, remote controls or plug-ins), zoom functions, various cabling types, optional inputs, type checking, and so on. Triana may generate Pegasus input files.

Kepler [34] is a Java-based open-source software framework providing a graphical user interface and a run-time engine that can execute workflows either from within the graphical interface or from a command line. It is developed and maintained by a team consisting of several key institutions at the University of California and has been used to design and execute various workflows in biology, ecology, geology, chemistry, and astrophysics.

Askalon [35] is an application development and runtime environment, developed at the University of Innsbruck, which allows the execution of distributed workflow applications in service-oriented Grids. Its SOA-based runtime environment uses Globus Toolkit as Grid middleware. Workflow applications in Askalon are described at a high level of abstraction using a custom XML-based language called abstract grid workflow anguage (AGWL).

The Askalon architecture includes the following components:

- *Resource broker*—responsible of the resources negotiation and reservation in Grid.
- *Resource monitoring*—rule-based monitoring; monitors Grid resources.
- *Information service*—discovery, organization, and maintenance of resources and data.
- *Workflow executor*—dynamic deployment and fault-tolerant execution of activities in the Grid nodes.
- *Metascheduler*—workflow applications mapping in the Grid.
- *Performance prediction*—estimates execution time of atomic activities and data transfers; Grid resource availability.
- *Performance analysis*—unifies the performance monitoring, instrumentation, and analysis for Grid applications; supports the interpretation of performance bottlenecks.
- *Askalon Scheduler*.

One of the Askalon architecture components that represents interest for us is the MetaScheduler and its architecture described in Fig. 2.3. The Scheduler consists of two major components: Workflow Converter and Scheduling Engine. Event Generator is a future extension for increasing dynamicity in workflow processing.

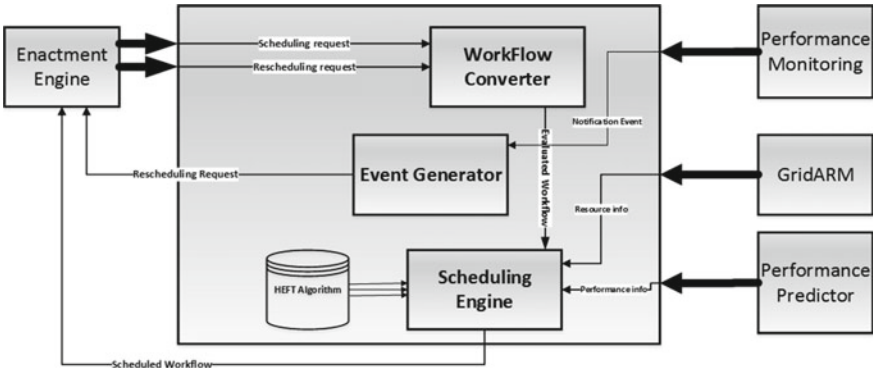


Fig. 2.3 Askalon components

The Workflow Converter is responsible for transforming all the sophisticated workflow graphs to simple DAGs. The Scheduling Engine for scheduling workflows into specific resources. It is based on a plug-in architecture, where different scheduling algorithms can be implemented. By default the HEFT algorithm is chosen as the primary scheduling algorithm for Askalon.

Karajan [36] is a JAVA written system that allows users to compose workflows through an XML scripting language and a custom language, called K, which is more user friendly. Both languages support hierarchical workflow descriptions based on DAGs and have the ability to use instructions such as if/while order to easily express concurrency. Also, it can be based on Grid tools such as Globus GRAM for distributed/parallel execution of workflows. The architecture of the Karajan framework contains the following components:

- *Workflow engine*—interacts with high-level components (GUI module for describing the workflows) and monitors the execution.
- *Checkpointing subsystem*—checkpoints the current state of the workflow.
- *Workflow service*—allows the execution of workflows; specific libraries enables the workflow engine to access specific functionalities.

2.4 Analysis of Workflow Management Systems

In the following table, we present a critical analysis of existing workflow management systems, by highlighting the strengths and weaknesses and a short description for each presented method.

WMS	Short description	Strengths	Weaknesses
Taverna	DAG based; Graphical specification	User-friendly interface for workflow description; easy to use by nontechnical scientists in their simulations	Relies on the user to make the choices of resources for mapping; It doesn't offer integration with public cloud environments; Adaptive workflows
Triana	Non-DAG based; Graphical specification	Graphical User Interface; Pegasus interoperability; Tool for editing/creating workflows; Workflow rewriting: creating sub-workflows that execute and feed back into the main workflow	Job submission made through GridLab GAT, which can make use of GRMS, GRAM or Condor for the actual job submission; Passive approach in case of a failure (it informs the users)
Pegasus	DAG based; Language specification	Focus on the mapping and execution capabilities and leave the higher level composition tasks to other tools; Amazon EC2/Google Cloud Support; Support for optimization decisions	It does not offer support for adaptive workflows
Kepler	Non-DAG based; Graphical specification	Graphical workflow specification; Both workflow specification support and execution engine; Local/Web/Grid services; Fault tolerance—smart rerun; Adaptive workflows—workflows can modify themselves during execution	Relies on the user to make the choices of resources for mapping; Adaptive Workflows; It doesnt offer integration with public cloud environments
Askalon	DAG; Language and graphical specification	Graphical User Interface; Support for optimization decisions; Complex Fault tolerance mechanism (checkpointing, task-level recovery)	Custom description language (AGWL); Common Public Cloud integration; Adaptive Workflows
Karajan	DAG; Language and graphical specification	Graphical User Interface; Support hierarchical workflows; Checkpoint and rollback assurance	Low support for interoperability between workflow management system (only XML and custom workflow specification); Cloud integration; Adaptive Workflows

2.5 Conclusions

In this chapter, we investigate the current solutions for managing workflow applications in grids and clouds, offering a critical analysis on existing scheduling algorithms and management systems. We presented an overview of the best-effort workflow scheduling models.

Acknowledgments The research presented in this paper is supported by projects: *DataWay*: Real-time Data Processing Platform for Smart Cities: Making sense of Big Data—PN-II-RU-TE-2014-4-2731; *MobiWay*: Mobility Beyond Individualism: an Integrated Platform for Intelligent Transportation Systems of Tomorrow—PN-II-PT-PCCA-2013-4-0321; *CyberWater* grant of the Romanian National Authority for Scientific Research, CNDI-UEFISCDI, project number 47/2012; *clueFarm*: Information system based on cloud services accessible through mobile devices, to increase product quality and business development farms—PN-II-PT-PCCA-2013-4-0870.

References

1. Pop, F., Zhu, X., Yang, L.T.: Midhdc: Advanced topics on middleware services for heterogeneous distributed computing. part 1. Future Gener. Comput. Syst. **56**, 734–735 (2016)
2. Pop, F., Potop-Butucaru, M.: Armco: Advanced topics in resource management for ubiquitous cloud computing: An adaptive approach. Future Gener. Comput. Syst. **54**, 79–81 (2016)
3. Simion, B., Leordeanu, C., Pop, F., Cristea, V.: A hybrid algorithm for scheduling workflow applications in grid environments (icpdp). In: OTM Confederated International Conferences "On the Move to Meaningful Internet Systems", pp. 1331–1348. Springer (2007)
4. Vasile, M.A., Pop, F., Tutueanu, R.I., Cristea, V., Kołodziej, J.: Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing. Future Gener. Comput. Syst. **51**, 61–71 (2015)
5. Lynch, C.: Big Data: How do your data grow? Nature **455**(7209), 28–29 (2008)
6. Pop, F., Iacono, M., Gribaudo, M., Kołodziej, J.: Advances in modelling and simulation for big-data applications (amsba). Concurrency Comput. Practice Experience **28**(2), 291–293 (2016)
7. Chen, M., Mao, S., Liu, Y.: Big Data: a survey. Mob. Networks Appl. **19**(2), 171–209 (2014)
8. Erl, T., Khattak, W., Buhler, P.: Big Data Fundamentals: Concepts. Prentice Hall Press, Drivers & Techniques (2016)
9. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-science: An overview of workflow system features and capabilities. Future Gener. Comput. Syst. **25**(5), 528–540 (2009)
10. Muresan, O., Pop, F., Gorgan, D., Cristea, V.: Satellite image processing applications in mediogrid. In: 2006 Fifth International Symposium on Parallel and Distributed Computing, pp. 253–262. IEEE (2006)
11. Gorgan, D., Bacu, V., Rodila, D., Pop, F., Petcu, D.: Experiments on esipenvironment oriented satellite data processing platform. Earth Sci. Inf. **3**(4), 297–308 (2010)
12. Masdari, M., ValiKardan, S., Shahi, Z., Azar, S.I.: Towards workflow scheduling in cloud computing: a comprehensive analysis. J. Network Comput. Appl. **66**, 64–82 (2016)
13. Taylor, I.J., Deelman, E., Gannon, D.B., Shields, M.: Workflows for e-Science: Scientific Workflows for Grids. Springer Publishing Company, Incorporated (2014)
14. Pop, F., Dobre, C., Cristea, V.: Performance analysis of grid dag scheduling algorithms using monarc simulation tool. In: 2008 International Symposium on Parallel and Distributed Computing, pp. 131–138. IEEE (2008)
15. Yu, J., Buyya, R., Ramamohanarao, K.: Workflow scheduling algorithms for grid computing. In: Metaheuristics for Scheduling in Distributed Computing Environments, pp. 173–214. Springer (2008)

16. Wieczorek, M., Prodan, R., Fahringer, T.: Scheduling of scientific workflows in the askalon grid environment. ACM SIGMOD Rec. **34**(3), 56–62 (2005)
17. Maheswaran, M., Ali, S., Siegal, H., Hensgen, D., Freund, R.F.: Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In: Heterogeneous Computing Workshop, 1999.(HCW'99) Proceedings. Eighth, pp. 30–44. IEEE (1999)
18. Topcuoglu, H., Hariri, S., Wu, M.Y.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002)
19. Sakellariou, R., Zhao, H.: A hybrid heuristic for dag scheduling on heterogeneous systems. In: Proceedings of the 18th International Parallel and Distributed Processing Symposium, 2004, p. 111. IEEE (2004)
20. Bajaj, R., Agrawal, D.P.: Improving scheduling of tasks in a heterogeneous environment. IEEE Trans. Parallel Distrib. Syst. **15**(2), 107–118 (2004)
21. Golberg, D.E.: Genetic algorithms in search, optimization, and machine learning. Addion Wesley **1989**, 102 (1989)
22. Hou, E.S., Ansari, N., Ren, H.: A genetic algorithm for multiprocessor scheduling. IEEE Trans. Parallel Distrib. Syst. **5**(2), 113–120 (1994)
23. YarKhan, A., Dongarra, J.J.: Experiments with scheduling using simulated annealing in a grid environment. In: International Workshop on Grid Computing, pp. 232–242. Springer (2002)
24. Menasce, D.A., Casalicchio, E.: A framework for resource allocation in grid computing. In: MASCOTS, pp. 259–267. Citeseer (2004)
25. Yu, J., Buyya, R., Tham, C.K.: Cost-based scheduling of scientific workflow applications on utility grids. In: First International Conference on e-Science and Grid Computing (e-Science'05), pp. 8–pp. IEEE (2005)
26. Sakellariou, R., Zhao, H., Tsiakkouri, E., Dikaiakos, M.D.: Scheduling workflows with budget constraints. In: Integrated Research in GRID Computing, pp. 189–202. Springer (2007)
27. Ramakrishnan, A., Singh, G., Zhao, H., Deelman, E., Sakellariou, R., Vahi, K., Blackburn, K., Meyers, D., Samidi, M.: Scheduling data-intensiveworkflows onto storage-constrained distributed resources. In: Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07), pp. 401–409. IEEE (2007)
28. Yu, Z., Shi, W.: A planner-guided scheduling strategy for multiple workflow applications. In: 2008 International Conference on Parallel Processing-Workshops, pp. 1–8. IEEE (2008)
29. Deelman, E., Singh, G., Su, M.H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G.B., Good, J., et al.: Pegasus: A framework for mapping complex scientific workflows onto distributed systems. Sci. Prog. **13**(3), 219–237 (2005)
30. Xu, M., Cui, L., Wang, H., Bi, Y.: A multiple qos constrained scheduling strategy of multiple workflows for cloud computing. In: 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications, pp. 629–634. IEEE (2009)
31. Durillo, J.J., Nae, V., Prodan, R.: Multi-objective energy-efficient workflow scheduling using list-based heuristics. Future Gener. Compu. Syst. **36**, 221–236 (2014)
32. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., et al.: Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics **20**(17), 3045–3054 (2004)
33. Taylor, I., Shields, M., Wang, I., Rana, O.: Triana applications within grid computing and peer to peer environments. J. Grid Comput. **1**(2), 199–217 (2003)
34. Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., Mock, S.: Kepler: an extensible system for design and execution of scientific workflows. In: Proceedings of the 16th International Conference on Scientific and Statistical Database Management, 2004, pp. 423–424. IEEE (2004)
35. Fahringer, T., Prodan, R., Duan, R., Nerieri, F., Podlipnig, S., Qin, J., Siddiqui, M., Truong, H.L., Villazon, A., Wieczorek, M.: Askalon: A grid application development and computing environment. In: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing, pp. 122–131. IEEE Computer Society (2005)
36. von Laszewski, G., Hategan, M.: Java Cog Kit Karajan/Gridant Workflow Guide. Tech. rep, Technical Report, Argonne National Laboratory, Argonne, IL, USA (2005)

Chapter 3
Cloud Technologies: A New Level for Big Data Mining

Viktor Medvedev and Olga Kurasova

3.1 Introduction

Conventional technologies for data storage and analysis are not efficient anymore in the Big Data era, as the data are of high-volumes, come with high-velocity, and have many varieties. Moreover, there is a need for discovering meaningful knowledge from a large amount of data. Big Data bring new challenges to data mining, because large volumes and different varieties must be taken into account. The common methods and tools for data processing and analysis are unable to manage such amounts of data, even if powerful computer clusters are used. To analyze Big Data, many new data mining and machine learning algorithms as well as technologies have been developed. Big Data do not only provide new data types and storage mechanisms, but also new methods of analysis. The aim of the research is to derive requirements for data mining systems suitable for Big Data with a view to bring conventional data mining to a new level in Big Data era using Cloud technologies.

3.2 Data Mining in Big Data Era

Data mining is an important issue in the data analysis process and knowledge discovery in medicine, economics, finance, telecommunication and other scientific fields. For this reason, for several decades, the attention has been focused on development of data mining techniques and their applications.

V. Medvedev (✉) · O. Kurasova
Institute of Mathematics and Informatics, Vilnius University,
Akademijos str. 4, 08663Vilnius, Lithuania
e-mail: Viktor.Medvedev@mii.vu.lt

O. Kurasova
e-mail: Olga.Kurasova@mii.vu.lt

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_3

In the modern world, we often deal with Big Data, because nowadays technologies are able to store and process larger and larger amount of data. Big Data is the term for a collection of data sets so large and complex that it becomes difficult to process and analyze using conventional data processing and mining tools. Big Data can be characterized by three V's: volume (large amounts of data), variety (includes different types of data), and velocity (constantly accumulating new data) [25, 27]. Data become Big Data when their volume, velocity, or variety exceed the abilities of IT systems to store, analyze, and process them [5, 16]. Big Data are not just about lots of data, they are actually a new concept providing an opportunity to find a new insight into the existing data. There are many applications of Big Data: business, technology, telecommunication, medicine, health care, bioinformatics (genetics), science, e-commerce, finance, the Internet (information search, social networks), etc. Some sources of Big Data are actually new. Big Data can be collected not only from computers, but also from billions of mobile phones, social media posts, different sensors installed in cars, utility meters, shipping and many other sources. In many cases, data are just being generated faster than they can be processed and analyzed. Clustering, classification, association rule learning and other data mining problems often arise in the modern world. There are a lot of applications in which extremely large or Big Data sets need to be explored, but which are too large to be processed by conventional data mining methods [21].

3.2.1 Cloud Computing Solutions

Common software has been based on the so-called *Service Oriented Architecture* (SOA). It is a set of principles used to build flexible, modular, and interoperable software applications. The implementation of SOA is represented by web services. A *Web Service* is a collection of functions that are packaged as a single entity and published in the network for use by other applications through a standard protocol [14]. The web service allows us to integrate heterogeneous platforms and applications. Services are running independently in the system, external components do not know how the services perform their functions. The components only care that the services would return the expected results. So, web services are widely used for on-demand computing [3].

Some important technologies related to web services are as follows: WSDL (Web Service Definition Language) , SOAP (Simple Object Access Protocol) and REST (REpresentational State Transfer) . WSDL is an XML format to describe web services as a set of endpoints operating on messages that consist of either document-oriented or procedure-oriented information. SOAP is a protocol for exchanging structured information in web services implementation in computer networks. REST is a style of software architecture that abstracts the architectural elements within distributed systems.

There are a large number of distributed and data-intensive applications and data mining algorithms that simply cannot be fully exploited without Grid or Cloud tech-

nologies [20, 29]. The main goal of Cloud computing is to give a possibility to access distributed computing environments that can utilize computing resources on demand. Cloud-based data mining allows us to distribute a compute-intensive data analysis among a large number of remote computing resources.

Special tools for building the Grid and Cloud infrastructure have been developed. The Globus toolkit (http://www.globus.org), provided by the Globus Alliance, includes various software for different purposes: security, information infrastructure, resource management, data management, fault detection [11]. Nimbus (http://www.nimbusproject.org) is an open-source toolkit focused on providing Infrastructure-as-a-Service (IaaS) capabilities to the scientific community. Amazon Elastic MapReduce (Amazon EMR) is a web service that enables researchers and data analysts to easily process large amounts of data. Using Amazon EMR, it is possible to use different capacities to perform data-intensive tasks for applications such as data mining, machine learning, scientific simulation, web indexing, and bioinformatics research (http://aws.amazon.com/whitepapers/). Google introduced an online service to process large volumes of data. The service supports ad hoc queries, reports, data mining, or even web-based applications (https://cloud.google.com).

Hadoop software (http://hadoop.apache.org) is widely used for distributed computations. The software includes some open-source software projects: MapReduce, HDFS (Hadoop Distributed File System), Hbase, Hive, Pig, etc. [32]. MapReduce is a programming model for processing large data sets as well as a running environment in large computer clusters. Due to HDFS Hadoop allows us to save the computing time, needed for sending data from one computer to another. The Hadoop Mahout (http://mahout.apache.org) library is developed for data mining, where some classification, clustering, regression, dimensionality reduction algorithms are implemented. However, there are not many data mining algorithms and, due to the MapReduce particularity, not always the existing data mining algorithm can be used easily and effectively.

Apache Spark (http://spark.apache.org) is another open-source parallel processing framework that supports in-memory processing to boost the performance of Big Data analytic tasks. Spark is designed to work with HDFS to improve MapReduce technology. Spark makes it easier for developers and data scientists to work with data and deliver advanced insights faster. For data mining purposes data scientists can use scalable machine learning library MLlib where common machine learning and statistical algorithms have been implemented. Spark is generally a lot faster than MapReduce because of the way it processes the data.

3.2.2 Scientific Workflows

Many recent data mining systems are implemented using scientific workflows. A scientific workflow is a specialized form of workflows, developed specifically to compose and execute a series of data analyses and computation procedures in scientific application. The development of scientific workflow-based systems is under the

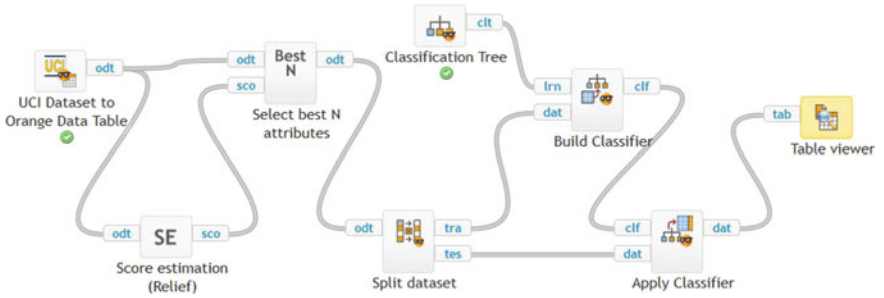


Fig. 3.1 Scientific workflow in ClowdFlows

influence of e-science technologies and applications [4]. The aim of e-science is to enable researchers to collaborate when carrying out a large scale of scientific experiments and knowledge discovery applications, using distributed systems of computing resources, devices, and data sets [1]. Scientific workflows play an important role in order to reach this aim. First of all, usage of scientific workflows allows researchers to compose convenient platforms for experiments by retrieving data from databases and data warehouses and running data mining algorithms in Cloud infrastructure. Secondly, web services can be easily imported as a new component of workflows.

Some merits of usage of scientific workflows are as follows: providing an easy-to-use drag-and-drop environment for researchers to create their own workflows for individual applications; providing interactive tools for the researchers that enable them to execute the workflows and view the results in real-time; simplifying the process of sharing and reusing workflows among the researchers (see Fig. 3.1).

Some systems for scientific workflow management are developed, for example, Pegasus workflow management system (http://pegasus.isi.edu), and Apache Airavata (http://airavata.apache.org).

3.2.3 An Overview of Web Service-Based Data Mining Solutions

Data mining is the centre of attention among various intelligent systems, because it focuses on extracting useful information from Big Data. Data mining methods were rapidly developed by extending mathematical statistics methods and creating new ones. Later on, the data mining systems, in which several methods were usually implemented, were developed in order to facilitate solving the data mining problems. Many of them are open source and available for free, therefore they have become very popular among researchers. Recently, software applications have been developed under a SOA paradigm. Thus, some new data mining systems are based on web services. Attempts are made to develop scalable, extensible, interoperable, modular and easy-to-use data mining systems.

Some popular data mining systems have been reoriented to web services. Here, at first, we discuss extensions of Weka [13], Orange [8], KNIME [23] and MATLAB (http://www.mathworks.com).

Weka4WS is an extension of the widely used Weka [13] to support distributed data mining on Grids [30]. The system should be installed on a computer, but there is a possibility to select computing resources in the Grid environment. Weka4WS adopts the web services resource framework (WSRF) technology, provided by Globus Toolkit, for running remote data mining algorithms and managing distributed computing. In Weka, the overall data mining process takes place on a single machine and the algorithms can be executed only locally. In Weka4WS, the distributed data mining tasks can be executed on various Grid nodes by exploiting the data distribution and improving the application performance. Weka4WS permits to analyze a single data set, using different data mining algorithms or using the same algorithm with different control parameters in parallel on multiple Grid nodes.

Orange4WS [26] is an extension of another well-known data mining system—Orange. Comparing with Orange, Orange4WS includes some new features. The ability to use web services as workflow components is implemented. The knowledge discovery ontology describes workflow components (data, knowledge and data mining services) in an abstract and machine-interpretable way. Usage of ontologies enables an automated composition of data mining workflows. In Orange4WS, there is a possibility to import external web services, only the WSDL file location should be specified.

The KNIME system [23] is also extended by a possibility to use web services. In KNIME labs, a web service client node is developed that allows us to import web services to KNIME workflows.

Usage of web services is implemented in a widely used programming and computing environment MATLAB (http://www.mathworks.com), too. There are implemented two types of web services RESTful and SOAP.

Considering to new technologies for data analysis, web service- and Cloud computing-based data mining systems have been designed and still underdeveloped. Some tools, related to web services, have been developed through the project myGrid (http://www.mygrid.org.uk). The tools support the creation of e-laboratories and have been used in domains of various systems: biology, social science, music, astronomy, multimedia, and chemistry. The tools have been adopted by a large number of projects: myExperiment [9], BioCatalogue [2], Taverna [24, 34], etc. MyExperiment is a collaborative environment where scientists can safely publish their workflows, share them with other scientists [9]. BioCatalogue is a registry of web services for live science [2]. Taverna is an open source and domain-independent workflow management system—a suite of tools used to create and execute scientific workflows [24, 34]. The system is oriented to web services, but the existing services are not suitable for data mining. However, there is implemented a possibility to import external data mining web service.

All the data mining systems previously reviewed (Weka4WS, Orange4WS, KNIME and Taverna) still remain desktop applications. Nowadays web applications become more popular due to the ubiquity of web browsers. ClowdFlows is a

web application based on a service oriented data mining tool [18, 19]. ClowdFlows is an open-source Cloud-based platform for composition, execution, and sharing of interactive machine learning and data mining workflows. The data mining algorithms from Orange and Weka are implemented as local services. Other SOAP services can be imported and used in ClowdFlows, too.

DAME (DAta Mining & Exploration) is another innovative web-based, distributed data mining infrastructure (http://dame.dsf.unina.it/), specialized in large data sets exploration by data mining methods [22]. The DAME is organized as a Cloud of web services and applications. The idea of DAME is to provide a user-friendly and standardized scientific gateway to ease the access, exploration, processing and understanding of large data sets. The DAME system includes not only web applications, but also several web services, dedicated to provide a wide range of facilities for different e-science communities.

The systems that provide platforms for the data analysis, using different data mining methods and technical frameworks for computing performances of learning algorithms, as well as standardize the testing procedure, recently have gained popularity. The key aim of such systems is to provide a service for comparing and testing a number of data mining methods on a large number of real data sets.

Using MLcomp (http://mlcomp.org) users can upload programs or data sets (or use the existing data sets) and run any available algorithms on any available data set through the web interface. TunedIT offers an online algorithm comparison, too [33]. TunedIT specializes in the fields of data mining, machine learning, computational intelligence and statistical modelling. TunedIT has a research platform where a user can run automated tests on data mining algorithms and get reproducible experimental results through a web interface. Galaxy is an open, web-based platform for accessible, reproducible, and transparent computational biomedical research [12]. Users without programming experience can easily specify parameters and run tools and workflows. Galaxy captures information so that any user can repeat and understand a complete computational analysis.

High performance computing (HPC) and distributed computing plays an important role in data mining. Data mining often requires huge amounts of resources in the storage space. Data are often distributed into several databases. Tasks of data mining are time consuming as well. It is important to develop mechanisms that distribute the workload of the tasks among several places in a flexible way. Distributed data mining techniques allow us to apply data mining in a non-centralized way [10]. Data mining algorithms and knowledge discovery processes are compute- and data-intensive, therefore Grids and Cloud offer a computing and data management infrastructure for supporting a decentralized and parallel data analysis [7]. So, the Cloud is an inseparable part of distributed data mining.

As mentioned before, the usage of Weka4WS allows us to distribute computations to some resources. Weka4WS is implemented using the WSRF libraries and services provided by Globus Toolkit [30]. KNIME Cluster Execution allows us to run the data mining task in computer clusters.

A review of systems and projects for Grid-based data mining is presented in [31]: Knowledge Grid, GridMiner, FAEHIM (Federated Analysis Environment for Hetero-

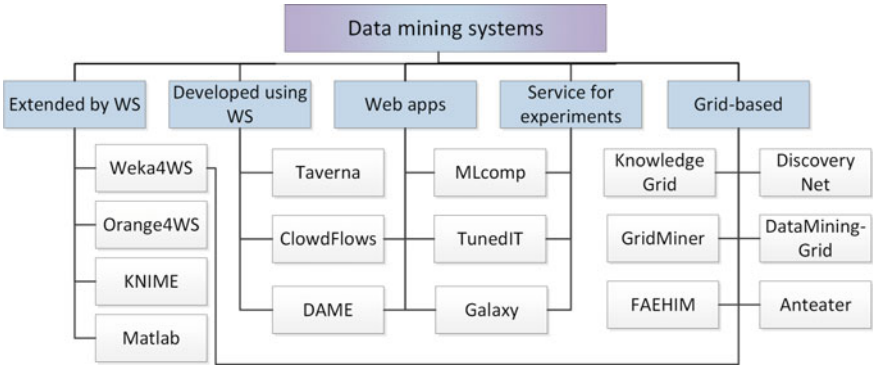


Fig. 3.2 Taxonomy of data mining systems

geneous Intelligent Mining), Discovery Net, DataMiningGrid, Anteater, etc. Most of them were created by some projects. Unfortunately, when the projects are completed, the systems are not supported anymore.

A taxonomy of the aforementioned web service-based data mining systems is performed where five classes are identified:

- The extensions of the existing popular data mining systems by web services are assigned to the class 'Extended by WS'. Their former properties remain and new ones are added.
- The systems developed using web services are assigned to the class 'Developed using WS'.
- The web application systems are assigned to the class 'Web apps'.
- The systems developed to running experiments are assigned to the class 'Services for experiments'.
- The systems based on Grid and HPC technologies are assigned to the class 'Grid-based'.

The reviewed data mining systems can be assigned to one or some classes (Fig. 3.2).

3.3 Comparative Analysis of Data Mining Systems

Some comparative analyses of data mining systems are made in [6, 15, 17, 28]. Usually, attention is focused on data mining problems and tasks, while a comparative analysis according to web services and Cloud computing is missing. The aim of this analysis is to compare web service-based underdeveloped data mining systems in order to determine the properties which should have a scalable, extensible, interoperable, modular and easy-to-use data mining systems.

Table 3.1 Grounding of evaluation criteria for data mining systems

No.	Criteria	Possible values	Grounding
C1	Web service	SOAP, RESTful	SOAP and RESTful are the most popular types of web services, thus it is important that both types would be implemented in the systems
C2	Operating system	Yes, No	It is important that the systems should support various common platforms (Windows, Linux, Mac OS)
C3	Web application	Yes, No	An advantage of web applications is that they are used and controlled by a web browser without additional installing
C4	External web services	Yes, No	It is important that it would be possible to import external web services without additional programming
C5	Cloud computing	Yes, No	Cloud and Grid computing allows us to solve complicated and time-consuming data mining problems
C6	Repository of user's data	Yes, No	Users can save the data file into the web repository and it allows performing the different experiments with the same data without uploading them
C7	Open source	Yes, No	Open source provides to extend and improve the systems
C8	Scientific workflow	Yes, No	Scientific workflows allow us to create an environment for experiments, to save that for further investigations
C9	Data mining methods	Classification, Clustering, Dimensionality reduction	The system universality is an important property in solving data mining problems, often it is necessary to apply some various data mining methods to the same problem

First of all, a set of the criteria needs to be selected, according to which the systems will be evaluated and compared. The selected criteria, their possible values and grounding are presented in Table 3.1.

The data mining systems, assigned to two classes 'Extended by WS' and 'Developed using WS' (Fig. 3.2), are selected for evaluation. Taverna is not included into the evaluation, because it has no web services for data mining. The systems assigned to other classes should be evaluated by other criteria and it is out of scope of this research. The evaluation results are presented in Table 3.2. Here the sign '+'/'–' means that a system satisfies/unsatisfies the criterion, respectfully. In the last column, the total numbers of the pluses for each system are presented. The numbers

Table 3.2 Evaluation of data mining systems based on web services

	C1		C2	C3	C4	C5	C6	C7	C8	C9			Total
	SOAP	RESTful	Multi-OS support	Web app	External WS	Cloud comp.	Web repository	Open source	Workflows	Classification	Clustering	Dim. reduction	
Weka4WS	+	–	–	–	–	+	–	+	+	+	+	+	7
Orange4WS	+	–	+	–	+	–	–	+	+	+	+	+	8
KNIME	+	–	+	–	+	–	–	+	+	+	+	+	8
MATLAB	+	+	+	–	+	+	–	–	–	+	+	+	8
ClowdFlows	+	–	+	+	+	+	–	+	+	+	+	–	9
DAME	–	+	+	+	–	–	+	–	–	+	+	–	6
Total	5	2	5	2	4	3	1	4	4	6	6	4	

of the systems, satisfying the corresponding criterion, are presented in the last row. From the results, presented in Table 3.2, the conclusions can be drawn:

- Almost all systems use only SOAP web services. The DAME system uses the RESTful protocol. MATLAB is the only system in which two types of web services are implemented.
- All systems with the exception of only Weka4WS run on Windows, Linux, and Mac operating systems.
- Only two systems ClowdFlows and DAME are implemented as web applications.
- Four systems Orange4WS, KNIME, MATLAB, and ClowdFlows have a capability to import external web services. This criterion is very important, as data mining experts can use the web services created by others.
- Four systems are open source, so they can be extended by adding new functionalities.
- Scientific workflows are implemented in four systems Weka4WS, Orange4WS, KNIME, and ClowdFlows. The main advantage of this option is that data mining scientists can choose different tools and create their workflows to solve a specific data analysis task.
- All the three groups of data mining methods are implemented in four systems (Weka4WS, Orange4WS, KNIME and MATLAB).

The total results of the analysis have showed that the system satisfying most of the criteria is ClowdFlows (9 out of 12 points). The main advantages of the ClowdFlows system are as follows: it supports multi-OS and Cloud computing, it is the open-source web application, there is possibility to import external web services and to create scientific workflows. Orange4WS, KNIME and MATLAB have got 8 points. DAME is rated poorly (only 6 out of 12 points).

From the results of the comparative analysis we can conclude that there is no data mining system that would satisfy all the criteria. Moreover, the existing systems usually cannot cope with Big Data mining problems due to large volumes and different varieties of data, as the used storage and computational resources are of limited capabilities. Thus, to process and analyze Big Data, more sophisticated data mining solutions should be developed to ensure a new level of data mining systems suitable for massive and complex data of different nature in Big Data era.

3.4 Conclusions

Big Data bring new challenges to data mining as the conventional methods and tools for data processing and analysis are unable to manage such data. This research deals with Cloud technologies for Big Data mining. The review of web service based data mining systems has been presented. Some criteria for evaluating the systems have been proposed. The analyzed systems are compared by the criteria and it has been noted that there is no data mining system that would satisfy all the criteria.

More sophisticated data mining solutions should be developed with a view to cope with challenges of time- and resource-consuming Big Data mining problems. There is a need for such a system that satisfies the following requirements:

- The system must be based on Cloud technologies. The usage of Cloud computing gives a possibility to access distributed computing environments that can utilize computing resources on demand. The system should be a multi-user web application with a possibility to set user priorities, when a task queue is formed, the tasks of users of a higher priority must be run first of all.
- The scientific workflow paradigm should be implemented in the system. Scientific workflow with drag-and-drop interface provides an easy-to-use environment for researchers to create their own workflows for individual applications. It ensures a possibility to make the results of data mining experiments and appropriate workflows accessible for other users.
- The state-of-the-art data mining methods suitable for Big Data must be implemented. It should assist to solve different problems arising in the real world such as prediction, classification, clustering, feature selection, dimensionality reduction, associative rules mining, etc.
- It should be provided a possibility through drag-and-drop interface to import external web services in order to extend the system functionality and to supplement a set of the implemented data mining methods.

The aforementioned requirements for the systems should bring the conventional data mining to a new level that should assist to cope with challenges of massive and complex data of different nature in Big Data era.

References

1. Barker, A., Van Hemert, J.I.: Scientific workflow: A survey and research directions. PPAM **4967**, 746–753 (2008). doi:10.1007/978-3-540-68111-3_78
2. Bhagat, J., Tanoh, F., Nzuobontane, E., Laurent, T., Orlowski, J., Roos, M., Wolstencroft, K., Aleksejevs, S., Stevens, R., Pettifer, S., Lopez, R., Goble, C.A.: Biocatalogue: A universal catalogue of web services for the life sciences. Nucleic Acid Res. **38** (2010). doi:10.1093/nar/gkq394
3. Birant, D.: Service-oriented data mining (2011). doi:10.5772/14066
4. Cerezo, N., Montagnat, J., Blay-Fornarino, M.: Computer-assisted scientific workflow design. J. Grid Comput. **11**(3), 585–612 (2013). doi:10.1007/s10723-013-9264-5
5. Che, D., Safran, M., Peng, Z.: From big data to big data mining: challenges, issues, and opportunities. In: Database Systems for Advanced Applications, Lecture Notes in Computer Science, pp. 1–15. Springer (2013). doi:10.1007/978-3-642-40270-8
6. Chen, X., Ye, Y., Williams, G., Xu, X.: A survey of open source data mining systems. Emerg. Technol. Knowl. Discov. Data Min. **4819**, 3–14 (2007). doi:10.1007/978-3-540-77018-3_2
7. Congiusta, A., Talia, D., Trunfio, P.: Service-oriented middleware for distributed data mining on the grid. J. Parallel Distrib. Comput. **68**, 3–15 (2008). doi:10.1016/j.jpdc.2007.07.007
8. Demšar, J., Curk, T., Erjavec, A., Črt Gorup, Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., Zupan,

B.: Orange: Data mining toolbox in Python. J. Mach. Learn. Res. **14**, 2349–2353 (2013). http://jmlr.org/papers/v14/demsar13a.html
9. De Roure, D., Goble, C., Stevens, R.: The design and realisation of the virtual research environment for social sharing of workflows (2009). doi:10.1016/j.future.2008.06.010
10. Domenico, T., Paolo, T.: Service-oriented distributed knowledge discovery. Chapman and Hall/CRC (2012)
11. Foster, I.: Globus toolkit version 4: Software for service-oriented systems. Netw. Parallel Comput. **3779**, 2–13 (2005). doi:10.1007/11577188_2
12. Goecks, J., Nekrutenko, A., Taylor, J.: Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol. **11**, R86 (2010). doi:10.1186/gb-2010-11-8-r86
13. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: An update. SIGKDD Explor. Newsl. **11**(1), 10–18 (2009). doi:10.1145/1656274.1656278
14. Heather, K.: Web services conceptual architecture (wsca 1.0). Architecture **5**, 6–7 (2001)
15. Hmida, M.B.H., Slimani, Y.: Meta-learning in grid-based data mining systems. Int. J. Commun. Networks Distrib. Syst. **5**(3), 214–228 (2010). 10.5121/ijcnc.2010.2514
16. Japkowicz, N., Stefanowski, J.: A machine learning perspective on big data analysis. In: Big Data Analysis: New Algorithms for a New Society, pp. 1–31. Springer (2016). doi:10.1007/978-3-319-26989-4
17. Jovic, A., Brkic, K., Bogunovic, N.: An overview of free software tools for general data mining. In: 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO 2014) **11**(3), 1112–1117 (2014). doi:10.1109/MIPRO.2014.6859735
18. Kranjc, J., Podpecan, V., Lavrac, N.: Clowdflows: A cloud based scientific workflow platform. In: Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, vol. 7524, pp. 816–819. Springer, Berlin, Heidelberg (2012). doi:10.1007/978-3-642-33486-3
19. Kranjc, J., Smailovič, J., Podpečan, V., Grčar, M., Žnidaršič, M., Lavrač, N.: Active learning for sentiment analysis on data streams: Methodology and workflow implementation in the clowdflows platform. Inf. Process. Manage. **51**(2), 187–203 (2014). doi:10.1016/j.ipm.2014.04.001
20. Kravtsov, V., Niessen, T., Stankovski, V., Schuster, A.: Service-based resource brokering for grid-based data mining. In: in: Proceedings of the International Conference on Grid Computing and Applications, pp. 163–169 (2006)
21. Kurasova, O., Marcinkevičius, V., Medvedev, V., Rapečka, A., Stefanovič, P.: Strategies for big data clustering. In: 26th International Conference on Tools with Artificial Intelligence (ICTAI2014), pp. 740–747. IEEE (2014). doi:10.1109/ICTAI.2014.115
22. Massimo, B., Giuseppe, L., Castellani, M., Cavuoti, S., D'Abrusco, R., Laurino, O.: Dame: A distributed web based framework for knowledge discovery in databases. Metnorie della Soc. Astron. Ital. Suppl. **19**, 324–329 (2012)
23. Meinl, T., Cebron, N., Gabriel, T.R., Dill, F., Kötter, T.: The konstanz information miner **2**, (2009). doi:10.1145/1656274.1656280
24. Missier, P., Soiland-Reyes, S., Owen, S., Tan, W., Nenadic, A., Dunlop, I., Williams, A., Oinn, T., Goble, C.: Taverna, reloaded. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 6187 LNCS, pp. 471–481 (2010). doi:10.1007/978-3-642-13818-8
25. Pattnaik, K., Mishra, B.S.P.: Introduction to big data analysis. In: Techniques and Environments for Big Data Analysis, pp. 1–20. Springer (2016). doi:10.1007/978-3-319-27520-8
26. Podpečan, V., Zemenova, M., Lavrač, N.: Orange4ws environment for service-oriented data mining. Comput. J. **55**, 82–98 (2012). doi:10.1093/comjnl/bxr077
27. Schmidt, S.: Data is exploding: the 3 versus of big data. Bus. Comput. World **15** (2012)
28. Stankovski, V., Swain, M., Kravtsov, V., Niessen, T., Wegener, D., Kindermann, J., Dubitzky, W.: Grid-enabling data mining applications with DataMiningGrid: An architectural perspective. Future Gener. Comput. Syst. **24**, 259–279 (2008). doi:10.1016/j.future.2007.05.004

29. Talia, D., Trunfio, P.: How distributed data mining tasks can thrive as knowledge services. Commun. ACM **53**, 132–137 (2010). doi:10.1145/1785414.1785451
30. Talia, D., Trunfio, P., Verta, O.: The weka4ws framework for distributed data mining in service-oriented grids. Concurrency Comput. Pract. Experience **20**, 1933–1951 (2008). doi:10.1002/cpe.v20:16
31. Werner, D.: Data Mining Meets Grid Computing: Time to Dance? John Wiley and Sons. Ltd (2009). doi:10.1002/9780470699904.ch1
32. White, T.: Hadoop: The definitive guide, vol. 54. O'Reilly Media (2012)
33. Wojnarski, M., Stawicki, S., Wojnarowski, P.: Tunedit.org: System for automated evaluation of algorithms in repeatable experiments. Rough Sets Current Trends Comput. **6086**, 20–29 (2010). doi:10.1007/978-3-642-13529-3_4
34. Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Bacall, F., Hardisty, A., Nieva de la Hidalga, A., Balcazar Vargas, M.P., Sufi, S., Goble, C.: The Taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. Nucleic Acids Res. **41**(W1), W557–W561 (2013). doi:10.1093/nar/gkt328

Chapter 4
Agent-Based High-Level Interaction Patterns for Modeling Individual and Collective Optimizations Problems

Rocco Aversa and Luca Tasquier

4.1 Introduction

Multi-agent systems are systems composed of multiple interacting computing elements, known as *agents*. Agents are computer systems with two important capabilities. First, they are at least to some extent capable of *autonomous action*—of deciding *for themselves* what they need to do in order to satisfy their design objectives. Second, they are capable of interacting with other agents—not simply by exchanging data, but by engaging in common social activities such as cooperation, coordination, negotiation, and the like [16]. In the context of social interaction, paradigms it is possible to identify two approaches that aim at different targets (Fig. 4.1): in the *Individual Intelligence* the interactions of an individual within the community are aimed at meeting the objectives of the individual, using a selfish approach; by the contrary in the *Collective Intelligence* the interaction of an individual with other entities of the same community, or with the external environment, is not only aimed at satisfying individual goals but also the ones of the community to which it belongs.

Multi-agent systems seem to be a natural metaphor for understanding and building a wide range of what is called *artificial social systems*. The ideas of multi-agent systems are not tied to a single application domain: several interaction protocols have been devised for systems of agents. In cases where the agents have conflicting goals or are simply self-interested (Individual Intelligence), the objective of the protocols is to maximize the payoffs (utilities) of the agents [11]. In cases where the agents have similar goals or common problems (Collective Intelligence), as in distributed problem solving (DPS), the objective of the protocols is to maintain globally coherent

R. Aversa · L. Tasquier (✉)
Department of Industrial and Information Engineering, Second University of Naples, Via Roma 29, Aversa, Italy
e-mail: luca.tasquier@unina2.it

R. Aversa
e-mail: rocco.aversa@unina2.it

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_4

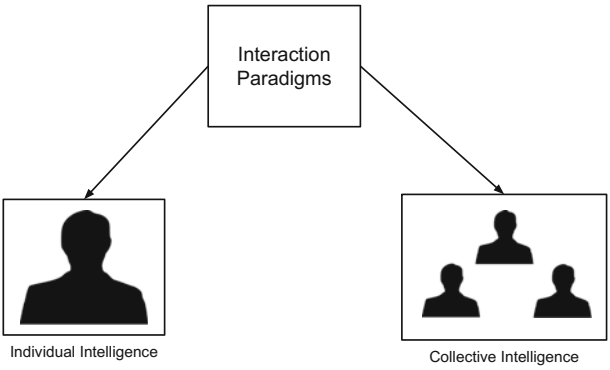


Fig. 4.1 Social interaction paradigms

performance of the agents without violating autonomy, i.e., without explicit global control [7].

Agent-based approaches are adopted for solving several types of optimization problems [2]; in [6] agent based and classic optimization approaches are compared: according to *size*, since agent based approaches support the dividing of the global problem into a number of smaller local allocation problems, large-sized problems could be handled well in such cases the problem is modular, due to the *modularity* of the agent based approaches. In terms of computational times, agent-based approaches can provide some advantages thanks to their ability to divide problems in several sub-problems: since agents are able to continuously monitor the state of local environment and typically do not have to make very complex decisions, they are able to react to changes fast, providing some advantages if the domain of the problem is characterized by a high *changeability/time scale* level. From the point of view of the *quality of solution*, since agent based approaches are distributed, they do not have a global view of the state of the system, which often is necessary in order to find a truly good solution. For these reasons, agent based approaches in optimization techniques tend to be preferable when the size of the problem is large, the domain is modular in nature and the structure of the domain changes frequently (i.e., high changeability). The application domains of multi agent systems designed for the implementation of optimization techniques are various: in [12] authors see agent based models useful for manufacturing problems because in this kind of environments the search space is very complex, with a high number of decision variables, parameters and constraints; thus the domain is multimodal and suitable for using cooperative intelligent agents. In [8] the autonomy and the intelligence of the agents are considered important elements, and the authors claim that agent based approaches can offer good features to deal with optimization problems. The research activities presented [10] led to the designing of an uncoordinated fully distributed scheduling scheme for federated cloud organizations, based on independent, competing, and

self-interested job/task execution agents, driven by optimum social welfare criteria toward a Nash equilibrium solution.

The presented work aims at defining an high-level interaction paradigm to model different optimization problems which rely on negotiation and collaboration mechanisms: the models will address both Individual and Collective Intelligence implementing them by means of agent based interaction paradigms. The proposed models will be based on general interaction protocols that can be easily specialized to fit a specific problem. Starting from high-level models, agent based architectures that maps these models will be presented in different scenarios. The first one is *Cloud Computing*, which represents a challenging test case for the presented model that implements the designed high-level model related to Individual Intelligence, due to the heterogeneity of resources, access technologies to services and providers. Cloud Computing is a computing paradigm wherein the capabilities of business applications are exposed as services that can be accessed over a network: this paradigm allows Cloud providers to increase their profits by charging consumers for accessing these services; on the other hand consumers, such as enterprises, are attracted by the opportunity for reducing or eliminating costs associated with "in-house" provision of these services [4]. The proposed multi-agent architecture allows the provisioning, management, monitoring, and reconfiguration of resources in a transparent way with respect to the heterogeneity of the infrastructures. The second scenario in which the designed model for Collective Intelligence are implemented is the context of *Smart Cities*: a city can be defined "smart" when investments in human and social capital and traditional (transport) and modern (ICT) communication infrastructure fuel sustainable economic development and a high quality of life, with a wise management of natural resources, through participatory action and engagement [5]. "Smart Cities" mainly focus on applying the next-generation information technology to all walks of life, embedding sensors and equipment to hospitals, power grids, railways, bridges, tunnels, roads, buildings, water systems, dams, oil and gas pipelines and other objects in every corner of the world. Within this context, it is presented an agent based environment that allows the collection of data about energy consumption in a neighbourhood and the negotiation of the whole produced/consumed energy among the involved parties in order to maximize the profits of the community: the collective optimization is implemented by maximizing the auto-consumption of energy by the actors in the community, and by maximizing the exchange of energy among peers in the neighbourhood, thus reducing the energy selling by external suppliers. The presented environment implements a high-level model specifically developed for the minimization of the individual and community energy costs, being compliant with the Collective Intelligence interaction paradigm.

This chapter is organized as follows: in Sect. 4.2 the model for Individual Intelligence is presented, while in Sect. 4.2.1 an application of the model within the Cloud Computing scenario is presented; the model for Collective Intelligence is described in Sect. 4.3 and its application in Smart Cities context is detailed in Sect. 4.3.1; conclusion is due in Sect. 4.4.

4.2 Individual Intelligence: An Interaction Model

Taking up from what described in Sect. 4.1, within the framework of the social interaction paradigms Individual Intelligence occurs when the interactions of an individual inside a community are aimed at meeting the objectives of the individual, using a selfish approach. For these reasons, a model that is compliant with this paradigm should have social skills in order to interact with the community, reasoning capabilities to elaborate data and information coming from individuals and environment and monitoring abilities to continuously check the satisfaction of individual requirements. According to these skills, it is proposed the interaction model depicted in Fig. 4.2; this schema is based on the "*supply and demand*" model [3], where the discriminant is a set of variables that represents the individual requirements. Being a general model, it can be easily specialized in different optimization problems that fits with these kind of operations, in particular where there is a limited market and the buyer's and seller's roles are clearly distinct (e.g., [9]).

The developed model is waterfall with feedback: it has also been designed in such a way that the individual has the ability to continuously monitor its state and the

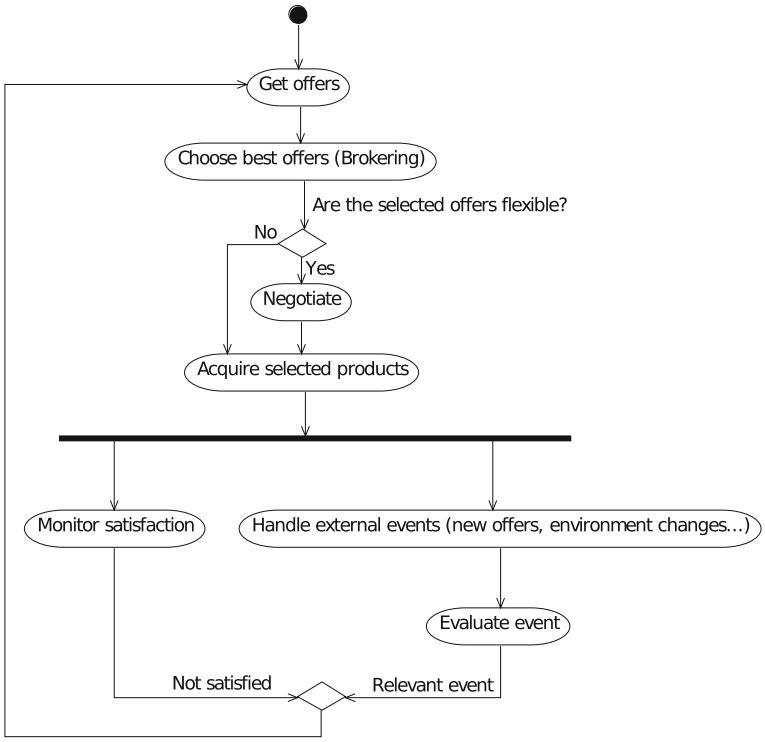


Fig. 4.2 Individual Intelligence interaction model

state of the environment in which it resides in order to continuously reach states that maximize the satisfaction of the goals, even in response to external events. Being compliant with the supply and demand schema, at the beginning of the model the individual starts asking for offers to the entities in the same community in order to fulfill its needings; after the receipt of the deals, the individual begins a *brokering* phase, where it selects the offers that best fit with its requirements among the required ones. This first optimization stage does not need interaction among the parties in order to make the choice; the brokering step is followed by an evaluation of the kind of chosen proposals: if the selected offers are not flexible, the individual proceeds in acquiring the selected products; if the deals are flexible, the individual starts a new optimization phase by interacting with the entities that made the offers in order to better fulfill its requirements: this step is called *negotiation* and involves interactions among the parties that are aimed at the satisfaction of the individual's objectives. After the negotiation part, it is possible to acquire the selected products. At this point, the model provides two parallel activities that are designed to target the optimization of the individual's objectives also after the first brokering-negotiation-purchasing step (also known as *provisioning*): one of the activity is the *satisfaction monitoring*: in this phase the fulfillment of the requirements is continuously controlled and, if the satisfaction of the objectives is no longer enough, the individual can start a new provisioning phase in order to retrieve better offers. On the other hand, the individual handles and evaluates external events that might change its requirements, such as the arrival of new entities in the community that bing with them new offers (maybe more favorable for individual's objectives), the environment's change, and so on. All these events are catched and evaluated: if one of this is relevant with respect the optimization fo the individual's requirements, the model allows a new provisioning phase.

As discussed in Sect. 4.1, due to its reactivity and proactivity characteristics and for its adaptability to the environment, the agent based model is one of the most suitable paradigms that can embody and implement the aforementioned interaction paradigms. For this reason, in the following it is presented an agent based architecture that implements the designed Individual Intelligence model: the chosen application scenario is is Cloud Computing, with particular application to IaaS level. Due to the heterogeneity of resources, access technologies to services and providers, Cloud Computing is a challenging test case for the developed model: the proposed multi-agent architecture allows the provisioning, management, monitoring and reconfiguration of resources in a transparent way with respect to the heterogeneity of the infrastructures.

By relating the developed model with respect to the Cloud Computing context, the goals of which the model is in charge of maximizing the satisfaction are:

1. *performance indexes*: the individual tries to maximize the fulfillment of its application parameters, in terms of performance indexes (such as CPU speed, memory consumption, etc.), with the ones provided by the Cloud vendors;

2. *QoS parameters*: as for the performance indexes, the goal is to maximize the meeting of application transversal requirements, such as QoS parameters like availability of the resources, throughput of the virtual network, and so on;
3. *cost*: the third parameter that is taken into account within Cloud Computing environment is the cost, that the individual tries to minimize.

On each of these three goals can be fixed some reference values that represent the thresholds for the fulfillment of the particular requirement. In the provisioning phase, the model tries to interpolate the three goals, which represent the variables of its multi-objectives utility function, in order to best meet the constraints. This step consists in brokering and negotiating offers coming from different Cloud providers, by relying on the good faith of the vendors for what concerns the effectiveness of the provided resources' parameters: the satisfaction of the requirements applied in a multi-Clouds environment. The model presented in Fig. 4.2 provides a satisfaction monitoring phase where the fulfillment of the requirements is continuously controlled. The application of this concept within multi-Clouds environments is translated in checking the effectiveness of the acquired resource's parameters in terms of performance indexes, QoS parameters and cost. If one or more of the agreed objectives and constraints are not met, the individual can start a new provisioning round, by evaluating new offers. This activity covers the case in which the vendors are not providing the resources with the agreed parameters and the case in which the requirements change while the application is running.

4.2.1 Agent Based Platform for Individual Multi-clouds Optimization

The high-level interaction model for Individual Intelligence has been implemented and validated through an agent-based platform that covers all the activities described in Sect. 4.2: Cloud Agency [15], that is a multi agent system (MAS) that accesses, on behalf of the user, the utility market of Cloud computing to maintain always the best resources configuration that satisfies the application requirements. It is in charge to provide the collection of Cloud resources, from different vendors, that continuously meets the requirements of users' applications. According to the available offers, it generates a service level agreement that represents the result of resource brokering and booking with available providers. The user is able to delegate to the Agency the monitoring of resource utilization, the necessary checks of the agreement fulfillment and eventually re-negotiations. Cloud Agency will supplement the common management functionalities which are currently provided by IaaS Private and Public infrastructure with new advanced services, by implementing transparent layer to IaaS Private and Public Cloud services. Cloud Agency architecture is depicted in Fig. 4.3.

One of the leading agents that composes Cloud Agency is the *Broker Agent* that receives the list of those resources that the application needs for its deployment and execution, asks to providers for available offers, brokers the best one and allows for

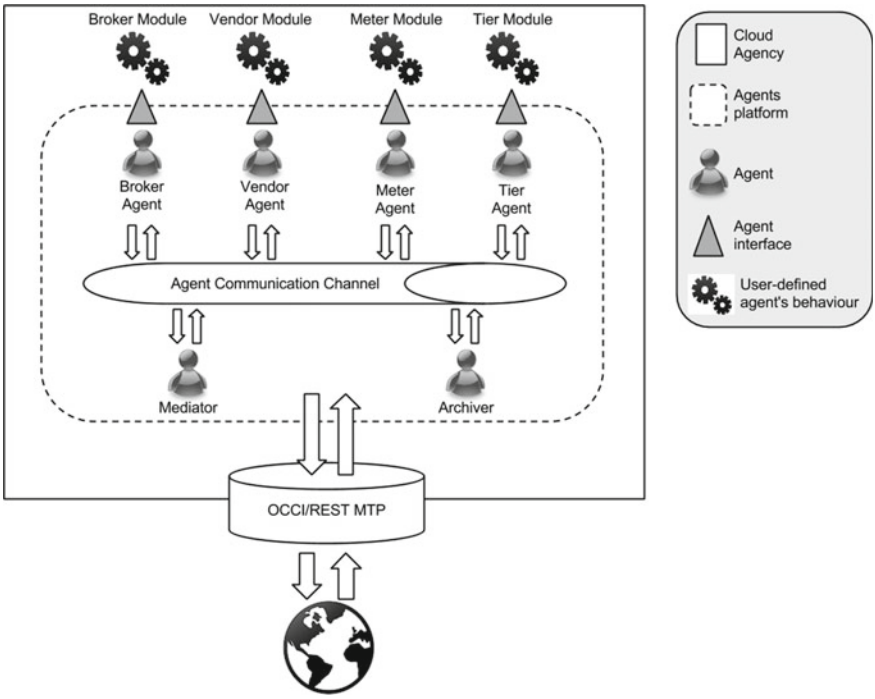


Fig. 4.3 Cloud agency architecture

closing the transaction. *Vendor Agents* implement a wrapper for a specific Cloud: they are used to get an offer for resource provisioning, to accept or refuse that offer, to get informations about a resource or to perform an action on it (start, stop, resume). Broker Agent and Vendor Agent are responsible of mapping the provisioning phase of the Individual Interaction model described in Fig. 4.2: the Vendor Agents represent the sellers queried by the user to retrieve the offers in the market and support him/her in acquiring the selected resources, while the Broker Agent embodies the user within the brokering and negotiation phase. *Meter Agents* perform measures of performance indexes at IaaS level and return their values to the *Archiver Agent*. The Archiver collects the measures and maintains statistics about the system. For the reconfiguration service the *Tier Agent* has been developed: it is triggered by the Archiver and uses policies defined by the user to apply the necessary reactions. Meter Agents, together with the Archiver, map the actions due to monitor the user's satisfaction, while the Tier Agents are responsible of handling external events, as described in Sect. 4.2. The *Mediator Agent* receives requests directly from the user: it is in charge of starting new transactions for provisioning by creating Broker Agents; it also starts new Tiers and Meters, and returns information about Cloud Agency configuration and services. Further details about Cloud Agency architecture and execution are stated in [1, 14].

4.3 Collective Intelligence: An Interaction Model

According to the definition given in Sect. 4.1, within the context of the social interaction paradigm, the dual approach to the Individual Intelligence concept is the Collective Intelligence. We talk about Collective Intelligence when the interaction of an individual with other entities of the same community, or with the external environment, is not only aimed at satisfying individual goals but also the ones of the community to which it belongs. On the basis of these skills, it has been developed an interaction model, as shown in Fig. 4.4. As the one described in Sect. 4.2, the proposed model is very general and it can well fit with the optimization problems that require an interaction and a collaboration among the individuals: it can be easily specialized within the context of the peer-to-peer networks, where the role of each individual often varies. With respect to the Individual Intelligence model, one of the main differences is that in this case the products' sellers are not passive entities within the community, but they are active players in the optimization model. For this reason, it is necessary to split the interaction model in two macro-behaviours: buyer behaviour and seller behaviour. Each individual can play both roles according to its utility function, that aims at maximizing its satisfaction and the community's

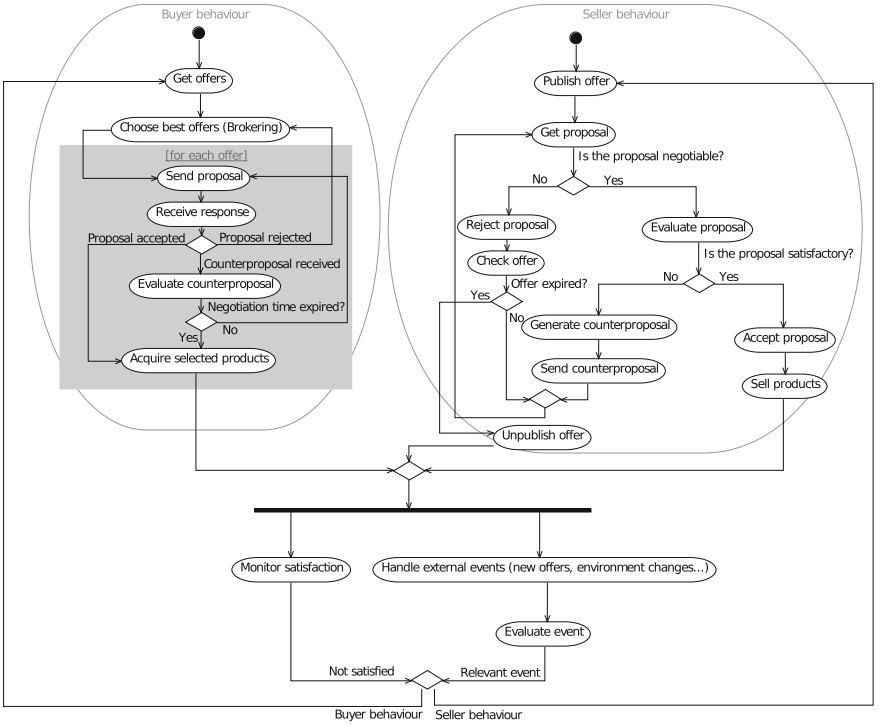


Fig. 4.4 Collective intelligence interaction model

requirements. As *buyer*, the individual starts the operations by collecting the offers coming from the belonging community; after that, it performs the brokering phase (as described for the Individual Intelligence) and, for each offer, starts a negotiation phase by providing a purchase proposal to the related seller. The received response can have three different shapes:

1. *proposal accepted*: in this case the buyer acquires the selected products from the related vendor;
2. *counterproposal received*: if the seller sends a counterproposal, the buyer evaluates it and if the negotiation time for the selected products is not expired, it sends another proposal to the vendor in order to negotiate the products' purchasing. If the negotiation time is expired, the buyer acquires the selected products without negotiating anymore;
3. *proposal rejected*: this event can occur, for example, if during the provisioning phase a vendor already sold the selected products to another buyer that provided a more convenient proposal (in terms of its objective function); in this case the buyer must re-tune its objectives by using a new brokering round.

As it is possible to understand, the behaviours of other peers in the community can impact on the behaviour of the individual that is forced to change its objectives in order to find another maximization point that maybe is not the absolute optimum for it but allows to maximize the objectives' satisfaction of the community. As *seller*, the individual starts the behaviour by publishing an offer within the community and by receiving a proposal. If the proposal is related to an offer that is not valid anymore (expired or bound to products already sold), the seller answers by rejecting the proposal. If the rejection is due to the offer's expiration, the seller unpublishes the offer; it receives a new proposal otherwise. If the proposal is negotiable, the seller evaluates it by applying its utility function (that takes into account individual and community satisfaction). If the proposal is acceptable, the individual agrees to the proposal and sells the products; if not, it generates a counterproposal, sends it to the buyer and waits for other proposals. At this point, the two macro-behaviours meet each other: in fact, every entity performs the monitoring and the external events handling activities, as well as the Individual Intelligence model. These parallel activities aim at evaluating the objectives' satisfaction: if one of these activities triggers a relevant event, the individual can decide to act as buyer or as seller according to its utility function.

In the following Section, the described Collective Intelligence interaction model will be applied within the scenario of Smart Cities, where it will be presented an agent based environment that allows the collection of data about energy consumption in a neighbourhood and the negotiation of the whole produced/consumed energy among the involved parties in order to maximize the profits of the community. The proposed formulation will be guided by two main criteria, where the role of individual is played by the house of the neighbourhood:

1. *individual optimization*: the first objective that the model will deal with is the individual optimization, that, in the context of energy trading in Smart Cities, is the minimization of house's energy cost;

2. *community optimization*: the second objective that will be considered in the model's development is the minimization of community energy cost. In order to understand deadlines and constraints for the interactions, it will be studied how the house can be characterized from an energetic point of view.

4.3.1 Agent Interaction Model for Cost Minimization

To implement the abovementioned strategy, we use the agent paradigm, building up an interaction model for Collective Intelligence that aims at minimizing the overall neighbourhood's cost; each house is modeled by an agent that adapts its behaviour in order to maximize auto-consumption of energy and minimize the exchange with the energy provider. Thus the neighbourhood is represented by a number of agents that are distributed within a "virtual" community and run autonomously in order to implement their own strategy. Thanks to its reactiveness and proactiveness capabilities, the agent paradigm is able to match the described selfish behaviour with on-demand collaboration in a distributed environment by using an asynchronous communication approach. The agent technology allows to easily react to environment's changes in order to reach the cost minimization goals; moreover, the architecture is highly scalable and can easily grow and decrease with the neighbourhood by simply adding and removing agents from the platform, thus exploiting the complete decoupling among the agents.

The minimization's strategy can be resumed in three agent's macro-beahviours

1. *maximize auto-consumption*: whatever state the agent is in, if the agent needs energy and an energy production's event occurs, this event triggers a series of state transitions that lead it to consume the produced energy;
2. *minimize energy requests to the provider*: if there is an energy request and the produced energy is not sufficient to completely satisfy the request, the agent asks for the needed energy to the neighbourhood;
3. *collaborative approach*: if the house has an excess of produced energy, the agent provides this energy to the neighbourhood.

The **agent interaction model** is drawn in Fig. 4.5 while the description of each state is provided below:

- *1: EVALUATE STATE*: in this state are performed all the operations aimed at evaluating if the agent has to acquire or to sell energy.
- *2: PUBLISH PROPOSAL*: if the house produces some exceeding energy, the agent publishes a proposal in order to sell the energy to other houses in the neighbourhood.
- *3: ASK FOR PROPOSAL*: if the house needs energy and it is not available at home, it asks the neighbourhood for energy to buy by using a Call for Proposal (CFP).
- *4: WAIT ACCEPTANCE*: in this state the house waits for the acceptance of a proposal published in state 2.

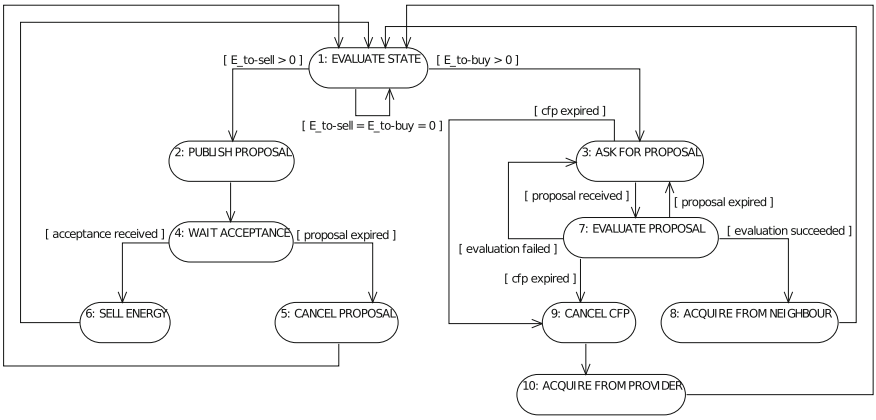


Fig. 4.5 Agent interaction model for energy cost minimization

- *5: CANCEL PROPOSAL*: if during the waiting of a proposal acceptance notification Δt passes, the proposal is canceled.
- *6: SELL ENERGY*: if a proposal acceptance notification has been received, the agent sells the agreed energy to the buyer.
- *7: EVALUATE PROPOSAL*: if a proposal is received, the agent evaluates it in order to the buy neighbour's energy.
- *8: ACQUIRE FROM NEIGHBOUR*: if a proposal evaluation succeeded, the agent buys the agreed energy from the seller.
- *9: CANCEL CFP*: if during a proposal evaluation or the waiting of proposals Δt passes, the CFP is canceled.
- *10: ACQUIRE FROM PROVIDER*: this is the worst situation; if the agent needs energy and no acceptable proposals come within Δt, the only thing that the agent can do is to acquire the needed energy from the energy supplier.

Even if the interaction model is event-based, the full set of operations is marked by Δt: every Δt the automata returns to its initial state, starting a new round of estimation-trading-purchasing/selling. As it is possible to understand, Δt becomes a crucial parameter for the algorithm performance: too small a value of Δt makes stressing the prediction algorithm and might be too short to complete the negotiation phase; too high a value of Δt makes the energy performance of the house too bind to the accuracy of the forecasting, since an incorrect prediction can impact on the energy behaviour of the house for a long period, thus reducing the performance. For these reasons, the tuning of Δt strongly impacts on the house cost minimization.
It is important to highlight that, in order to implement the high-level model, each agent can act as both buyer and seller: it is not bound to a specific role but it can adapt its behaviour to the specific requirements and execution state.

A detailed description of the specific model used to minimize the neighbourhood's energy cost together with experimental results used to validate the proposed model are presented in [13].

4.4 Conclusion

This work, starting from commonly used interaction models, proposed high-level interaction paradigms easy to specialize in order to guide an user in solving his/her optimization problem on the basis of his/her requirements: moreover, the presented models have been implemented by means of agent based paradigms and technologies. In the context of Individual Intelligence, has been presented an interaction model (waterfall with feedback) based on on the "*supply and demand*" mechanism. The application of this model to a Cloud Computing scenario led to the design and implementation of an agent based platform for the provisioning, management and monitoring of Cloud infrastructures. This framework covers all the interactions proposed in the high-level model by using the agent capabilities in terms of reactiveness, proactiveness and social attitudes. In the context of Collective Intelligence, it has been proposed a community-oriented optimization model for energy exchange in Smart Cities and, in order to validate it, it has been presented an agent based interaction model that aims at maximizing the auto-consumption of the produced energy and at buying the needed one from neighbours instead of supplier.

References

1. Aversa, R., Tasquier, L., Venticinque, S.: Cloud agency: A guide through the clouds. Mondo Digitale **13**(49) (2014)
2. Barbati, M., Bruno, G., Genovese, A.: Applications of agent-based models for optimization problems: A literature review. Expert Syst. Appl. **39**(5), 6020–6028 (2012)
3. Besanko, D., Braeutigam, R.: Microeconomics. John Wiley & Sons (2010)
4. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. **25**(6), 599–616 (2009)
5. Caragliu, A., Del Bo, C., Nijkamp, P.: Smart cities in Europe. J. urban Technol. **18**(2), 65–82 (2011)
6. Davidsson, P., Persson, J.A., Holmgren, J.: On the integration of agent-based and mathematical optimization techniques. In: Agent and multi-agent systems: technologies and applications, pp. 1–10. Springer (2007)
7. Durfee, E.H.: Coordination of distributed problem solvers. Kluwer Academic Publishers (1988)
8. Kornienko, S., Kornienko, O., Priese, J.: Application of multi-agent planning to the assignment problem. Comput. Ind. **54**(3), 273–290 (2004)
9. Kozat, U.C., Harmanci, O., Kanumuri, S., Demircin, M.U., Civanlar, M.R.: Peer assisted video streaming with supply-demand-based cache optimization. IEEE Trans. Multimedia **11**(3), 494–508 (2009)
10. Palmieri, F., Buonanno, L., Venticinque, S., Aversa, R., Di Martino, B.: A distributed scheduling framework based on selfish autonomous agents for federated cloud environments. Future Gener. Comput. Syst. **29**(6), 1461–1472 (2013)
11. Rosenschein, J.S., Zlotkin, G.: Designing conventions for automated negotiation. AI magazine **15**(3), 29 (1994)
12. Shen, W., Wang, L., Hao, Q.: Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. **36**(4), 563–577 (2006)

13. Tasquier, L., Aversa, R.: An agent-based collaborative platform for the optimized trading of renewable energy within a community. J. Telecommun. Inf. Technol. **2014**(4) (2014)
14. Venticinque, S., Tasquier, L., Di Martino, B.: Agents based cloud computing interface for resource provisioning and management. In: 2012 Sixth International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), pp. 249–256. IEEE (2012)
15. Venticinque, S., Tasquier, L., Di Martino, B.: A restfull interface for scalable agents based cloud services. Int. J. Ad Hoc Ubiquitous Comput. **16**(4), 219–231 (2014)
16. Wooldridge, M.: An introduction to multiagent systems. John Wiley & Sons (2009)

Chapter 5
Maximize Profit for Big Data Processing in Distributed Datacenters

Weidong Bao, Ji Wang and Xiaomin Zhu

5.1 Introduction

The volume of data increases drastically in recent years, which leads to a high demand for Big Data processing. Efficient processing and analysis of Big Data has exhibited its tremendous potential to facilitate decision-making, production sale and so on. Despite the huge values, Big Data processing, however, requires big infrastructures and big costs, which often exceeds the capacity most enterprises and institutes are able to afford. Fortunately, cloud computing, a revolutionary computing service provision scheme, provides a promising solution for the enterprises and institutes to process Big Data in a highly effective and economic manner. Enterprises and institutes who have the demand for Big Data processing submit their service requests to the cloud service providers (CSPs) such as Amazon, Google, and Microsoft to rent computing resources for executing Big Data processing applications.

Cloud service providers are supported by large-scale cloud datacenters that are usually distributed in different geographic regions across the world [4]. For example, Google operates 13 datacenters across eight countries. The explosion of the Big Data processing demands also incur a high cost to operate distributed datacenters for these CSPs. How to manager distributed datacenters and maximize the profit has become the major concern of CSPs. Nonetheless, there are four-fold challenges to optimize this problem. (1) The service requests in cloud environments are highly dynamic and unpredictable. It is nearly impossible for CSPs to make an accurate prediction of the

W. Bao · J. Wang (✉) · X. Zhu
College of Information System and Management, National University of Defense Technology, Changsha, Hunan, People's Republic of China
e-mail: wangji@nudt.edu.cn

W. Bao
e-mail: wdbao@nudt.edu.cn

X. Zhu
e-mail: xmzhu@nudt.edu.cn

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_5

future workload. (2) The electricity price and bandwidth price, two major operation costs of datacenters, are stochastic in practice. Combined with the first challenge, it makes the optimization in distributed datacenters far more difficult. (3) The huge amounts of computing resources and service requests incur over-burdened complexity when optimizing this problem, which requires the solution to be a distributed scheme with high time efficiency. (4) Datacenters in clouds are usually divers in terms of energy efficiency due to the different geographic locations, infrastructures, and cooling mechanisms, which make the cost different even processing the same application among the distributed datacenter.

In this work, to overcome the above-mentioned challenges and attain the optimum of CSPs' profit, we jointly take the electricity price, bandwidth price, and geographic differences into consideration, and then model service requests acceptance control, requests dispatching, and VM provisioning as an integrated optimization framework based on Lyapunov optimization theory [8]. An efficient online algorithm is proposed to help CSPs obtain maximal time-averaged profit over the long run. The rigorous theoretic analysis shows that our framework is able to arbitrarily close to optimum.

The rest of the paper is organized as follows. Section 5.2 summaries the related work. The system model and problem formulation is given in Sect. 5.3. We propose an online optimization framework in Sect. 5.4, and the performance of this framework is analyzed in Sect. 5.5. Finally, Sect. 5.6 concludes this paper.

5.2 Related Work

Due to the increasing operation costs of distributed datacenters with the explosion of Big Data processing, it is imperative to propose a highly efficient approach for CSPs to reduce operation costs and maximize profits. Consequently, a series of technologies and frameworks have been studied to cut expenses of datacenters.

Considering the geographic differences among distributed datacenters, Xu et al. [10] proposed a temperature aware workload management approach that was solved by an m-block alternating direction method of multipliers algorithms. In order to reduce energy costs and environmental impact, Liu et al. [7] studied a holistic approach of workload management integrated with renewable supply, dynamic pricing, and cooling supply. An optimal workload control and balancing by considering latency, carbon footprint, and energy cost was proposed by Gao et al. [2]. Zhang et al. [12] studied a framework for dynamic service placement in geographically distributed clouds based on control- and game-theoretic models. However, all the above works require a prior knowledge about the arrival rates of requests to achieve the long-run optimization, which is usually impossible in cloud environments.

Lyapunov optimization framework is a promising approach to solve the optimization problems in cloud computing for its advantage of requiring no knowledge about the arrival rates in advance to achieve a long-run optimization. Zhou et al. [6] designed and optimal control framework to make online decisions on request admission control, routing, and VM scheduling based on the Lyapunov techniques. However, this

work is designed for a single datacenter, and hence not suitable for the environment of multiple distributed datacenters. An online server and workload management across datacenters is proposed by Abbasi et al. [1] for minimizing operational cost while satisfying the carbon footprint reduction goal. Zhao et al. [13] proposed an online algorithm for dynamic pricing of VMs across datacenters, together with job scheduling and server provisioning in each datacenter to maximize the profit over the long run. In addition, a contribution was made in this work to allow the execution time of each job to be longer than the interval of decision making in the Lyapunov framework. Yao et al. [11] developed a two-time-scale decision strategy based on the Lyapunov optimization framework for delay tolerant workloads. These works have partly solved the optimization problem in distributed datacenter environments. Nonetheless, it is assumed that only one VM can be used by a service request over several time slots in these works, which is not appropriate for Big Data processing as the Big Data processing usually demands multiple VMs simultaneously to accelerate the processing speed. To overcome this problem, we extend the service model in this work, and propose an corresponding optimization framework for Big Data processing in distributed datacenters based on the Lyapunov optimization technology.

5.3 System Model and Problem Formulation

In this section, we first give the system model and then formulate the workload management problem for distributed cloud datacenters. We consider a cloud service provider that possesses a set $\mathscr{D}$ cloud data centers. Each datacenter $d \in \mathscr{D}$ has N^d homogeneous servers, which can dynamically change according to the system workload. It should be noted that although our model assumes that the servers in one datacenter are homogeneous, it is straightforward to extend the model to incorporate the heterogeneous case with some additional notations. A cloud service provider usually provide several types of virtual machine (VM), such as Amazon EC2. Correspondingly, a set $\mathscr{V}$ of different types of VMs are modeled in our work. A server in a datacenter can be virtualized into several VMs of the same type. n_v^d denote the number of type-v VMs that a server in datacenter d can host. The system operates in a time-slotted scheme. In every time slot t, the system make the workload management decisions to dispatch customers' service request and control the number of servers in distributed datacenters.

5.3.1 Service Request Model

Customers submit a set $\mathscr{S}$ of distinct types of service request to process the Big Data applications. Each service request $s \in \mathscr{S}$ is depicted as a tuple $(v_s, n_s, \omega_s, d_s, r_s)$, where $v_s \in \mathscr{V}$ is the type of VM requested; $n_s \in [n^{min}, n^{max}]$ is the amount of type-v_s VM requested; $\omega_s \in [\omega_s^{min}, \omega_s^{max}]$ is the amount of time slots requested;

$d_s \in [d^{min}, d^{max}]$ is the data volume processed by this kind of application; and r_s is the revenue obtained when the type-s service request is provided.

The amount of type-s service requests submitted to the CSP is denoted as $A_s(t)$. We assume that $A_s(t)$ is independent and identically distributed (i.i.d) over time slots. In addition, it is assumed that there is a peak amount of service requests A_s^{max}, such that $\{A_s(t) \leq A_s^{max}, \forall s \in \mathscr{S}, \forall t\}$.

5.3.2 Service Dispatch and Server Operation Model

Service requests first are submitted to the CSP's front-end proxy server by customers. Then, at the beginning of each time slot, the front-end server determines the requests $R_s(t)$ that are accepted by the CSP ($R_s^{min} \leq R_s(t) \leq A_s(t)$) and decides how to dispatch these requests to the distributed datacenters. We use $R_s^d(t)$ to denote the amount of type-r requests dispatched to datacenter d, $\sum_{d \in \mathscr{D}} R_s^d(t) = R_s(t)$. Once a service request s is dispatched to datacenter d, the requested VMs will run in this datacenter for ω_s time slots. $\mu_s^d(t) \in [0, \mu^{max}]$ denotes the amount of type-s requests that are scheduled to be served in datacenter d at time slot t. Each datacenter maintains a queue $Q_s^d(t)$ denoting the total size of type-s requests queued at the datacenter for service at time slot t (Initially, $Q_s^d(0) = 0, \forall s, \forall d, \forall t$). It is updated over time as follows:

$$Q_s^d(t+1) = \max\{Q_s^d(t) - n_s \mu_s^d(t), 0\} + \omega_s n_s R_s^d(t) \tag{5.1}$$

The amount of servers in a datacenter can dynamically change to make a good tradeoff between performance and cost. With the advanced intra-datacenter VM migration technique [5], the VMs can be consolidated to the minimal number of servers. Hence, we use $N_v^d(t)$ to denote the amount of servers in datacenter d that host type-v VMs at time slot t. It satisfies the following inequation:

$$N_v^d(t) \geq \sum_{s:v_s=v} n_s \mu_s^d(t)/n_v^d \tag{5.2}$$

5.3.3 Cost Model

In this paper, the operating cost of CSPs we considered consists of electricity cost and bandwidth cost. The electricity cost model is given first in the following part, and then the bandwidth cost model is described.

Considering the diversity of geo-locations and cooling infrastructures among distributed datacenters, their PUEs, representing the ratio of the total power consumption

to the power delivered to the computing infrastructures, are different. We use η^d to denote the PUE of datacenter d. Then, the power consumption of datacenter d at time slot t can be calculated as $\eta^d p_s \sum_{v \in \mathscr{V}} N_v^d(t)$, where p_s is the power consumption of one server. In addition, the electricity price varies with time and regions in practice. To capture this characteristic, $e^d(t)$ is used to denote the electricity price of datacenter d at time slot t. Consequently, the electricity cost of datacenter d at time slot t can be modeled as:

$$c_e^d(t) = e^d(t)\eta^d p_s \sum\nolimits_{v \in \mathscr{V}} N_v^d(t) \tag{5.3}$$

The bandwidth price exhibits location diversity [9]. Besides, bandwidth is usually charged based on the basic 95/5 billing scheme. So the price also varies with time in real-life market. To reflect such characteristics, we use $b^d(t)$ to denote the bandwidth cost for one unit in datacenter d at time slot t. Therefore, the bandwidth cost of datacenter d at time slot t can be modeled as:

$$c_b^d(t) = d_s \sum_{s \in \mathscr{S}} b^d(t)\mu_s^d(t) \tag{5.4}$$

5.3.4 *Profit Maximization Problem*

The CSP's profit is the difference between the revenue obtained by providing services and the operating cost. The revenue obtained at time slot t is $\sum_{s \in \mathscr{S}} r_s R_s(t)$, while the operating cost of one datacenter is $c_e^d(t) + c_b^d(t)$. Then, the CSP's profit at time slot t is

$$p(t) = \sum_{s \in \mathscr{S}} r_s R_s(t) - \sum_{d \in \mathscr{D}} c_e^d(t) + c_b^d(t) \tag{5.5}$$

The objective of this paper is to maximize the time-averaged expected profit as described below

$$maximize\ P = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} E[p(t)] \tag{5.6}$$

Then, the profit maximization problem studied in this paper can be formulated as the following Problem I:

$$\max : P \tag{5.7}$$

$$\text{s.t.} : \sum_{v \in \mathcal{V}} N_v^d(t) \leq N^d, \forall d, \forall t \tag{5.8}$$

$$\sum_{s:v_s=v} n_s \mu_s^d(t)/n_v^d \leq N_v^d(t), \forall d, \forall s, \forall t \tag{5.9}$$

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} E[\omega_s n_s R_s^d(t)] < \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} E[n_s \mu_s^d(t)], \forall d, \forall s \tag{5.10}$$

To maximize the time-averaged profit, the CSP need to strategically determines the best amount of accepted service requests $R_s(t)$, an appropriate request dispatching scheme $R_s^d(t)$, the optimal amount of service requests that are scheduled to be served $\mu_s^d(t)$, and the best amount of active servers. Constraint (8) ensures that the amount of total active servers is bounded by the amount of servers in a datacenter. Constraint (9) guarantees the amount of VMs providing service does not exceeds the amount of active servers. Constraint (10) specifies that the queue $Q_s^d(t)$ is stable.

5.4 The Online Optimization Framework

There are two main challenges for the CSP to solve Problem I in practise. First, the service requests in cloud environment are highly dynamic and unpredictable, which makes it impossible to get the key parameters such as $A_s(t)$. Second, the huge amounts of servers and service requests incur over-burdened complexity when solving the problem with a centralized method. Therefore, we have to find out an online and distributed optimization framework to make decisions efficiently.

In response to the above challenges, we exploit the advantages of Lyapunov optimization techniques [8] to design an online optimization framework that is able to achieve a time-averaged profit arbitrarily close to optimum, while keeps the system stable. To begin with, we define the Lyapunov function as follows:

$$L(t) = \frac{1}{2} \sum_{s \in \mathcal{S}} \sum_{d \in \mathcal{D}} Q_s^d(t)^2 \tag{5.11}$$

The Lyapunov function represents a scalar metric of queue congestion [8] of all the distributed datacenters. A small value of $L(t)$ suggests that the cloud system has strong stability satisfying Constraint (10). To push the Lyapunov function towards a congestion state, we then define the one-slot conditional Lyapunov drift

$$\Delta(Q(t)) = E\{L(t+1) - L(t) | Q(t)\} \tag{5.12}$$

Meanwhile, in order to maximize the time-averaged profit, the drift-plus-penalty technique in Lyapunov framework is introduced, which transforms the problem into the minimization of the upper bound for the following expression in each time slot:

$$\Delta(Q(t)) - VE\{\sum_{s\in\mathcal{S}} r_s R_s(t) - \sum_{d\in\mathcal{D}} c_e^d(t) + c_b^d(t)\} \tag{5.13}$$

The parameter V ($\geq$0) chosen by the CSP is designed to control the tradeoff between the system stability and the profit. A lower value of V implies that the CSP prefers to maintain the cloud system stable rather than obtain higher time-averaged profit. To derive the upper bound of the drift-plus-penalty expression, we provide the following lemma:

Lemma 1 *In each time slot t, for any value of $Q(t)$, the drift-plus-penalty expression is up bounded by:*

$$\begin{aligned}
&\Delta(Q(t)) - VE\{\sum_{s\in\mathcal{S}} r_s R_s(t) - \sum_{d\in\mathcal{D}} c_e^d(t) + c_b^d(t)\} \\
&\leq B_1 + E\{\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}} \omega_s n_s Q_s^d(t) R_s^d(t)\} - VE\{\sum_{s\in\mathcal{S}} r_s R_s(t)\} \\
&+VE\{\sum_{d\in\mathcal{D}} p_s e^d(t)\eta^d \sum_{v\in\mathcal{V}} N_v^d(t)\} \\
&+E\{\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}} (Vd_s b^d(t) - n_s Q_s^d(t))\mu_s^d(t)\},
\end{aligned} \tag{5.14}$$

where $B_1 = \frac{\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}} {n_s}^2 {\mu_s^{max}}^2 + \sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}} {\omega_s}^2 {n_s}^2 {A_s^{max}}^2}{2}$ *is a positive constant.*

Proof Leveraging the fact that $(\max[a-b,0]+c)^2 \leq a^2+b^2+c^2-2a(b-c)$, we have:

$$\begin{aligned}
Q_s^d(t+1)^2 \leq\ & Q_s^d(t)^2 + {n_s}^2 \mu_s^d(t)^2 + {\omega_s}^2 {n_s}^2 R_s^d(t)^2 \\
& -2Q_s^d(t)(n_s \mu_s^d(t) - \omega_s n_s R_s^d(t))
\end{aligned} \tag{5.15}$$

Based on Eqs. (5.11), (5.12) and (5.15), we can further derive:

$$\begin{aligned}
\Delta(Q(t)) \leq\ & \tfrac{1}{2} E\{\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}} {n_s}^2 \mu_s^d(t)^2\} + \tfrac{1}{2} E\{\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}} {\omega_s}^2 {n_s}^2 R_s^d(t)^2\} \\
& -E\{\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}} Q_s^d(t)(n_s \mu_s^d(t) - \omega_s n_s R_s^d(t))\}
\end{aligned} \tag{5.16}$$

Note that $\mu_s^d(t)^2$ is bounded by ${\mu_s^{max}}^2$, and $R_s^d(t)^2$ is bounded by ${A_s^{max}}^2$. By defining $B_1 = \frac{\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}} {n_s}^2 {\mu_s^{max}}^2 + \sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}} {\omega_s}^2 {n_s}^2 {A_s^{max}}^2}{2}$, and subtracting VP, the above expression can be simplified to (5.14).

Instead of minimizing the drift-plus-penalty expression directly, our approach strives to minimize the upper bound given above, and thus to maximize the lower

bound of the profit P. The stability of $Q_s^d(t)$ also can be guaranteed in this process, and hence Constraint (10) is satisfied. Then, by introducing the drift-plus-penalty technique, Problem I is transformed into the following Problem II:

$$\begin{aligned} \min: & \sum_{s\in\mathscr{S}}\sum_{d\in\mathscr{D}} \omega_s n_s Q_s^d(t) R_s^d(t) - V \sum_{s\in\mathscr{S}} r_s R_s(t) + V \sum_{d\in\mathscr{D}} p_s e^d(t) \eta^d \sum_{v\in\mathscr{V}} N_v^d(t) \\ & + \sum_{s\in\mathscr{S}}\sum_{d\in\mathscr{D}} (V d_s b^d(t) - n_s Q_s^d(t)) \mu_s^d(t) \\ \text{s.t.}: & \text{ Constraints(8)(9)} \end{aligned} \tag{5.17}$$

Based on the optimization objective of Problem II, it can be investigated that Problem II can be equivalently decoupled into two independent phases of decisions, including: (a) Request Acceptance Control and Dispatching, and (b) VM provisioning.

(a) Request Acceptance Control and Dispatching: In each time slot, the CSP decides the amount of accepted requests $R_s(t)$ and the dispatching scheme $R_s^d(t)$ for each type of service requests. To minimize (5.17), the part related to $R_s(t)$ and $R_s^d(t)$ is $\sum_{s\in\mathscr{S}}\sum_{d\in\mathscr{D}} \omega_s n_s Q_s^d(t) R_s^d(t) - V \sum_{s\in\mathscr{S}} r_s R_s(t)$. In addition, as the decision parameters of different types of requests are independent from each other, this optimization problem can be decoupled to be computed concurrently as follows:

$$\begin{aligned} \min: & \sum_{d\in\mathscr{D}} \omega_s n_s Q_s^d(t) R_s^d(t) - V r_s R_s(t) \\ \text{s.t.}: & R_s^{min} \le R_s(t) \le A_s(t) \\ & \sum_{d\in\mathscr{D}} R_s^d(t) = R_s(t) \end{aligned} \tag{5.18}$$

The above problem is a joint optimization of request acceptance control and dispatching, we first solve a simple case: if the value of $R_i(t)$ is given, then the problem (5.18) is equivalent to the problem below:

$$\begin{aligned} \min: & \sum_{d\in\mathscr{D}} \omega_s n_s Q_s^d(t) R_s^d(t) \\ \text{s.t.}: & \sum_{d\in\mathscr{D}} R_s^d(t) = R_s(t) \end{aligned} \tag{5.19}$$

To minimize (5.19), the optimal operation tends to maximally dispatch type-s requests to the datacenter d^* whose observed $Q_s^d(t)$ if the smallest among all the datacenters, i.e., $d^* = \arg\max_{d\in D} Q_s^d(t)$. Hence, all the accepted type-s requests $R_s(t)$ should be dispatched to d^*:

$$R_s^d(t) = \begin{cases} R_s(t), & d = d^*, \\ 0, & \text{else.} \end{cases} \tag{5.20}$$

Then, based on the optimal dispatching scheme, the request acceptance control decision can be converted into the following equivalent problem:

$$\begin{aligned} \min &: \omega_s n_s Q_s^{d^*}(t) R_s(t) - V r_s R_s(t) \\ \text{s.t.} &: R_s^{min} \leq R_s(t) \leq A_s(t) \end{aligned} \tag{5.21}$$

The objective function of the above problem is linear in $R_s(t)$. Hence, we can get the optimal request acceptance decision as follow:

$$R_s(t) = \begin{cases} A_s(t), & Q_s^{d^*} < \frac{V r_s}{\omega_s n_s}, \\ R_s^{min}, & \text{else}. \end{cases} \tag{5.22}$$

According to the above analysis, in each time slot, for each type of service requests, the CSP first find the datacenter with the shortest queue, and then compare the value of $Q_s^{d^*}$ with the threshold $\frac{V r_s}{\omega_s n_s}$ to decide the amount of accepted requests. After that, all the accepted requests will be dispatched to the datacenter with the shortest queue.

(b) VM Provisioning: In each time slot, each datacenter makes decisions about when and how to serve the requests dispatched to it by determining the parameter $N_v^d(t)$ and $\mu_s^d(t)$. Again, because $N_v^d(t)$ and $\mu_s^d(t)$ are independent among different datacenters, the related part in (5.17) can be decoupled as follows:

$$\begin{aligned} \min &: V p_s e^d(t) \eta^d \sum_{v \in \mathscr{V}} N_v^d(t) + \sum_{s \in \mathscr{S}} (V d_s b^d(t) - n_s Q_s^d(t)) \mu_s^d(t) \\ \text{s.t.} &: \text{Constraints}(8)(9) \end{aligned} \tag{5.23}$$

The above problem also is a joint optimization problem. Similarly, we first assume one decision parameter $N_v^d(t)$ is known in advance. Then, the problem is converted to

$$\begin{aligned} \min &: \sum_{s \in \mathscr{S}} (V d_s b^d(t) - n_s Q_s^d(t)) \mu_s^d(t) \\ \text{s.t.} &: \text{Constraints}(9) \end{aligned} \tag{5.24}$$

To minimize the objective expression of (5.26), it is straightforward to maximally serve the type-s^* requests whose observed value of $V d_s b^d(t) - n_s Q_s^d(t)$ is the smallest among all types of requests, i.e., $s^* = \arg\min_{s \in \mathscr{S}} V d_s b^d(t) - n_s Q_s^d(t)$. Therefore, the optimal value of $\mu_s^d(t)$ is

$$\mu_s^d(t) = \begin{cases} \frac{n_v^d N_v^d(t)}{n_s}, & s = s^*, \\ 0, & \text{else}. \end{cases} \tag{5.25}$$

Then, we determine $N_v^d(t)$ by solving the following equivalent problem:

$$\begin{aligned} &\min: V p_s e^d(t)\eta^d \sum_{v\in\mathcal{V}} N_v^d(t) + (V d_{s^*} b^d(t) - n_{s^*} Q_{s^*}^d(t)) \frac{n_{v_{s^*}}^d N_{v_{s^*}}^d(t)}{n_{s^*}} \\ &\text{s.t.}: \text{Constraints(8)} \end{aligned} \tag{5.26}$$

Since all the parameters are non-negative, it is intuitive to minimize all types of servers except $N_{v_{s^*}}^d(t)$. For $N_{v_{s^*}}^d(t)$, we can get the optimal value as follows:

$$N_{v_{s^*}}^d(t) = \begin{cases} N^d, & Q_{s^*}^d(t) > \frac{V p_s e^d(t)\eta^d}{n_{v_{s^*}}^d} + \frac{V d_{s^*} b^d(t)}{n_{s^*}} \\ 0, & \text{else.} \end{cases} \tag{5.27}$$

5.5 Performance Analysis

In this section, we will give two theorems to present the time-averaged performance of the system in terms of stability and profit.

Before the analysis, we first introduce the following thereom:

Theorem 1 (Existence of Optimal Randomized Stationary Policy) *For arbitrary arrival rates of service requests, there exits a randomized stationary control policy* π *that chooses feasible control decisions* $R_s^\pi(t)$, $R_s^{d^\pi}(t)$, $\mu_s^{d^\pi}(t)$ *and* $N_v^{d^\pi}(t)$ *for* $\forall t, \forall d, \forall s, \forall v$, *independent of the current queue, and gets the following steady state values:*

$$\begin{aligned} &E\{R_s^\pi(t)\} = r^*, \\ &E\{R_s^{d^\pi}(t)\} = r_s^{d^*}, \\ &E\{\omega_s R_s^{d^\pi}(t)\} > E\{\mu_s^{d^\pi}(t)\}, \\ &E\{N_v^{d^\pi}(t)\} = N_v^{d^*}. \end{aligned} \tag{5.28}$$

where r^*, $r_s^{d^*}$, *and* $N_v^{d^*}$ *is the optimal solution to the Problem I.*

Because this theorem can be proved by the similar techniques in [3], we omit the details for brevity here.

Theorem 2 (Profit Optimality) *For arbitrary arrival rates of service requests, the gap between the time averaged profit achieved by the proposed online optimization framework and the optimal profit* ξ^* *is*

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} E\{\sum_{s\in S} r_s R_s(t) - \sum_{d\in D} c_e^d(t) + c_b^d(t)\} \geq \xi^* - \frac{B_1}{V} \tag{5.29}$$

where ξ^* *is the optimal time averaged profit, and* B_1 *is a constant defined in Lemma 1.*

Proof Let $\xi(t)$ denote the profit obtained in time slot t. As our proposed method strives to choose those variables that can minimize the objective of Problem II among all the feasible decisions, including the control policy π in each time slot, we can derive

$$\begin{aligned}
&\Delta(Q(t)) - VE\{\xi(t)\} \le B_1 + E\{\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}} \omega_s n_s Q_s^d(t) R_s^{d\pi}(t)\} - VE\{\sum_{s\in\mathcal{S}} r_s R_s{}^{\pi}(t)\} \\
&+VE\{\sum_{d\in\mathcal{D}} p_s e^d(t)\eta^d \sum_{v\in\mathcal{V}} N_v^{d\pi}(t)\} \\
&+E\{\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}} (Vd_s b^d(t) - n_s Q_s^d(t))\mu_s^{d\pi}(t)\},
\end{aligned} \tag{5.30}$$

Take the expectations of both sides of the above yields

$$\begin{aligned}
&E\{L(t+1) - L(t)\} - VE\{\xi(t)\} \\
&\le B_1 + E\{\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}} \omega_s n_s Q_s^d(t) R_s^{d*}(t)\} - VE\{\sum_{s\in\mathcal{S}} r_s R_s{}^{*}(t)\} \\
&+VE\{\sum_{d\in\mathcal{D}} p_s e^d(t)\eta^d \sum_{v\in\mathcal{V}} N_v^{d*}(t)\} \\
&+E\{\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}} (Vd_s b^d(t) - n_s Q_s^d(t))\mu_s^{d*}(t)\},
\end{aligned} \tag{5.31}$$

Based on Theorem 1, we can derive

$$E\{L(t+1) - L(t)\} - VE\{\xi(t)\} \le B_1 - \xi^*, \tag{5.32}$$

By summing the above over time slots $\tau \in \{0, 1, \ldots, t-1\}$ and then divide the result by t, we get

$$\frac{E\{L(t+1) - L(0)\}}{t} - \frac{V}{t}\sum_{\tau=0}^{t-1} E\{\xi(t)\} \le B_1 - V\xi^* \tag{5.33}$$

Considering the fact that $L(t) \ge 0$ and $L(0) = 0$, we can get

$$\frac{1}{t}\sum_{\tau=0}^{t-1} E\{\xi(t)\} \ge \xi^* - \frac{B_1}{V} \tag{5.34}$$

Now, (5.29) follows by taking a lim as $t \to \infty$.

Theorem 3 (Queue Stability Bound) *For arbitrary arrival rates of service requests, there exits an $\varepsilon > 0$, the system using the proposed online optimization framework with any $V \ge 0$ can guarantee that the time averaged queue satisfy:*

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}} Q_s^d(t) \le \frac{B_1 - V\xi^*}{\varepsilon} \tag{5.35}$$

Proof The proof is similar to that of Theorem 2 by the combination usage of Lemma 1 and Theorem 1. We omit it for brevity.

5.6 Conclusion

The increasing demand of Big Data processing in distributed datacenters calls for a highly efficient framework to optimization profit of the service providers, CSPs. In this work, we jointly consider the key parameters of datacenter operations to propose an online optimization based on the Lyapunov optimization theory.

The system model and the problem formulation is first presented, based on which we transform the problem with the Lyapunov optimization theory to make a trade-off between the system stability and the profit. Then, the framework makes three independent decisions on three important control phase, including service requests acceptance control, requests dispatching, and VM provisioning. Through a rigorous mathematical analysis, the proposed framework can approach a time averaged profit that is arbitrarily close to optimum, while keeping the system stable.

References

1. Abbasi, Z., Pore, M., Gupta, S.K.S.: Online server and workload management for joint optimization of electricity cost and carbon footprint across data centers. In: 2014 IEEE International Parallel Distributed Processing Symposium, pp. 317–326 (2014)
2. Gao, P.X., Curtis, A.R., Wong, B., Keshav, S.: Its not easy being green. In: Proceedings of the ACM SIGCOMM 2012 Conference, pp. 211–222 (2012)
3. Georgiadis, L., Neely, M.J., Tassiulas, L.: Resource allocation and cross-layer control in wireless networks. Found. Trends Networking **1**(1) (2006)
4. Gu, L., Zeng, D., Guo, S., Xiang, Y., Hu, J.: A general communication cost optimization framework for big data stream processing in geo-distributed data centers. IEEE Trans. Comput. Line (2015). doi:10.1109/TC.2015.2417566
5. Hines, M.R., Deshpande, U., Gopalan, K.: Post-copy live migration of virtual machines. Sigops Operating Syst. Rev. **43**, 14–26 (2009)
6. Liu, F., Zhou, Z., Jin, H., Li, B., Li, B., Jiang, H.: On arbitrating the power-performance tradeoff in saas clouds. IEEE Trans. Parallel Distrib. Syst. **25**(10), 2648–2658 (2014)
7. Liu, Z., Chen, Y., Bash, C., Wierman, A., Gmach, D., Wang, Z., Marwah, M., Hyser, C.: Renewable and cooling aware workload management for sustainable data centers. Perform. Eval. Rev. **40**(1), 175–186 (2012)
8. Neely, M.: Stochastic network optimization with application to communication and queueing systems. Synth. Lect. Commun. Netw. **3**(1) (2010)
9. Valancius, V., Lumezanu, C., Feamster, N., Johari, R., Vazirani, V.V.: How many tiers? pricing in the internet transit market. In: Proceedings of the ACM SIGCOMM 2011 Conference, pp. 194–205 (2011)
10. Xu, H., Feng, C., Li, B.: Temperature aware workload management in geo-distributed datacenters. Acm Sigmetrics Perform. Eval. Rev. **41**(1), 373–374 (2013)
11. Yao, Y., Huang, L., Sharma, A., Golubchik, L., Neely, M.: Data centers power reduction: A two time scale approach for delay tolerant workloads. In: 2012 Proceedings IEEE INFOCOM, pp. 1431–1439 (2012)

12. Zhang, Q., Zhu, Q., Zhani, M.F., Boutaba, R.: Dynamic service placement in geographically distributed clouds. In: 2012 IEEE International Conference on Distributed Computing Systems, pp. 526—535 (2012)
13. Zhao, J., Li, H., Wu, C., Li, Z., Zhang, Z., Lau, F.: Dynamic pricing and profit maximization for the cloud with geo-distributed data centers. In: 2014 Proceedings IEEE INFOCOM, pp. 118–126 (2014)

Chapter 6
Energy and Power Efficiency in Cloud

Michał Karpowicz, Ewa Niewiadomska-Szynkiewicz, Piotr Arabas and Andrzej Sikora

6.1 Introduction

Progress in high energy physics, chemistry, and biology depends on energy-efficient data mining and computing technologies. Indeed, data centers, supporting both cloud services and high performance computing (HPC) applications, consume enormous amounts of electrical energy. In the period from 2005 to 2010 the energy consumed by data centers worldwide rose by 56 %, which was accounted between 1.1 and 1.5 % of the total electricity use in 2010. The growth of energy consumption rises operating costs of data centers but also contributes to carbon dioxide (CO2) production. According to the analysis of current trends (gesi.org/SMARTer2020), carbon dioxide emissions of the ICT industry are expected to exceed 2 % of the global emissions, a level equivalent to the contribution of the aviation [76]. Energy usage in data centers grow rapidly with the climbing demand for cloud and HPC services. However, as the analysis of current trends clearly indicates, the growth rate of ICT cannot be sustained unless the power consumption problem is addressed [1, 39, 66, 113]. In response to the created momentum new computing elements, i.e., CPUs/GPUs, memory units, disks, network interface cards (NICs), have been designed to operate in multiple (performance and idle) modes of differentiated energy-consumption levels (ACPI).

M. Karpowicz · E. Niewiadomska-Szynkiewicz (✉) · P. Arabas
Warsaw University of Technology, Warsaw, Poland
e-mail: ens@ia.pw.edu.pl

M. Karpowicz
e-mail: a.karpowicz@elka.pw.edu.pl

P. Arabas
e-mail: p.arabas@elka.pw.edu.pl

M. Karpowicz · E. Niewiadomska-Szynkiewicz · P. Arabas · A. Sikora
Research and Academic Computer Network (NASK), Warsaw, Poland
e-mail: andrzej.sikora@nask.pl

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_6

Energy efficiency (FLOPS/W) of ICT systems continues to improve (www.green500.org). However, the rate of improvement does not match the growth rate of demand for computing capacity. Unless radically new energy-aware technologies are introduced, both in hardware and software domain, it will not be possible to meet DARPAs 20-MW exaflop goal (50 GFLOPS/W) by the year 2020 [1, 39]. Computational power improvements are, in fact, heavily constrained by the energy budget that is necessary for driving data centers and cloud infrastructures. Limiting power consumption and related thermal emission has therefore become a key problem. Based on the projections of technology development, it has been argued that the continued scaling of available systems will eventually lead to a data center consuming more than a gigawatt of electrical power (at Exaflop level), a level that violates economic rationale for providing cloud or HPC services. Therefore, in order to meet the challenging goals of modern cloud and high performance computing, advances in hardware layer development require immediate improvements in the design of data center and computer network control software. It is obvious that the optimisation of energy consumption in cloud computing infrastructures is necessary and must be addressed in response to the market and environment protection needs [108]. Energy-aware infrastructure components as well as new control and optimisation strategies may save the energy utilized by the whole system through adaptation of a computing system capacity and resources to the actual demands and traffic and computing load, while ensuring quality of service.

Recently, various activities and research projects aimed at developing energy-efficient computing devices and networks have been undertaken. Approaches ranging from "green" computing and network devices, optimisation and control strategies to traffic engineering and routing protocols have been developed and investigated. In this chapter, the attention is focused on power and energy management methodologies in distributed computing environments. In general, they can be classified into two main categories [74, 118]

- static energy management (SEM),
- dynamic energy management (DEM).

The methodologies from both classes can operate on the hardware and software levels. In the static mode at the hardware level, the energy can be optimised by using low-power devices or nano-processors. Two common techniques for dynamic energy management that utilize the power supply modulation or deactivation of the idle devices can be distinguished as

- *smart standby* that leverages on the concept of introducing idle mode capabilities, i.e., the whole device or its component is automatically switched off when it is idle—there is no data to process or transmit,
- *dynamic power scaling* that adopts the power consumption of the devices to the current load—the energy demands are decreased by changing the performance of the device.

Adaptive rate and low power idle are two common techniques of power scaling approaches. The *adaptive rate* (AR) method reduces the energy demands in

a network by scaling the processing capabilities of a given device or the transmission or reception speed of the network interface. The *low power idle* (LPI) allows reducing the energy requirements by putting the device or its component into a low power mode during short inactivity periods. While dynamic power scaling approaches often involve deep modifications in the design of software and hardware components of computing and network devices, the smart standby method requires only coordination among these devices to carefully re-distribute the computing and traffic load that results from switching off selected devices or their components.

Most personal computers implement both AR and LPI techniques. The ACPI (Advanced Configuration and Power Interface) specification described in [54] defines a number of energy-aware states attained via voltage and clock frequency scaling and idle states in which the processor is in the standby mode. Development of APIs and management tools is, without a doubt, essential for optimal utilization of computing resources. On the other hand, system-wide regulation of power consumption needs to be commanded by a centralized management framework, capable of collecting and processing detailed measurements, and taking real-time coordinated actions across the data computing cloud infrastructure. The detailed taxonomy of the energy and power management in highly parametrized distributed environments can be found in [18, 20, 21, 85, 91, 115]. The selected approaches are described in this chapter.

The remainder of this chapter is organized as follows. The power consumption measurement capabilities and control techniques are described in Sect. 6.2. Approaches to server-level and network-level power consumption modeling are presented in Sect. 6.3. Energy efficient servers, data centers, and networks are described in Sects. 6.4 and 6.5. We conclude this chapter in Sect. 6.6.

6.2 Power Consumption Measurement and Control

Monitoring of cluster performance is fundamental for its efficient management. In order to keep track of how well the computing tasks are processed, cluster control systems need to collect accurate measurements of activities of cluster components. The collected measurements, including both data processing and power consumption metrics, provide feedback for management operations and serve as a basis for the design of new cluster control systems.

Figure 6.1 presents an overview of a cluster control system architecture. The racks, supplied with electric power by power distribution units (PDUs), are filled with blade servers. The racks are connected into a data center network with a hierarchy of switches (SW). The management layer is responsible for allocation of resources, job submission, adjustments of the interconnect settings, power budgeting, and system monitoring. These tasks are executed by dedicated resource allocation and job management systems (RJMS) and system-wide energy management systems (SEM). The lower control layer, composed of operating systems controlling servers, is responsible for job execution and enforcement of resource usage constraints in computing nodes [48, 65, 117].

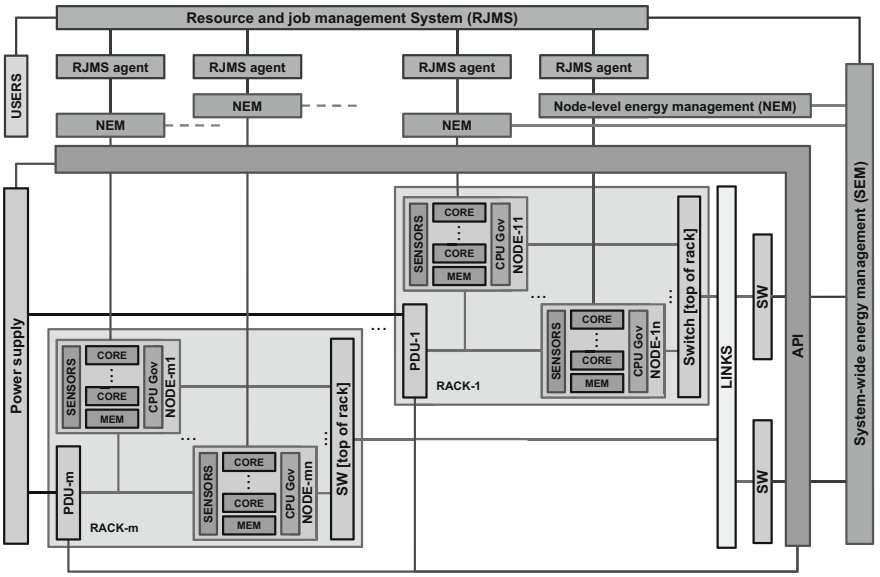


Fig. 6.1 An overview of cluster control system architecture

6.2.1 Performance Metrics and Benchmarks

Energy consumption management is a multi-objective optimisation problem in which multiple performance- and energy-related metrics are considered [14, 79, 89, 119]. Usually at least the following two objectives are considered:

- minimization of peak power consumption,
- maximization of energy-efficiency.

Indeed, limiting peak power consumption is critical to maintaining reliability of data center, avoiding power capacity overloads and system overheating. At the same time, since economically feasible power consumption levels are strongly correlated with the costs of electricity and power provisioning, it is important to maximize efficiency of operations performed in data center [39, 112, 113].

Energy-efficiency is defined as a number of operations performed per energy unit, i.e.,

$$\text{energy efficiency} = \frac{\text{computing performance}}{\text{total energy consumed}}. \tag{6.1}$$

This universal metric has been in the center of research focused on energy-aware control of data processing systems. In order for the metric to be improved, it is necessary to increase the number of operations performed per unit of energy consumed or to decrease the amount of energy required per operation. Based on the

above observations, various strategies of power management have been developed. Consequently, the metric has also been used in many benchmarking methodologies.

Basic industry-standard methodology for power and performance benchmarking of a computing server is SPECpower [77, 93, 110]. The benchmark measures power consumption of a server running an appropriately designed application (Java application server) at workload ranging from 10 to 100 % of peak achievable level. Namely, a steady flow of work requests is submitted to the server under test to determine the number of requests that can be satisfied in a given period of time. The benchmark drivers request work at intermediate points between zero and the maximum throughput value. The related toolset can be used with other cluster-wide benchmarks.

Energy-efficiency has been used as a default benchmarking metric. In the case of HPC systems (or batch processing systems), the performance metric is typically defined by the number of GFLOPS performed on average per Watt while executing a selected benchmark [40, 61]. Transaction processing systems, composed of application and web servers, as well as networks of routers have been evaluated in terms of served requests per Watt during throughput-based benchmarks [9, 28, 73]. Dedicated tests reporting transaction throughput per Watt have been developed for storage systems [41, 109], as well.

Finally, from the perspective of data center management energy-efficiency is viewed as a product of [14]:

- PUE (Power Usage Effectiveness)—facility efficiency, the ratio of total amount of energy used by a data center facility to the energy delivered to computing equipment,
- SPUE (Server Power Usage Effectiveness)—server power conversion efficiency, the ratio of total server input power to its useful power consumed by the electronic components directly involved in the computation,
- server's architectural efficiency, the ratio of computing performance metric to total amount of energy used by electronic components.

6.2.2 Power Monitoring and Profiling

Monitoring of power and energy consumption in computing clusters is a complex problem [37, 48, 51, 87]. Two general approaches can be distinguished that allow to perform the required measurements. The first one is based on power metering devices connected to the servers. Basic system-wide measurements are usually provided at rate ranging from 0.01 to several samples per second by power supply units (PSUs) and power distribution units (PDUs) through the Intelligent Platform Management Interface (IPMI) [59]. More accurate and detailed measurements, collected at high sampling rate and covering selected components of servers, may be provided by additional and dedicated metering devices [38, 52].

Whenever IPMI-based monitoring systems directly communicate with the Baseboard Management Controllers (BMC) or metering devices, there is no direct

overhead on the observed servers caused by the measurements. Otherwise, perturbations of measurements should be expected. In practice, IPMI is often used with server management software running under the local operating system. This allows to access hardware-specific function, exposed by available APIs, and conveniently deal with local measurements, control commands execution, error handling, and alerting.

Monitoring, configuration and control of devices that support IPMI on Linux systems can be performed with `ipmitool` utility. A list of sensors visible in the system and their records can be viewed with commands

```
$ ipmitool sensor
$ ipmitool sdr -v
$ ipmitool sdr elist full
```

The second approach to monitoring of power consumption exploits hardware-level counters provided by selected computing elements, including CPU, GPU, and DRAM. The most commonly available counters are exposed through Intel's Running Average Power Limiting (RAPL) interface and NVIDIA's Management Library (NVML) functions [60, 99]. Performance counters allow to collect measurements at rate ranging from 100 to 1000 samples per second with high accuracy [58]. When used together with benchmarking probes of the operating system kernel, for example, based on PAPI or `perf_events` functions allowing to observe micro-architectural event (such as instructions completed per cycle and cache-misses), a runtime estimation of power consumption and performance can be realized on a per-application basis; cf. [2, 3, 73, 114, 122]. This paves the way for high resolution identification of data processing dynamics and its energy-efficiency, and potentially for the design of energy-aware application-specific server controllers. Novel autotuning systems are also developed that allow to introduce server energy control instructions into application source code [49].

The following listing shows how Linux `stress` benchmark can be analyzed with `perf` tool based on `perf_events` subsystem of the Linux kernel.

```
$ perf stat -a \
  stress --cpu 32 --io 32 --vm 32 --vm-bytes 512M \
         --hdd 10 --timeout 30s
```

Listing 1 illustrates how an average power consumption of CPU and DRAM can be calculated in Linux systems. Listing 2 illustrates how desired RAPL MSRs (`address` variable) can be accessed from application level.[1] In order to get energy

[1] Appropriate permissions should be setup to access `/dev/cpu/*/msr` interface.

Listing 1 Simple power consumption monitoring script

```
#!/bin/bash
SAMPLING_RATE=1                  # seconds
MSR_PKG_ENERGY_STATUS="0x611"    # CPU energy counter
MSR_DRAM_ENERGY_STATUS="0x619" # DRAM energy counter
# Energy Status Units (ESU)
ESU=`echo "ibase=16;\
␣␣␣␣␣1/2^$(rdmsr␣-X␣0x606␣-f␣12:8)" | bc -l`
# Calculate number of CPU energy status
# counter incremants during sampling period
ESPKG=`a=$(rdmsr-X $MSR_PKG_ENERGY_STATUS);\
       sleep $SAMPLING_RATE; echo "ibase=16;\
␣␣␣␣␣␣␣$(rdmsr␣-X␣$MSR_PKG_ENERGY_STATUS)-$a"|bc`
# Calculate DRAM energy status
# counter incremants during sampling period
ESDRAM=`a=$(rdmsr -X $MSR_DRAM_ENERGY_STATUS);\
        sleep $SAMPLING_RATE; echo "ibase=16;\
␣␣␣␣␣␣␣␣$(rdmsr␣-X␣$MSR_DRAM_ENERGY_STATUS)-$a"|bc`
# Calculate power consumption [W] CPUPOW=`echo
"$ESPKG␣*␣ $ESU" | bc -l`
DRAMPOW=`echo "$ESDRAM ␣*␣
$ESU" | bc -l` echo CPU: $CPUPOW W echo
DRAM: $DRAMPOW W
```

consumption information from the Intel's RAPL model specific registers (MSRs)[2] it is necessary to multiply increments of appropriate energy status counters, stored in `MSR_*_ENERGY_STATUS` registers, by scaled energy status unit, stored in `MSR_RAPL_POWER_UNIT` register. Energy status MSRs are updated approximately every 1 msec, with wraparound time of around 60 s when power consumption is high [58, 60].

It is important to point out that care must be taken when MSR-based measurements are used for performance benchmarking. Since readouts are taken on the system under test the measurements may be significantly perturbed by the measurement process itself, especially under high sampling rate.

Both methods of monitoring are conveniently integrated by the resource allocation and job scheduling systems [48]. As a result it is not only possible to perform energy accounting and power profiling per job but also to setup system power-saving configuration for the purpose of job execution. Along with the scheduled batch of jobs appropriately defined control server control policies can be submitted to the computing nodes, thereby optimising energy efficiency.

[2] Intel's SandyBridge processors

Listing 2 Example of RAPL MSR read function

```
int read_msr(int cpu, unsigned int address, uint64_t *value)
{
  int   err = 0;
  char  msr_path[32];
  FILE *fp;
  sprintf(msr_path, "/dev/cpu/%d/msr", cpu);
  err = ((fp = fopen(msr_path, "r")) == NULL);
  if (!err) err = (fseek(fp, address, SEEK_CUR) != 0);
  if (!err) err = (fread(value, sizeof(uint64_t), 1, fp) != 1);
  if (fp != NULL) fclose(fp);
  return err;
}
```

6.2.3 *Power Control Programming Interfaces*

Power management capabilities of hardware layer are exposed in the form of Application Programming Interfaces (APIs). The foundations of power control APIs were built by the Advanced Configuration and Power Interface (ACPI) specification [54]. The specification defines hardware dependent energy saving (idle) and performance (active) states that can be adjusted on demand from the software level. This allows to control power saving and data processing efficiency according to a designed policy.

An attempt to design a vendor-neutral API dedicated for power measurement and control in HPC systems resulted in development of Power API specification [36, 78]. The Power API describes cluster as a collection of objects forming a discoverable hierarchy. Objects in the system are characterized by a set of attributes which allow for measurement (reading attributes) and control (overwriting attributes) of their power saving capabilities. Functions providing gathering of statistics are provided for objects and groups of objects. Both ACPI and PAPI can be adapted to the Power API abstract model.

The similar approach has been proposed by the ETSI GAL (Green Abstraction Layer) standard [26]. This standard describes a general concept of programming interface for energy state configuration of energy-aware telecommunication fixed network nodes. A hierarchical representation of a network device is proposed, which allows to control the available energy-aware states of its internal components. The innovation is not only in the described unification of control but also in the ability to query energy-aware capabilities of the components.

6.3 Power Consumption Modeling

Data center management systems require accurate power consumption and workload models for control purposes. The models can be identified based on data collected from the monitoring systems and interfaces. Two classes of models are applied, i.e., static models, taking into account only current values of system statistics, and

dynamic models, predictive models with input variables expressed by historical workload.

6.3.1 Server-Level Modeling

Commonly used static models estimate server power consumption as a polynomial function of workload w [10, 23, 105]

$$P_s(w) = \sum_{i=0}^{n} a_i w^i. \tag{6.2}$$

In its simplest form the model represents power as a linear function:

$$P_s(w) = a_0 + a_1 w, \tag{6.3}$$

where $0 \leq w \leq 1, a_0 = P_{s1}$ is the fixed power consumed by server in deep sleep mode and a_1 is the rate at which power consumption rises with the workload. Despite its simplicity Eq. (6.3) to a reasonable extent reflects internal structure and power characteristic of a server. A machine that can only operate either in idle or active state, without any power-saving mechanisms, consumes similar amount of energy at any workload, which corresponds to $a_1 \approx 0$. On the other hand, for perfectly power proportional device $P_{s1} \approx 0$. It must be noted that due to some physical phenomena (e.g., leakage currents) it is impossible to reduce the power consumption to zero, even in the no load conditions. However, P_{s1} may be lowered significantly by application of appropriate energy-saving mechanism.

Empirical studies clearly show that power consumption grows quickly between low and middle workloads, and saturates close to the capacity of the system. These nonlinear effects can be accurately represented by polynomial models of higher orders. Specialized models, designed for particular applications, have been proposed as well. For example, in [105] it is proposed to extend the linear model by introducing a power exponent term

$$P_s(w) = a_0 + a_1(2w - w^\beta), \ 1 \leq \beta \leq 2. \tag{6.4}$$

Power exponent β allows to model concavity of power characteristic. For an overview of other approaches, see e.g. [32, 50, 92].

The models discussed above treat server as a black-box adjusting its capabilities autonomously. More detailed models are required in order to describe impact of control inputs on the system workload and power consumption. In this case CPU operating modes, i.e., ACPI-compliant performance and idle states, are usually taken into account. An external control input f, describing CPU frequency corresponding to a server operating state, can be introduced to Eq. (6.3) as follows:

$$P_s(w, f) = a_0(f) + a_1(f)w \tag{6.5}$$

It should be noticed that in this case the model coefficients are parametrized by the applied control input.

Stochastic models have been used to describe average power consumption of the server switching its CPU operating mode [23, 27]. The probability $p(w, f)$ of the system being in idle state depends on CPU frequency and workload:

$$p(w, f) = 1 - \frac{w}{c(f)} \tag{6.6}$$

where $c(f)$ describes how computing power depends on CPU clock frequency. With idle probability defined as above the total expected power consumption may be described as follows:

$$P_s(w, f) = p(w, f)a_0(f) + (1 - p(w, f))a_1(f) \tag{6.7}$$

where f is CPU frequency, $a_0(f) = P_{s1}(f)$ describes power consumption during idle state and $a_1(f) = P_{s2}(f)$ is the power consumed during active state.

Network traffic processing may be an important part of workload experienced by servers in data centers. Even if request responses require complex calculations, the overall workload is usually correlated with the rate of incoming traffic. The general form of this relationship may be nonlinear, however in many cases (e.g., for network management and control) it is sufficient to approximate it with linear model of the form (6.3) [7, 71]. In this case, the overall server workload w may be attributed to the total traffic incoming on all network interfaces of the server or the rate at which requests arrive.

Finally, server power consumption may be estimated based on measurements of CPU power consumption. It can be demonstrated that the two values are correlated (see Fig. 6.2), however the character of this relation depends on a kind of operations carried out by the server [7, 71]. The results of experiments show that server power profiles may be successfully described by polynomial models of low degree

$$P_s(P_{cpu}) = a_0 + a_1 P_{cpu} + a_2 (P_{cpu})^2, \tag{6.8}$$

where P_{cpu} is the CPU power read from MSRs and a_i, $i = 0, 1, 2$ are model coefficients. More specifically, linear models may be used for CPU intensive workloads. More complicated CPU and memory intensive services, such as video streaming, exhibits nonlinearity in the form of saturation effects [46]. These, however, can be quite accurately described with second order polynomial function. Figure 6.3 illustrates both relations.

The second class of models used in power management includes dynamic models with inputs expressed by samples of historical workload estimates and commanded control signals. These models allow to design controllers adjusting system operations to the predicted average system workload. Indeed, identification of server

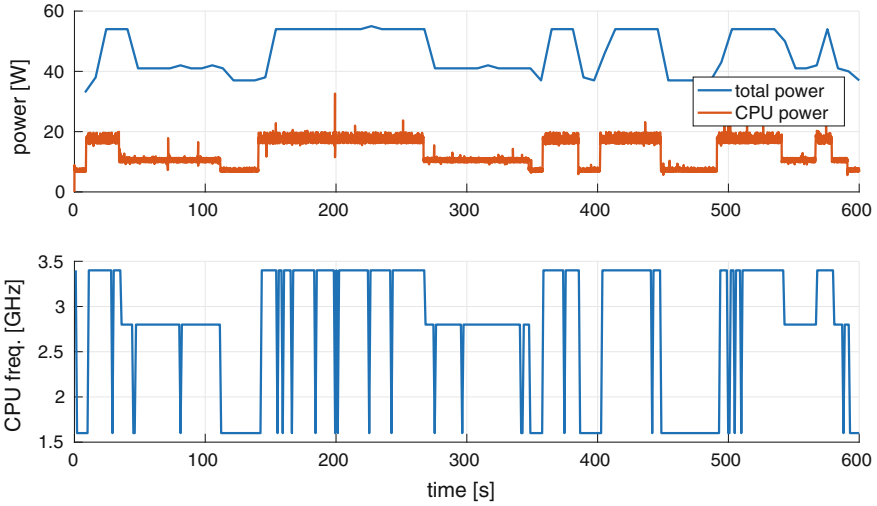
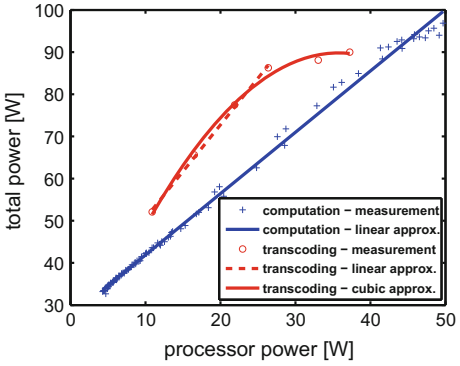


Fig. 6.2 Comparison of total power consumed by the server, processor power read from MSRs and cpu clock frequency

Fig. 6.3 Server power consumption as a function of CPU power consumption



performance dynamics is an essential step in the controller design procedure. In most typical settings linear models (ARMAX) are used [11, 84]

$$\begin{aligned} A(q)y(t) &= B(q)u(t) + C(q)e(t), \\ A(q) &= q^{n_a} + a_1 q^{n_a-1} + \cdots + a_{n_a}, \\ B(q) &= b_0 q^{n_b} + b_1 q^{n_b-1} + \cdots + b_{n_b}, \\ C(q) &= q^{n_c} + c_1 q^{n_c-1} + \cdots + c_{n_c}, \end{aligned} \tag{6.9}$$

where A, B and C represent polynomials of forward shift operator q (i.e., $qf(t) = f(t+1)$) acting on sequences of system outputs $y(t)$, control inputs $u(t)$ and random

disturbances $e(t)$. A general linear controller in the corresponding polynomial form may be defined as follows:

$$R(q)u(t) = T(q)u_c(t) - S(q)y(t), \tag{6.10}$$

where $u_c(t)$ is an external command signal representing desired setpoint. System output may represent power consumption, however, it is often related with data processing performance, workload dynamics and energy efficiency. Control inputs usually correspond to server operating modes, whereas disturbances are related with unpredictable background operations of operating systems.

To take advantage of the above formulation, the model coefficients should be identified experimentally with reasonable accuracy. For this purposes benchmarking techniques and metering systems may be used, as well as appropriate identification procedures [90]. Inherent complexities, such as multiple cores, hidden device states, and large dynamic power components, all shaping the system's dynamics, make the system difficult to describe. Appropriate choice of modeling technique, both minimizing prediction errors and insensitive to measurement noises, is indeed challenging.

Several applications of the above approach may be mentioned. In [81, 82], the above model was used to design PID controller that uses server power measurements to periodically select the highest performance state while keeping the system within a fixed power constraint. System identification approach to server workload modeling is presented in [100], together with the controller design. The model estimator captures the relationship between application performance and resource allocations, the controller allocates the right amount of computing resources to keep application performance at the required level. In [47], a design of performance optimising controller is proposed for a single application server. The paper describes the use of MIMO techniques to track desired CPU and memory utilization while capturing the related interactions between CPU and memory. An extensive survey of applied approaches to controller design exploiting dynamic models of performance and power consumption can be found in [103].

6.3.2 Network-Level Modeling

The architecture of modern network devices in many aspects resembles that of general computers. Each network node is a multi-chassis device which is composed of many entities, i.e., chassis, cpu, memory, switching fabric, line cards with communication ports, power supply, fans, etc. The only specific element—switching fabric—may be considered as a kind of specialized processor connected to common buses. The main difference is however in number and power consumption of line cards. While, in a general purpose PC there are from one to several network interfaces consuming

only a fraction of power in an average switch or router there are usually from tens to hundreds network ports occupying several line cards. Therefore, a hierarchical view of the internal organization of network devices is employed in commonly used power consumption models. It is represented through several layers, namely a device itself (the highest level), cards (the middle level) and communication interfaces (the lowest level). Each component is energy powered. The hierarchical composition is important when adjusting power states is considered—obviously it is not possible to lower energy state of a line card without lowering energy states of its ports [96].

Let us consider a computer network formed by the following components: R routers ($r = 1, \ldots, R$), C line cards ($c = 1, \ldots, C$) and I communication interfaces $i = 1, \ldots, I$). As mentioned above, the hierarchical representation of a router is assumed, i.e., each router is equipped with a number of line cards, and each card contains a number of communication interfaces. All pairs of interfaces from different cards are connected by E links ($e = 1, \ldots, E$). In case of modern networks equipped with mechanisms for dynamic power management all network components can operate in K energy states (EASs) defined as power settings, and labeled with $k = 1, \ldots, K$. Two ports connected by the eth link are in the same state k. In general, the corresponding power $P_{net}(q)$ consumed by a backbone network transfering a total traffic q can be calculated as a sum of power consumed by all network devices. In the network built of standard devices, the energy usage is relatively constant regardless of network traffic. The legacy equipment can operate only in two energy states: deep sleep ($k = 1$) and active with full power ($k = 2$). Thus, the corresponding power consumption $P_d(q)$ for total traffic q served by the network device d, i.e. a router ($d = r$), line card ($d = c$) or link connecting two interfaces ($d = e$) can be described as follows [106], (see Fig. 6.4a):

$$P_d(q) = \begin{cases} P_{d1} & \text{if } q = 0, \\ P_{d2} & \text{if } q > 0, \end{cases} \tag{6.11}$$

where P_{d1} and P_{d2} denote fixed power levels associated to the device d in deep sleep and active states, respectively.

Novel devices equipped with mechanisms for dynamic power management—e.g. InfiniBand cards switching between $1\times$ and $4\times$ mode or bundled WAN links [22, 56]—can operate in a number of dynamic modes ($k = 1, \ldots, K$), which differ in power usage [96] (see Fig. 6.4b):

$$P_d(q, k) = \begin{cases} P_{d1} & \text{if } q = 0, \\ P_{dk} & \text{if } q > 0, \end{cases} \tag{6.12}$$

where P_{d1} denotes fixed power level associated to the device d in deep sleep state and P_{dk} a fixed power level in the k-th active energy state, $k = 2, \ldots, K$.

The 802.3az standard [57] defines the implementation of low power idle for Ethernet interfaces. Numerous formal models that describe the correlation between an amount of transmitted data and energy consumption are provided in the literature.

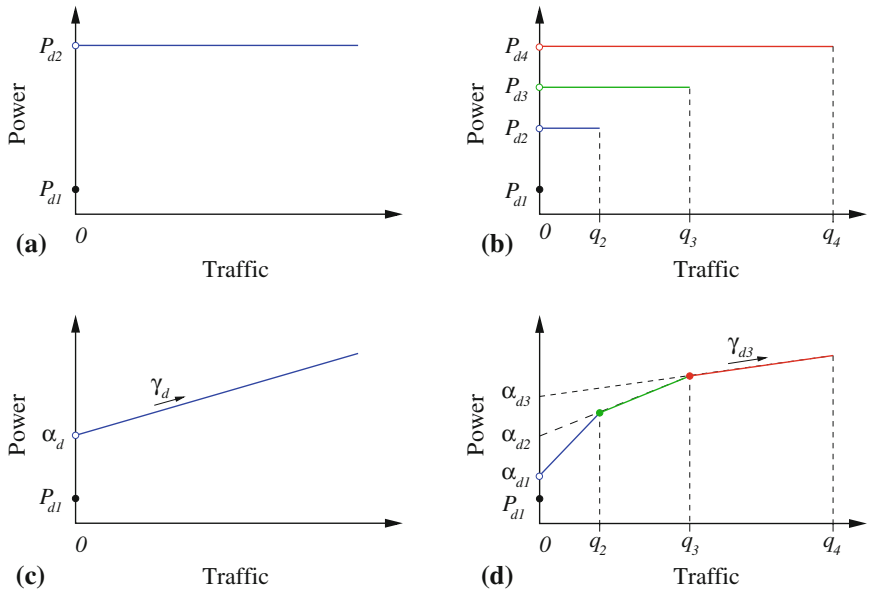


Fig. 6.4 Power consumption models **a** two energy states, **b** multiple energy states, **c** linear approximation, **d** multiple energy states, and piece-wise linear approximation

The simplest approximation is a linear modification of (6.11), see Fig. 6.4c:

$$P_d(q) = \begin{cases} P_{d1} & \text{if } q = 0, \\ \alpha_d + \gamma_d q & \text{if } q > 0, \end{cases} \tag{6.13}$$

where α_d denotes a constant offset, γ_d a coefficient defining the slope of approximated characteristic of the power consumed by the device d. The results of application of linear power consumption models for traffic engineering are presented in [62]. More precise models require nonlinear functions [23, 53].

Finally, a combined power consumption model can be formulated in which a device can operate in a number of energy states and power consumption is modeled as a piece-wise linear function of throughput (see Fig. 6.4d)

$$P_d(q,k) = \begin{cases} P_{d1} & \text{if } q = 0, \\ \alpha_{dk} + \gamma_{dk} q & \text{if } q > 0, \end{cases} \tag{6.14}$$

where α_{dk} and γ_{dk} denote coefficients for k-th active energy state, $k = 2, \ldots, K$.

6.4 Energy-Efficient Servers

According to statistical data [13, 14], utilization of servers in data centers rarely approaches 100 %. Most of the time servers operate at 10–50 % of their full capacity, which results from the requirements of providing sufficient quality of service provisioning margins. Over-subscription of computing resources is applied as a sound strategy to eliminate potential breakdowns caused by traffic fluctuations or internal disruptions, e.g., hardware or software faults. A fair amount of slack capacity is also required for the purpose of maintenance tasks. Since the strategy of resource over-provisioning is, clearly, a source of energy waste, the provisioned power supply is less than the sum of the possible peak power demands of all the servers combined. This, however, rises the problem of power distribution in data center. To keep the actual total power use within the available power range, servers are equipped with power budgeting mechanisms (e.g., ACPI-based) capable of limiting their power usage. The challenge of energy-efficient data center control is, therefore, to design control structure improving the utilization of servers and reducing energy consumption subject to quality of service constraints in highly stochastic environment (random stream of submitted jobs, OS kernel task scheduling), capable of providing fast responses to fluctuations in application workloads.

Server power control in data centers is a coordinated process that is carefully designed to reach multiple data center management objectives. As argued above, the main objectives are related with maintaining reliability and quality of data center services. These include avoiding power capacity overloads and system overheating, as well as fulfilling service-level agreements (SLAs). In addition to the primary goals, server control process aims to maximize various energy efficiency metrics subject to reliability constraints. Mechanisms designed for these purposes exploit closed-loop controllers dynamically adjusting operating modes of server components to the variable system-wide workload. Structures of currently developed control systems usually consist of two layers. The system-wide control layer adjusts power consumption of computing nodes in order to keep data center power consumption within the required bounds. Quality of services provided by the application layer is controlled by server-level control layer, see e.g. [45, 83, 120, 121].

Power consumption controllers are usually designed according to well-established methods of feedback control theory in order to reach specified objectives and to provide sufficient guarantees for performance, accuracy and stability of server operations [11, 12, 16, 44]. Apart from these goals additional requirements are introduced by hardware- and software-related constraints.

Since power consumption of CMOS circuit is proportional to its switching frequency and to the square of its operating voltage, performance, and power consumption of a processor can be efficiently controlled with dynamic voltage and frequency scaling (DVFS) mechanisms. These standard mechanisms, simultaneously adapting frequency and voltage of voltage/frequency islands (VFIs) present on the processor die [55], are commonly used as a basic method for server power control. First,

contribution of CPU in server power consumption spans from 40 to 60 % of total, which shows the dominant role CPU plays in the server power consumption profile [14, 68, 112]. Furthermore, since the difference between the maximal and minimal power usage of CPU is high, DVFS allows to compensate for the power variation of other server components. Second, most servers support DVFS of processors, power throttling of memory or other computing components is not commonly available [120].

In the Linux systems, DVFS is implemented by ACPI-based controllers. Translation of commands responsible for the CPU frequency control is provided by the `cpufreq` kernel module [72, 102]. The module allows to adjust performance of CPU by activating a desired software-level control policy implemented as a CPU *governor*. Typically, the calculated control inputs are mapped to admissible performance ACPI P-states and passed to a processor driver (e.g., `acpi_cpufreq`). Each governor of the `cpufreq` module implements a frequency switching policy. There are several standard build-in governors available in the recent versions of the Linux kernel. The governors named `performance` and `powersave` keep the CPU at the highest and the lowest processing frequency, respectively. The `userspace` governor permits user-space control of the CPU frequency. Finally, the default energy-aware governor, named `ondemand`, dynamically adjusts the frequency to the observed variations of the CPU workload. The sleeping state, or ACPI C-state, which CPU enters during idle periods, is independently and simultaneously determined by the `cpuidle` kernel module [101].

Configuration of CPU DVFS control on Linux systems can be performed with `cpufreq` configuration files or `cpupower` utility [4]. The following commands allows to retrieve available `cpufreq` kernel information and manually setup CPU frequency

```
$ cpupower frequency-info
$ cpupower frequency-set -g userspace
$ cpupower frequency-set -f $FrequencyInkHz -r
```

A straightforward DVFS-based policy of energy-efficient server control can be derived immediately from the definition of energy efficiency metric. In order to increase the number of computing operations performed per Watt it is necessary to reduce the amount of time the processor spends running idle loops or stall cycles [79]. Therefore, energy-efficiency maximizing controller should implement a workload following policy dynamically adjusting CPU performance state to the observed short-term CPU utilization or application-related latency metrics.

The above control concept is implemented in the CPU frequency governors of the Linux kernel, namely `intel_pstate` and `cpufreq_ondemand`. The output signal used in the feedback loops is an estimated CPU workload [4]. In the case of Intel IA32 architectures, it can be calculated as the ratio of the MPERF counter,

IA32_MPERF (0xe7), running at a constant frequency during active periods (ACPI C0 state), and the time stamp counter, TSC (0F 31), incremented every clock cycle at the same frequency during active and inactive periods (ACPI C-states). Another commonly applied CPU workload estimate, though somewhat less accurate, is given by the ratio of the amount of time the CPU was executing instructions since the last measurement was taken, wall_time − idle_time, to the length of the sampling period, wall_time. Given the CPU workload estimate the intel_pstate governor applied PID control rule to keep the workload at the default reference level of 97 %. The ondemand governor calculates CPU frequency according to the following policy. If the observed CPU workload is higher than the upper-threshold value then the operating frequency is increased to the maximal one. If the observed workload is below the lower threshold value, then the frequency is set to the lowest level at which the observed workload can be supported.

Many application-specific designs of the energy-efficient policy have been investigated as well. Design of decoding rate control based on DVFS, applied in multimedia-playback systems, is presented in [86]. The proposed DVFS feedback-control algorithm is designed for portable multimedia system to save power while maintaining a desired playback rate. In [75], a technique is proposed to reduce memory bus and memory bank contention by DVFS-based control of thread execution on each core. The proposed control mechanism deals with the problem of performance gap between processors and main memory, and the scenario in which memory requests simultaneously generated by multiple cores result in memory accesses delay for computing threads. A process identification technique applied for the purpose of CPU controller design is presented in [123]. Based on stochastic workload characterization, a feedback control design methodology is developed that leads to stochastic minimization of performance loss. The optimal design of the controller is formulated as a problem of stochastic minimization of runtime performance error for selected classes of applications. Adaptive DVFS governors proposed in [111] take the number of stalls introduced in the machine due to non-overlapping last-level cache misses as their input, calculate performance/energy predictions for all possible voltage-frequency pairs and select the one that yields the minimum energy-delay performance. In [67], a supervised learning technique is used in DVFS to predict the performance state of the processor for each incoming task and to reduce the overhead of the state observation. Finally, in [72] an attempt is made to characterize a possibly general structure of energy-efficient DVFS policy as a solution to stochastic control problem. The obtained structure is characterized in terms of short-term best-response function minimizing weighted sum of average costs of server power-consumption and performance. It is also compared to the ondemand governor, a default DVFS mechanism for the Linux system. The following interpretation can be given to the derived policy. Whenever it is possible for the server to process workload with the CPU frequency minimizing short-term operational cost, then frequency minimizing short-term costs should be selected. Otherwise, the controller should set the frequency optimising long-term operational cost, this however being bounded from below by the frequency minimizing short-term costs.

Finally, power consumption controllers are designed and implemented directly in the server firmware [43, 55, 80, 94, 95]. The proposed solutions allow to keep the reference power consumption set point, control peak power usage, reduce of power over-allocation and maximize energy-efficiency by following the workload demand. It should be noted that in this case hardware producers often provide mechanisms that optimise operations of all interacting server components directly during runtime.

Currently, observed trends in server power control exploit increasing possibilities provided by high-resolution sensors of modern computing hardware and software. The related research efforts seem to be drifting beyond CPU control towards increasing power proportionality of other subsystems, including memory and network interfaces. More advanced control algorithms and structures, outperforming standard PID controllers, are also proposed for device drivers and kernels of modern operating systems.

6.5 Energy-Efficient Networks

Reduction of power consumption by a network infrastructure, including both datacenter interconnect networks and wide area network, is another key aspect in the development of modern computing clouds. Recently, new solutions both in hardware and software have been developed to achieve the desired trade-off between power consumption and the network performance according to the network capacity, current traffic, and requirements of the users. In general, new network equipment is more energy effective because of modern hardware technology including moving more tasks to optical devices. However, even greater savings are obtained by employing energy-aware traffic management and modulating the energy consumption of routers, line cards and communication interfaces. In the following two subsections the techniques developed for keeping the connectivity and saving the energy that can be used for energy-efficient dynamic management in backbone and datacenter interconnect networks and LANs are surveyed and discussed.

6.5.1 Low Energy Consumption Backbone Networks

6.5.1.1 Network Energy Saving Problem Formulation

Energy consumption trends in the next generation networks have been widely discussed and the optimisation of total power consumption in today's computer networks has been a considerable research issue. Apart from improving the effectiveness of network equipment itself, it is possible to adopt energy-aware control strategies and algorithms to manage dynamically the whole network and reduce its power consumption by appropriate traffic engineering and provisioning. The aim is to reduce the gap between the capacity provided by a network for data transfer and the requirements,

especially during off-peak periods. Typically backbone network infrastructure is to some degree redundant to provide the required level of reliability. Thus, to mitigate power consumption some parts of the network may be switched off or the speed of processors and links may be reduced. According to recent studies concerning Internet service providers networks [17, 107], the total energy consumption may be substantially reduced by employing such techniques. The common approach to power management is to formulate an optimisation problem that resembles traditional network design problem [104] or QoS provisioning task [64, 88], but with a cost function defined as a sum of energy consumed by all components of the network. In contrary to the traditional network design problem, data transfers should be aggregated along as few devices as possible instead of balancing traffic in a whole computer network. The major drawback is complexity of the optimisation problem that is much more difficult to solve than typical shortest path calculation task. The complexity has roots in NP-completeness of flow problems formulated as mixed integer programming, dependencies among calculated paths and requirement for flow aggregation. Furthermore, energy consumption models are often non-convex making even continuous relaxation difficult to solve and introducing instability to suboptimal solutions [116].

Various formulations of a network energy saving problem are provided and discussed in the literature; starting from a mixed integer programming (MIP) formulation to its relaxation in order to obtain a continuous problem formulation and employ simple heuristics. The aim is to calculate optimal energy states of all network devices and corresponding flow transformation through the network. In general, due to high dimensionality and complexity of the optimisation problem, mentioned above, linear power consumption models are preferred. Some authors limit the number of energy states of network equipment (routers and cards) into active and switched off and use power consumption model (6.11), [31]. Furthermore, they propose to use multi-path routing, which is typically avoided. However, the recent trend in green networking is to develop devices with the ability to independently adapt performance of their components by setting them to one of a number of energy-aware states (power consumption model (6.12) or (6.14)) [24, 25]. It is obvious that modeling such situation implies larger dimensionality of the optimisation problem and more complicated dependencies among network components. Another approach is to exploit properties of optical transport layer to scale link rates by selectively switching off fibres composing them [42] or even build two level model with IP layer set upon optical devices layer [56]. L. Chiaraviglio et al. describe in [31] an integer linear programming formulation to determine network nodes and links that can be switched off. J. Chabarek et al. in [29] solve a large mixed-integer linear problem for a specific traffic matrix to detect the idle links and line cards. The common solution is to aggregate nodes and flows to decrease the dimension of the optimisation problem [31]. P. Arabas et al. describe in [8] four formulations of the network energy saving problem, starting from an exact MIP formulation, including complete routing and energy-state decisions, and presenting subsequent aggregations and simplifications in order to obtain a continuous problem formulation

LNPb: Link-Node Problem: a complete network management problem stated in terms of binary variables assuming full routing calculation and energy state assignment to all devices and links in a network.
LPPb: Link-Path Problem: a formulation stated in terms of binary variables assuming predefined paths (simplification of LNPb).
LNPc: Link-Node Problem: a complete network management problem stated in terms of continuous variables assuming full routing calculation.
LPPc: Link-Path Problem: a formulation stated in terms of continuous variables assuming predefined paths (simplification of LNPc).

Given the notation from Sect. 6.3.2, a complete network management problem LNPb stated in terms of binary variables k assuming energy state assignment to routers ($k = 1, \ldots, K_r$), line cards ($k = 1, \ldots, K_c$) and communication interfaces ($k = 1, \ldots, K_e$), and full routing calculation for recommended network configuration can be formulated as follows:

$$\min_{x_{rk}, x_{ck} x_{ek}, u_{ed}} \left[P_{net}^{LNPb} = \sum_{r=1}^{R} \sum_{k=1}^{K_r} P_{rk} x_{rk} + \sum_{c=1}^{C} \sum_{k=1}^{K_c} P_{ck} x_{ck} + \sum_{e=1}^{E} \sum_{k=1}^{K_e} P_{ek} x_{ek} \right], \tag{6.15}$$

subject to the set of constraints presented and discussed in [96, 98]. The power consumption and corresponding throughput are presented in Fig. 6.5: The objective function P_{net}^{LNPb} is the total power consumed by all network devices calculated for energy models of all network devices expressed by (6.12). Variables and constants used in above formulas denote: $x_{rk} = 1$, $x_{ck} = 1$, $x_{ek} = 1$ if the router r, card c, link e, respectively is in the state k (0 otherwise), $l_{ci} = 1$ if the interface (port) i belongs to the card c (0 otherwise), $u_{ed} = 1$ if the path d belongs to the link e (0 otherwise). P_{rk}, P_{ck}, P_{ek} denote the fixed power consumed by router, card, and link in the state k. To calculate the optimal energy states of network equipment the optimisation problem (6.15) has to be solved for number of assumed demands ($d = 1, \ldots, D$) imposed on the network and transmitted by means of flows allocated to given paths. The above-presented formulation requires predictions of the assumed rate V_d of each flow d that is associated with a link connecting any two ports: ports of the source and the destination nodes for the demand d.

The problem formulation obtained after flows aggregation (LPPb) is provided in [8]. Although LPPb is easier to solve due to smaller number of constraints, but is still too complex for medium-sized networks. As stated previously the widely used direction to reduce complexity of a network optimisation problem is to transform the LNPb problem stated in terms of binary variables to the LNPc with continuous variables x_{rk}, x_{ck}, x_{ek}, u_{ed}

$$\min_{x_{rk}, x_{ck}, x_{ek}, u_{ed}} \left[P_{net}^{LNPc} = \sum_{r=1}^{R} \sum_{k=1}^{K_r} P_{rk} x_{rk} + \sum_{c=1}^{C} \sum_{k=1}^{K_c} P_{ck} x_{ck} + \sum_{e=1}^{E} \sum_{k=1}^{K_e} P_{ek} x_{ek} \right] \tag{6.16}$$

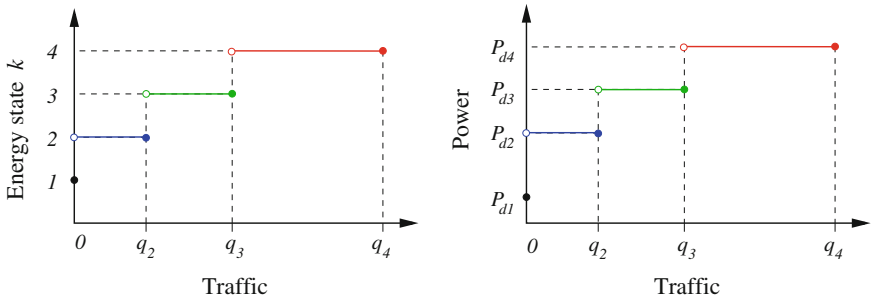


Fig. 6.5 Optimized energy-aware state switching policy (*left*) and corresponding power consumption profile (*right*)

and solve it subject to the set of constraints presented and discussed in [96]. In the above-formulation, the energy consumption and throughput utilization of the link e in the state k are described in the form of incremental model. The current values of fixed power consumption (P_{ek}) and the throughput of the link e (q_{ek}), both in the state k are calculated as follows: $P_{ek} = P_e(k) - P_e(k-1)$ and $q_{ek} = q_e(k) - q_e(k-1)$; where, respectively, $P_e(k)$ denotes power used by the link e in the state k and $q_e(k)$ denotes load of the link e in the state k. Furthermore, it is assumed that a given link can operate in more than one energy-aware state. The additional constraints are provided to force binary values of variables x_{rk}, x_{ck} in case when x_{ek} takes a binary value. The efficient heuristic to solve relaxed optimisation task (6.16) was developed and presented in [96].

The energy-aware network management problem defined above has an important disadvantage. The vector of assumed flow rates V_d of flows $d = 1, \ldots, D$, being crucial for the problem in practice is difficult to predict. When operating close to capacity limits of the network, which is often the case, a poor estimation of V_d can lead to the infeasible solution. The two criteria optimisation problem defined in [63, 70] use utility functions instead of a demand vector. The modified formulation of the optimisation problem is presented below. The modification consists in relaxing flow rates denoted by v_d. The original objective function P_{net}^{LNPb} (6.15) is augmented with a QoS-related criterion Q_d, which represents a penalty for not achieving the assumed flow rate V_d by the flow d. $Q_d(v_d)$ is a convex and continuous function, decreasing on interval $[0, V_d]$. It is reaching minimum (zero) at V_d, the point in which user expectations are fully satisfied. Finally, a two criteria—i.e., reflecting energy costs and QoS—mixed integer network problem of simultaneous optimal bandwidth allocation and routing is formulated as follows

$$\min_{x_{rk}, x_{ck}, x_{ek}, u_{dl}, v_d} \left\{ P_{net}^{2C} = \alpha P_{net}^{LNPb} + (1-\alpha) \sum_{d=1}^{D} Q_d(v_d) = \right.$$

$$= \alpha \left[\sum_{e=1,3,5,\ldots}^{E-1} \sum_{k=1}^{K_e} P_{ek} x_{ek} + \sum_{c=1}^{C} \sum_{k=1}^{K_c} P_{ck} x_{ck} + \sum_{r=1}^{R} \sum_{k=1}^{K_r} P_{rk} x_{rk} \right] + \\ + (1 - \alpha) \sum_{d=1}^{D} Q_d(v_d) \Bigg\}, \tag{6.17}$$

subject to a set of constraints defined and discussed in [70]. In the above formulation $\alpha \in [0, 1]$ is a scalarizing weight coefficient, which can be altered to emphasize any of the objectives.

6.5.1.2 Control Frameworks for Dynamic Power Management

Various control frameworks for resource consolidation and dynamic power management of the backbone network through energy-aware routing, traffic engineering and network equipment activity control have been designed and investigated [19, 22, 69, 96, 97]. They utilize smart standby and dynamic power scaling, i.e., the energy consumed by the network is minimized by deactivation of idle devices (routers, line cards, communication ports) and by reduction of the speed of link transfers. The implementation of such framework providing two variants of network-wide control, i.e., centralized and hierarchical (with two coordination schemes) both with central decision unit is described and discussed in [96]. In the presented approach, it is assumed that network devices can operate in different energy-aware states (EAS), which differ in the power usage (energy model (6.12)). The implementations of both centralized and hierarchical frameworks provide two control levels

- Local control level—control mechanisms that are implemented in the network devices level.
- Central control level—network-wide control strategies implemented in a single node controlling the whole infrastructure.

The objective of the central unit is to optimise the network performance to reduce power consumption. The decisions about activity and power status of network equipment are determined by solving LNPb or LNPc problems of minimizing the power consumption utilizing a holistic view of the system. Each local control mechanism implements adaptive rate and low power idle techniques on a given networking device. Technologies for dynamic configuration setting of the energy-saving capabilities of the network devices, e.g., described in Sect. 6.4 can be employed.

The outcomes of the levels of the control frame depend on the implementation. In the centralized scenario, the suggested power status of network devices are calculated by the optimisation algorithm executed by the central unit, and then sent to adequate network devices. Furthermore, the routing tables for the MPLS protocol for recommended network configuration are provided. Hence, in this scenario, the activity of each local controller is reduced to comply with the recommendations cal-

culated by the central unit, taking into account constraints related to current local load and incoming traffic. In the hierarchical scenario, the central unit does not directly force the energy configuration of the devices. The outcome of the central controller is reduced to routing tables for the MPLS protocol that are used for routing current traffic within a given network. The objective of the local algorithm implemented in the devices level is to optimise the configuration of each component of a given device in order to achieve the desired trade off between energy consumption and performance according to the incoming traffic and measured load of a given device. The CPU frequency controller described in Sect. 6.4 can be used.

Utility and efficiency of control frameworks employing LNPb and LNPc schemes for calculating optimal energy states of devices for small-, medium- and large-size network topologies were verified by simulation and by laboratory tests. The test cases were carried on a number of synthetic as well as on real network topologies, giving encouraging results. The results are presented and discussed in [96].

In the presented control frameworks, the total power utilized in a network for finalizing all required operations is minimized and end-to-end QoS is ensured. However, both in centralised and hierarchical variants the holistic view of a network is utilized to calculate the optimal performance of a system. Furthermore, most calculations are conducted on a selected node. Due to networks scalability and reliability distributed control is recommended. Distributing energy-aware control mechanisms extend existing mechanisms—typically routing protocols, e.g., OSPF and MPLS [19, 33, 35], BGP and MPLS. Important profit of close cooperation with signaling protocols is that the observed state of the network may be used to estimate flows and to reconstruct the traffic matrix [30].

An agent-based heuristic approach to energy-efficient traffic routing has been presented in [69]. Identical agents are running on top of OSPF processes, that activate or deactivate local links, and the decisions are based on common information about whole network state. Individual decisions affect in turn OSPF routing. Simulations show that perfect knowledge about the origin-destination matrix improves energy savings not much more than when simple local heuristics are applied by agents. On the other hand, imperfect information about origin-destination matrix can make the result worse than in case when there is no energy-saving algorithm running at all. The proposed approach is viable to implement on any routing device, through command line and basic OSPF protocol.

Bianzino et al. describe in [19] the distributed mechanism GRiDA (Green Distributed Algorithm) adopting the sleep mode approach for dynamic power management. GRiDA is an on line algorithm designed to put into sleep mode links in an IP-based network. Contrary to the centralised and hierarchical control schemes utilizing LNPb and LNPc for energy saving in computer networks, GRiDA does neither require a centralized controller node, nor the knowledge of the expected traffic matrix. This solution is based on a reinforcement learning technique that requires only the exchange of periodic Link State Advertisements (LSA) in the network. Thus, the switch off decision is taken considering the current load of incident links and the learning based on past decisions. GRiDA is fully distributed among the nodes to: (i) limit the amount of shared information, (ii) limit coordination among nodes,

and (iii) reduce the problem complexity. It is a robust and efficient solution. The simulation results of its application to various network scenarios are presented and discussed in [19].

6.5.2 Datacenter Interconnect Network

The networks in datacenters and clusters must fill specific requirements for data transfer rate, latency and reliability. Another important goal is to provide a clear cabling structure allowing to pack equipment in cabinets effectively. Typical topologies are highly regular, usually hierarchical. The most popular technologies are Ethernet and InfiniBand [15]. Consistent technology used across datacenter allows to build switched network limiting delays and complexity. Usually, datacenter networks are constructed with three layers of switches. Lowest level switches (ToR—*Top of the Rack*) are installed in the same cabinet as the servers connected directly to them. To attain high reliability and multiply bandwidth, the servers use more than one network interface connected to different switches. Similarly, ToR switches are connected with two upper-level aggregation devices. The pairs of aggregating switches are interconnected using redundant links. Due to limited number of devices the aggregation layer can be interconnected in full mesh via the highest layer core switches [34].

The resulting multiple tree topologies are substantially redundant in order to separate traffic. Paths leading through disjoint physical links allow to attain high throughput, reliability and, if necessary security. On the other hand, such an architecture implies relatively high energy consumption. It seems natural that this can be reduced in the periods of lower load by switching off some links or switches. Furthermore, as a whole network is managed by the same institution and collocated with the rest of equipment it is possible and in many aspects favorable to use a centralized network management controller. This way energy-efficient interconnect control may be calculated and implemented. Unfortunately large number of nodes and links makes mathematical programming task (e.g., of the form known from [104]) formidable and impossible to solve in acceptable time.

The above-described topology offers high reliability and bandwidth at the cost of energy demanding provider grade equipment in upper layers. Recent works propose a number of simplified topologies based on commodity switches at least at two lower layers. Moreover, limited number of redundant links and regular topology makes it possible to apply effective heuristic algorithms to manage power consumption. An example of such an approach is a combined Clos with fat-tree topology [6]. A large number of commodity switches can be used in all layers to multiply their capacity (in terms of number of ports and bandwidth) and bisection bandwidth of the whole network. Another approach is to use a highly specialized equipment and flatten the network structure to reduce graph diameter, like in the Flattened Butterfly topology proposed in [5]. The idea is to use switches with relatively high number of ports linked along several dimensions. In a simplest case a group of n switches forms a dimension, each of them connects to $n - 1$ switches in all dimensions it belongs to, it also serves

n computing nodes. The topology is flat in the sense that the longest shortest path (diameter) may be shorter than in the fat-tree network (3 vs. 4 when basic examples are considered). Despite the resulting complicated cabling the network requires less equipment, which allows to save energy. Further savings are possible using link rate adaptation assisted by adaptive routing.

6.6 Summary and Conclusions

Energy awareness is an important aspect of modern computing systems design and management, especially in the case of data intensive large-scale distributed computing systems and Internet-scale networks. This chapter presents an overview of selected technologies, architectures and methods that allow to reduce energy consumption in such infrastructures, which is also the main reason for reducing the total cost of running a data center and a network. The attention is focused on measurement technologies and modeling of energy consumption by computing and data transmitting devices. Advantages and disadvantages of various models, control structures, and mechanisms are presented and discussed.

Acknowledgments This research was partially supported by the National Science Centre (NCN) under the grant no. 2015/17/B/ST6/01885.

References

1. ETP4HPC Strategic Research Agenda Achieving HPC leadership in Europe. www.etp4hpc.eu
2. www.icl.cs.utk.edu/papi
3. www.web.eece.maine.edu/~vweaver/projects/perf_events
4. www.kernel.org/doc/Documentation/
5. Abts, D., Marty, M.R., Wells, Ph.M., Klausler, P., Liu, H.: Energy proportional datacenter networks. SIGARCH Comput. Archit. News **38**(3), 338–347 (2010). June
6. Al-Fares, M., Loukissas, M., Vahdat, A.: A scalable, commodity data center network architecture. In: Proceedings SIGCOMM 2008 Conference on Data Communications, Seattle, WA, pp. 63–74 (2008)
7. Arabas, P., Karpowicz, M.: Server power consumption: measurements and modeling with msrs. In: Proceedings AUTOMATION-2016, March 2–4, 2016, Warsaw, Poland, pp. 233–244. Springer International Publishing (2016)
8. Arabas, P., Malinowski, K., Sikora, A.: On formulation of a network energy saving optimization problem. In: Proceedings of 4th International Conference on Communications and Electronics (ICCE 2012), pp. 122–129 (2012)
9. Arabas, P., Karpowicz, M.: Server power consumption: measurements and modeling with MSRs. In: Challenges in Automation, Robotics and Measurement Techniques, pp. 233–244. Springer (2016)
10. Arjona Aroca, J., Chatzipapas, A., Fernández Anta, A., Mancuso. V.: A measurement-based analysis of the energy consumption of data center servers. In: Proceedings 5th International Conference on Future Energy Systems, pp. 63–74. ACM (2014)

11. Åström, K.J., Wittenmark, B.: Computer-controlled systems: theory and design. Dover Publications, Mineola (2011)
12. Åström, K.J., Hägglund, T.: Advanced PID control. ISA-The Instrumentation, Systems, and Automation Society; Research Triangle Park, NC 27709 (2006)
13. Andr Barroso, L., Hlzle, U.: The case for energy-proportional computing. IEEE Comput. **40**(12), 3337 (2007)
14. André Barroso, L., Clidaras, J., Hölzle, U.: The datacenter as a computer: an introduction to the design of warehouse-scale machines. Morgan & Claypool Publishers (2013)
15. Benito, M., Vallejo, E., Beivide, R.: On the use of commodity ethernet technology in exascale hpc systems. In: Proceedings IEEE 22nd International Conference on High Performance Computing (HiPC), pp. 254–263 (2015)
16. Bertsekas, D.P.: Dynamic Programming and Optimal Control, 3rd edn. Athena Scientific, Belmont (2005)
17. Bianco, F., Cucchietti, G., Griffa, G.: Energy consumption trends in the next generation access network – a telco perspective. In: Proceedings 29th International Telecommunication Energy Conference (INTELEC 2007), pp. 737–742 (2007)
18. Bianzino, A.P., Chaudet, C., Rossi, D., Rougier, J.-L.: A survey of green networking research. IEEE Commun. Surveys Tutorials **2** (2012)
19. Bianzino, A.P., Chiaraviglio, L., Mellia, M.: GRiDA: a green distributed algorithm for backbone networks. In: Online Conference on Green Communications (GreenCom 2011), pp. 113–119. IEEE (2011)
20. Bolla, R., Bruschi, R.: Energy-aware load balancing for parallel packet processing engines. In: Online Conference on Green Communications (GreenCom), pp. 105–112. IEEE (2011)
21. Bolla, R., Bruschi, R., Davoli, F., Cucchietti, F.: Energy efficiency in the future internet: a survey of existing approaches and trends in energy-aware fixed network infrastructures. IEEE Commun. Surveys Tutorials **13**, 223–244 (2011)
22. Bolla, R., Bruschi, R., Davoli, F., Lago, P., Bakay, A., Grosso, R., Kamola, M., Karpowicz, M., Koch, L., Levi, D., Parladori, P., Suino, D.: Large-scale validation and benchmarking of a network of power-conservative systems using etsi's green abstraction layer. Trans. Emerg. Tel. Tech. **2016**(27), 451–468 (2016)
23. Bolla, R., Bruschi, R., Ranieri. A.: Green support for pc-based software router: performance evaluation and modeling. In: ICC'09 Communications International Conference, pp. 1–6. IEEE (2009)
24. Bolla, R., et al.: Econet deliverable d2.1 end-user requirements, technology, specifications and benchmarking methodology. https://www.econet-project.eu/Repository/DownloadFile/291 (2011)
25. Bolla, R., et al.: Econet deliverable d4.1 definition of energy-aware states. https://www.econet-project.eu/Repository/Document/331 (2011)
26. Bolla, R., Bruschi, R., Davoli, F., Gregorio, L.D., Donadio, P., Fialho, L., Collier, M., Lombardo, A., Recupero, D.R., Szemethy, T.: Green abstraction layer (GAL): power management capabilities of the future energy telecommunication fixed network nodes. Technical Report ES 203 237, ETSI, 2014
27. Bolla, R., Bruschi, R., Lago, P.: Energy adaptation in multi-core software routers. Comput. Netw. **65**, 111128 (2014)
28. Bradner, S., McQuaid, J.: RFC 2544: benchmarking methodology for network interconnect devices (1999)
29. Chabarek, J., Sommers, J., Barford, P., Estan, C., Tsiang, D., Wright, S.: Power awareness in network design and routing. In: Proceedings 27th Conference on Computer Communications (INFOCOM 2008), pp. 457–465 (2008)
30. Chiaraviglio, L., Mellia, M., Neri, F.: Energy-aware backbone networks: a case study. In: Proceedings 1st International Workshop on Green Communications, IEEE International Conference on Communications (ICC'09), pp. 1–5. IEEE (2009)
31. Chiaraviglio, L., Mellia, M., Neri, F.: Minimizing ISP network energy cost: formulation and solutions. IEEE/ACM Trans. Netw. **20**, 463–476 (2011)

32. Choi, J., Govindan, S., Urgaonkar, B., Sivasubramaniam, A.: Profiling, prediction, and capping of power consumption in consolidated environments. In: MASCOTS 2008. IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems, 2008, pp. 110, Sept 2008
33. Cianfrani, A., Eramo, V., Listani, M., Marazza, M., Vittorini, E.: An energy saving routing algorithm for a Green OSPF protocol. In: Proceedings IEEE INFOCOM Conference on Computer Communications, pp. 1–5. IEEE (2010)
34. Cisco Systems, Inc.: Cisco Data Center Infrastructure 2.5 Design Guide (2011)
35. Cuomo, F., Abbagnale, A., Cianfrani, A., Polverini, M.: Keeping the connectivity and saving the energy in the Internet. In: Proceedings IEEE INFOCOM 2011 Workshop on Green Communications and Networking, pp. 319–324. IEEE (2011)
36. DeBonis, D., Grant, R.E., Olivier, S.L., Levenhagen, M., Kelly, S.M., Pedretti, K.T., Laros, J.H.: A power api for the hpc community. Sandia Report SAND2014-17061, Sandia National Laboratories (2014)
37. Diouri, M.E.M., Dolz, M.F., Glück, O., Lefèvre, L., Alonso, P., Catalán, S., Mayo, R., Quintana-Ortí, E.S.: Assessing power monitoring approaches for energy and power analysis of computers. Sustain. Comput.: Inf. Syst. **4**(2), 68–82 (2014)
38. Dolz, M.F., Heidari, M.R., Kuhn, M., Ludwig, T., Fabregat, G.: ARDUPOWER: a low-cost wattmeter to improve energy efficiency of HPC applications. In: Green Computing Conference and Sustainable Computing Conference (IGSC), 2015 Sixth International, pp. 1–8, Dec 2015
39. Dongarra, J. et al.: The international exascale software project roadmap. Int. J. High Perform. Comput. Appl. **25**, 3–60 (2011)
40. Dongarra, J.J., Luszczek, P., Petitet, A.: The LINPACK benchmark: past, present and future. Concurrency Comput.: Pract. Experience **15**(9):803–820 (2003)
41. SNIA Emerald: SNIA emerald power efficiency measurement specification. www.snia.org
42. Fisher, W., Suchara, M., Rexford, J.: Greening backbone networks: reducing energy consumption by shutting off cables in bundled links. In: Proceedings 1st ACM SIGCOMM Workshop on Green Networking (Green Networking'10), pp. 29–34. ACM (2010)
43. Floyd, M., Allen-Ware, M., Buyuktosunoglu, A., Rajamani, K., Brock, B., Lefurgy, C., Drake, A.J., Pesantez, L., Gloekler, T., Tierno, J.A., et al.: Introducing the adaptive energy management features of the Power7 chip. IEEE Micro **2**, 60–75 (2011)
44. Franklin, G.F., David Powell, J., Workman, M.L.: Digital control of dynamic systems, vol. 3. Addison-Wesley Menlo Park (1998)
45. Gandhi, A., Harchol-Balter, M., Das, R., Lefurgy, C.: Optimal power allocation in server farms. In: ACM SIGMETRICS Performance Evaluation Review, vol. 37, pp. 157–168. ACM (2009)
46. Gandhi, A., Harchol-Balter, M., Ram, R., Kozuch, M.A.: Autoscale: dynamic, robust capacity management for multi-tier data centers. ACM Trans. Comput. Syst. **30**(4), 14 (2012)
47. Gandhi, N., Tilbury, D.M., Diao, Y., Hellerstein, J., Parekh, S.: MIMO control of an apache web server: modeling and controller design. Proc. Am. Control Conf. **6**, 4922–4927 (2002)
48. Georgiou, Y., Cadeau, T., Glesser, D., Auble, D., Jette, M., Hautreux, M.: Energy accounting and control with SLURM resource and job management system. In: Distributed Computing and Networking, pp. 96–118. Springer (2014)
49. Gerndt, M., César, E., Benkner, S. (eds.): Automatic Tuning of HPC Applications. Shaker Verlag (2015)
50. Gu, C., Heng, H., Xiuping, J.: Power metering for virtual machine in cloud computing-challenges and opportunities. IEEE Access **2**, 1106–1116 (2014)
51. Hackenberg, D., Ilsche, T., Schone, R., Molka, D., Schmidt, M., Nagel, W.E.: Power measurement techniques on standard compute nodes: a quantitative comparison. In: IEEE International Symposium on Performance Analysis of Systems and Software, pp. 194–204. IEEE (2013)
52. Hackenberg, D., Ilsche, T., Schuchart, J., Schone, R., Nagel, W.E., Simon, M., Georgiou, Y.: HDEEM: high definition energy efficiency monitoring. In: Energy Efficient Supercomputing Workshop, pp. 1–10. IEEE (2014)

53. Hays, R.: Active/Idle toggling with low-power idle. Presentation at IEEE802.3az Task Force Group Meeting. http://www.ieee802.org/3/az/public/jan08/hays_01_0108.pdf (2008)
54. Hewlett-Packard Corp., Intel Corp., Microsoft Corp., Phoenix Technologies Ltd., and Toshiba Corp.: Advanced Configuration and Power Interface Specification, Revision 5.0 (2011)
55. Howard, J., Dighe, S., Vangal, S.R., Ruhl, G., Borkar, N., Jain, S., Erraguntla, V., Konow, M., Riepen, M., Gries, M., et al.: A 48-core IA-32 processor in 45 nm CMOS using on-die message-passing and DVFS for performance and power scaling. IEEE J. Solid-State Circuits **46**(1), 173–183 (2011)
56. Idzikowski, F., Orlowski, S., Raack, Ch., Rasner, H., Wolisz, A.: Saving energy in IP-over-WDM networks by switching off line cards in low-demand scenarios. In: Proceedings 14th Conference on Optical Network Design and Modeling (ONDM'10). IEEE (2010)
57. IEEE, Institute of Electrical and Electronics Engineers, IEEE 802.3az Energy Efficient Ethernet Task Force. http://grouper.ieee.org/groups/802/3/az/public/index.html (2012)
58. Ilsche, T., Hackenberg, D., Graul, S., Schöne, R., Schuchart, J.: Power measurements for compute nodes: improving sampling rates, granularity and accuracy. In: Sixth International Green and Sustainable Computing Conference (2015)
59. Intel. Intel Intelligent Power Node Manager. www.intel.com
60. Intel Corp.: Intel 64 and IA-32 Architectures Software Developers Manual Combined Volumes: 1, 2A, 2B, 2C, 3A, 3B and 3C (2015)
61. Iosup, A., Ostermann, S., Yigitbasi, M.N., Prodan, R., Fahringer, T., Epema, D.H.J.: Performance analysis of cloud computing services for many-tasks scientific computing. IEEE Trans. Parallel Distrib. Syst. **22**(6), 931–945 (2011)
62. Jaskóła, P., Arabas, P., Karbowski, A.: Combined calculation of optimal routing and bandwidth allocation in energy aware networks. In: Proceedings 26th International Teletraffic Congress, Karlskrona, pp. 1–6. IEEE (2014)
63. Jaskóła, P., Arabas, P., Karbowski, A.: Simultaneous routing and flow rate optimization in energy-aware computer networks. Int. J. Appl. Math. Comput. Sci. **26**(1), 231–243 (2016)
64. Jaskóła, P., Malinowski, K.: Two methods of optimal bandwidth allocation in TCP/IP networks with QoS differentiation. In: Proceedings Summer Simulation Multiconference (SPECTS'04), pp. 373–378 (2004)
65. Jha, S., Qiu, J., Luckow, A., Mantha, P., Fox, G.C.: A tale of two data-intensive paradigms: applications, abstractions, and architectures. In: IEEE International Congress on Big Data, pp. 645–652. IEEE (2014)
66. Jing, S.-Y., Ali, S., She, K., Zhong, Y.: State-of-the-art research study for green cloud computing. J. Supercomput. **65**(1), 445–468 (2013)
67. Jung, H., Pedram, M.: Supervised learning based power management for multicore processors. IEEE Trans. Comput.-Aid. Des. Integrat. Circuits Syst. **29**(9), 1395–1408 (2010)
68. Melanie, K., Martha, A.K.: An experimental survey of energy management across the stack. In: Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications, pp. 329–344. ACM (2014)
69. Kamola, M., Arabas, P.: Shortest path green routing and the importance of traffic matrix knowledge. In: 2013 24th Tyrrhenian International Workshop on Digital Communications - Green ICT (TIWDC), pp. 1–6, Sept 2013
70. Karbowski, A., Jaskóła, P.: Two approaches to dynamic power management in energy-aware computer networks - methodological considerations. In: Proceedings of Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 1177–1182, Sept 2015
71. Karpowicz, M., Arabas, P.: Energy-aware multi-level control system for a network of linux software routers: design and implementation. IEEE Syst. J. PP(99):1–12 (2015)
72. Karpowicz, M.P.: Energy-efficient CPU frequency control for the Linux system. Concurrency Comput.: Pract. Experience **28**(2):420–437 (2016). cpe.3476
73. Karpowicz, M.P., Arabas, P.: Preliminary results on the Linux libpcap model identification. In: 20th International Conference on Methods and Models in Automation and Robotics (MMAR), pp. 1056–1061. IEEE (2015)

74. Kołodziej, J., Khan, S.U., Wang, L., Zomaya, A.Y.: Energy efficient genetic-based schedulers in computational grids. Concurrency Comput.: Pract. Experience (2012). doi:10.1002/cpe.2839
75. Kondo, M., Sasaki, H., Nakamura, H.: Improving fairness, throughput and energy-efficiency on a chip multiprocessor through DVFS. ACM SIGARCH Comput. Architect. News **35**(1), 31–38 (2007)
76. Koomey, J.: Growth in data center electricity use 2005 to 2010. Analytics Press, Oakland, Aug, 1, 2010, 2011
77. Lange, K.-D.: Identifying shades of green: the SPECpower benchmarks. IEEE Comput. **42**(3), 95–97 (2009)
78. Laros, J.H., III, DeBonis, D., Grant, R., Kelly, S.M., Levenhagen, M., Olivier, S., Pedretti, K.: High performance computing-power application programming interface specification. Technical Report SAND2014-17061, Sandia National Laboratories (2014)
79. Lefurgy, C., Rajamani, K., Rawson, F., Felter, W., Kistler, M., Keller, T.W.: Energy management for commercial servers. Computer **36**(12), 39–48 (2003)
80. Lefurgy, C., Wang, X., Ware, M.: Server-level power control. In: The 4th IEEE International Conference on Autonomic Computing. IEEE (2007)
81. Lefurgy, C., Wang, X., Ware, M.: Power capping: a prelude to power shifting. Cluster Comput. **11**(2), 183–195 (2008)
82. Lefurgy, C.R., Drake, A.J., Floyd, M.S., Allen-Ware, M.S., Brock, B., Tierno, J.A., Carter, J.B., Berry, R.W.: Active guardband management in Power7+ to save energy and maintain reliability. IEEE Micro **33**(4), 35–45 (2013)
83. Lim, H., Kansal, A., Liu, J.: Power budgeting for virtualized data centers. In: 2011 USENIX Annual Technical Conference (USENIX ATC'11), p. 59 (2011)
84. Ljung, L.: System Identification. Prentice Hall, Upper Saddle River (1998)
85. Lorch, J.R., Smith, A.J.: Improving dynamic voltage scaling algorithms with pace. In: Proceedings ACM SIGMETRICS 2001 International Conference on Measurement and Modeling of Computer Systems, p. 5061 (2001)
86. Lu, Z., Hein, J., Humphrey, M., Stan, M., Lach, J., Skadron, K.: Control-theoretic dynamic frequency and voltage scaling for multimedia workloads. In: Proceedings of the 2002 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, pp. 156–163. ACM (2002)
87. Mair, J., Eyers, D., Huang, Z., Zhang, H.: Myths in power estimation with performance monitoring counters. Sustain. Comput.: Inf. Syst. **4**(2), 83–93 (2014)
88. Malinowski, K., Niewiadomska-Szynkiewicz, E., Jaskóła, P.: Price method and network congestion control. J. Telecommun. Inf. Technol. **2**, 73–77 (2010)
89. Mastelic, T., Oleksiak, A., Claussen, H., Brandic, I., Pierson, J.-M., Vasilakos, A.V.: Cloud computing: survey on energy efficiency. ACM Comput. Surv. **47**(2):33 (2015)
90. McCullough, J.C., Agarwal, Y., Chandrashekar, J., Kuppuswamy, S., Snoeren, A.C., Gupta, R.K.: Evaluating the effectiveness of model-based power characterization. In: USENIX Annual Technical Conference (2011)
91. Min, R., Furrer, T., Chandrakasan, A.: Dynamic voltage scaling techniques for distributed microsensor networks. In: Proceedings IEEE Workshop on VLSI, pp. 43–46 (2000)
92. Mobius, C., Dargie, W., Schill, A.: Power consumption estimation models for processors, virtual machines, and servers. IEEE Trans. Parallel Distrib. Syst. **25**(6), 1600–1614 (2014)
93. Molka, D., Hackenberg, D., Schöne, R., Minartz, T., Nagel, W.E.: Flexible workload generation for HPC cluster efficiency benchmarking. Comput. Sci.-Res. Dev. **27**(4):235–243 (2012)
94. Nakai, M., Akui, S., Seno, K., Meguro, T., Seki, T., Kondo, T., Hashiguchi, A., Kawahara, H., Kumano, K., Shimura, M.: Dynamic voltage and frequency management for a low-power embedded microprocessor. IEEE J. Solid-State Circuits **40**(1), 28–35 (2005)
95. Naveh, A., Rajwan, D., Ananthakrishnan, A., Weissmann, E.: Power management architecture of the 2nd generation Intel Core microarchitecture, formerly codenamed Sandy Bridge. In: Hot Chips, vol. 23, p. 0 (2011)

96. Niewiadomska-Szynkiewicz, E., Sikora, A., Arabas, P., Kamola, M., Mincer, M., Koodziej, J.: Dynamic power management in energy-aware computer networks and data intensive systems. Future Gener. Comput. Syst. **37**, 284–296 (2014)
97. Niewiadomska-Szynkiewicz, E., Sikora, A., Arabas, P., Kołodziej, J.: Control framework for high performance energy aware backbone network. In: Proceedings of European Conference on Modelling and Simulation (ECMS 2012), pp. 490–496 (2012)
98. Niewiadomska-Szynkiewicz, E., Sikora, A., Arabas, P., Kołodziej, J.: Control system for reducing energy consumption in backbone computer network. Concurrency Comput.: Pract. Experience **25**, 1738–1754 (2013)
99. NVIDIA. NVML API Reference Manual. www.developer.nvidia.com (2012)
100. Padala, P., Hou, K.-Y., Shin, K.G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., Merchant, A.: Automated control of multiple virtualized resources. In: Proceedings of the 4th ACM European Conference on Computer Systems, pp. 13–26. ACM (2009)
101. Pallipadi, V., Li, S., Belay, A.: cpuidle: do nothing, efficiently. Proc. Linux Symp. **2**, 119–125 (2007)
102. Pallipadi, V., Starikovskiy, A.: The ondemand governor. Proc. Linux Symp. **2**, 215–230 (2006)
103. Patikirikorala, T., Colman, A., Han, J., Wang, L.: A systematic survey on the design of self-adaptive software systems using control engineering approaches. In: ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 33–42. IEEE (2012)
104. Pióro, M., Mysłek, M., Juttner, A., Harmatos, J., Szentesi, A.: Topological design of MPLS networks. In: Proceedings GLOBECOM'2001 (2001)
105. Qureshi, A., Weber, R., Balakrishnan, H.: Cutting the electric bill for internet-scale systems. In: SIGCOMM'09, pp. 123–134. ACM, Aug 17–21 2009
106. Restrepo, J., Gruber, C., Machuca, C.: Energy profile aware routing. In: Proceedings 1st International Workshop on Green Communications, IEEE International Conference on Communications (ICC'09), pp. 1–5 (2009)
107. Roy, S.N.: Energy logic: a road map to reducing energy consumption in telecom munications networks. In: Proceedings 30th International Telecommunication Energy Conference (INTELEC 2008) (2008)
108. Sikora, A., Niewiadomska-Szynkiewicz, E.: A federated approach to parallel and distributed simulation of complex systems. Appl. Math. Comput. Sci. **17**(1), 99–106 (2007)
109. Storage Performance Council (SPC): Storage Performance Council SPC Benchmark 2/Energy Extension. www.storageperformance.org
110. Standard Performance Evaluation Corporation (SPEC): SPEC Power and Performance Benchmark Methodology. www.spec.org/power_ssj2008/
111. Spiliopoulos, V., Kaxiras, S., Keramidas, G.: Green governors: a framework for continuously adaptive DVFS. In: 2011 International Green Computing Conference and Workshops (IGCC), pp. 1–8. IEEE (2011)
112. Subramaniam, B., Feng, W.: Towards energy-proportional computing for enterprise-class server workloads. In: Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering, pp. 15–26. ACM (2013)
113. Subramaniam, B., Saunders, W., Scogland, T., Feng, W.: Trends in energy-efficient computing: a perspective from the Green500. In: 2013 International Green Computing Conference (IGCC), pp. 1–8. IEEE (2013)
114. Taniça, L., Ilic, A., Tomás, P., Sousa, L.: Schedmon: a performance and energy monitoring tool for modern multi-cores. In: Euro-Par 2014: Parallel Processing Workshops, pp. 230–241. Springer (2014)
115. Valentini, G.L., Lassonde, W., Khan, S.U., Min-Allah, N., Madani, S.A., Li, J., Zhang, L., Wang, L., Ghani, N., Kołodziej, J., Li, H., Zomaya, A.Y., Xu, C.-Z., Balaji, P., Vishnu, A., Pinel, F., Pecero, J.E., Kliazovich, D., Bouvry, P.: An overview of energy efficiency techniques in cluster computing systems. Cluster Comput. (2011). doi:10.1007/s10586-011-0171-x
116. Vasić, N., Kostić, D.: Energy-aware traffic engineering. In: Proceedings 1st International Conference on Energy-Efficient Computing and Networking (E-ENERGY 2010) (2010)

117. Vavilapalli, V.K., Murthy, A.C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., et al.: Apache hadoop yarn: yet another resource negotiator. In Proceedings of the 4th Annual Symposium on Cloud Computing, p. 5. ACM (2013)
118. Wang, L., Khan, S.U.: Review of performance metrics for green data centers: a taxonomy study. J. Supercomput. (2011). doi:10.1007/s11227-011-0704-3:1-18
119. Wang, L., Khan, S.U.: Review of performance metrics for green data centers: a taxonomy study. J. Supercomput. **63**(3), 639–656 (2013)
120. Wang, X., Wang, Y.: Coordinating power control and performance management for virtualized server clusters. IEEE Trans. Parallel Distrib. Syst. **22**(2), 245–259 (2011)
121. Wang, Y., Wang, X., Chen, M., Zhu, X.: Partic: power-aware response time control for virtualized web servers. IEEE Trans. Parallel Distrib. Syst. **22**(2), 323–336 (2011)
122. Weaver, V.M., Johnson, M., Kasichayanula, K., Ralph, J., Luszczek, P., Terpstra, D., Moore, S.: Measuring energy and power with PAPI. In: 41st International Conference on Parallel Processing Workshops (ICPPW), 2012, pp. 262–268. IEEE (2012)
123. Wu, B., Li, P.: Load-aware stochastic feedback control for DVFS with tight performance guarantee. In: 2012 IEEE/IFIP 20th International Conference on VLSI and System-on-Chip (VLSI-SoC), pp. 231–236, Oct 2012

Chapter 7
Context-Aware and Reinforcement Learning-Based Load Balancing System for Green Clouds

Ionut Anghel, Tudor Cioara and Ioan Salomie

7.1 Introduction

With the recent technological advancements in the areas of mobile devices and wireless sensor networks, the research efforts are moving towards developing complex distributed systems able to administrate heterogeneous pervasive environments composed of sensors, actuators, smart devices and software agents which interact and collaborate. To offer support for the human agents during their interactions and to allow them to concentrate on their tasks, it is essential for the distributed systems to become aware of their execution context and to adapt their behaviour to different context situations.

In these circumstances, a new type of complex distributed systems emerges: the context-aware adaptive systems. They define a special class of distributed systems which are able to sense and understand the changes from their execution environment and adapt or optimize their operation accordingly. The development of such systems poses significant research problems regarding relevant context identification, acquisition, representation, awareness, and adaptation for operation optimization.

1. *Context Definition* Unlike traditional systems which do not consider and use the context to drive their operation, for context-aware systems the context data becomes a valuable asset for understating the situation in which they evolve and for adapting their behaviour accordingly. There are many ways in which the context may be defined, but the most comprehensive one is given in [1]. Dey considers

I. Anghel (✉) · T. Cioara · I. Salomie
Technical University of Cluj-Napoca, Memorandumului 28, 400114
Cluj-Napoca, Romania
e-mail: ionut.anghel@cs.utcluj.ro

T. Cioara
e-mail: tudor.cioara@cs.utcluj.ro

I. Salomie
e-mail: ioan.salomie@cs.utcluj.ro

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_7

the context as any information relevant for characterizing the situation of an entity or its interactions. Starting from this definition we consider the systems' execution context as consisting of all context data relevant for assessing the systems current operation situation and its interactions. The context data may refer both to the systems' execution environment and to the systems' internal state and available computational resources.

2. *Context Monitoring and Abstraction* Context data is usually collected from various heterogeneous sources thus being difficult to be automatically processed and interpreted at run-time. Some of the sensors needed for context-aware computing differ in their programming interfaces and even more, sensors for collecting similar information may provide different raw data. This is why, in our vision, a key component in developing context-aware adaptive systems is the context model whose main goal is to programmatically represent, fusion and abstract the acquired raw data so that it can be processed, evaluated, understood, and shared by all the modules of a context-aware system.
3. *Context Dynamicity, Awareness, and Adaptation* The context-aware system needs to respond to context changes by adapting their observable behaviour to optimize their operation. In our approach, a system becomes aware of its execution context if it is able to understand the changes in its execution context and to identify those context situations in which adaptation is needed. To achieve these goals, the development of context management techniques that automatically process and interpret the context without human factor intervention becomes a key element. After identifying a situation requiring adaptation, the context-aware adaptive system must automatically decide and select the appropriate adaptation actions to be executed such that system's observable behaviour is changed according to the new context.

Lately, the energy consumption of Data Centres (DCs) has dramatically emerged as one of the most critical environmental challenges to be dealt with. The EU Code of Conduct for DC Energy Efficiency estimates that DCs consume around 56 TWh per year in EU as a whole and forecasts that this will increase to 104 TWh per year by 2020 [2]. The amount of energy consumed by world's DCs in one year is similar to the total amount consumed by a single country in the same time period and greatly affects both the environment and the economy. The global recession of the last years has not stopped the DCs expansion and in 2014 they consumed up to 3 % of all global electricity production while producing 200 million tons of CO2 and generating about 60 billion dollars costs [3].

Currently, many researches, from both management and technical aspects, are striving to reduce the energy consumption of DCs. Besides the cooling infrastructure, the low utilization of computing resources (i.e., servers) is one of the main factors contributing to their high energy consumption [4]. The adoption of virtualization for DC resources provision allows the hosting of multiple virtualized applications on the same physical server and the allocation of desired computational capacity on-demand. The cloud-based DCs are configured to ensure that the peak workload

is satisfied, leading to a mean usage ration of DC servers between 10 and 50 % and in consequence to a waste of energy of about 50 % [5].

In this chapter, we address the cloud-based DCs energy efficiency problem by proposing a context-aware adaptive load balancing system capable of dynamically taking management or adaptation decisions to decrease the DCs energy consumption.

Each of the problems identified for developing context-aware systems are addressed considering the specific case of a load balancing system for energy efficient cloud computing. The *context definition and abstraction* is approached by considering the load levels and energy consumption values as relevant contextual data and by deploying relevant physical and virtual sensors for acquiring this data at cloud level. This allows our context-aware adaptive load balancing system to understand the current cloud situation and potential changes and automatically decide on the actions than need to be executed to schedule and consolidate the virtualized work-load in an energy efficient manner. To *enact the situation and contextual awareness* techniques are developed to allow our load balancing system to make assumptions upon and understand the efficiency of current cloud situation (i.e., snapshot). We aim at identifying energy inefficient, non-green, cloud situations by defining a set of green policies for clouds DCs and evaluating if they hold for the current cloud situation. Another important challenge here is to *define the context for the domain of green clouds* which is addressed by considering, among others, the energy consumption and workload levels as relevant context features. To *automatically decide on context adaptation actions as to scale up or down the cloud computing resources allocation*, a reinforcement learning-based approach is used. The reinforcement learning process considers all broken green policies, one by one, according to their weight and determines the workload balancing actions to enforce them.

The rest of the chapter is organized as follows: Sect. 7.2 presents the relevant related work in the area of energy efficient resources balancing in the context of cloud-based DCs, Sect. 7.3 presents the context-aware workload balancing system, Sect. 7.4 shows details on the implementation of such context-aware load balancing system as an extension of the OpenNebula clouds management middleware [6] (Green Cloud Scheduler OpenNebula component [7]) while Sect. 7.5 concludes the chapter.

7.2 Related Work

Hardware resources *virtualization is the fundamental technology that powers up cloud computing*. It aims at logically dividing the DCs hardware resources into segments that can be managed independently [8].

There are several advantages coming with the use of Virtual Machines (VMs) [9]. The *first advantage of virtualization* is the fact that the server's hardware resources will be reproduced through software such that multiple operating systems can share a single physical server. In non-virtualized DCs, each psychical server may host only one task (e.g., a web server application or a file server application). By using resources virtualization, both the previously mentioned applications may run independently

on the same psychical server, therefore reducing the costs and energy. The *second advantage of virtualization* is the applications isolation [10]. The context in which a VM is running is completely transparent to the applications, so an error in one VM will not affect the physical server running it, or any other machines deployed on that server. The *third advantage of virtualization* is the fact that the applications contained in the VMs are no longer tied to a physical system with certain hardware of software configuration [10]. Instead, they can be migrated from one host to another. In a DC, VMs are moved between servers to achieve higher computational density and higher workload consolidation during periods with low requests.

The use of *virtualization technique itself increases the energy efficiency* of cloud-based DCs compared to nonvirtualized ones (i.e., collocation) as it is advocated by Intel [11] in a white paper about the strategies to reduce energy consumption implemented at the CERN physics laboratory. Similarly in [12] the authors argue that using virtualization the ratio between the computing efficiency and power consumption of DCs is increased. Authors of [13] state that by using virtualization the DCs density increases and the storage will be more optimally used resulting in a decrease of maintenance and electricity costs. Besides the decreasing of energy consumption of the IT components, additional energy savings can be further obtained by optimizing cooling infrastructure operation.

However, virtualization alone is not enough because even if the scheduling decision for a VM to be hosted into a specific DC server is correct when the VM is created and deployed, during the lifecycle of the cloud, the distribution of the VMs may get worse and the usage of resources may become inefficient. *This is why load balancing systems need to be developed to better consolidate the deployment of VMs on DCs and reduce the fragmentation of computing resources and the ratio of unused servers*. The load balancing and consolidation systems take advantage of virtualization, by proposing models to migrate the VMs in a DC from one server/cluster to another so that they better fit on the DCs computing resources. By VMs migration the workload distribution can be balanced and consolidated on smaller number of physical machines allowing for servers, or even entire operation nodes, to be completely shut down [14]. This increases the overall energy efficiency of the DC but also brings with it the downside of additional overhead in terms of migration costs, latency and Quality of Service (QoS)/Service Level Agreement (SLA) penalties. For example, VMwares published benchmark on ESX latency found an average I/O latency of 13 % [15].

For developing of such systems, two challenges need to be addressed: (1) *the effective monitoring of current distribution, usage and energy consumption of cloud VMs/servers* and (2) *the energy-aware server consolidation decision process*.

The lack of monitoring software or power meters for VMs is one of the main challenges for using virtualization. Unlike physical servers that provide in-hardware or outlet level power measurement functions, it is very difficult to estimate the power required by a VM when hosted on a certain hardware configuration [16]. The performance and health monitoring counters exposed by the VM hypervisor, correlated with relevant power information about the hosting hardware and simple adaptive power models can offer relatively accurate information [17]. In [16], Joulemeter is designed

and proposed as the solution for providing power metering functionality for VMs while in [17] the authors acknowledge the challenges of power management in virtualized servers and describe an alternative framework oriented towards hypervisor systems. The latter framework relies on special drivers at the host OS level, at hypervisor level and at guest OS level, coordinating their activities in order to maximize energy savings. In [18] an alternative for power budgeting of a virtualized server is defined by means of control theory and used to drive system parameters under power constraints. A proportional-integral-derivative controller is used to enforce system-level CPU usage constraints based upon monitored power consumption and a specified power limit. The output of this controller, along with feedback from VMs based on a congestion pricing scheme, is used by a resource manager to disperse CPU allocations amongst guest VMs.

The load balancing and consolidation algorithms can be classified in three different types taking into account the consolidation time [19]: static, semi-static and dynamic consolidation. In the static consolidation the workload VMs are dispatched in a consolidated manner without any further migration [20]. This approach is recommended to be used in case of static workload that doesn't change in time. In semi-static consolidation, the server workload is balanced and consolidated once every day or every week [21]. The dynamic consolidation deals with workloads that frequently change in time. It requires a run-time manager that should deploy/migrate workload applications and/or wake-up/turn-off servers as a response to workload variation [22]. In the rest of this section, we will detail the most relevant state of the art approaches for dynamic load balancing and consolidation which is the subject of our context-aware-based approach.

To enable energy efficient dynamic workload balancing , the inter-relationships between energy consumption, resource utilization, and performance of consolidated workloads must be considered [23]. In [14], the authors reveal that energy performance trade-offs for consolidation and optimal operating points exist. As shown in [19], the greatest challenge for cloud balancing methods is deciding which workloads should be combined on a common physical server since resource usage, performance, and energy consumption are not additive. In case of resources consolidation [24], many small physical servers are replaced by one larger physical server, to increase the utilization of costly hardware resources such as CPU. Although hardware is consolidated, typically operating systems are not. Instead, each OS running on a physical server is replaced by an equivalent OS running inside a VM. In [25] a framework called Energy-Aware Grid is proposed and used to increase the energy efficiency of federated DCs connected in a grid. The central component of the framework is Globus Resource Allocation Manager an energy-aware co-allocator, which takes workload consolidation, decisions across the grid based on a set of energy efficiency coefficients. In [26] the problem of power and performance management in virtualized DCs is approached by dynamically provisioning VMs, consolidating the workload, and turning on and off servers. To select the appropriate consolidation decisions a predictive control approach based on the limited look ahead control technique is proposed. Authors of [27] model the problem of energy-aware workload balancing in DCs as a multi-dimensional bin-packing problem and to solve it they designed

a swarm-inspired algorithm based on the Ant Colony Optimization meta-heuristic. In [28] an energy-aware scheduling and consolidation algorithm for VMs in DCS is proposed. The algorithm is based on a fitness metric to evaluate the consolidation decision effectiveness.

7.3 Context-Aware Load Balancing System

Figure 7.1 presents the conceptual architecture of our context-aware adaptive load balancing system for cloud-based DCs. To be able to take informed decisions our context-aware load balancing system needs to define, identify and acquire data from the cloud-based DCs relevant for taking adaptation/optimization decisions. The information is fused (if needed) and abstracted in the form of cloud snapshots representing the current situation of the cloud-based DC.

The cloud snapshots are analysed and different green policies or indicators are evaluated to detect those inefficient situations in which the workload needs to be rearranged. If such situations are encountered an action plan is constructed to change the workload distribution, to consolidate existing workload and to avoid servers fragmentation. The decision process uses a reward penalty approach in which the reward is given by potential energy saving while the penalty is given by the workload migration overhead cost.

We define the load balancing problem for cloud-based DCs as a function which associates to each cloud snapshot that fails to fulfil a predefined *set of policies or indicators*, an adaptation action plan that should be executed in order to re-balance the workload, change its distribution and re-enforce the defined indicators:

$$Balance_{Workload}(Cloud_{snapshot}, Indicators) \rightarrow VMs\ Redistribution\ Plan. \tag{7.1}$$

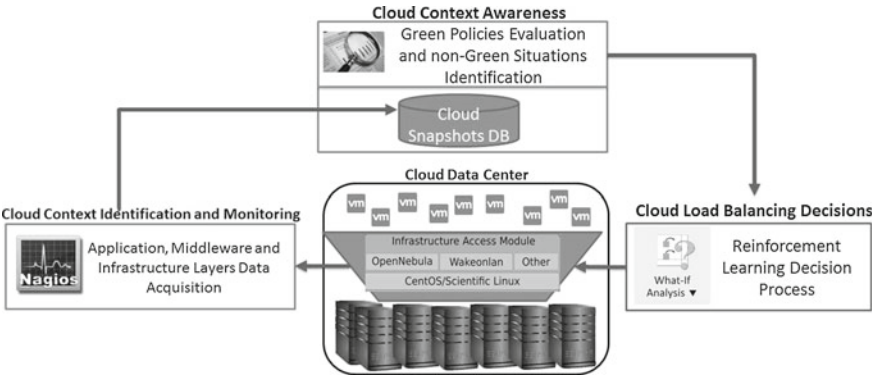


Fig. 7.1 Load balancing system conceptual architecture

7.3.1 Cloud Context Identification and Monitoring

This module is responsible for collecting cloud context data and constructing the Cloud Snapshots. The cloud context data describes the cloud DC workload, its distribution on the DC servers, the computing resources usage and the associated energy consumption values. This information is fundamental input for our load balancing system to understand the context and efficiency of the current investigated Cloud Snapshot and to take workload balancing and consolidation decisions in an energy-aware manner.

- At the Application Layer reside business processes oriented applications that are executed by the cloud-based DCs. This layer provides information regard-ing QoS and customers agreed SLA levels;
- At the Middleware Layer reside the VMs through which the business processes activities are executed. This layer provides, data regarding the IT computing resources allocated to/used by the VMs, VMs distribution and their state (active, idle, etc.);
- At the Infrastructure Layer reside the DCs hardware resources. This layer provides data regarding servers load levels and energy consumption values as well as temperature and energy consumption of the cooling system.

The above-presented situation poses an unavoidable problem: how to effectively monitor and acquire context data from all cloud-based DC layers? The techniques for collecting context data are completely distinct for each layer. For instance, the elements located at the infrastructure layer are mostly physical devices, thus the relevant data regarding the energy consumption in this layer can be directly measured by deploying sensors (i.e., power meters for measuring instant power consumption) and using their APIs. On the contrary, the parameters residing at the application layer are difficult to be monitored and they are usually collected from configuration files. Table 7.1 shows the context data we found to be relevant for our load balancing system and the associated data sources.

As a consequence, the selection of a proper monitoring tool capable of gathering all this heterogeneous data in a uniform and integrated manner was a challenging task. The selected monitoring tool in our approach is Nagios [29], a leading open-source monitoring tool for systems, networks, and applications. The major advantage of Nagios consists of its plug-in tool box that enables Nagios to be configurable, implementable and customizable to meet different real and virtual sensors characteristics and APIs. Nagios relies on external programs (plug-ins) to perform actual checks and provides a plug-in API allowing new plug-ins to be created in different programming languages for specific data acquisitions. This aspect enables Nagios to monitor the whole IT infrastructure including system metrics, network services (e.g., SMTP, POP3, HTTP, etc.), applications, host resources (e.g., HDD space, RAM and CPU utilization, server exhaust temperatures, etc.) and the network infrastructure. Nagios-based monitoring infrastructure defined and used to collect the cloud-based

Table 7.1 Context data relevant for load balancing at different cloud-based DC layers

DC layer	Context data	Data source
Infrastructure layer	Total facility/IT Equipment instant power (W)	Power meter device
	Ambient temperature (°C)	Sensor
	CPU temperature (°C)	
	Server CPU fan speed (rpm), usage (%), frequency (Mhz)	Cpufreq Linux kernel module
	MEM usage (%)	Linux command line utility
	Internal HDD usage (%) and speed (rpm)	Hdparm Linux utility
	Shared Storage usage (%) and speed (rpm)	
	Server Status (ON/OFF)	OpenSSH
Middleware layer	VM State (Active, Idle)	VM Hypervisor
	VM allocated CPU (%), MEM (% and Mb) and HDD (% and Mb)	
	VM usage of CPU (%), MEM (% and Mb), HDD (% and Mb)	
Application layer	Task requests for CPU (% and Mhz), MEM (% and Mb) and HDD (% and Mb)	Configuration files
	Maximum response time (s)	SLA files

DC con-text data is presented in Fig. 7.2. NDOUtils (Nagios Data Out) add-on allows the users to store all data collected by Nagios in a Cloud Snapshot database.

7.3.2 Cloud Context Awareness

The goal of this module is to detect those Cloud Snapshots that are inefficient in terms of workload distribution and power/energy usage. To achieve this, state of the art indicators defined for each cloud-based DC layer are evaluated for the current Cloud Snapshot. If some of the indicators fail below the predefined thresholds actions need to be taken to rebalance the workload as to enforce those indicators. Table 7.2 summarizes the indicators we have used at each cloud-based DC layers to detect the inefficient Cloud Snapshots.

The *Power Usage Effectiveness (PUE)* indicator was proposed by Green Grid [30] and measures how much power is used by the IT Equipment in contrast to IT Facility. The objective is to balance the cloud DC workload distribution to obtain more efficient cooling and bring the indicator value as close as possible to 1. The computing formula is presented in Eq. 7.2:

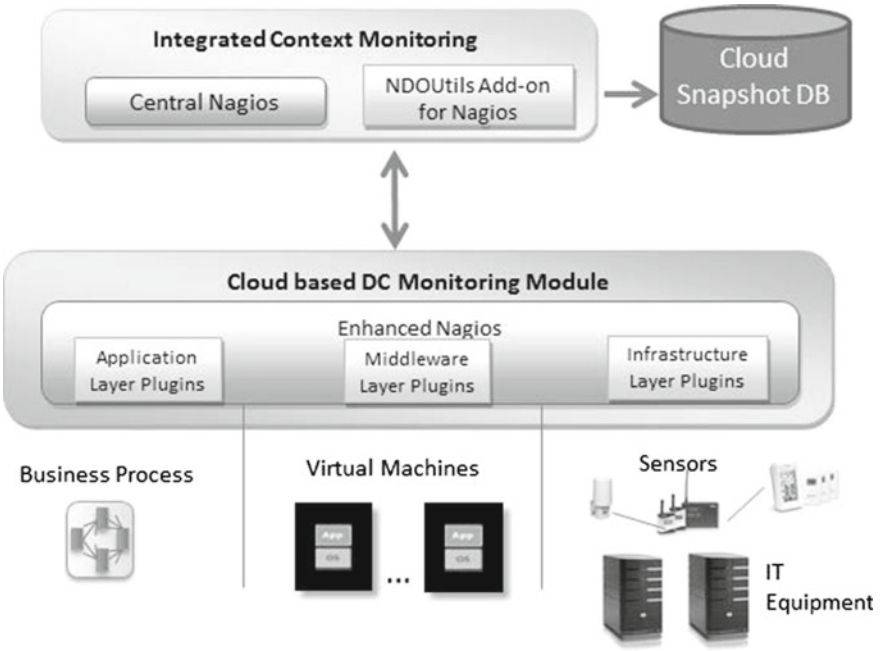


Fig. 7.2 Cloud context monitoring infrastructure

Table 7.2 Cloud-based DC indicators/policies

DC layer	Indicator objectives	Used indicators
Infrastructure layer	Impose restrictions about the DC power usage and server utilization	Power Usage Effectiveness (PUE), Server CPU & Memory Usage
Middleware layer	Specify the optimal workload distribution values	VMs CPU&MEM allocation, Deployed Hardware Utilization Ratio (DH-UR), Deployed Hardware Utilization Efficiency (DH-UE)
Application layer	Describe the QoS and SLA requirements agreed with customers for applications execution	Response time

$$PUE = \frac{Total\ Facility\ Power}{IT\ Equipment\ Power}. \tag{7.2}$$

However, a small PUE value is not enough, because this metric does not consider the actual utilization of hardware computational resources. Thus we have utilized it in conjunction with resources usage indicators described below.

The *Server CPU & Memory Usage* indicators describe the optimal load levels for a cloud DC server. This value depends on server hardware specification but usually it is desired to have load levels between 50 and 80 %. Under 50 % the server will be underutilized thus will waste energy, while over 80 % the client may experience significant performance penalties.

$$U_{server} = \frac{Load\ Of\ The\ Server\ Processor}{Maximum\ Load\ At\ The\ Highest\ Frequency\ State}. \tag{7.3}$$

The *Deployed Hardware Utilization Ratio (DH-UR)* indicator was proposed by the Uptime Institute [31] and measures the power drained by the idle servers not executing any workload or in other words, the amount of power waste. The calculation method is presented in Eq. 7.4:

$$DH - UR = \frac{No\ Servers\ Running\ Live\ Apps}{Total\ No\ Servers\ Deployed}. \tag{7.4}$$

In a similar manner, DH-UR can be used for assessing storage systems efficiency: number of terabytes containing frequently accessed data divided by the total number of terabytes of the storage system.

The *Deployed Hardware Utilization Efficiency (DH-UE)* indicator was also proposed Uptime Institute [31] and measures the power efficiency of the operating servers. The calculation method is provided in Eq. 7.5

$$DH - UE = \frac{Min\ No\ Servers\ Necessary\ For\ Handling\ Peak\ Load}{Total\ No\ Servers\ Deployed}. \tag{7.5}$$

The DH-UR and DH-UE indicators need to be as close as possible to 1. If this is not fulfilled it means that are servers that are not optimally utilized, the workload is fragmented and workload needs to be re-balanced by means of consolidations and migrations.

VMs CPU & MEM allocation and applications response time are constraints that are mandatory to be fulfilled in all Cloud Snapshots since they are usually enforced by bilateral contracts. They must be taken into account during workload balancing decisions because their violations will result in serious penalties and costs for the DC.

7.3.3 Cloud Load Balancing Decisions

The decision process is implemented by means of a reinforcement leaning technique which starts from the current Cloud Snapshot that was assessed as inefficient and builds a decision (learning) tree by simulating the execution of all potential load balancing actions with the goal of reducing the load fragmentation on DC servers. We have considered the following actions: deploy/migrate VM and turn on/off server.

The reinforcement learning is a non-supervised type of learning. In Artificial Intelligence the reinforcement learning model consists of the following elements [32]: (i) a set of environment states S, (ii) a set of actions A, (iii) rules of transitioning between states, (iv) a transition reward and (v) a reinforcement learning agent. The reinforcement learning algorithm has the following main steps:

1. The reinforcement learning agent receives an environment observation which corresponds to the current environment state ($s_t \in$ S);
2. The agent executes an action ($a_t \in$ A) from the set of available actions;
3. As a consequence a new environment state and the associated transition Tr=(s_t, a_t, s_{t+1}) is generated;
4. The reward is computed for the determined transition Tr;
5. The reinforcement learning agent moves forward and considers the new generated environment state (s_{t+1}) as current state.

We have mapped the above-presented reinforcement learning algorithm to our context-aware load balancing decision process as follows:

- The environment states set S corresponds to the potential Cloud Snapshots;
- The set of actions A includes the deploy/migrate VM and turn on/off server actions;
- The transition rules between states will define how the Cloud Snapshot will be changed as result of executing the defined actions.

The decision process may generate different Cloud Snapshots with different reward values by means of whatif analysis of potential sequence of balancing actions. As a consequence, the context-aware load balancing reinforcement learning algorithm has the following steps:

1. A reinforcement learning agent receives a Cloud Snapshot which breaks the conditions imposed by a defined indicator, resulting a faulty context situation (s_t) in which workload balancing is needed;
2. The agent simulates the execution of all available adaptation actions (a_t) on the current Cloud Snapshot using the defined transition rules;
3. As a consequence, a new Cloud Snapshot is virtually generated and the associated transitions Tr=(s_t, a_t, s_{t+1}) are generated. The transitions are stored in a learning tree as follows: s_t- parent node, s_{t+1}- children node, a_t- annotates the link between the parent and the children node;
4. For the transition Tr the reward is computed based on the values of a predefined indicator before ($Indicator\ Value_{s_t}$) and after ($Indicator\ Value_{s_{t+1}}$) the action execution (see Eq. 7.4 where *ActionCost* represents the cost of each possible

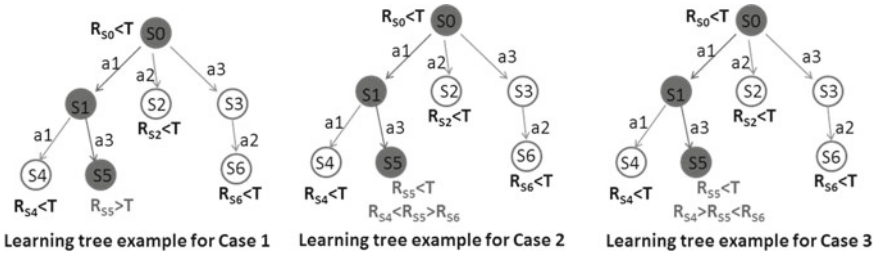


Fig. 7.3 Potential cases in the load balancing decision tree and their ending conditions

actions defined in design-time and γ-is a parameter whose value is used to cut-off the reinforcement learning search process);

$$Reward_{s_{t+1}} = Reward_{s_t} + \gamma * (Indicator\ Value_{s_{t+1}} - Indicator\ Value_{s_t} - Action\ Cost). \tag{7.6}$$

5. The reinforcement learning agent considers s_{t+1} as the current Cloud Snapshot and re-executes all algorithm steps until one of the ending conditions are met.

Figure 7.3 presents the ending conditions for the context-aware load balancing decision process. During decision process execution we have identified the following three special cases that need to be handled:

- *Case 1*: A generated Cloud Snapshot has its reward higher than a defined threshold ($R_{S5} > T$) and the faulty indicator is enforced. In this case the sequence of load balancing actions leading to this potential Cloud Snapshot (*S0, S1, S5*) is selected and the search is stopped;
- *Case 2*: A generated Cloud Snapshot has its reward smaller than the defined threshold, but higher than the maximum reward determined so far ($R_{S5} < T$ and $R_{S4} < R_{S5} > R_{S6}$). We replace the maximum reward with the new reward and continue the decision process;
- *Case 3*: A generated cloud snapshot has its reward value lower than both the threshold and the maximum reward encountered so far ($R_{S5} <$ T and $R_{S4} > R_{S5} < R_{S6}$); the reinforcement learning algorithm continues the search process. If all exercised paths of the decision tree are cycles, the algorithm stops and chooses the path leading to a state with the maximum reward.

7.4 Green Cloud Scheduler: A Load Balancer for OpenNebula

We have used the context-aware load balancing system presented in Sect. 7.3 to implement the Green Cloud Scheduler (GCS) as a proof of concept implementation which augments OpenNebula [6] Cloud Management Middleware with energy-

aware task management features. OpenNebula is a middleware for managing virtualized clouds. OpenNebula offers functionalities for VM creation and deployment on a specific server and VM migration from a server to another, etc. GCS is available on the OpenNebula Ecosystem official page [7], from which potential DCs managers can download, install and use it.

GCS substitutes the default OpenNebula scheduler and:

- Maintains low fragmentation of the available cloud computing resources by balancing the cloud VMs in an energy efficient manner;
- Optimizes the number of servers needed to host the cloud VMs;
- Turns on/off servers considering the workload characteristics.

The decrease in available resources fragmentation reduces the number of servers needed to host the VMs, leaving room for turning off the unused servers. For obtaining low fragmentation, VMs are migrated between servers to better balance the avail-able workload and exploit the available computing resources.

For *cloud context monitoring* GCS takes advantage of the *OpenNebula Java OCA API* [33] to access the OpenNebula servers pool and VMs pool for retrieving characteristics such as: (i) memory, CPU number of cores, CPU current frequency, etc., (ii) information about the clouds VMs (pending VMs, their current state, etc.), and (iii) clouds available computational resources. For finding the maximum CPU frequency for a server the *cpufreq-utils tool* is used. To retrieve the MAC addresses for each server defined in OpenNebula servers pool, the GCS reads the data from *./config/arpTable* and if the MAC for a server is not defined, GCS pings the server, issues an *arp* and parses the output. The retrieved MAC address is then stored in the *./config/arpTable* for later use.

For *load balancing actions enforcement* GCS uses the following stack of software resources: CentOS 5.5 Linux operating system [34], KVM (Kernel-based Virtual Machine) [35], OpenNebula Middleware [6], Wake-on-LAN [36] and OpenSSH [37]. CentOS is the Linux distribution used by the GCS as servers operating system. GCS interacts directly with CentOS for enforcing server Turn-off adaptation action. KVM is a hypervisor which deals with the virtualization of the server hardware resources so that the workload applications may be executed and is installed on top of CentOS. The hypervisor also offers functionalities for monitoring and managing a VM. GCS uses the Wake-on-LAN tool to remotely wake up the DC servers, while OpenSSH is used to turn off the DC servers remotely. Table 7.3 shows (i) the actions considered by the GCS and (ii) examples on how these actions are enforced at the cloud level.

Table 7.3 GCS actions enforcement

Action	Enforcement	
	Tool	Example
Deploy VM	OpenNebula	VM vm = new VM(VM_ID, openNebulaClient) vm.deploy(DESTINATION_SERVER_ID)
Migrate VM		vm.migrate(DESTINATION_SERVER_ID)
Wakeup server	Wake-on-LAN	>wakeonlan SERVER_MAC
Turnoff server	CentOS/OpenSSH	>ssh IP echo disk >/sys/power/state

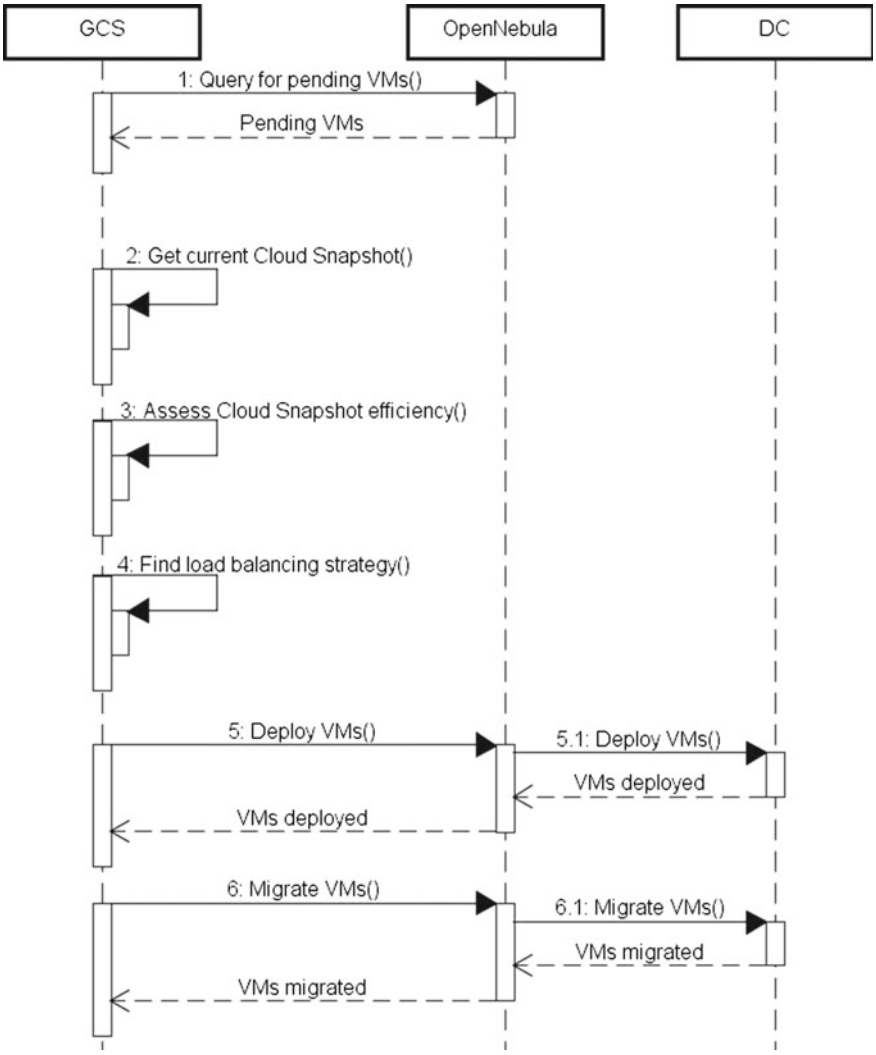


Fig. 7.4 GCS load balancing sequence diagram

GCS is deployed behind the OpenNebula Middleware and bypasses the default OpenNebula Scheduler and solely manages and balances the cloud-based DC workload (i.e., VMs). In this case, the DC clients are unaware of GCS (see Fig. 7.4).

The VMs are created using the OpenNebula API and placed as pending in the OpenNebula VM Pool. GCS periodically queries the OpenNebula VM pool for pending VMs, finds the most appropriate server on which the pending VMs should be deployed and executes the deployment actions using the OpenNebula API (see Fig. 7.4 sequence diagram).

At the same time, when inefficient Cloud Snapshots are detected, VMs are re-balanced on servers by means of migration actions aiming to reduce the workload and resources fragmentation and to consolidate the available servers.

7.5 Conclusions

In this chapter, a context-aware adaptive load balancing system for cloud-based DCs is presented. The system leverages on concepts and techniques specific to context-aware computing and state of the art metrics and indicators for detecting inefficient Cloud Snapshots and the workload needed to be re-balanced. The work-load balancing decision process is based on reinforcement learning which conducts a what-if analysis based on defined rewards and penalties. The Green Cloud Scheduler is presented as a proof of concept implementation of the load balancing system. The scheduler has been accepted as an official OpenNebula ecosystem component for energy efficient balance of workload in cloud-based DCs.

References

1. Abowd, G.D., Dey, A.K., Brown, P.J., et al.: Towards a Better Understanding of Context and Context-Awareness. In: Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing, pp. 304–307 (2000). http://dl.acm.org/citation.cfm?id=743843
2. Data Centres Energy Efficiency Code of Conduct. http://iet.jrc.ec.europa.eu/energyefficiency/ict-codes-conduct/data-centres-energy-efficiency
3. Industry Outlook: Data Center Energy Efficiency (2014). http://www.datacenterjournal.com/it/industry-outlook-data-center-energy-efficiency/
4. Data Center Efficiency Assessment. https://www.nrdc.org/energy/files/data-center-efficiency-assessment-IP.pdf
5. Ardito, L.: Green IT—Available data and guidelines for reducing energy consumption in IT systems. Sustainable Comput. **4**(1), 24–32 (2013)
6. OpenNebula middleware. http://opennebula.org/
7. Green Cloud Scheduller OpenNebula component. http://community.opennebula.org/ecosystem:green_cloud_scheduler/
8. Murphy, A.: Virtualization defined: eight different ways. White paper (2007). http://www.f5.com/pdf/white-papers/virtualization-defined-wp.pdf
9. Brasol, S.M.: Analysis of Advantages and Disadvantages to Server Virtualization. Master Thesis (2009)
10. Sharif, M.I., Lee, W., Cui, W., Lanzi, A.: Secure In-VM monitoring using hardware virtualization. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 477–487 (2009). http://dx.doi.org/10.1145/1653662.1653720
11. Reducing Data Center Energy Consumption, Intel Whitepaper (2010). http://software.intel.com/file/6577/
12. Talaber, R., Brey, T., Lamers, L.: Using Virtualization to Improve Data Center Efficiency, Green Grid White paper (2009). http://www.thegreengrid.org/Global/Content/white-papers/Using-Virtualization-to-Improve-Data-Center-Efficiency

13. Niles, S., Donovan, P.: Virtualization and Cloud Computing: Optimized Power, Cooling, and Management Maximizes Benefits, White Paper Published by APC Schneider Electric (2012). http://www.apcmedia.com/salestools/SNIS-7AULCP_R3_EN.pdf
14. Srikantaiah, S., Kansal, A., Zhao, F.: Energy Aware Consolidation for Cloud Computing, Microsoft Research (2009)
15. Wolf, C.: The myths of virtual machine consolidation (2006). www.SearchServerVirtualization.com
16. Kansal, A., Zhao, F., Liu, J., Kothari, N., Bhattacharya, A.: Virtual Machine Power Metering and Provisioning, SOCC (2010)
17. Stoess, J., Lang, C., Bellosa, F.: Energy management for hypervisor-based virtual machines. In: USENIX Annual Technical Conference (2007)
18. Nathuji, R., England, P., Sharma, P., Singh, A.: Feedback Driven QoS-Aware Power Budgeting for Virtualized Servers, Microsoft Research (2010)
19. Verma, A., Dasgupta, G., Kumar Nayak, T., et al.: Server workload analysis for power minimization using consolidation. In: USENIX Annual Technical Conference (2009)
20. Zhu, Q., Zhu, J., Agrawal G.: Power-aware consolidation of scientific workflows in virtualized environments. High Performance Comput. Networking, Storage Anal. 1–12 (2010). http://dx.doi.org/10.1109/SC.2010.43
21. Uddin, M., Rahman, A.A.: Server consolidation: an approach to make data centers energy efficient. Green Int. J. Sci. Eng. Res. **1**(1), (2010). http://arxiv.org/abs/1010.5037
22. Borgetto, D., Stolf, P., Da Costa, G., Pierson, J.M.: Energy aware autonomic manager. In: 1st International Conference on Energy-Efficient Computing and Net-working (2010)
23. Torres, J., Carrera, D., Beltran, V.: Tailoring resources: the energy efficient consolidation strategy goes beyond virtualization. In: International Conference on Autonomic Computing, pp. 197–198 (2008). http://dx.doi.org/10.1109/ICAC.2008.11
24. Jerger, N.E., Vantrease, D., Lipasti, M.: An Evaluation of server consolidation workloads for multi-core designs. In: Proceedings of the 2007 IEEE 10th International Symposium on Workload Characterization, pp. 47–56 (2007). http://dx.doi.org/10.1109/IISWC.2007.4362180
25. Patel, C., Sharma, R., Bash, C., Graupner, S.: Energy aware grid: global workload placement based on energy efficiency. In: International Mechanical Engineering Congress and Exposition (2003). http://www.hpl.hp.com/techreports/2002/HPL-2002-329.html
26. Kusic, D., Kephart, J.O., Hanson, J.E., Kandasamy, N.: Power and per-formance management of virtualized computing environments via lookahead control. In: Proceedings of the 2008 International Conference on Autonomic Computing (2008). http://dx.doi.org/10.1109/ICAC.2008.31
27. Feller, E., Rillingy, L., Morin, C.: Energy-aware ant colony based workload placement in clouds. In: Proceedings of the IEEE/ACM 12th International Conference on Grid Computing, pp. 26–33 (2011). http://dx.doi.org/10.1109/Grid.2011.13
28. Sharifi, M., Salimi, H., Najafzadeh, M.: Power-efficient distributed scheduling of virtual machines using workload-aware consolidation techniques. J. Supercomputing (2011). http://dx.doi.org/10.1007/s11227-011-0658-5
29. Nagios, the Industry Standard in IT Infrastructure Monitoring. http://www.nagios.org/
30. The Green Grid Data Center Power Efficiency Metrics: PUE and DCiE, Green Grid White Paper (2007). http://www.thegreengrid.org/Global/Content/white-papers/The-Green-Grid-Data-Center-Power-Efficiency-Metrics-PUE-and-DCiE
31. Stanley, J.R., Brill, K.G., Koomey, J.G.: Four Metrics Define Data Center Greenness, Uptime Institute Whitepaper (2007). http://www.dcxdc.ru/files%5C4ede4eff-13b0-49d9-b4da-b0406bfc190e.pdf
32. Dayan, P., Watkins, C.: Reinforcement learning. Encycl. Cogn. Sci. (1999). http://www.gatsby.ucl.ac.uk/~dayan/papers/dw01.pdf
33. OpenNebula OCA API. http://archives.opennebula.org/documentation:archives:rel2.0:java
34. CentOS Overview. http://www.centos.org/
35. Kernel Based Virtual Machine. http://www.linux-kvm.org/page/Main_Page
36. Wake-On-LAN. http://wakeonlan.me/
37. OpenSSH. http://www.openssh.com/

Part II
Big Data Analysis

Chapter 8
High-Performance Storage Support for Scientific Big Data Applications on the Cloud

Dongfang Zhao, Akash Mahakode, Sandip Lakshminarasaiah and Ioan Raicu

8.1 Introduction

While cloud computing has become one of the most prevailing paradigms for big data applications, many legacy scientific applications are still struggling to leverage this new paradigm. One challenge for scientific big data applications to be deployed on the cloud lies in the storage subsystem. Popular file systems such as HDFS [1] are designed for many workloads in data centers that are built with commodity hardware. Nevertheless, many scientific applications deal with a large number of small files [2]—a workload that is not well supported by the data parallelism provided by HDFS. The root cause to the storage discrepancy between scientific applications and many commercial applications on cloud computing stems from their original design goals. Scientific applications assume their data to be stored in remote parallel file systems, and cloud platforms provide node-local storage available on each virtual machine.

This chapter shares our views on how to design storage systems for scientific big data applications on the cloud. Based on the literature and our own experience on big data, cloud computing, and high-performance computing (HPC) in the last decade, we believe that cloud storage would need to provide the following three essential services for scientific big data applications:

D. Zhao (✉)
University of Washington, Seattle, WA, USA
e-mail: dzhao@cs.washington.edu

A. Mahakode · S. Lakshminarasaiah · I. Raicu
Illinois Institute of Technology, Chicago, IL, USA
e-mail: amahakod@hawk.iit.edu

S. Lakshminarasaiah
e-mail: slakshm6@hawk.iit.edu

I. Raicu
e-mail: iraicu@cs.iit.edu

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_8

1. Scalable metadata accesses. Conventional centralized mechanisms for managing metadata on cloud computing, such as GFS [3] and HDFS [1], would not suffice for the extensive metadata accesses of scientific big data applications.

2. Optimized data write. Due to the nature of scientific big data applications, checkpointing is the de facto approach to achieve fault tolerance. This implies that the underlying storage system is expected to be highly efficient on data write as checkpointing itself involves frequent data write.

3. Localized file read. When a failure occurs, some virtual machines (VM) need to restart. Instead of transferring VM images from remote file systems, it would be better to keep a local copy of the image and load it from the local disk if at all possible.

In order to justify the above arguments, we analyze four representative file systems. Two of them are originated from cloud computing (S3FS [4], HDFS [1]). S3FS is built on top of the S3 storage offered by Amazon EC2 cloud as a remote shared storage with the added POSIX support with FUSE [5]. HDFS is an open-source clone of Google File System (GFS [3]) without POSIX support. The other two file systems were initially designed for high-performance computing (Ceph [6], FusionFS [7, 8]). Ceph employs distributed metadata management and the CRUSH [9] algorithm to balance the load. FusionFS is first introduced in [10] and supports several unique features such as erasure coding [11], provenance [12], caching [13, 14], compression [15, 16], and serialization [17]. This study involves two test beds: a conventional cluster Kodiak [18] and a public cloud Amazon EC2 [19].

The remainder of this chapter is organized as follows. Section 8.2 discusses the scalability of metadata accesses. We present the design and performance of achieving optimized file write and localized file read in Sects. 8.3 and 8.4, respectively. Section 8.5 details a real system that employs the proposed design principles as well as unique features in caching, compression, GPUs, provenance, and serialization. We review important literature in big data systems and HPC systems in Sect. 8.6 and finally conclude this chapter in Sect. 8.7.

8.2 Scalable Metadata Accesses

State-of-the-art distributed file systems on cloud computing, such as HDFS [1], still embrace the decade-old design of a centralized metadata server. The reason of such a design is due to the workload characteristic in data centers. More specifically, a large portion of workloads in data centers involve mostly large files. For instance, HDFS has a default 64 MB chunk size (typically 128 MB though), which implicitly implies that the target workload has many files larger than 64 MB; HDFS is not designed or optimized for files smaller than 64 MB. Because many large files are expected, the metadata accesses are not intensive and one single metadata server in many cases is sufficient. In other words, a centralized metadata server in the conventional workloads of cloud computing is not a performance bottleneck.

The centralized design of metadata service, unfortunately, would not meet the requirement of many HPC applications that deal with a larger number of

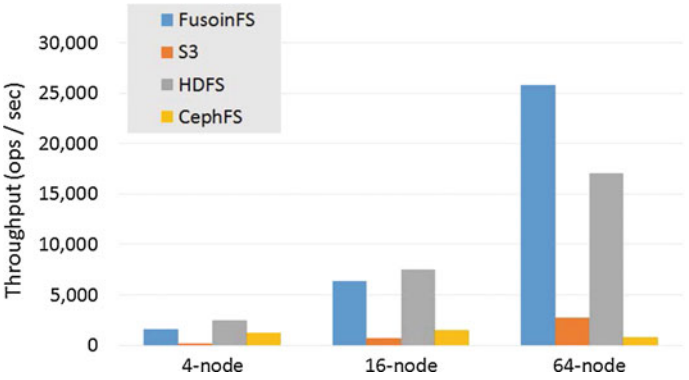


Fig. 8.1 Metadata performance comparison

concurrent metadata accesses. HPC applications are, in nature, highly different than those conventionally deployed on cloud platforms. One of the key differences is file sizes. For instance, Welch and Noer [20] report that 25–90 % of all the 600 million files from 65 Panasas [21] installations are 64 KB or smaller. Such a huge number of small files pose a significantly higher pressure to the metadata server than the cloud applications. A single metadata server would easily become the bottleneck in these metadata-intensive workloads.

A distributed approach to manage metadata seems to be the natural choice for scientific applications on the cloud. Fortunately, several systems (for example, [6, 7]) have employed this design principle. In the remainder of this section, we pick FusionFS and HDFS as two representative file systems to illustrate the importance of a distributed metadata service under intensive metadata accesses. Before discussing the experiment details, we provide a brief introduction of the metadata management of both systems.

HDFS, as a clone of the Google File System [3], has a logically[1] single metadata server (i.e., namenode).The replication of the namenode is for fault tolerance rather than balancing the I/O pressure. That is, all the metadata requests are directed to the single namenode—a simple, yet effective design decision for the cloud workloads. FusionFS is designed to support extremely high concurrency of metadata accesses. It achieves this goal by dispersing metadata to as many nodes as possible. This might be overkill for small- to medium-scale applications, but is essential for those metadata-intensive workloads that are common in scientific applications.

On Amazon EC2, we compare the metadata performance of all four file systems, i.e., FusionFS, S3, HDFS, and CephFS. The workload we use is asking each client to write 10,000 empty files to the according file system. Results are reported in Fig. 8.1.

There are a few observations worth further discussing. First, HDFS outperforms other peers on four nodes and scales well toward 16 nodes. And yet, its scalability is not as good as FusionFS, whose metadata throughput is significantly higher than

[1] because it gets replicated on multiple nodes, physically.

HDFS although the former delivers a lower throughput on four nodes. Second, S3 scales well but is hardly competitive compared to other systems because only with 64 nodes its performance becomes comparable to others on four nodes. Third, CephFS's scalability is poor even from 4 to 16 nodes. Even worse, its performance is degraded when scaling from 16 to 64 nodes.

8.3 Optimized Data Write

Data write is one of the most common I/O workloads in scientific applications due to their de facto mechanism to achieve fault tolerance—checkpointing. Essentially, checkpointing asks the system to periodically persist its memory states to the disks, which involves a larger number of data writes. The persisted data only need to be loaded (i.e., read) after a failure occurs in a completely nondeterministic manner. As the system is becoming increasingly larger, the time interval between consecutive checkpoints is predicted to be dramatically smaller in future systems. [22] From storage's perspective, cloud platform will have to provide highly efficient data write throughput for scientific applications.

Unfortunately, HDFS could hardly provide optimized data write due to the metadata limitation discussed in Sect. 8.2. Figure 8.2 shows the write throughput of FusionFS and HDFS on Kodiak. Similarly to the metadata trend, the write throughput of HDFS also suffers poor scalability beyond 128 nodes.

Another option in cloud platforms is the remote shared storage. It usually provides a unified interface and scalable I/O performance for applications. One example is the S3 storage on Amazon EC2 cloud. S3 does not only provide a set of API but also leverages FUSE [5] to serve as a fully POSIX-compliant file system named S3FS. Therefore S3FS is becoming a popular replacement of the conventional remote shared file systems [23, 24] in HPC.

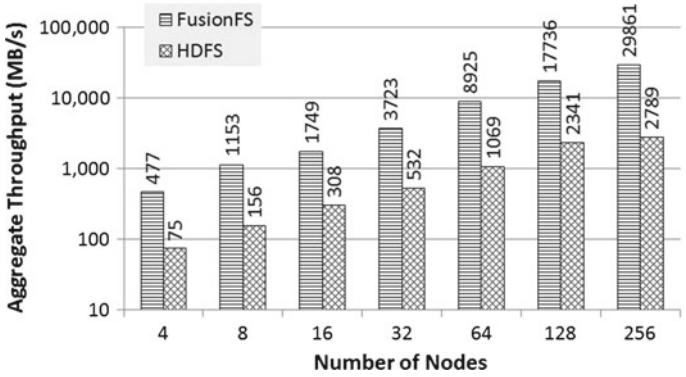


Fig. 8.2 Write throughput of FusionFS and HDFS are compared

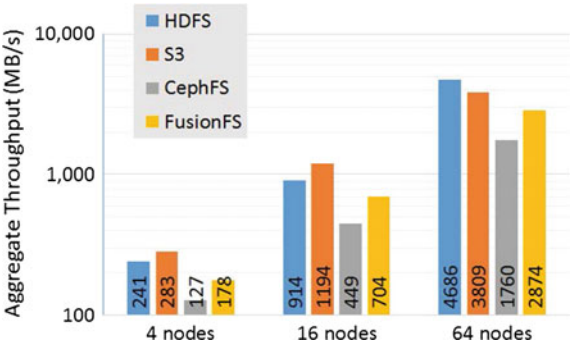


Fig. 8.3 Scalable write throughput on Amazon EC2

We compare all the file systems in discussion so far on the same testbed, i.e., m3.large instance on Amazon EC2. The experiment is in modest scale, from four nodes to 16 nodes, and to 64 nodes in a weak-scaling manner. That is, every node works on the same amount of data—in this case, writing a hundred of 100 MB files to the respective file system.

From Fig. 8.3 we observe that all these systems scale well up to 64 nodes. Note that HDFS and S3 were designed for data centers and cloud computing, while CephFS and FusionFS targeted at scientific applications and high-performance computing. While CephFS is relatively slower than others, FusionFS performs faster comparatively to HDFS and S3 on Amazon EC2 even though FusionFS was not originally designed for data centers. With FusionFS as an example, we believe in the near future a gradual convergence, from the perspective of storage and file system, is emerging between communities of cloud computing and high-performance computing.

We also compare these systems with respect to different file sizes, as shown in Fig. 8.4. Our results show that starting from 1 MB, file size affects little to the write performance on all systems. However, we observe two dramatical extremes on 1 KB

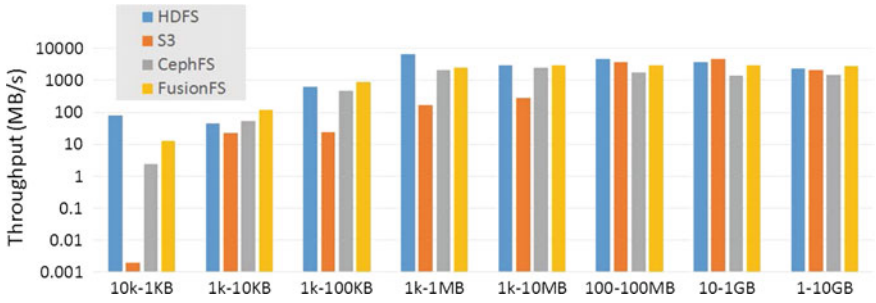


Fig. 8.4 Write throughput of different file sizes

files: HDFS achieves an impressing overall throughput on these small files while S3 is extremely slow. This indicates that for applications where small files dominate, HDFS is in favor with regards to performance.

8.4 Localized File Read

File read throughput is an important metric and is often underestimated since a lot of effort is put on data write as discussed in Sect. 8.3. When a VM is booted or restarted, the image needs to be loaded into the memory and this is becoming a challenging problem in many cloud platforms [25, 26]. Therefore a scalable read throughput is highly desirable for the cloud storage, which urges us to revisit the conventional architecture where files are typically read from remote shared file systems. In HPC this means that the remote parallel file system such as GPFS and Lustre, and in cloud platforms such as Amazon EC2 it implies the remote S3 storage, or the S3FS file system.

We compare the read performance of all the file systems on the m3.large instance of Amazon EC2. Similarly, we scale the experiment from four nodes to 16 nodes, and to 64 nodes in a weak-scaling manner. Every node read a hundred of 100 MB files from the respective file system. Figure 8.5 shows that all these systems scale well up to 64 nodes.

We also compare the systems of interest with respect to their block sizes. In Fig. 8.6, we let each system read different number of files of various sizes, all on 64 Amazon EC2 m3.large instances. For example, "1 k–100 KB" means the system writes 1,000 files of 100 KB.

We observe that for all systems, once the file size is 1 MB and beyond, the read throughput is relatively stable, meaning the I/O bandwidth is saturated. We also note that S3 performs significantly worse than others for files of size 1 KB, although it quickly catches up others at 10 KB and beyond. This suggests that S3 would not be an ideal medium for applications where small files dominate.

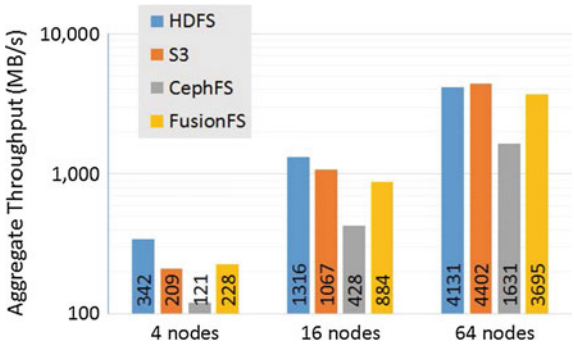


Fig. 8.5 Scalable read throughput on Amazon EC2

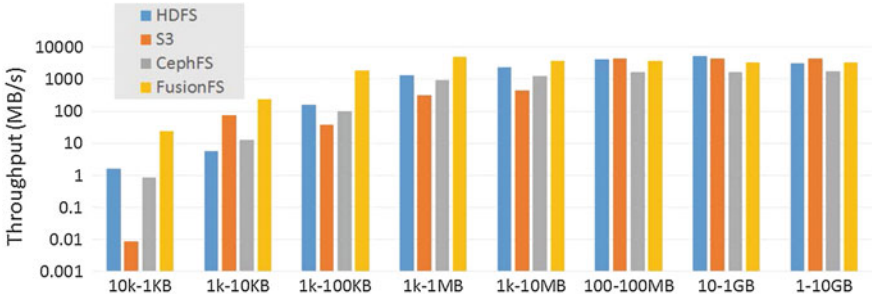


Fig. 8.6 Read throughput of various file sizes

8.5 Put It Altogether: The FusionFS Filesystem

In the previous three sections we discuss three main design criteria for the next-generation HPC storage system on the cloud. This section will present a real system, namely FusionFS, that implements all the aforementioned designs as well as its unique features such as cooperative caching, GPU acceleration, dynamic compression, lightweight provenance, and parallel serialization.

8.5.1 Metadata Management

FusionFS has different data structures for managing regular files and directories. For a regular file, the field *addr* stores the node where this file resides. For a directory, there is a field *filelist* to record all the entries under this directory. This *filelist* field is particularly useful for providing an in-memory speed for directory read such as "ls/mnt/fusionfs". Nevertheless, both regular files and directories share some common fields, such as timestamps and permissions, which are commonly found in traditional i-nodes.

The metadata and data on a local node are completely decoupled: a regular file's location is independent of its metadata location. This flexibility allows us to apply different strategies to metadata and data management, respectively. Moreover, the separation between metadata and data has the potential to plug in alternative components to metadata or data management, making the system more modular.

Besides the conventional metadata information for regular files, there is a special flag in the value indicating if this file is being written. Specifically, any client who requests to write a file needs to set this flag before opening the file, and will not reset it until the file is closed. The atomic compare-swap operation supported by DHT [27, 28] guarantees the file consistency for concurrent writes.

Another challenge on the metadata implementation is on the large-directory performance issues. In particular, when a large number of clients write many small files

on the same directory concurrently, the value of this directory in the key-value pair gets incredibly long and responds extremely slowly. The reason is that a client needs to update the entire old long string with the new one, even though the majority of the old string is unchanged. To fix that, we implement an atomic append operation that asynchronously appends the incremental change to the value. This approach is similar to Google File System [3], where files are immutable and can only be appended. This gives us excellent concurrent metadata modification in large directories, at the expense of potentially slower directory metadata read operations.

8.5.2 File Write

Before writing to a file, the process checks if the file is being accessed by another process. If so, an error number is returned to the caller. Otherwise the process can do one of the following two things: If the file is originally stored on a remote node, the file is transferred to the local node in the *fopen()* procedure, after which the process writes to the local copy. If the file to be written is right on the local node, or it is a new file, then the process starts writing the file just like a system call.

The aggregate write throughput is obviously optimal because file writes are associated with local I/O throughput and avoids the following two types of cost: (1) the procedure to determine to which node the data will be written, normally accomplished by pinging the metadata nodes or some monitoring services, and (2) transferring the data to a remote node. It should be clear that FusionFS works at the file level, thus chunking the file is not an option. Nevertheless, we will support chunk-level data movement in the next release of FusionFS. The downside of this file write strategy is the poor control on the load balance of compute node storage. This issue could be addressed by an asynchronous re-balance procedure running in the background, or by a load-aware task scheduler that steals tasks from the active nodes to the more idle ones.

When the process finishes writing to a file that is originally stored in another node, FusionFS does not send the newly modified file back to its original node. Instead, the metadata of this file is updated. This saves the cost of transferring the file data over the network.

8.5.3 File Read

Unlike file write, it is impossible to arbitrarily control where the requested data reside for file read. The location of the requested data is highly dependent on the I/O pattern. However, we could determine which node the job is executed on by the distributed workflow system such as Swift [29]. That is, when a job on node A needs to read some data on node B, we reschedule the job on node B. The overhead of rescheduling the job is typically smaller than transferring the data over the network, especially

for data-intensive applications. In our previous work [30], we detailed this approach, and justified it with theoretical analysis and experiments on benchmarks and real applications.

Indeed, remote readings are not always avoidable for some I/O patterns such as merge sort. In merge sort, the data need to be joined together, and shifting the job cannot avoid the aggregation. In such cases, we need to transfer the requested data from the remote node to the requesting node.

8.5.4 Hybrid and Cooperative Caching

When the node-local storage capacity is limited, remote parallel filesystems should coexist with FusionFS to store large-sized data. In some sense, FusionFS is regarded as a caching middleware between the main memory and remote parallel filesystems. We are interested in what placement policies (i.e., caching strategies) are beneficial to HPC workloads.

Our first attempt is a user-level caching middle where on every compute node, assuming a memory-class device (for example, SSD) is accessible along with a conventional spinning hard drive. That is, each compute node is able to manipulate data on hybrid storage systems. The middleware, named HyCache [14], speeds up HDFS by up to 28 %.

Our second attempt is a cooperative caching mechanism across all the compute nodes, called HyCache+ [13]. HyCache+ extends HyCache in terms of network storage support, higher data reliability, and improved scalability. In particular, a two-stage scheduling mechanism called 2-Layer Scheduling (2LS) is devised to explore the data locality of cached data on multiple nodes. HyCache+ delivers two orders of magnitude higher throughput than the remote parallel filesystems, and 2LS outperforms conventional LRU caching by more than one order of magnitude.

8.5.5 Accesses to Compressed Data

Conventional data compression embedded in filesystems naively applies the compressor to either the entire file or every block of the file. Both methods have limitations on either inefficient data accesses or degraded compression ratio. We introduce a new concept called virtual chunks, which enable efficient random accesses to the compressed files while retaining high compression ratio.

The key idea [16] is to append additional references to the compressed files so that a decompression request could start at an arbitrary position. Current system prototype [15] assumes the references are equidistant, and experiments show that virtual chunks improve random accesses by 2× speedup.

8.5.6 Space-Efficient Data Reliability

The reliability of distributed filesystems is typically achieved through data replication. That is, a primary copy serves most requests, and there are a number of backup copies (replicas) that would become the primary copy upon a failure.

One concern with the conventional approach is its space efficiency; for example, two replicas imply poor 33 % space efficiency. On the other hand, erasure coding has been proposed to improve the space efficiency; unfortunately it is criticized on its computation overhead. We integrated GPU-accelerated erasure coding to FusionFS and report the performance in [11]. Results showed that erasure coding could improve FusionFS performance by up to 1.82×.

8.5.7 Distributed Data Provenance

The traditional approach to track application's provenance is through a centralized database. To address this performance bottleneck on large-scale systems, in [31] we propose a lightweight database on every compute node. This allows every participating node to maintain its own data provenance, and results in highly scalable aggregate I/O throughput. Admittedly, an obvious drawback of this approach is on the interaction among multiple physical databases: the provenance overhead becomes unacceptable when there is heavy traffic among peers.

To address the above drawback, we explore the feasibility of tracking data provenance in a completely distributed manner in [12]. We replace the database component by a graph-like hashtable data structure, and integrate it into the FusionFS filesystem. With a hybrid granularity of provenance information on both block- and file-level, FusionFS achieves over 86 % system efficiency on 1,024 nodes. A query interface is also implemented with small performance overhead as low as 5.4 % on 1,024 nodes.

8.5.8 Parallel Serialization

We have explored how to leverage modern computing systems' multi-cores to improve the serialization and deserialization speed of large objects. [17] Rather than proposing new serialization algorithms, we tackle the problem from a system's perspective. Specifically, we propose to leverage multiple CPU cores to split a large object into smaller sub-objects, so to be serialized in parallel. While data parallelism is not a new idea in general, it has never been applied to data serialization and poses new problems. For instance, serializing multiple chunks of a large object incurs additional overhead such as metadata maintenance, thread and process synchronization, resource contention. In addition, the granularity (i.e., the number of sub-objects)

is a machine-dependent choice: the optimal number of concurrent processes and threads might not align with the available CPU cores.

In order to overcome these challenges and better understand whether the proposed approach could improve the performance of data serialization of large objects, we provide detailed analysis on the system design, for example, how to determine the sub-object's granularity for optimal performance and how to ensure that the performance gain is larger than the cost. To demonstrate the effectiveness of our proposed approach, we implemented a system prototype called parallel protocol buffers (PPB) by extending a widely used open-source serialization utility (Google's Protocol Buffers [32]). We have evaluated PPB on a variety of test beds: a conventional Linux server, the Amazon EC2 cloud, and an IBM Blue Gene/P supercomputer. Experimental results confirm that the proposed approach could significantly accelerate the serialization process. In particular, PPB could accelerate the metadata interchange 3.6× faster for FusionFS.

8.6 Related Work

Conventional storage in HPC systems for scientific applications are mainly remote to compute resources. Popular systems include GPFS [23], Lustre [24], PVFS [33]. All these systems are typically deployed on a distinct cluster from compute nodes. The architecture with separated compute- and storage-resources, which was designed decades ago, has shown its limitation for modern applications that are becoming increasingly data-intensive [7].

Cloud computing, on the other hand, is built on the commodity hardware where local storage is typically available for virtual computing machines. The de facto node-local file system (Google File System [3], HDFS [1]), however, can be hardly leveraged by scientific applications out of the box due to the concerns on small file accesses, POSIX interface, and so forth. Another category of storage in the cloud is similar to the conventional HPC solution—a remote shared storage such as Amazon S3. A POSIX-compliant file system built on S3 is also available named S3FS [4]. Unfortunately its throughput performance usually becomes a bottleneck of the applications and thus limits its use in practice.

Fortunately, researchers have made a significant amount of effort [34, 35] to bridge the gap between two extremes (HPC and cloud computing) of storage paradigms, particularly in terms of scalability [36]. We observe more and more node-local and POSIX-compliant storage systems (Ceph [6], FusionFS [7]) being tested on the cloud.

We will briefly review the unique features provided by FusionFS as follows.

8.6.1 Filesystem Caching

To the best of our knowledge, HyCache is the first user-level POSIX-compliant hybrid caching for distributed file systems. Some of our previous work [30, 37] proposed data caching to accelerate applications by modifying the applications and/or their workflow, rather than at the filesystem level. Other existing work requires modifying OS kernel, or lacks of a systematic caching mechanism for manipulating files across multiple storage devices, or does not support the POSIX interface. Any of the these concerns would limit the system's applicability to end users. We will give a brief review of previous studies on hybrid storage systems.

Some recent work reported the performance comparison between SSD and HDD in more perspectives [38, 39]. Hystor [40] aims to optimize of the hybrid storage of SSDs and HDDs. However it requires to modify the kernel which might cause some issues. A more general multitiering scheme was proposed in [41] which helps decide the needed numbers of SSD/HDDs and manage the data shift between SSDs and HDDs by adding a "pseudo device driver," again, in the kernel. iTransformer [42] considers the SSD as a traditional transient cache in which case data needs to be written to the spinning hard disk at some point once the data is modified in the SSD. iBridge [43] leverages SSD to serve request fragments and bridge the performance gap between serving fragments and serving large sub-requests. HPDA [44] offers a mechanism to plug SSDs into RAID in order to improve the reliability of the disk array. SSD was also proposed to be integrated to the RAM level which makes SSD as the primary holder of virtual memory [45]. NVMalloc [46] provides a library to explicitly allow users to allocate virtual memory on SSD. Also for extending virtual memory with Storage Class Memory (SCM), SCMFS [47] concentrates more on the management of a single SCM device. FAST [48] proposed a caching system to pre-fetch data in order to quicken the application launch. In [49] SSD is considered as a read-only buffer and migrate those random-writes to HDD.

A thorough review of classical caching algorithms on large-scale data-intensive applications is recently reported in [50]. HyCache+ is different from the classical cooperative caching [51] in that HyCache+ assumes persistent underlying storage and manipulates data at the file level. As an example of distributed caching for distributed file systems, Blue Whale Cooperative Caching (BWCC) [52] is a read-only caching system for cluster file systems. In contrast, HyCache+ is a POSIX-compliant I/O storage middleware that transparently interacts with the underlying parallel file systems. Even though the focus of this chapter lies on the 2-layer hierarchy of a local cache (e.g., SSD) and a remote parallel file system (e.g., GPFS [23]), the approach presented in HyCache+ is applicable to multitier caching architecture as well. Multilevel caching gains much research interest, especially in the emerging age of cloud computing where the hierarchy of (distributed) storage is being redefined with more layers. For example, Hint-K [53] caching is proposed to keep track of the last K steps across all the cache levels, which generalizes the conventional LRU-K algorithm concerned only on the single-level information.

There are extensive studies on leveraging data locality for effective caching. Block Locality Caching (BLC) [54] captures the backup and always uses the latest locality information to achieve better performance for data deduplication systems. The File Access corRelation Mining and Evaluation Reference model (FARMER) [55] optimizes the large-scale file system by correlating access patterns and semantic attributes. In contrast, HyCache+ achieves data locality with a unique mix of two principles: (1) write is always local, and (2) read locality depends on the novel 2LS mechanism which schedules jobs in a deterministic manner followed by a local heuristic replacement policy.

While HyCache+ presents a pure software solution for distributed cache, some orthogonal work focuses on improving caching from the hardware perspective. In [56], a hardware design is proposed with low overhead to support effective shared caches in multicore processors. For shared last-level caches, COOP [57] is proposed to only use one bit per cache line for re-reference prediction and optimize both locality and utilization. The REDCAP project [58] aims to logically enlarge the disk cache using a small portion of main memory, so that the read time could be reduced. For Solid-State Drive (SSD), a new algorithm called lazy adaptive replacement cache [59] is proposed to improve the cache hit and prolong the SSD lifetime.

Power-efficient caching has drawn a lot of research interests. It is worth mentioning that HyCache+ aims to better meet the need of high I/O performance for HPC systems, and power consumption is not the major consideration at this point. Nevertheless, it should be noted that power consumption is indeed one of the toughest challenges to be overcome in future systems. One of the earliest work [60] tried to minimize the energy consumption by predicting the access mode and allowing cache accesses to switch between the prediction and the access modes. Recently, a new caching algorithm [61] was proposed to save up to 27 % energy and reduce the memory temperature up to 5.45 °C with negligible performance degradation. EEVFS [62] provides energy efficiency at the file system level with an energy-aware data layout and the prediction on disk idleness.

While HyCache+ is architected for large-scale HPC systems, caching has been extensively studied in different subjects and fields. In cloud storage, Update-batched Delayed Synchronization (UDS) [63] reduces the synchronization cost by buffering the frequent and short updates from the client and synchronizing with the underlying infrastructure in a batch fashion. For continuous data (e.g., online video), a new algorithm called Least Waiting Probability (LWP) [64] is proposed to optimize the newly defined metric called user waiting rate. In geoinformatics, the method proposed in [65] considers both global and local temporal-spatial changes to achieve high cache hit rate and short response time.

The job scheduler proposed in this work takes a greedy strategy to achieve the optimal solution for the HyCache+ architecture. A more general, and more difficult, scheduling problem could be solved in a similar heuristic approach [66, 67]. For an even more general combinatorial optimization problem in a network, both precise and bound-proved low-degree polynomial approximation algorithms were reported in [68, 69]. Some incremental approaches [70–72] were proposed to

efficiently retain the strong connectivity of a network and solve the satisfiability problem with constraints.

8.6.2 Filesystem Compression

While the storage system could be better designed to handle more data, an orthogonal approach is to address the I/O bottleneck by squeezing the data with compression techniques. One example where data compression gets particularly popular is checkpointing, an extremely expensive I/O operation in HPC systems. In [73], it showed that data compression had the potential to significantly reduce the checkpointing file sizes. If multiple applications run concurrently, a data-aware compression scheme [74] was proposed to improve the overall checkpointing efficiency. Recent study [75] shows that combining failure detection and proactive checkpointing could improve 30 % efficiency compared to classical periodical checkpointing. Thus data compression has the potential to be combined with failure detection and proactive checkpointing to further improve the system efficiency. As another example, data compression was also used in reducing the MPI trace size, as shown in [76]. A small MPI trace enables an efficient replay and analysis of the communication patterns in large-scale machines.

It should be noted that a compression method does not necessarily need to restore the absolutely original data. In general, compression algorithms could be categorized into to two groups: lossy algorithms and lossless algorithms. A lossy algorithm might lose some (normally a small) percentage of accuracy, while a lossless one has to ensure the 100 % accuracy. In scientific computing, studies [77, 78] show that lossy compression could be acceptable, or even quite effective, under certain circumstances. In fact, lossy compression is also popular in other fields, e.g. the most widely compatible lossy audio and video format MPEG-1 [79]. This section presents virtual chunks mostly by going through a delta-compression example based on XOR, which is a lossless compression. It does not imply that virtual chunks cannot be used in a lossy compression. Virtual chunk is not a specific compression algorithm, but a system mechanism that is applicable to any splittable compression, no matter if it is lossy or lossless.

Some frameworks are proposed as middleware to allow applications call high-level I/O libraries for data compression and decompression, e.g., [80–82]. None of these techniques take consideration of the overhead involved in decompression by assuming the chunk allocated to each node would be requested as an entirety. In contrast, virtual chunks provide a mechanism to apply flexible compression and decompression.

There is much previous work to study the file system support for data compression. Integrating compression to log-structured file systems was proposed decades ago [83], which suggested a hardware compression chip to accelerate the compressing and decompressing. Later, XDFS [84] described the systematic design and implementation for supporting data compression in file systems with BerkeleyDB [85].

MRAMFS [86] was a prototype file system to support data compression to leverage the limited space of nonvolatile RAM. In contrast, virtual trunks represent a general technique applicable to existing algorithms and systems.

Data deduplication is a general inter-chunk compression technique that only stores a single copy of the duplicate chunks (or blocks). For example, LBFS [87] was a networked file system that exploited the similarities between files (or versions of files) so that chunks of files could be retrieved in the client's cache rather than transferring from the server. CZIP [88] was a compression scheme on content-based naming that eliminated redundant chunks and compressed the remaining (i.e., unique) chunks by applying existing compression algorithms. Recently, the metadata for the deduplication (i.e., file recipe) was also slated for compression to further save the storage space [89]. While deduplication focuses on inter-chunk compressing, virtual chunk focuses on the I/O improvement within the chunk.

Index has been introduced to data compression to improve the compressing and query speed, e.g., [90, 91]. The advantage of indexing is highly dependent on the chunk size: large chunks are preferred to achieve high compression ratios in order to amortize the indexing overhead. Large chunks, however, would cause potential decompression overhead as explained earlier in this chapter. Virtual chunk overcomes the large-chunk issue by logically splitting the large chunks with fine-grained partitions while still keeping the physical coherence.

8.6.3 GPU Acceleration

Recent GPU technology has drawn much research interest of applying these many-cores for data parallelism. For example, GPUs are proposed to parallelize the XML processing [92]. In high-performance computing, a GPU-aware MPI was proposed to enable the inter-GPU communication without changing the original MPI interface [93]. Nevertheless, GPUs do not necessarily provide superior performance; GPUs might suffer from factors such as small shared memory and weak single-thread performance as shown in [94]. Another potential drawback of GPUs lies in the dynamic instrumentation that introduces runtime overhead. Yet, recent study [95] shows that the overhead could be alleviated by information flow analysis and symbolic execution. In this paper, we leverage GPUs in key-value stores—a new domain for many-cores.

8.6.4 Filesystem Provenance

As distributed systems become more ubiquitous and complex, there is a growing emphasis on the need for tracking provenance metadata along with file system metadata. A thorough review is presented in [96]. Many Grid systems like Chimera [97] and the Provenance Aware Service Oriented Architecture (PASOA) [98] provide

provenance tracking mechanisms for various applications. However these systems are very domain-specific and do not capture provenance at the filesystem level. The Distributed Provenance Aware Storage System (DPASS) tracks the provenance of files in a distributed file system by intercepting filesystem operations and sending this information via a netlink socket to user-level daemon that collects provenance in a database server [99]. The provenance is however, collected in a centralized fashion, which is a poor design choice for distributed file systems meant for extreme scales. Similarly in efficient retrieval of files, provenance is collected centrally [100].

PASS describes global naming, indexing, and querying in the context of sensor data [101], which is a challenging problem also from system's perspective [36]. PA-NFS [102] enhances NFS to record provenance in local area networks but does not consider distributed naming explicitly. SPADE [103] addresses the issue using storage identifiers for provenance vertices that are unique to a host and requiring distributed provenance queries to disambiguate vertices by referring to them by the host on which the vertex was generated as well as the identifier local to that host.

Several storage systems have been considered for storing provenance. ExSPAN [104] extends traditional relational models for storing and querying provenance metadata. SPADE supports both graph and relational database storage and querying. PASS has explored the use of clouds [101]. Provbase uses Hbase to store and query scientific workflow provenance [105]. Further compressing provenance [104], indexing [106] and optimization techniques [107] have also been considered. However, none of these systems have been tested for exascale architectures. To give adequate merit to the previous designs, we have integrated FusionFS with SPADE as well as considered FusionFS's internal storage system for storing audited provenance.

8.6.5 Data Serialization

Many serialization frameworks are developed to support transporting data over distributed systems. XML [108] represents a set of rules to encoding documents or text-based files. Another format, namely JSON [109], is treated as a lightweight alternative to XML in web services and mobile devices as well. While XML and JSON are the most widely used data serialization format for text-based files, binary format is also gaining its popularity. A binary version of JSON is available called BSON [110]. Two other famous binary data serialization frameworks are Google's Protocol Buffers [32] and Apache Thrift [111]. Both frameworks are designed to support lightweight and fast data serialization and deserialization, which could substantially improve the data communication in distributed systems. The key difference between Thrift and Protocol Buffers is that the former has the built-in support for RPC.

Many other serialization utilities are available at the present. Avro [112] is used by Hadoop for serialization. Internally, it uses JSON [109] to represent data types and protocols and improves the performance of the Java-based framework. Etch [113] supports more flexible data models (for example, trees), but it is slower and generates

larger files. BERT [114] supports data format compatible with Erlang's binary serialization format. Message Pack [115] allows both binary data and non UTF-8 encoded strings. Hessian [116] is a binary web service protocol that is 2× faster than the Java serialization with significantly smaller compressed data size. ICE [117] is a middleware platform that supports object-oriented RPC and data exchange. CBOR [118] is designed to support extremely small message size.

None of the aforementioned systems, however, support data parallelism. Thus they suffer the low efficiency problem when multiple CPU cores are available particularly when the data is large in size. PPB, on the other hand, takes advantage of the idle cores and leverage them for parallelizing the compute-intensive process of data serialization

Many frameworks are recently developed for parallel data processing. MapReduce [119] is a programming paradigm and framework that allows users to process terabytes of data over massive-scale architecture in a matter of seconds. Apache Hadoop [120] is one of the most popular open-source implementations of MapReduce framework. Apache Spark [121] is an execution engine which supports more types of workload than Hadoop and MapReduce.

Several parallel programming models and paradigms have been existing for decades. Message Passing Interface (MPI) a standard for messages exchange between processes. It greatly reduces the burden from developers who used to consider detailed protocols in multiprocessing programs and tries to optimize the performance in many scenarios. The major implementation includes MPICH [122] and Open MPI [123]. OpenMP [124] is a set of compiler directives and runtime library routines that enable the parallelization of code's execution over shared memory multiprocessor computers. It supports different platforms and processor architectures, programming languages, and operating systems. Posix Threads (Pthread) is defined as a set of C programming types and function calls. It provides standardized programming interface to create and manipulate threads, which allow developers to take full advantage of the capabilities of threads. Microsoft's Parallel Patterns Library (PPL) [125] gives an imperative programming model that introduces parallelism to applications and improves scalability.

Numerous efforts have been devoted to utilizing or improving data parallelism in cluster and cloud computing environment. Jeon et al. [126, 127] proposed adaptive parallelization and prediction approaches for search engine query. Lee et al. [128] presented how to reduce data migration cost and improve I/O performance by incorporating parallel data compression on the client side. Klasky et al. [129] proposed a parallel data-streaming approach with multi-threads to migrate terabytes of scientific data across distributed supercomputer centers. Some work [130–133] proposed data-parallel architectures and systems for large-scale distributed computing. In [134, 135], authors exploited the data parallelism in a program by dynamically executing sets of serialization codes concurrently.

Unfortunately, little study exists on data parallelism for data serialization, mainly because large messages are usually not the dominating cost by convention. PPB [17] for the first time identifies that large message is a challenging problem from our

observations on real-world applications at Google. We hope our PPB experience could provide the community insights for designing the next-generation data serialization tools.

8.7 Conclusion

This chapter envisions the characteristics of future cloud storage systems for scientific applications that used to be running on HPC systems. Based on the literature and our own FusionFS experience, we believe the key designs of future storage system comprise the fusion of compute and storage resources as well as completely distributed data manipulation (both metadata and data), namely (1) distributed metadata accesses, (2) optimized data write, and (3) localized file read.

To make matters more concrete, we then detail the design and implementation of FusionFS, whose uniqueness lies in its highly scalable metadata and data throughput. We also discuss its integral features such as cooperative caching, efficient accesses to compressed data, space-efficient data reliability, distributed data provenance, and parallel data serialization. All the aforementioned features enable FusionFS to nicely bridge the storage gap between scientific big data applications and cloud computing.

References

1. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: Proceedings of IEEE Symposium on Mass Storage Systems and Technologies (2010)
2. Carns, P., Lang, S., Ross, R., Vilayannur, M., Kunkel, J., Ludwig, T.: Small-file access in parallel file systems. In: Proceedings of IEEE International Symposium on Parallel and Distributed Processing (2009)
3. Ghemawat, S., Gobioff, H., Leung, S.T.: The Google file system. In: ACM Symposium on Operating Systems Principles (2003)
4. S3FS: https://code.google.com/p/s3fs/. Accessed 6 March 2015
5. FUSE: http://fuse.sourceforge.net. Accessed 5 Sept 2014
6. Weil, S.A., Brandt, S.A., Miller, E.L., Long, D.D.E., Maltzahn, C.: Ceph: a scalable, high-performance distributed file system. In: Proceedings of the 7th Symposium on Operating Systems Design and Implementation (2006)
7. Zhao, D., Zhang, Z., Zhou, X., Li, T., Wang, K., Kimpe, D., Carns, P., Ross, R., Raicu, I.: FusionFS: Toward supporting data-intensive scientific applications on extreme-scale distributed systems. In: Proceedings of IEEE International Conference on Big Data, pp. 61–70 (2014)
8. Zhao, D., Liu, N., Kimpe, D., Ross, R., Sun, X.H., Raicu, I.: Towards exploring data-intensive scientific applications at extreme scales through systems and simulations. IEEE Trans. Parallel Distrib. Syst. 1–14 (2015). doi:10.1109/TPDS.2015.2456896
9. Weil, S.A., Brandt, S.A., Miller, E.L., Maltzahn, C.: Crush: controlled, scalable, decentralized placement of replicated data. In: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing (2006)

10. Zhao, D., Raicu, I.: Distributed file systems for exascale computing. In: International Conference for High Performance Computing, Networking, Storage and Analysis (SC '12), doctoral showcase (2012)
11. Zhao, D., Burlingame, K., Debains, C., Alvarez-Tabio, P., Raicu, I.: Towards high-performance and cost-effective distributed storage systems with information dispersal algorithms. In: IEEE International Conference on Cluster Computing (2013)
12. Zhao, D., Shou, C., Malik, T., Raicu, I.: Distributed data provenance for large-scale data-intensive computing. In: IEEE International Conference on Cluster Computing (2013)
13. Zhao, D., Qiao, K., Raicu, I.: Hycache+: towards scalable high-performance caching middleware for parallel file systems. In: Proceedings of the 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 267–276 (2014)
14. Zhao, D., Raicu, I.: HyCache: a user-level caching middleware for distributed file systems. In: Proceedings of IEEE 27th International Symposium on Parallel and Distributed Processing Workshops and PhD Forum (2013)
15. Zhao, D., Yin, J., Qiao, K., Raicu, I.: Virtual chunks: on supporting random accesses to scientific data in compressible storage systems. In: Proceedings of IEEE International Conference on Big Data, pp. 231–240 (2014)
16. Zhao, D., Yin, J., Raicu, I.: Improving the i/o throughput for data-intensive scientific applications with efficient compression mechanisms. In: International Conference for High Performance Computing, Networking, Storage and Analysis (SC '13), poster session (2013)
17. Zhao, D., Qiao, K., Zhou, Z., Li, T., Zhou, X., Wang, K., Raicu, I.: Exploiting multi-cores for efficient interchange of large messages in distributed systems. Concurrency Comput.: Pract. Experience 2015 (accepted)
18. Kodiak: https://www.nmc-probe.org/wiki/Machines:Kodiak. Accessed 5 Sept 2014
19. Amazon EC2: http://aws.amazon.com/ec2. Accessed 6 March 2015
20. Welch, B., Noer, G.: Optimizing a hybrid SSD/HDD HPC storage system based on file size distributions. In: IEEE 29th Symposium on Mass Storage Systems and Technologies (2013)
21. Nagle, D., Serenyi, D., Matthews, A.: The Panasas activescale storage cluster: delivering scalable high bandwidth storage. In: Proceedings of ACM/IEEE Conference on Supercomputing (2004)
22. Zhao, D., Zhang, D., Wang, K., Raicu, I.: Exploring reliability of exascale systems through simulations. In: Proceedings of the 21st ACM/SCS High Performance Computing Symposium (HPC) (2013)
23. Schmuck, F., Haskin, R.: GPFS: a shared-disk file system for large computing clusters. In: Proceedings of the 1st USENIX Conference on File and Storage Technologies (2002)
24. Schwan, P.: Lustre: building a file system for 1,000-node clusters. In: Proceedings of the Linux Symposium (2003)
25. Wu, H., Ren, S., Garzoglio, G., Timm, S., Bernabeu, G., Chadwick, K., Noh, S.-Y.: A reference model for virtual machine launching overhead. IEEE Trans. Cloud Comput. (pp. 99), 1–1 (2014)
26. Wu, H., Ren, S., Garzoglio, G., Timm, S., Bernabeu, G., Noh, S.-Y.: Modeling the virtual machine launching overhead under fermicloud. In: 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), May 2014
27. Li, T., Zhou, X., Brandstatter, K., Zhao, D., Wang, K., Rajendran, A., Zhang, Z., Raicu, I.: ZHT: A light-weight reliable persistent dynamic scalable zero-hop distributed hash table. In: Proceedings of IEEE International Symposium on Parallel and Distributed Processing (2013)
28. Li, T., Ma, C., Li, J., Zhou, X., Wang, K., Zhao, D., Raicu, I.: Graph/z: a key-value store based scalable graph processing system. In: IEEE International Conference on Cluster Computing (2015)
29. Zhao, Y., Hategan, M., Clifford, B., Foster, I., von Laszewski, G., Nefedova, V., Raicu, I., Stef-Praun, T., Wilde, M.: Swift: Fast, reliable, loosely coupled parallel computation. In: IEEE Congress on Services (2007)
30. Raicu, I., Foster, I.T., Zhao, Y., Little, P., Moretti, C.M., Chaudhary, A., Thain, D.: The quest for scalable support of data-intensive workloads in distributed systems. In: Proceedings of ACM International Symposium on High Performance Distributed Computing (2009)

31. Shou, C., Zhao, D., Malik, T., Raicu, I.: Towards a provenance-aware distributed filesystem. In: 5th Workshop on the Theory and Practice of Provenance (TaPP) (2013)
32. Protocol Buffers: http://code.google.com/p/protobuf/. Accessed 5 Sept 2014
33. Carns, P.H., Ligon, W.B., Ross, R.B., Thakur, R.: PVFS: a parallel file system for linux clusters. In: Proceedings of the 4th Annual Linux Showcase and Conference (2000)
34. Li, T., Zhou, X., Wang, K., Zhao, D., Sadooghi, I., Zhang, Z., Raicu, I.: A convergence of key-value storage systems from clouds to supercomputer. Concurrency Comput.: Pract. Experience (2016)
35. Zhao, D., Yang, X., Sadooghi, I., Garzoglio, G., Timm, S., Raicu, I.: High-performance storage support for scientific applications on the cloud. In: Proceedings of the 6th Workshop on Scientific Cloud Computing (ScienceCloud) (2015)
36. Li, T., Keahey, K., Wang, K., Zhao, D., Raicu, I.: A dynamically scalable cloud data infrastructure for sensor networks. In: Proceedings of the 6th Workshop on Scientific Cloud Computing (ScienceCloud) (2015)
37. Raicu, I., Zhao, Y., Foster, I.T., Szalay, A.: Accelerating large-scale data exploration through data diffusion. In: Proceedings of the 2008 International Workshop on Data-aware Distributed Computing (2008)
38. Li, S., Huang, H.H.: Black-box performance modeling for solid-state drives. In: 2010 IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS) (2010)
39. Rizvi, S., Chung, T.-S.: Flash SSD vs HDD: High performance oriented modern embedded and multimedia storage systems. In: 2nd International Conference on Computer Engineering and Technology (ICCET) (2010)
40. Chen, F., Koufaty, D.A., Zhang, X.: Hystor: making the best use of solid state drives in high performance storage systems. In: Proceedings of the International Conference on Supercomputing (2011)
41. Guerra, J., Pucha, H., Glider, J., Belluomini, W., Rangaswami, R.: Cost effective storage using extent based dynamic tiering. In: Proceedings of the 9th USENIX Conference on File and Stroage Technologies (2011)
42. Zhang, X., Davis, K., Jiang, S.: iTransformer: using SSD to improve disk scheduling for high-performance I/O. In: Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium (2012)
43. Zhang, X., Ke, L., Davis, K., Jiang, S.: iBridge: improving unaligned parallel file access with solid-state drives. In: Proceedings of the 2013 IEEE 27th International Parallel and Distributed Processing Symposium (2013)
44. Mao, B., Jiang, H., Feng, D., Wu, S., Chen, J., Zeng, L., Tian, L.: HPDA: a hybrid parity-based disk array for enhanced performance and reliability. In: 2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS) (2010)
45. Badam, A., Pai, V.S.: SSDAlloc: hybrid SSD/RAM memory management made easy. In: Proceedings of the 8th USENIX Conference on Networked systems design and implementation (2011)
46. Wang, C., Vazhkudai, S.S., Ma, X., Meng, F., Kim, Y., Engelmann, C.: Nvmalloc: exposing an aggregate ssd store as a memory partition in extreme-scale machines. In: Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium (2012)
47. Wu, X., Narasimha Reddy, A.L.: SCMFS: a file system for storage class memory. In: Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (2011)
48. Joo, Y., Ryu, J., Park, S., Shin, K.G.: FAST: quick application launch on solid-state drives. In: Proceedings of the 9th USENIX Conference on File and Stroage Technologies (2011)
49. Yang, Q., Ren, J.: I-CASH: intelligently coupled array of SSD and HDD. In: Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture (2011)
50. Fares, R., Romoser, B., Zong, Z., Nijim, M., Qin, X.: Performance evaluation of traditional caching policies on a large system with petabytes of data. In: 2012 IEEE 7th International Conference on Networking, Architecture and Storage (NAS) (2012)

51. Podlipnig, S., Böszörmenyi, L.: A survey of web cache replacement strategies. ACM Comput. Surv. **35**(4) (2003)
52. Shi, L., Liu, Z., Xu, L.: Bwcc: a fs-cache based cooperative caching system for network storage system. In: Proceedings of the 2012 IEEE International Conference on Cluster Computing (2012)
53. Wu, C., Xubin, H., Qiang, C., Changsheng, X., Shenggang, W.: Hint-k: an efficient multi-level cache using k-step hints. IEEE Trans. Parallel Distrib. Syst. **99** (2013)
54. Meister, D., Kaiser, J., Brinkmann, A.: Block locality caching for data deduplication. In: Proceedings of the 6th International Systems and Storage Conference (2013)
55. Xia, P., Feng, D., Jiang, H., Tian, L., Wang, F.: Farmer: a novel approach to file access correlation mining and evaluation reference model for optimizing peta-scale file system performance. In: Proceedings of the 17th International Symposium on High Performance Distributed Computing (2008)
56. Lin, J., Lu, Q., Ding, X., Zhang, Z., Zhang, X., Sadayappan, P.: Enabling software management for multicore caches with a lightweight hardware support. In: Proceedings of the 2009 ACM/IEEE Conference on Supercomputing (2009)
57. Zhan, D., Jiang, H., Seth, S.C.: Locality & utility co-optimization for practical capacity management of shared last level caches. In: Proceedings of the 26th ACM International Conference on Supercomputing (2012)
58. Gonzalez-Ferez, P., Piernas, J., Cortes, T.: The ram enhanced disk cache project (redcap). In: Proceedings of the 24th IEEE Conference on Mass Storage Systems and Technologies (2007)
59. Huang, S., Wei, Q., Chen, J., Chen, C., Feng, D.: Improving flash-based disk cache with lazy adaptive replacement. In: 2013 IEEE 29th Symposium on Mass Storage Systems and Technologies (MSST) (2013)
60. Zhu, Z., Zhang, X.: Access-mode predictions for low-power cache design. IEEE Micro **22**(2) (2002)
61. Yue, J., Zhu, Y., Cai, Z., Lin, L.: Energy and thermal aware buffer cache replacement algorithm. In: Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST) (2010)
62. Manzanares, A., Ruan, X., Yin, S., Xie, J., Ding, Z., Tian, Y., Majors, J., Qin, X.: Energy efficient prefetching with buffer disks for cluster file systems. In: Proceedings of the 2010 39th International Conference on Parallel Processing (2010)
63. Li, Z., Wilson, C., Jiang, Z., Liu, Y., Zhao, B., Jin, C., Zhang, Z.L., Dai, Y.: Efficient batched synchronization in dropbox-like cloud storage services. In: Proceedings of the 14th International Middleware Conference (2013)
64. Xu, Y., Xing, C., Zhou, L.: A cache replacement algorithm in hierarchical storage of continuous media object. In: Advances in Web-Age Information Management: 5th International Conference (2004)
65. Li, R., Guo, R., Xu, Z., Feng, W.: A prefetching model based on access popularity for geospatial data in a cluster-based caching system. Int. J. Geogr. Inf. Sci. **26**(10) (2012)
66. Qiao, K., Tao, F., Zhang, L., Li, Z.: A ga maintained by binary heap and transitive reduction for addressing psp. In: 2010 International Conference on Intelligent Computing and Integrated Systems (ICISS) (2010)
67. Tao, F., Qiao, K., Zhang, L., Li, Z., Nee, A.: GA-BHTR: an improved genetic algorithm for partner selection in virtual manufacturing. Int. J. Prod. Res. **50**(8) (2012)
68. Calinescu, G., Qiao, K.: Asymmetric topology control: exact solutions and fast approximations. In: IEEE International Conference on Computer Communications (INFOCOM '12) (2012)
69. Calinescu, G., Kapoor, S., Qiao, K., Shin, J.: Stochastic strategic routing reduces attack effects. In: Global Telecommunications Conference (GLOBECOM 2011), 2011. IEEE (2011)
70. Zhao, D,, Yang, L.: Incremental isometric embedding of high-dimensional data using connected neighborhood graphs. IEEE Trans. Pattern Anal. Mach. Intell. **31**(1), 86–98 (2009)
71. Lohfert, R., Lu, J., Zhao, D.; Solving sql constraints by incremental translation to sat. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (2008)

72. Zhao, D., Yang, L.: Incremental construction of neighborhood graphs for nonlinear dimensionality reduction. In: Proceedings of 18th International Conference on Pattern Recognition, vol. 3, pp. 177–180 (2006)
73. Ferreira, K.B., Riesen, R., Arnold, D., Ibtesham, D., Brightwell, R.: The viability of using compression to decrease message log sizes. In: Proceedings of International Conference on Parallel Processing Workshops (2013)
74. Zerin Islam, T., Mohror, K., Bagchi, S., Moody, A., de Supinski, B.R., Eigenmann, R.: McrEngine: a scalable checkpointing system using data-aware aggregation and compression. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC) (2012)
75. Slim Bouguerra, M., Gainaru, A., Gomez, L.B., Cappello, F., Matsuoka, S., Maruyam, N.: Improving the computing efficiency of hpc systems using a combination of proactive and preventive checkpointing. In: IEEE International Symposium on Parallel Distributed Processing (2013)
76. Noeth, M., Marathe, J., Mueller, F., Schulz, M., de Supinski, B.: Scalable compression and replay of communication traces in massively parallel environments. In: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing (SC) (2006)
77. Laney, D., Langer, S., Weber, C., Lindstrom, P., Wegener, A.: Assessing the effects of data compression in simulations using physically motivated metrics. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (2013)
78. Lakshminarasimhan, S., Jenkins, J., Arkatkar, I., Gong, Z., Kolla, H., Ku, S.-H., Ethier, S., Chen, J., Chang, C.S., Klasky, S., Latham, R., Ross, R., Samatova, N.F.: ISABELA-QA: query-driven analytics with ISABELA-compressed extreme-scale scientific data. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11) (2011)
79. MPEG-1: http://en.wikipedia.org/wiki/MPEG-1. Accessed 5 Sept 2014
80. Bicer, T., Yin, J., Chiu, D., Agrawal, G., Schuchardt, K.: Integrating online compression to accelerate large-scale data analytics applications. In: Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing (IPDPS) (2013)
81. Schendel, E.R., Pendse, S.V., Jenkins, J., Boyuka, D.A., II, Gong, Z., Lakshminarasimhan, S., Liu, Q., Kolla, H., Chen, J., Klasky, S.,Ross, R., Samatova, N.F.: Isobar hybrid compression-i/o interleaving for large-scale parallel i/o optimization, In: Proceedings of International Symposium on High-Performance Parallel and Distributed Computing (2012)
82. Jenkins, J., Schendel, E.R., Lakshminarasimhan, S., Boyuka, D.S., II, Rogers, T., Ethier, S., Ross, R., Klasky, S., Samatova, N.F.: Byte-precision level of detail processing for variable precision analytics. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC) (2012)
83. Burrows, M., Jerian, C., Lampson, B., Mann, T.: On-line data compression in a log-structured file system. In: Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) (1992)
84. Joshua, P.: MacDonald. File system support for delta compression. Technical report, University of California, Berkley (2000)
85. Olson, M.A., Bostic, K., Seltzer M.: db. In: Proceedings of the Annual Conference on USENIX Annual Technical Conference (1999)
86. Edel, N.K., Tuteja, D., Miller, E.L., Brandt S.A.: Mramfs: a compressing file system for non-volatile ram. In: Proceedings of the the IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS) (2004)
87. Muthitacharoen, A., Chen, B., Mazières, D.: A low-bandwidth network file system. In: Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles (SOSP) (2001)
88. Park, K.S., Ihm, S., Bowman, M., Pai, V.S.: Supporting practical content-addressable caching with czip compression. In: 2007 USENIX Annual Technical Conference (2007)

89. Meister, D., Brinkmann, A., Süß, T.: File recipe compression in data deduplication systems. In: Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST) (2013)
90. Lakshminarasimhan, S., Boyuka, D.A., Pendse, S.V., Zou, X., Jenkins, J., Vishwanath, V., Papka, M.E., Samatova, N.F.: Scalable in situ scientific data encoding for analytical query processing. In: Proceedings of the 22nd International Symposium on High-performance Parallel and Distributed Computing (HPDC) (2013)
91. Gong, Z., Lakshminarasimhan, S., Jenkins, J., Kolla, H., Ethier, S., Chen, J., Ross, R., Klasky, S., Samatova, N.F.: Multi-level layout optimization for efficient spatio-temporal queries on isabela-compressed data. In: Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium (IPDPS) (2012)
92. Shnaiderman, L., Shmueli, O.: A parallel twig join algorithm for XML processing using a GPGPU. In: International Workshop on Accelerating Data Management Systems Using Modern Processor and Storage Architectures (2012)
93. Wang, H., Potluri, S., Bureddy, D., Rosales, C., Panda, D.K.: Gpu-aware mpi on rdma-enabled clusters: design, implementation and evaluation. IEEE Trans. Parallel Distrib. Syst. **25**(10) (2014)
94. Bordawekar, R., Bondhugula, U., Rao. R.: Believe it or not!: mult-core cpus can match gpu performance for a flop-intensive application! In: Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques, PACT '10, (2010)
95. Farooqui, N., Schwan, K., Yalamanchili, S.: Efficient instrumentation of gpgpu applications using information flow analysis and symbolic execution. In: Proceedings of Workshop on General Purpose Processing Using GPUs, GPGPU-7 (2014)
96. Muniswamy-Reddy, K.-K.: Foundations for provenance-aware systems (2010)
97. Foster, I.T., Vckler, J.S., Wilde, M., Zhao, Y.: The virtual data grid: a new model and architecture for data-intensive collaboration. In: CIDR'03 (2003)
98. Provenance aware service oriented architecture. http://twiki.pasoa.ecs.soton.ac.uk/bin/view/PASOA/WebHome. Accessed 6 July 2015
99. Parker-Wood, A., Long, D.D.E., Miller, E.L., Seltzer, M., Tunkelang, D.: Making sense of file systems through provenance and rich metadata. Technical Report UCSC-SSRC-12-01, University of California, Santa Cruz, March 2012
100. Muniswamy-Reddy, K.-K., Holland, D.A., Braun, U., Seltzer, M.: Provenance-aware storage systems. In: Proceedings of the annual conference on USENIX '06 Annual Technical Conference (2006)
101. Muniswamy-Reddy, K.-K., Macko, P., Seltzer, M.: Making a cloud provenance-aware. In: 1st Workshop on the Theory and Practice of Provenance (2009)
102. Muniswamy-Reddy, K.-K., Braun, U., Holland, D.A., Macko, P., Maclean, D., Margo, D., Seltzer, M., Smogor, R.: Layering in provenance systems. In: Proceedings of the 2009 USENIX Annual Technical Conference (2009)
103. Gehani, A., Tariq, D.: SPADE: support for provenance auditing in distributed environments. In: Proceedings of the 13th International Middleware Conference (2012)
104. Zhou, W., Sherr, M., Tao, T., Li, X., Thau Loo, B., Mao, Y.: Efficient querying and maintenance of network provenance at internet-scale. In: Proceedings of the 2010 International Conference on Management of Data, pp. 615–626 (2010)
105. Abraham, J., Brazier, P., Chebotko, A., Navarro, J., Piazza, A.: Distributed storage and querying techniques for a semantic web of scientific workflow provenance. In: 2010 IEEE International Conference on Services Computing (SCC), pp. 178–185. IEEE (2010)
106. Malik, T., Gehani, A., Tariq, D., Zaffar, F.: Sketching distributed data provenance. In: Data Provenance and Data Management in eScience, pp. 85–107 (2013)
107. Heinis, T., Alonso, G.: Efficient lineage tracking for scientific workflows. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1007–1018 (2008)
108. Extensible Markup Language (XML): http://www.w3.org/xml/. Accessed 13 Dec 2014
109. JSON: http://www.json.org/. Accessed 8 Dec 2014

110. Binary JSON: http://bsonspec.org/. Accessed 13 Dec 2014
111. Apache Thrift: https://thrift.apache.org/. Accessed 8 Dec 2014
112. Apache Avro: http://avro.apache.org/. Accessed 13 Dec 2014
113. Apache Etch: https://etch.apache.org/. Accessed 13 Dec 2014
114. BERT: http://bert-rpc.org/. Accessed 13 Dec 2014
115. Message Pack: http://msgpack.org/. Accessed 13 Dec 2014
116. Hessian: http://hessian.caucho.com/. Accessed 13 Dec 2014
117. ICE: http://doc.zeroc.com/display/ice34/home. Accessed 13 Dec 2014
118. CBOR: http://cbor.io/. Accessed 13 Dec 2014
119. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: Proceedings of USENIX Symposium on Opearting Systems Design & Implementation (2004)
120. Apache Hadoop: http://hadoop.apache.org/. Accessed 5 Sept 2014
121. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing (2010)
122. MPICH: http://www.mpich.org/. Accessed 10 Dec 2014
123. Open MPI: http://www.open-mpi.org/. Accessed 10 Dec 2014
124. OpenMP: http://openmp.org/wp/. Accessed 9 Dec 2014
125. PPL: http://msdn.microsoft.com/en-us/library/dd492418.aspx. Accessed 13 Dec 2014
126. Jeon, M., He, Y., Elnikety, S., Cox, A.L., Rixner, S.: Adaptive parallelism for web search. In: Proceedings of the 8th ACM European Conference on Computer Systems, EuroSys '13 (2013)
127. Jeon, M., Kim, S., Hwang, S., He, Y., Elnikety, S., Cox, A.L., Rixner, S.: Predictive parallelization: taming tail latencies in web search. In: Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR '14 (2014)
128. Lee, J., Winslett, M., Ma, X., Yu, S.: Enhancing data migration performance via parallel data compression. In: Proceedings of the 16th International Parallel and Distributed Processing Symposium, IPDPS '02 (2002)
129. Klasky, S., Ethier, S., Lin, Z., Martins, K., McCune, D., Samtaney, R.: Grid-based parallel data streaming implemented for the gyrokinetic toroidal code. In: Proceedings of the 2003 ACM/IEEE Conference on Supercomputing, SC '03 (2003)
130. Warneke, D., Kao, O.: Nephele: efficient parallel data processing in the cloud. In: Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers, MTAGS '09 (2009)
131. Yu, Y., Isard, M., Fetterly, D., Budiu, M., Erlingsson, U., Gunda, P.K., Currey, J.: Dryadlinq: a system for general-purpose distributed data-parallel computing using a high-level language. In: Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation, OSDI'08 (2008)
132. Ronnie, C., Bob, J., Per-Ake, L., Bill, R., Darren, S., Simon, W., Jingren, Z.: Scope: easy and efficient parallel processing of massive data sets. Proc. VLDB Endow. **1**(2), 1265–1276 (2008)
133. Ahrens, J., Brislawn, K., Martin, K., Geveci, B., Charles Law, C., Papka, M.: Large-scale data visualization using parallel data streaming. In: Computer Graphics and Applications. IEEE, **21**(4), July 2001
134. Allen, M.D., Sridharan, S., Sohi, G.S.: Serialization sets: a dynamic dependence-based parallel execution model. In: Proceedings of the 14th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP '09 (2009)
135. Voss, M., Eigenmann, R.: Reducing parallel overheads through dynamic serialization. In: Proceedings of the 13th International Symposium on Parallel Processing and the 10th Symposium on Parallel and Distributed Processing, IPPS '99/SPDP '99 (1999)

Chapter 9
Information Fusion for Improving Decision-Making in Big Data Applications

Nayat Sanchez-Pi, Luis Martí, José Manuel Molina and Ana C. Bicharra García

9.1 Introduction

Nowadays, occupational health and safety (OHS) compliance frameworks must rely upon information technology to be able to cope with the complexity of designing effective actions to prevent accidents in any industrial domains.

OHS is a priority concern for offshore oil and gas industry and a determining factor in its overall success. This is because successful offshore operations must take into account the implications of oil industry to health, safety, and the environment. Therefore, there are important investment efforts directed toward accident prevention.

Although the "learning from previous accidents" seems a mantra, the actual drilling down old episodes to find the reasons that permeate the facts represent a challenge. Intelligent information system technology may be used to help companies and employees to assess the risks related to places, activities, and even actions in a working environment. A dynamic risk picture for accident detection and recognition involves reasoning on contextual information such as past events, work environment information, work force's behavior, activities' requirements and premises and task goals. The situation assessment or situation awareness (SA) is a key component of any intelligent information system to support OHS decision-makers [1].

N. Sanchez-Pi (✉)
Institute of Mathematics and Statistics, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, Brazil
e-mail: nayat@ime.uerj.br

L. Martí · A.C. Bicharra García
Institute of Computing, Universidade Federal Fluminense, Rio de Janeiro, Brazil
e-mail: lmarti@ic.uff.br

A.C. Bicharra García
e-mail: bicharra@ic.uff.br

J.M. Molina
Computer Science Department, Universidad Carlos III de Madrid, Leganes, Spain
e-mail: molina@ia.uc3m.es

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_9

Traditional approaches using observational data and a priori models are insufficient to deal with real-world complex problems, so there is a need of Big Data techniques to deal with these approaches.

Historical data represents a key source of knowledge that must be analyzed to find interesting patterns. For this reason, data mining is the an essential part of the knowledge discovery process which combines databases, artificial intelligence, machine-learning, and statistics techniques. The basic techniques for data mining include: decision trees and forests induction, rules induction, artificial neural networks, clustering and association rules, among many other. Data mining can be applied to any domain where sufficiently large databases are available. Some applications areas are: failure prediction [2], biomedical applications [3], process and quality control [4], to name a few.

Association rule learning [5] is a popular and well-understood approach for discovering interesting relations between entities in a large databases. It is intended to discover rules that reflects strong associations among data in databases using different distance measurements such as novelty and lift. Many algorithms for generating association rules have been proposed over time as improvements of the classic Apriori algorithm [6], Eclat [7], and FP-Growth [8].

In this work, we describe an approach to deal with the OHS problem involving high-level information fusion, domain ontology and association rule data mining technique. Our proposed architecture provides an integrative picture about how to integrate historical data in risk analysis that has to do with an employee given its activities, profile, location, and time in a real-world environment. We have named this process as cognitive analysis. In particular, we build a causality model for accidents investigation by means of a well-defined spatiotemporal constraints on offshore oil industry domain.

The paper is organized as follows. Section 9.2 provides an introduction to the OHS problem and the role of information fusion processes, along with a brief description of the state of the art and some application domains. Section 9.3 describes the proposed fusion architecture, the knowledge representation model, and the cognitive analysis framework. Subsequently, Sect. 9.4 is centered in the knowledge extraction process; which is then applied in the case study discussed in Sect. 9.5. Finally, Sect. 9.6 puts forward some conclusive remarks and outlines the current and future work been carried out in this area.

9.2 Foundations

Data fusion has been defined in [9] as:

> "A multi-level process dealing with the association, correlation, combination of data and information from single and multiple sources to achieve refined position, identify estimates and complete and timely assessments of situations, threats and their significance."

Data fusion (DF) and information fusion (IF) has been treated more or less homogeneously in the literature but there are some conceptual differences between them. Data fusion is represents and deals with raw data, and, on the other hand, when referring to information fusion it implies a higher semantic level of fusion. The problem of information fusion has attracted significant attention in the artificial intelligence community, trying to innovate in the techniques used for combining the data and to refine state estimates and predictions.

Information fusion can be classified depending on the level of abstraction [10]: low-level fusion, medium-level fusion, high-level fusion, and multilevel fusion. In low-level fusion, raw data are directly provided as input to the data fusion process. The medium-level fusion is a feature level where features are fused to obtain other features that could be employed for other higher level tasks. In high-level fusion the input of the fusion process is a combination of symbolic representation. Finally, in multi-level fusion, the input comes from different levels of abstractions.

Others classifications or taxonomies of information fusion have been proposed. For example, the Dasarathy's Functional Model [11] or Joint Directors of Laboratories conceptual model (JDL) proposed by the American Department of Defense [9]. The JDL classification model consists into five processing levels in the transformation of input signals to decision-ready knowledge. These levels are: level 0 or source preprocessing; level 1 or object refinement; level 2 or situation assessment; level 3 or impact assessment, and level 4 or process refinement.

High-level fusion starts at level 2. Situation assessment (SA) aims to identify the most likely situations given the observed events and obtained data. It establishes relationships between the objects. Relations (i.e., proximity, communication, etc.) are evaluated to determine the significance of the entities or objects in a specific environment. The purpose of this level includes performing high-level inferences and identifying significant activities and events (patterns in general). The output is a set of high-level inferences. Situation assessment is an important part of the information fusion process because it is the purpose for the use of IF to synthesize the multitude of information, it provides an interface between the user and the automation, and focuses data collection and management.

Intensive research has been done in past years focused on low-level information fusion, nowadays the focus is currently shifting towards high-level information fusion [12]. Compared to the increasingly mature field of low-level IF, theoretical and practical challenges posed by high-level IF are more difficult to handle. Some of the applications that involve high-level fusion are: Defense [13–17], Computer and Information Security [18, 19], Disaster Management [20–23], Fault Detection [24–26], Environment [27–29]. Also techniques for using contextual information in high-level information fusion architectures has been studied at [30].

In the context of oil and gas industry there is an increasing concern with achieving and demonstrating good performance with regards to occupational health and safety (OHS), through the control of its OHS risks, which is consistent with its policy and objectives. In oil industry exist standards to identify and record workplace accidents and incidents to provide guiding means on prevention efforts, indicating specific failures or reference, means of correction of conditions or circumstances that cul-

minated in accident. So, events recognition is central to OHS, since the system can selectively start proper prediction services according to the user current situation and past knowledge taken from huge databases. In this sense, a fusion framework that combines data from multiples sources to achieve more specific inferences is needed [31, 32].

In fact, our proposal is inspired in the semantic strategy of Gomez et al. [30]. In our case, we propose a machine-learning algorithm to learn from past anomaly events and to predict accidents events in time and space. It also uses additional knowledge, like the contextual knowledge: user profile, event location and time, etc. Our proposed model provides the big picture about risk analysis for that employee, at that place, in that moment in a real-world environment. Our contribution is to build a causality model for accidents investigation by means of a well-defined spatiotemporal constraints on offshore oil industry domain. Also, we use ontological constraints in the postprocessing mining stage to prune resulting rules.

9.3 A Big Data Solution: A Data Fusion Framework for OHS

We now focus on the general architecture of the framework being put forward in this work. We outline the different components and how they are integrated.

9.3.1 Data Fusion Architecture for OHS Environments

The architecture of our context-based fusion framework is depicted in Fig. 9.1. The context-aware system developed has a hierarchical architecture with the following layers: Services layer, Context Acquisition layer, Context Representation layer, Context Information Fusion layer, and Infrastructure layer. The hierarchical architecture reflects the complex functionality of the system as shown in the following brief description of the functionality of particular layers:

- *Infrastructure Layer* The lowest level (level 0 in the JDL Model) of the location management architecture is the Sensor Layer which represents the variety of physical and logical location sensor agents producing sensor-specific location information.
- *Context Acquisition* The link between sensors (lowest layer) and the representation layer (level 1 in the JDL Model).
- *Context Representation* This is where the situation is represented by means of an ontology (level 2 in the JDL Model).
- *Context Information Fusion Layer* This is where the high-level information fusion occurs. It is here where reasoning about context and events of the past takes place (level 3 in JDL Model).

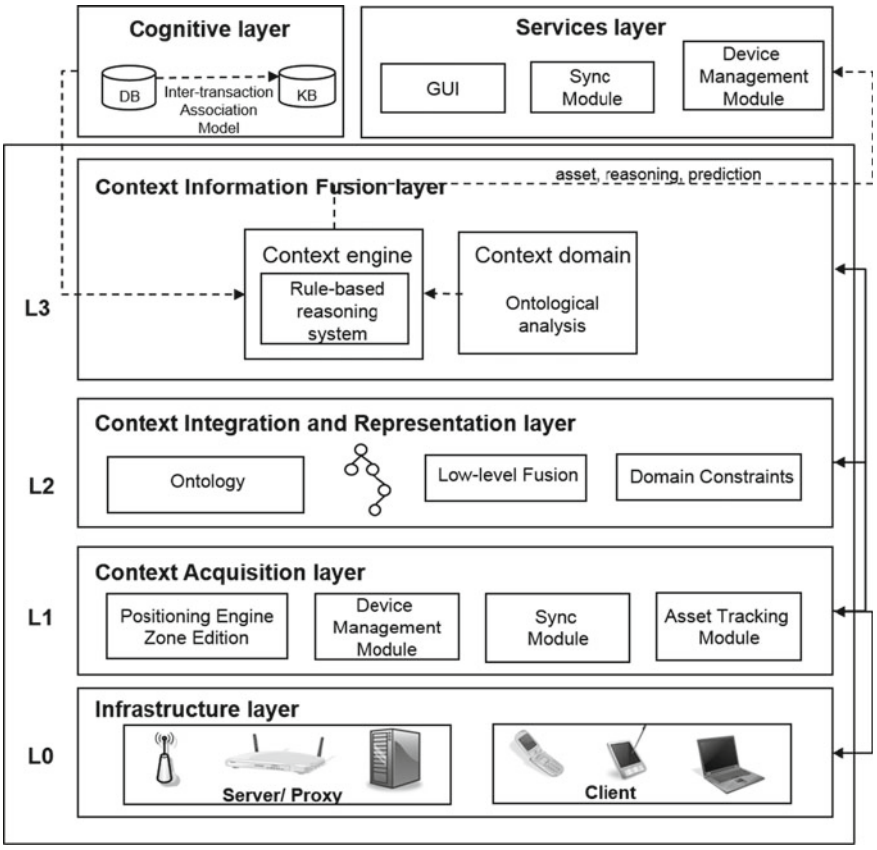


Fig. 9.1 Information fusion framework architecture

- *Cognitive Layer* This is where the Big Data techniques are applied and the intra- and inter-transaction association rules are extracted from the database.
- *Services Layer* This layer interacts with the variety of users of the system (employees) and therefore needs to address several issues including access rights to location information (who can access the information and to what degree of accuracy), privacy of location information (how the location information can be used), and security of interactions between users and the system.

9.3.2 Ontology Definition

Normally, ontology represents a conceptualization of particular domains. In our case, we will use the ontology for representing the contextual information of the offshore

oil industry environment. Ontologies are particularly suitable to project parts of the information describing and being used in our daily life onto a data structure usable by computers.

Using ontologies provides an uniform way for specifying the model's core concepts as well as an arbitrary amount of subconcepts and facts, altogether enabling contextual knowledge.

An ontology can be defined as an explicit specification of a conceptualization [33]. An ontology created for a given domain includes a set of concepts as well as relationships connecting them within the domain. Collectively, the concepts and the relationships form a foundation for reasoning about the domain.

A comprehensive, well-populated ontology with classes and relationships closely modeling a specific domain represents a vast compendium of knowledge in the domain.

Furthermore, if the concepts in the ontology are organized into hierarchies of higher level categories, it should be possible to identify the category (or a few categories) that best classify the context of the user.

Within the context of computer science, ontological concepts are frequently regarded as classes that are organized into hierarchies. The classes define the types of attributes, or properties common to individual objects within the class. Moreover, classes are interconnected by relationships, indicating their semantic interdependence (relationships are also regarded as attributes).

As part of our framework we built a domain ontology for the OHS environment of oil and gas domain [34] (see Fig. 9.2). We also obtained the inferences that reflect the dynamic side and we grouped the inferences sequentially to form tasks.

The principal concepts of the ontology are the following:

- *Anomaly* Undesirable event or situation which results or may result in damage or faults that affect people, the environment, equity (own or third party), the image of the industry partner, products, or production processes. This concept includes accidents, illnesses, incidents, deviations and non-conformances.
  - *Neglect* Any action or condition that has the potential to lead to, directly or indirectly, damage to people, to property (own or third party) or environmental impact, which is inconsistent with labor standards, procedures, legal or regulatory requirements, requirements management system or practice.
    Behavioral neglect Act or omission which, contrary provision of security, may cause or contribute to the occurrence of accidents.
    Non-behavioral neglect Environmental condition that can cause an accident or contribute to its occurrence. The environment includes adjective here, everything that relates to the environment, from the atmosphere of the workplace to the facilities, equipment, materials used and methods of working employees who is inconsistent with labor standards, procedures, legal requirements, or normative requirements of the management system or practice.
  - *Incident* Any evidence, personal occurrence or condition that relates to the environment and/or working conditions, can lead to damage to physical and/or mental.

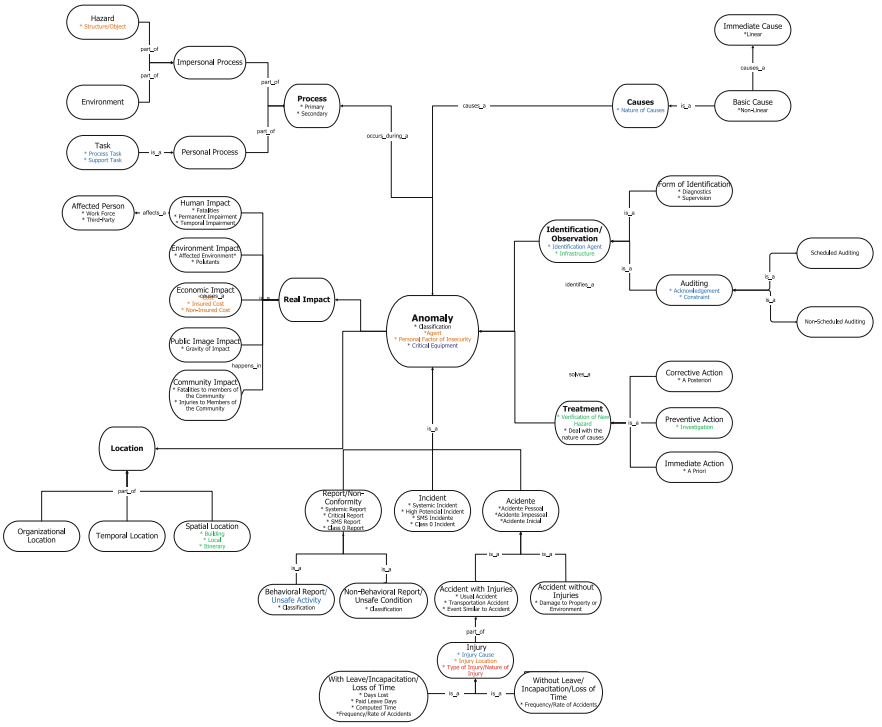


Fig. 9.2 Occupational Health and Security (OHS) ontology

– *Accident* Occurrence of unexpected and unwelcome, instant or otherwise, related to the exercise of the job, which results or may result in personal injury. The accident includes both events that may be identified in relation to a particular time or occurrences as continuous or intermittent exposure, which can only be identified in terms of time period probable. A personal injury includes both traumatic injuries and illnesses, as damaging effects mental, neurological, or systemic, resulting from exposures or circumstances prevailing at the year's work force. In the period for meal or rest, or upon satisfaction of other physiological needs at the workplace or during this, the employee is considered in carrying out the work.

  Accident with injury It is all an accident in which the employee suffers some kind of injury. Injury: Any damage suffered by a part of the human organism as a consequence of an accident at work.

  …with leave Personal injury that prevents the injured from returning to work the day after the accident or resulting in permanent disability. This injury can cause total permanent disability, permanent partial disability, total temporary disability, or death.

…without leave Personal injury that does not prevent the injured to return to work the day after the accident, since there is no permanent disability. This injury, not resulting in death, permanent total or partial disability or total temporary disability, requires, however, first aid or emergency medical aid. Expressions should be avoided "lost-time accident" and "accident without leave," used improperly to mean, respectively, "with leave injury" and "injury without leave."
Accident without injury Accident causes no personal injury.

9.3.3 Cognitive Analysis

Standard ontology reasoning procedures can be performed within the ontologies to infer additional knowledge from the explicitly asserted facts. Using an inference engine, tasks such as classification or instance checking can be performed. Figure 9.3 outlines how this process was carried out in our case.

Risk prevention is a paradigmatic case of inductive reasoning. Inductive reasoning begins with observations that are specific and limited in scope, and proceeds to a generalized conclusion that is likely, but not certain, in light of accumulated evidence. You could say that inductive reasoning moves from the specific to the general. Much scientific research is carried out by the inductive method: gathering evidence, seeking patterns, and forming a hypothesis or theory to explain what is seen.

In our framework, inductive rules formally represent contextual, heuristic, and common sense knowledge to accomplish high-level scene interpretation and low-level location refinement.

Once an employee enters the network, it immediately connects with a local proxy, which evaluates the position of the client device and assign a role to the employee. A preprocessing step begins then filtering the relevant features that are selected to

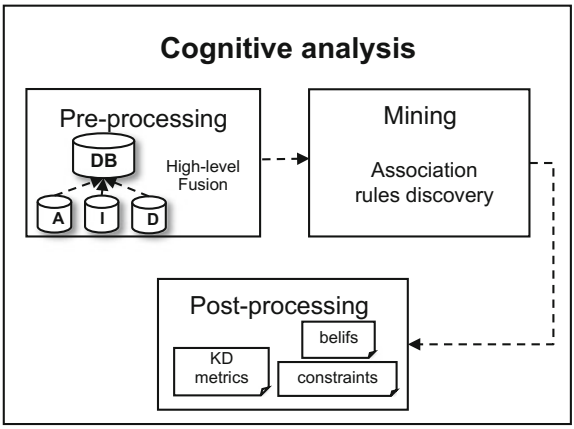


Fig. 9.3 Cognitive analysis

participate in the process of knowledge discovery by the type of employee (role). The association rules mining process starts with the selected configuration and the set of resulting rules can be analyzed. After that, a postprocessing step starts. This is an important component of consisting meant for eliminating redundancy, pruning, and filtering the resulting rules.

The fusion engine implements an association rules model that dynamically combines feature selection relying on the profile of the user in order to find spatiotemporal patterns between different types of anomalies (or event sequence, e.g., neglects, incidents, accidents) that match with the current location of the user.

In this case, the data preprocessing before mining is pretty straightforward, as the interest is to discover relationships between the values of different attributes and the possible presence of probabilistic implication rules between them. In particular, each anomaly is treated as a transaction whose items are the non-null values.

Two categories of association mining are employed: *intra-anomaly* and *inter-anomaly* [35]. Intra-anomaly associations are the associating among items within the same type of anomaly, where the notion of the transaction could be events where the same user participates. However, inter-anomaly describes relationships among different transactions; that means, between incidents, accidents and neglects. Further details about these two processes are given in the subsequent sections.

9.4 Mining Anomaly Information

As already explained, the task of providing context-based information calls for the processing and extraction of information in the form of rules. One of the possible ways of obtaining those rules is to apply an association rule algorithm. In this work, we employ Aprori and FP-Growth algorithms in parallel in order to mutually validate the results from each other.

As also explained in the above section, the fusion engine implements an association rules model that combines dynamic feature selection based on the role of the user in order to find spatiotemporal patterns between different types of anomalies (or event sequence, e.g., neglects, incidents, accidents) that match with the current location of the user.

The dataset of anomalies, $\mathscr{S}$, is composed by anomaly instances,

$$\mathscr{S} := \{A_1, A_2, \ldots, A_n\}, n \in \mathbb{N}, \tag{9.1}$$

with the instances defined as

Definition 1 (*Anomaly instance*) An anomaly instance can be defined as the tuple,

$$A := \langle t, c, \mathscr{L}, \mathscr{O}, \mathscr{N}, \mathscr{F} \rangle, \tag{9.2}$$

that is composed by:

- t, a time instant that marks when the anomaly took place;
- $c \in$ {accident, incident, neglect}, that sets the class of anomaly, and, therefore, its associated gravity;
- $\mathscr{L}$, a set of geo-location description attributes, which describe the geographical localization of the anomaly at different levels of accuracy;
- $\mathscr{O}$, a set of organizational location attributes that represent where in terms of organization structure the anomaly took place;
- $\mathscr{N}$, a set of descriptive nominal attributes that characterize the anomaly with a predefined values, and;
- $\mathscr{F}$, a set of free-text attributes that are used to complement or improve the descriptive power reachable with $\mathscr{N}$ attributes.

In order to make the rules produced interesting for the user the mining dataset, $\mathscr{S}$, must be preprocessed to meet the her/his needs. Using the above-described problem ontology, the set of anomalies relevant for mining can be (i) filtered and (ii) its attributed selected.

For the first task we defined a function

$$\texttt{filter\_anomalies}\,(u, \mathscr{S}) \rightarrow \mathscr{S}', \mathscr{S}' \subseteq \mathscr{S}, \tag{9.3}$$

which determines the subset, $\mathscr{S}'$, of the anomalies dataset, $\mathscr{S}$, that are of interest for a given user, u. For the second task we created the function

$$\texttt{filter\_attributes}\left(u, \mathscr{S}'\right) \rightarrow \mathscr{S}^*, \tag{9.4}$$

where $\forall A' =\in \mathscr{S}'$, $\exists A^* \in \mathscr{S}^*$ such that $t^* = t'$, $c^* = c'$, $\mathscr{L}^* \subseteq \mathscr{L}'$, $\mathscr{O}^* \subseteq \mathscr{O}'$, $\mathscr{N}^* \subseteq \mathscr{N}'$ and $\mathscr{F}^* \subseteq \mathscr{F}'$.

Relying on the $\mathscr{S}^*$ dataset customized to the user profile two classes of data mining operations can be carry out to extract knowledge rules. The first mines for rules regarding the relations of different attribute values in anomalies, and hence was called *intra-anomaly rule mining*. The other, more complex one, mines for relationships between anomalies, that take place in a same location—either geographical or organizational—and in similar dates. Because of that, this operation was denominated *spatiotemporal* or *inter-anomaly rule mining*. In the subsequent sections we describe both mining processes.

9.4.1 Mining for Intra-Anomaly Rules

In this case, the data preprocessing before mining is pretty straightforward, as the interest is to discover relationships between the values of different attributes and the possible presence of probabilistic implication rules between them. In particular, each anomaly in $\mathscr{S}^*$ is treated as a transaction whose items are the non-null values of the

corresponding $\mathscr{N}^*$. The descriptive attributes that take part of the mining process depend in the user profile. In order to model this we created a function the function

$$\mathscr{N}_{\text{sel}} = \text{select\_attributes}_{\text{intra}}\left(\mathscr{N}, u\right), \tag{9.5}$$

which returns the subset of attributes, $\mathscr{N}_{\text{sel}} \subseteq \mathscr{N}$, that are of interest to a given user, u.

The results of applying the rule mining algorithms need to be postprocessed to eliminate cyclic rules and to sort them according to an interestingness criterion. The outcome from this process should uncover relations between different values of the attributes. Some of those relationships might have a trivial.

9.4.2 Mining for Inter-Anomaly Rules

Mining spatiotemporal rules calls for a more complex preprocessing. As the most relevant anomalies are the accidents mining is centered around them. In this case, transactions will be constituted by anomalies that took place in the same location (deduced from the user profile) and with a given amount of time of precedence.

More formally, having the set of all accidents $\Lambda = \{A \in \mathscr{S}^* | A.c = \text{accident}\}$, for each element $\lambda \in \Lambda$, we construct the set of co-occurring anomalies, $\mathscr{C}(\lambda)$ as,

$$\begin{aligned}\mathscr{C}(\lambda) := &\left\{\kappa \in \mathscr{S}^* | \lambda.t - \kappa.t \leq \Delta t; \text{loc}(\lambda, u) = \text{loc}(\kappa, u)\right\} \\ &\cup \{\lambda\},\end{aligned} \tag{9.6}$$

with loc$(\cdot)$, a function that for a given anomaly and user returns the value of the location attribute of interest for that user according to her/his role, and Δt, a time interval for maximum co-occurrence.

The set of co-occurring anomalies $\{\mathscr{C}(\lambda) | \forall \lambda \in \Lambda\}$ is used as transactions dataset for the mining algorithms. However, anomalies cannot be used as-is, as it is necessary to express them in abstract form, in order to achieve sufficient generalization as to yield results that not are excessively particular or refined.

For this task, again depending on the user profile, a group of elements of each $\mathscr{N}^*$ is selected to create the abstract anomaly. This reduced set of attribute values are then used to construct the transactions. Therefore, as in the intra-anomaly case, we can construct a function

$$\mathscr{N}_{\text{sel}} = \text{select\_attributes}_{\text{inter}}\left(\mathscr{N}, u\right), \tag{9.7}$$

that having given a user, u, returns the subset of attributes, $\mathscr{N}_{\text{sel}} \subseteq \mathscr{N}$, that are of interest to u. Relying on $\mathscr{N}_{\text{sel}}$, the abstracted anomaly A_{abstract} as the concatenation of the attribute/value pairs,

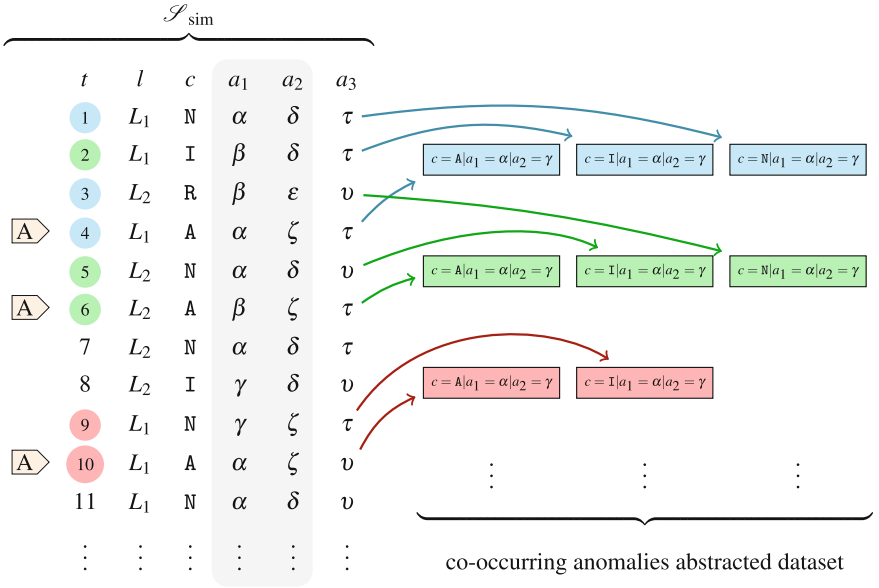


Fig. 9.4 Schematic representation of a co-occurrence and abstraction process example. The example is replies in a simplified anomalies dataset $\mathscr{S}_{\text{sim}}$ where, having a given user, u, the location attribute is $l = \text{loc}(\lambda, u), \forall \lambda \in \mathscr{S}_{\text{sim}}$ and the time of anomaly is denoted by t. For brevity reasons, the anomaly class is represented as $c = \{\text{A}, \text{I}, \text{N}\}$, for representing accidents, incidents, and neglects, respectively, and the set of descriptive nominal attributes $\mathscr{N} = \{a_1, a_2, a_3\}$. In this sample, there are three anomalies, which are marked with the A symbol. Assuming that $\mathscr{N}_{\text{sel}} = \{a_1, a_2\}$ (shaded in gray in the schema) and a $\Delta t = 2$, three transactions are created in the co-occurring anomalies dataset. This means that, for every accident, λ, $\lambda.c = \text{A}$ the anomalies that took place in the same location and with time in the interval $[\lambda.t - \Delta t, \lambda.t]$ are abstracted and added as a transaction to the mining dataset

$$A_{\text{abstract}} = \bigoplus_{a \in \mathscr{N}_{\text{sel}}} a \oplus A.a, \tag{9.8}$$

where $\oplus$ is the concatenation operator. As this concatenated representation is inefficient from a computational point of view, they can be transformed into a reduced form by applying a hashing [36] or a compression [37] operator.

This process is better understood with an illustrative example. Figure 9.4 puts forward such co-occurrence and abstraction process example. In this case, we have a simplified anomalies dataset $\mathscr{S}_{\text{sim}}$ where, having a given user, u, we also have the location attribute $l = \text{loc}(\lambda, u), \forall \lambda \in \mathscr{S}_{\text{sim}}$ and the time of anomaly is denoted by t. For brevity reasons in the figure, the anomaly class is represented as $c = \{\text{A}, \text{I}, \text{N}\}$, for representing accidents, incidents and neglects, respectively, and the set of descriptive nominal attributes $\mathscr{N} = \{a_1, a_2, a_3\}$.

In this sample, there are three anomalies, which are marked with the A symbol. Assuming that $\mathcal{N}_{\text{sel}} = \{a_1, a_2\}$ (shaded in gray) and a $\Delta t = 2$, three transactions are created in the co-occurring anomalies dataset. This is because of that, for every accident, λ, $\lambda.c = \mathbb{A}$ the anomalies that took place in the same location and with time in the interval $[\lambda.t - \Delta t, \lambda.t]$ are abstracted and added as a transaction to the mining dataset.

The resulting inter-anomaly mining dataset is composed by transactions that contain the abstracted version of the co-occurring anomalies for a given accident—or other class of anomaly of interest. As in the previous case, postprocessing is necessary to filter out possible irrelevant and/or cyclic rules. For this, a set of domain-principled filtering rules were proposed by the experts in order to define the most interesting consequences—accidents and incidents—and the preferred form of rules.

As this is part is a sensitive element of the solution, involving trade decisions, we are not discussing it in detail.

9.5 Case Study

In this section, we present a case study that was carried out with the intention of asserting from an experimental point of view, the viability of the solution put forward in this work.

In order to create a controlled context for the tests it is required to (i) select a subset of the fused dataset, which contains all available anomalies, of such a size that can be directly handled by an expert and with such properties that guaranties the existence of rules (ii) create a custom user role that when applied selects a group of features for intra-anomaly mining and other for inter-anomaly mining.

In order to obtain the data set, we applied a complex data query that filtered all accidents in a given time interval and their corresponding co-occurring anomalies. From that set, the accidents that did not have at least one more accident with the same abstracted co-occurring set were eliminated. This action produced about 2000 anomalies set in which it was certain that there were latent rules relating some of them.

As the amount of anomalies in the dataset is of a manageable size the application of data visualization and inspection of software, along with the use of basic statistical tools allow to uncover at least some of the rules that are latent in the dataset. Therefore, two sets of expected rules were manually extracted with the purpose of verify that the mining algorithms were capable of discovering rules known to exist beforehand.

In all experiments the threshold parameters of the rule mining algorithm were set as: support, 0.2, and confidence, 0.8.

Table 9.1 Similarity of the results produced by Apriori and FP-Growth in the intra- and inter-anomaly mining scenarios

	Number shared	Shared Apriori results (%)	Shared FP-growth results (%)
Intra-anomaly mining			
Freq. item sets	64	100.0000	90.1408
Rules	64	100.0000	84.2105
Inter-anomaly mining			
Freq. item sets	230	100.0000	100.0000
Rules	2670	100.0000	100.0000

9.5.1 Intra-Anomaly Rule Mining Results

The application of FP-Growth in the intra-anomaly case yielded 71 frequent sets of items and 76 rules. The Apriori algorithm, in the other hand, generated 64 frequent item sets and 64 rules. An important analysis is to compared at what degree the frequent itemsets and rules generated by each approach overlaps the other. This can be posed as counting the number of itemsets and rules that have been generated by both methods. This comparison is presented in Table 9.1. There it can be perceived that all itemsets and rules discovered by the Apriori method were also found by FP-Growth. This fact is a fundamental step to assert the validity of results.

Using the semi-automatic method explained above six rules extracted with the semi-automatic procedure. There rules were found to five of those rules were detected by Apriori, while only one by FP-Growth.

9.5.2 Inter-Anomaly Rule Mining Results

After carrying out the process of abstraction and anomaly co-occurrence, grouping a dataset with 1025 transactions was passed to the rule mining algorithms. Similarly, in that dataset, six rules were extracted by the semi-automatic method.

The results of both algorithms in this case are interesting. Table 9.1 show that Apriori and FP-Growth found the same number of frequent itemsets, 230, and of rules (before filtering), 2670. Again, this results validate the approach proposed.

When determining how many manually extracted rules were actually found by the algorithms, the results are also encouraging. All six rules were found by both algorithms as demonstrated in Table 9.2.

Table 9.2 Presence of the rules previously extracted by a semi-automatic procedure in the results of Apriori and FP-Growth algorithms

	Apriori results	FP-growth results
Intra-anomaly mining		
Expected rules	5	1
Coverage of expected rules	62.5000 %	12.5000 %
Percent true positives	7.8125 %	≈ 0.0000 %
Percent non-expected rules	92.1875 %	100.0000 %
Inter-anomaly mining		
Expected rules	6	6
Coverage of expected rules	100.0000 %	100.0000 %
Percent true positives	0.2247 %	0.2247 %
Percent non-expected rules	99.7753 %	99.7753 %

9.6 Final Remarks

In this work, we have presented a Big Data solution based on an information fusion framework for providing context-aware services related to risk prevention in offshore oil industry environment. The proposal put forward aims at providing context-based information related to accidents and their causes to users depending on their profiles and location.

Our approach relies on a domain ontology to capture the relevant concepts of the application and the semantics of the context in order to create a high-level fusion of information. Along with that we have introduced an innovative use of rule mining for provisioning knowledge for situation assessment and decision-making regarding risk an accidents prevention. This form of rule mining is capable of an online high-level knowledge extraction that represents relations between different kinds of anomalies that have taken place at the user location and that the system has determined that had lead to an accident.

This feature has the potential of lowering at great length the development of accidents and incidents as it allows the users to directly act on the causes and conditions that have prompted such situations in the past. It empowers the users with the tools that help them to modify their routine and to avoid possible hazards or dangers.

This work is of particular relevance when taking into account the significant human, social, economical, and environmental impact of accidents in this application context. From human and social points of view, the class of application described here is important as the remoteness and isolation of the installations render any assisting action more complicated and risky than usual. Similarly, oil industry is a heavily cost-minded industry, where accidents trend to have a important economical repercussions derived from the stop of production and the cost of the equipment and repair activities. Last, but certainly not least, the dramatic environmental impact of this industry has been sadly verified in the last years. Accidents, in the form of oil spills

and fires are one of the main risks and one of the main dangers perceived by society regarding this industry. The environmental issue implies damages that are frequently impossible to assess in quantitative terms and whose footprint can potentially remain latent for future generations. It also has human, social, and economic ramifications that fall in the above-mentioned areas.

Acknowledgments This work was partially funded by CNPq PVE 314017/2013-5, FAPERJ APQ1 Project 211.500/2015, FAPERJ APQ1 Project 211.451/2015 and by projects MINECO TEC2012-37832-C02-01, CICYT TEC2011-28626-C02-02.

References

1. Endsley, M.R.: Toward a theory of situation awareness in dynamic systems. Human Factors: J. Human Factors Ergon. Soc. **37**(1), 32–64 (1995)
2. Borrajo, M.L., Baruque, B., Corchado, E., Bajo, J., Corchado, J.M.: Hybrid neural intelligent system to predict business failure in small-to-medium-size enterprises. Int. J. Neural Syst. **21**(04), 277–296 (2011)
3. De Paz, J.F., Bajo, J., López, V.F., Corchado, J.M.: Biomedic organizations: an intelligent dynamic architecture for KDD. Inf. Sci. **224**, 49–61 (2013)
4. Conti, M., Pietro, R.D., Mancini, L.V., Mei, A.: Distributed data source verification in wireless sensor networks. Inf. Fusion **10**(4), 342–353 (2009)
5. Piatetsky-Shapiro, G.: Discovery, analysis and presentation of strong rules. In: Piatetsky-Shapiro G., Frawley W.J. (eds.) Knowledge Discovery in Databases, pp. 229–248. AAAI Press (1991)
6. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994, pp. 487–499. http://dl.acm.org/citation.cfm?id=645920.672836
7. Zaki, M.J.: Scalable algorithms for association mining. IEEE Trans. Knowl. Data Eng. **12**(3), 372–390 (2000)
8. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM SIGMOD Record, vol. 29, pp. 1–12. ACM (2000)
9. White, F.E.: Data Fusion Lexicon. Tech. Rep, DTIC Document (1991)
10. Luo, R.C., Chou, Y.C., Chen, O.: Multisensor fusion and integration: algorithms, applications, and future research directions. In: International Conference on Mechatronics and Automation, 2007. ICMA 2007, pp. 1986–1991. IEEE (2007)
11. Dasarathy, B.V.: Sensor fusion potential exploitation-innovative architectures and illustrative applications. Proc. IEEE **85**(1), 24–38 (1997)
12. Blasch, E., Llinas, J., Lambert, D., Valin, P., Das, S., Chong, C., Kokar, M., Shahbazian, E.: High level information fusion developments, issues, and grand challenges: fusion 2010 panel discussion. In: 2010 13th Conference on Information Fusion (FUSION), pp. 1–8. IEEE (2010)
13. Chong, C.-Y., Liggins, M., et al.: Fusion technologies for drug interdiction. In: IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI'94), pp. 435–441. IEEE (1994)
14. Gad, A., Farooq, M.: Data fusion architecture for maritime surveillance. In: Proceedings of the Fifth International Conference on Information Fusion (FUSION'02), vol. 1, pp. 448–455. IEEE (2002)
15. Liggins, M.E., Bramson, A., et al.: Off-board augmented fusion for improved target detection and track. In: 1993 Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers, pp. 295–299. IEEE (1993)

16. Ahlberg, S., Hörling, P., Johansson, K., Jöred, K., Kjellström, H., Mårtenson, C., Neider, G., Schubert, J., Svenson, P., Svensson, P., et al.: An information fusion demonstrator for tactical intelligence processing in network-based defense. Inf. Fusion **8**(1), 84–107 (2007)
17. Aldinger, T., Kao, J.: Data fusion and theater undersea warfare-an oceanographer's perspective. In: OCEANS'04. MTTS/IEEE TECHNO-OCEAN'04, vol. 4, pp. 2008–2012. IEEE (2004)
18. Corona, I., Giacinto, G., Mazzariello, C., Roli, F., Sansone, C.: Information fusion for computer security: State of the art and open issues. Inf. Fusion **10**(4), 274–284 (2009)
19. Giacinto, G., Roli, F., Sansone, C.: Information fusion in computer security. Inf. Fusion **10**(4), 272–273 (2009)
20. Little, E.G., Rogova, G.L.: Ontology meta-model for building a situational picture of catastrophic events. In: 8th International Conference on Information Fusion (FUSION'05), vol. 1, pp. 1–8. IEEE (2005)
21. Llinas, J.: Information fusion for natural and man-made disasters. In: Proceedings of the Fifth International Conference on Information Fusion (FUSION'02), vol. 1, pp. 570–576. IEEE (2002)
22. Llinas, J., Moskal, M., McMahon, T.: Information fusion for nuclear, chemical, biological & radiological (NCBR) battle management support/disaster response management support. Tech. Rep., Center for MultiSource Information Fusion, School of Engineering and Applied Sciences, University of Buffalo, USA (2002)
23. Mattioli, J., Museux, N., Hemaissia, M., Laudy, C.: A crisis response situation model. In: 10th International Conference on Information Fusion (FUSION'07), pp. 1–7. IEEE (2007)
24. Bashi, A.: Fault detection for systems with multiple unknown modes and similar units. Ph.D. Thesis, University of New Orleans (2010)
25. Bashi, A., Jilkov, V.P., Li, X.R.: Fault detection for systems with multiple unknown modes and similar units-part i. In: 12th International Conference on Information Fusion (FUSION'09), pp. 732–739. IEEE (2009)
26. Basir, O., Yuan, X.: Engine fault diagnosis based on multi-sensor information fusion using dempster-shafer evidence theory. Inf. Fusion **8**(4), 379–386 (2007)
27. Heiden, U., Segl, K., Roessner, S., Kaufmann, H.: Ecological evaluation of urban biotope types using airborne hyperspectral hymap data. In: 2nd GRSS/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas, pp. 18–22. IEEE (2003)
28. Khalil, A., Gill, M.K., McKee, M.: New applications for information fusion and soil moisture forecasting. In: 8th International Conference on Information Fusion (FUSION'05), vol. 2, p. 7. IEEE (2005)
29. Hubert-Moy, L., Corgne, S., Mercier, G., Solaiman, B.: Land use and land cover change prediction with the theory of evidence: a case study in an intensive agricultural region of france. In: Proceedings of the Fifth International Conference on Information Fusion (FUSION'02), vol. 1, pp. 114–121. IEEE (2002)
30. Gómez-Romero, J., Garcia, J., Kandefer, M., Llinas, J., Molina, J., Patricio, M., Prentice, M., Shapiro, S.: Strategies and techniques for use and exploitation of contextual information in high-level fusion architectures. In: 2010 13th Conference on Information Fusion (FUSION), pp. 1–8. IEEE (2010)
31. Sanchez-Pi, N., Martí, L., Molina, J.M., Garcia, A.C.B.: High-level information fusion for risk and accidents prevention in pervasive oil industry environments. In: Highlights of Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection, pp. 202–213. Springer (2014)
32. Sanchez-Pi, N., Martí, L., Molina, J.M., Garcia, A.C.B.: An information fusion framework for context-based accidents prevention. In: 2014 Proceedings of the 17th International Conference on Information Fusion (FUSION), pp. 1–8. IEEE (2014)
33. Gómez-Romero, J., Patricio, M.A., García, J., Molina, J.M.: Ontological representation of context knowledge for visual data fusion. In: 12th International Conference on Information Fusion (FUSION'09), pp. 2136–2143. IEEE (2009)
34. Sanchez-Pi, N., Martí, L., Bicharra Garcia, A.C.: Text classification techniques in oil industry applications. In: Herrero A., Baruque B., Klett F., Abraham A., Snášel V., Carvalho A.C.,

García Bringas P., Zelinka I., Quintián H., Corchado E. (eds.) International Joint Conference SOCO'13-CISIS'13-ICEUTE'13, vol. 239 of Advances in Intelligent Systems and Computing, pp. 211–220. Springer International Publishing (2014). http://dx.doi.org/10.1007/978-3-319-01854-6_22
35. Berberidis, C., Angelis, L., Vlahavas, I.: Inter-transaction association rules mining for rare events prediction. In: Proceedings 3rd Hellenic Conference on Artificial Intelligence (2004)
36. Tharp, A.L.: File organization and processing. Wiley (1988)
37. Sayood, K.: Introduction to Data Compression, 2nd edn. Morgan Kaufmann Publishers Inc., San Francisco (2000)

Chapter 10
Load Balancing and Fault Tolerance Mechanisms for Scalable and Reliable Big Data Analytics

Nitin Sukhija, Alessandro Morari and Ioana Banicescu

10.1 Introduction

In the past few years, a number of large-scale graph database systems have been proposed for storing, analyzing, processing and querying unprecedented amount of structured or unstructured data from multiple application sources, such as, social networks, web 2.0, government chronicles, medical and scientific knowledge bases, and cyber networks. However, serving large and dynamically changing OLTP (Online transaction processing) workloads from millions of users involving distributed processing of terabytes and even petabytes of data is extremely demanding and pushes these graph databases to their limits [1]. Moreover, frequently occurring resource failures drastically affect the execution performance of big data applications running on these big data systems. Therefore, in order to cope up with the most diverse and challenging *workloads* and *failures*, many solutions featured by the graph database management systems are focused on enhancing the following *aspects*:

N. Sukhija (✉)
Department of Computer Science, Slippery Rock University of Pennsylvania,
275 Advanced Technology & Science Hall Slippery Rock University, Slippery Rock,
Pennsylvania 16057, USA
e-mail: nitin.sukhija@sru.edu

A. Morari
Pacific Northwest National Laboratory, Richland, USA
e-mail: alessandro.morari@pnnl.gov

I. Banicescu
Mississippi State University, Starkville, USA
e-mail: ioana@cse.msstate.edu

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_10

10.1.1 Performance

Performance is a prerequisite and is an extremely challenging issue for ensuring the efficient management of big data systems in highly *unpredictable* computing environments. The ability of big database systems to manage its performance is mainly dependent on its capability to optimize the utilization of the underlying computational resources. The optimizations, such as, *sync operation* [2], *accumulators* [3] and others, employed by big data systems enable them to increase the overall throughput and to minimize the contention to manage and process the diverse workloads defined by combinations OLTP transactions, analytic queries and ad hoc system utility and commands.

10.1.2 Scalability

Scalability is an important feature of a big database system as this determines the ability of the system in consideration to *preserve* its *performance* behavior when computing resources are added to distribute and manage both the growing data volumes and the transaction workload. In general, adaptiveness as well as elasticity of these database systems in highly unpredictable data environments is achieved via *horizontal* or *vertical scaling* [4]. The term horizontal scaling means that the big database systems scales by adding more hardware or software components (data servers) to its existing pool of computing infrastructure. Herein, both data and the operational workload are distributed over many data servers according to the load balancing techniques with no RAM or disk shared among these data servers. In comparison with horizontal scaling, the vertical scaling of database system is achieved by adding more power (i.e., CPU, RAM) to the existing computing machines [5]. Therefore, in vertical scaling the data reside on one computational node and scaling is done by utilizing the multi-core, which spreads the workload among many cores and/or CPUs that shares RAM and disks.

10.1.3 Fault Tolerance

Fault tolerance is the ability of a big database system to *maximize* the *reliability* of its data operations in order to hide the occurrence of faults, or the sudden unavailability of computing resources. With the increasing heterogeneity and scales of today's high performance computing environments, failures of computing nodes, disks or data centers are becoming a norm rather than an exception, thus achieving reliability is becoming extremely challenging for big data systems [6]. Consequently, the big data community is looking at various fault-tolerance approaches to alleviate the issue of resilience against faults, errors and failures [7]. In general, various fault-tolerance

strategies, such as, *checkpointing* and *rollback recovery*, *replication*, are commonly employed to circumvent the inability of the execution platform to guarantee a failure-free execution and to insure high performance of the big databases running on the complex computational systems with unpredictable conditions. The de-facto general purpose fault tolerance mechanism employed by the big data systems makes use of the master–worker paradigm that uses checkpoints or message logs along with rollback and replication recovery mechanisms [8]. Herein, the master nodes are responsible for the global coordination among the worker nodes performing data computations when a failures strikes some computational component.

With the exponential growth in production of data generated by modern day applications, there has been significant research efforts geared towards the development of the *distributed NoSQL* database systems [9] in order to deal with the issues of scalability and throughput of traditional relational databases. The distributed big data systems employ several data store replicas to achieve load balancing, fault tolerance etc., thus improving their performance and ability to process increasingly growing user's workloads. However, as the distributed big data system scales, the *Brewer's CAP Theorem* [10] can be applied to describe the trade-offs one is subject to while working with these distributed data storage systems. The theorem illustrates three basic systemic requirements of *Consistency*, *Availability*, and *Partition tolerance* that exist in a special relationship when it comes to designing and deploying distributed big data systems. Herein, the Consistency requirement implies that all servers will have the same data and all the users will receive the same data as a response to each request (operation, transaction, and query) regardless of which server responds to the request. The Availability requirement infers that a request made by the user will always receive a response by the system (which can be even an inconsistent data or message saying that system not working). Furthermore, the Partition tolerance requirement states that the system will continue to perform as expected even if an individual server fails or cannot be reached. This characteristic in the CAP theorem focuses on how the system will respond if there is a delay or failure in communication between two servers or nodes (a delay in communication may be considered as a failure in these complex systems). As big database systems scales out, it is theoretically impossible to achieve all the above-mentioned requirements. For instance, partitioning a network to achieve fault tolerance against network failures makes it challenging to achieve both availability and consistency requirements at the same time. Therefore, a *combination* of two requirements must be chosen to avoid the *performance* degradation of these distributed big database systems.

In this chapter, we profile some popular *graph database management systems (DBMS)*, such as, *Titan* [11], *OrientDB* [12], *ArangoDB* [13], *Giraph* [14], and *Neo4j* [15], along the *load balancing* and *fault tolerance* dimensions. The *descriptions* of the frameworks highlight significant research pertaining to the existing solutions for accomplishing *scalability* and *resiliency* of big data analytics in such systems.

10.2 Titan

Titan is a java-based open-source graph database that provides a highly scalable solution capable of managing extreme transactional workloads interacting in real-time with massive-scale graphs distributed across cluster of several servers. Titan incorporates a wide range of building blocks to distinguish it among other graph databases. The *flexibility* to support and choose among the pluggable third party adapters (as shown in Fig. 10.1), such as, Apache Cassandra [16], Apache HBase [17], or Oracle BerkeleyDB [18] as *storage backends* helps to cater individually the contrasting demands pertaining to the transaction and the scalability aspects of the graph database systems. The *integration* of Titan with *Cassandra* promotes high availability of data *without any point of failure*. Furthermore, it alleviates the constraint on read and/or writes since there is *no master/slave* architecture. Moreover, it supports *elastic scalability* which allows joining or removing computational resources without degrading performance. In case, there is a need for real-time real/write access to data, integration with Apache HBase is recommended. The major advantage of using *Apache HBase* is its support for reliable strong *consistency* of data. In comparison with Apache Cassandra, HBase supports consistent reads and writes. The benefit of using *BerkeleyDB* as a storage backend is its ability to work on the *same JVM* (Java Virtual Machine) as Titan and local persistence of data with *no overhead* of administration. The BerkeleyDB is promising if all the graph data fits on a local disk and all the frequently accessed graph elements also fits in main memory. Thus, using BerkeleyDB puts a limit to how much graph data can be worked on as it will not reveal high performance when the size of the graph exceeds the local disk and cannot be accessed locally within the same JVM.

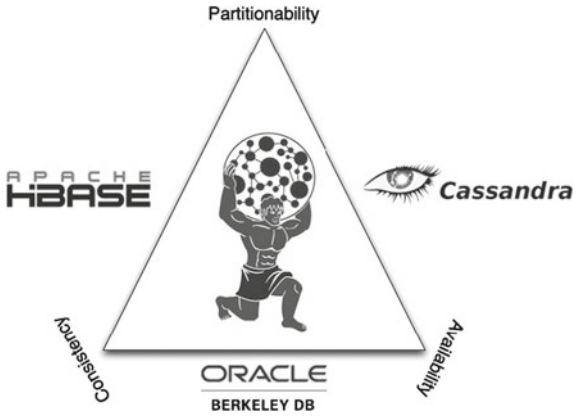


Fig. 10.1 Storage backends of Titan graph database and their respective preference toward two of the three requirements defined by CAP theorem

10.2.1 Load Balancing

With the increase in the volume of graph data and in the user base interacting with such graphs, balancing workloads is extremely important for preserving performance and is achieved in Titan by utilizing its *multiple storage backends*. In Titan database system the graph is stored as a collection of list (Adjacency lists). Therefore, the amount of vertices that are distributed among machines determines how well a graph is partitioned across the cluster. While *partitioning* a graph in Titan the below mentioned methods can be employed:

- **Edge Cut** This method partitions a graph and optimizes the assignment by placing the adjacent vertices of *frequently traversed* edges on the *same machine*. This assignment of adjacent vertices in the same partition leads to balanced load and to a lower communication overhead in graph queries. The performance of processing a graph query degrades if the graph traversal involves the vertices of frequently traversed edges that belong to different partitions on different machines.
- **Vertex Cut** This method focuses on addressing the issue of *hotspots* on very large graphs that are caused by vertices with a large number of incident edges. The Vertex Cut method optimizes the *balancing* of load by *partitioning* a vertex and its adjacency list across the *whole cluster*, thus removing hot spots. In Titan, the Vertex Cut is achieved by labeling a high degree vertex and defining it as partitioned, such that the label will be distributed across the cluster to avoid hot spots.

Titan database uses a *random partitioning* by default due to its configuration simplicity and efficiency. This approach works well and will result in balanced partitions, cxcept when the Titan cluster size and communication over graph queries increases resulting in performance degradation. In order to accommodate more graph data and to address the bottleneck associated with communication overhead, Titan offers an *explicit graph partitioning* option. However, this option necessitates calculation of the number of virtual partitions and configuration of these partitions beforehand. Thus, using a large number to partition the cluster can lead to excessive cluster fragments resulting in poor performance. Moreover, the number of the virtual partitions cannot be changed unless the graph is reloaded. The explicit partitioning can only be configured with storage backend that support ordered key storage (Hbase and Cassandra can both support this). Unlike Hbase, Cassandra requires an additional configuration of using *ByteOrderedPartitioner* in order to support explicit partitioning. The ByteOrderedPartitioner is a partitioner used by Cassandra to order rows lexically by key bytes [19].

10.2.2 Fault Tolerance

With multiple users, it is important that a database system is robust and can serve numerous concurrent requests. Titan database system is highly robust and supports

data distribution and *replication* for performance and fault tolerance. Titan facilitates continuously availability with *no single point of failure* using Cassandra as a storage backend. Titan renders a *master-master replication* setup using the Cassandra's replication strategy configuration which stores replicas on multiple nodes to ensure reliability and fault tolerance. The *replication strategies*, such as RackUnaware, RackAware, and DatacenterAware [20], configures the nodes where replicas can be placed and the replication factor decides the total number of replicas at nodes across the storage system. The *replication factor* determines the robustness of Titan in presence of system failures at the cost of data duplication. For instance, a replication factor of 3 implies that three replicas of each row will be configured and each replica will be placed on a different node, thus avoiding a single point of failure. Furthermore, a *Gossip protocol-based Accrual Failure Detector* is employed by Apache Cassandra for detecting the failures of computational nodes. This detector accrues the numeric values assigned to each node representing the suspicion level that another node might be down. These numeric values can be dynamically changed over time and can be used to trigger action plans. The *Hinted Handoff* recovery mechanism is employed by Cassandra for executing partition writes targeting the temporary failed nodes. Titan also employs a *transaction write-ahead log* which can be utilized to handle and repair the indexing and logging inconsistencies caused due to the transaction failures [21].

10.3 OrientDB

OrientDB is a distributed open source *Multi-Model* database management system which provides a wide-range NoSQL solution and an integrated scalable document and graph database [22].

10.3.1 Load Balancing

OrientDB database can be distributed across different servers and load-balancing can be applied among the active servers that are valid candidates to serve the requests for one user. Furthermore, OrientDB uses *sharding* (as shown in Fig. 10.2) to spread the load in order achieve the maximum of performance, scalability and robustness [4]. In OrientDB's distributed architecture, sharding is employed to *partition* the data of the database into partitions and these partitions are spread across different nodes. Herein, sharding of data is performed at class level where a developer can define multiple clusters per class and each cluster is comprised of one or multiple server nodes where data is replicated. If there are multiple servers available to connect, the following load balancing strategies can be used in order to deal with the high volume of user requests:

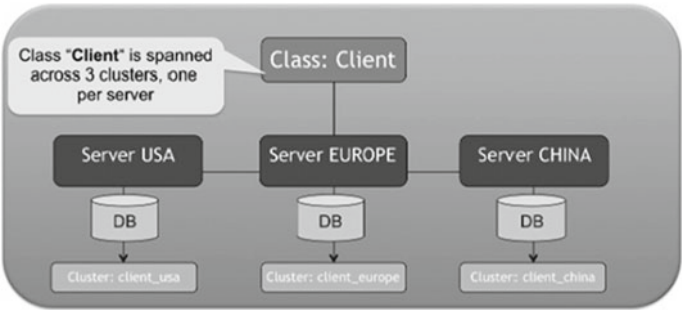


Fig. 10.2 Shards of data at class level: Multiple clusters defined per class and each cluster comprises of one or multiple server nodes for data replication

- **Sticky** It is the OrientDB default strategy and is a connecting characteristic that enables client to interact (connect) with one server and to switch only to another server if the current one is unreachable.
- **ROUND_ROBIN_CONNECT** This strategy allows a client to connect to a server from the list of all the available servers at connect time. The round-robin algorithm cycles through a list of available servers in order and the client will connect to available ones.
- **ROUND_ROBIN_REQUEST** This strategy allows a client to connect to a server from the list of all the available servers before every request in a round robin manner. This strategy should be used with caution when dealing with strong consistency.

10.3.2 Fault Tolerance

OrientDB's supports *Multi-Master replication* and *sharded architecture* for assuring the *reliability* of the ACID transactions [23]. In Multi-Master replication approach, all the nodes (servers) in OrientDB are considered as master nodes, which give every server the ability to read and write and perform queries on the database. When a node failure occurs or any server becomes unavailable, the *Auto Discovery* feature (as shown in Fig. 10.3) of OrientDB assist in automatically linking all the clients connected to the failed node to an available server with no fail-over to the application level. Moreover, OrientDB uses *Write Ahead Logging* (WAL) as a recovery mechanism for restoring the contents of the database once failure occurs, guaranteeing that all database transactions are processed accurately.

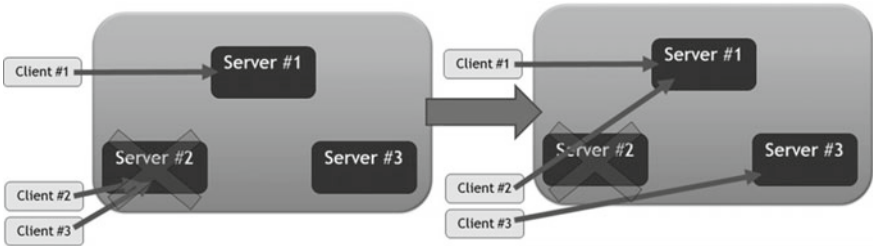


Fig. 10.3 Fail over Management in OrientDB: Automatic switching of clients and configuration update is performed when a server node fails

10.4 Giraph

Apache Giraph is a highly scalable distributed graph processing system that follows Google's Pregel [24] *Bulk Synchronous model* (BSP) [25]. Giraph provides a platform for enabling *iterative* large-scale graph *computation* on the data stored on Hadoop Distributed File System (HDFS) [26].

10.4.1 Load Balancing

The Giraph framework uses *BSP programming model*, where the operations are executed on graph data comprising vertices and edges. All operations in Giraph framework execute as a *Hadoop job* that utilizes the Hadoop cluster infrastructure. Moreover, Giraph follows a *master/worker architecture* and applies a *Zookeeper* [27] to select a master node to coordinate synchronization, computation and partitioning. In Giraph, the graph *computations* are expressed as sequence of iterations called *supersteps*, where each superstep is separated by a global synchronization barrier. During a superstep, the Giraph framework computes a function on each vertex in parallel and each vertex reads and processes the messages from previous iteration. Thus, balancing between computation and communication steps is extremely important for guaranteeing the performance and efficiency of the Giraph framework. In order to achieve balanced workload, Giraph uses *graph partitioning* to distribute work across the worker nodes (as shown in Fig. 10.4). In general, Giraph uses graph partitioning strategies, like *hash-based* or *range-based* partitioning that partitions the dataset based on a simple heuristic with the goal of distributing the vertices evenly among the partitions across computing nodes, irrespective of their edge connectivity. For instance the HashRange partitioner in Giraph framework allows the vertices to be divided into partitions by their hash code using ranges of their hash space [28]. Furthermore, the master node can become a performance bottleneck if it has to perform all computation processing and message communication at end of each superstep. To address this problem, Giraph employs *sharded aggregators* [29]. Herein, at the end

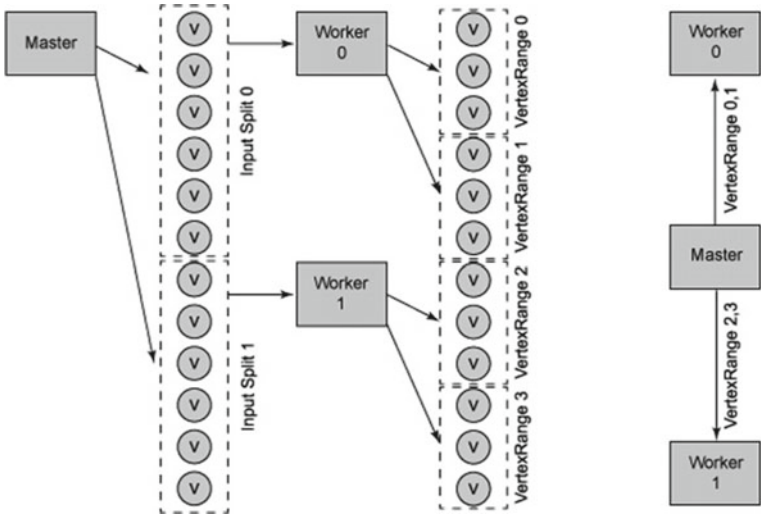


Fig. 10.4 Giraph applies master/worker architecture where a master node coordinates synchronization, computation and partitioning among the worker nodes

of the superstep in the BSP model each worker is assigned an aggregator. Thereafter, each worker is responsible for obtaining and aggregating the values for that aggregator from rest of the workers and sending all its aggregators to the master. Thus, using sharded aggregators help to balance the aggregation responsibilities among all the worker nodes and to alleviate the I/O traffic congestion at the master.

10.4.2 Fault Tolerance

The Giraph system is designed to have *no single point of failure* to perform reliable graph processing on large dataset which can be represented as a graph. The Giraph system adds fault tolerance to its infrastructure by employing *Apache Zookeeper* as its centralized coordination process which aids in *checkpointing paths* and in maintaining the global application state [30]. The Zookeeper maintains a *queue* of multiple master threads which is used by the workers nodes to choose a master that will serve as a computation coordinator. If the current master node fails then a new master is chosen automatically from the multiple masters' queue in order to carry out the operations reliably. In BSP model, any superstep can be restarted from a checkpoint. The Giraph framework enables implementation of *Checkpointing mechanism* at user-defined intervals which aids in automatically restarting the application when any worker node fails. The master node also uses the Zookeeper service to detect failure of a worker node. Once the failed node is detected, the master decides upon

the last committed superstep of the application. Thereafter, the master coordinates restart of the workers from the last committed superstep and the application is rolled back to its last committed superstep automatically.

10.5 ArangoDB

ArangoDB is a high performance *multi-model* NoSQL database system with flexible data modeling, where the data can be modeled as combination of graphs, key-value pairs or documents.

10.5.1 Load Balancing

ArangoDB uses *sharding* that allows usage of multiple computing nodes to execute a cluster of ArangoDB instances, which constitutes a single database. This facilitates automatic distribution of data to different servers, thus balancing the load among the different database servers and improving the data throughput. The ArangoDB cluster comprises of *two processes*: (1) DBservers that stores the data; (2) coordinators that act as links to the clients via REST interface. The infrastructure is designed to serve multiple client requests in parallel manner to obtain good performance. However, some client request scenarios can also lead to performance degradation; for instance if all the client requests are geared towards one single document. Nevertheless, *distributing* the highly requested/interacted documents uniformly over a *big sharded* collection will result in balancing the load of client requests/interaction across different DBservers and different coordinators, thus optimizing the parallel execution performance [31].

10.5.2 Fault Tolerance

The ArangoDB infrastructure supports fault tolerance for the actual DBserver nodes by employing asynchronous *master–slave replication*. Herein, pairs of DBservers are executed, where the primary DBserver cater to the incoming requests from the coordinators and the secondary DBserver perform replication of primary DBserver's data in a synchronous way. Furthermore, in addition to DBserver and coordinator processes (as shown in Fig. 10.5), an ArangoDB's cluster also contain *agents* processes, together called as "agency". The agent processes are used to manage the cluster configuration, and to synchronize reconfiguration and fail-over operations resulting in a high availability. The agency uses an external program *etcd* [32] that uses *RAFT* [33] consensus protocol as an alternative to *Paxos* [34] protocol to provide highly consistent and reliable hierarchical key value store.

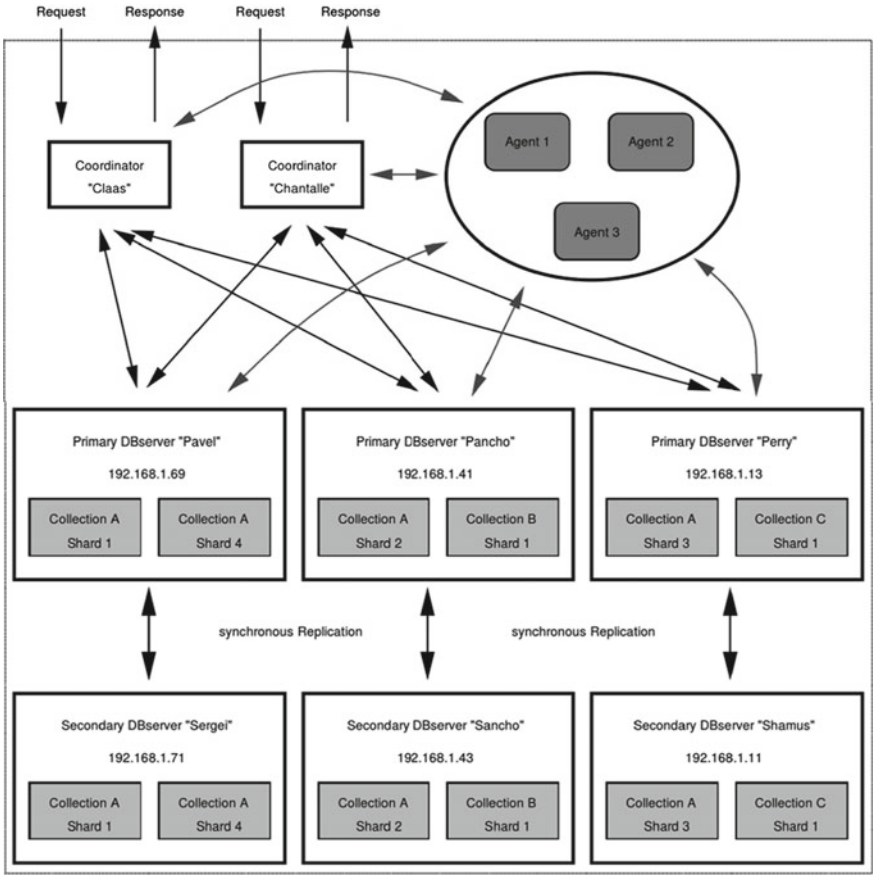


Fig. 10.5 Processes in the ArangoDB cluster: Coordinators managing the client requests, DBservers storing and replicating the data along with serving the requests from coordinators, and Agents performing the fail-over operations

10.6 Neo4j

Neo4j is an open source java-based *scalable* graph database management system. It is *leading* graph database among the current DB-Engines [35] and is ACID complaint transactional database with high availability and online backup.

10.6.1 Load Balancing

The Neo4j database system scales both *horizontally* and *vertically* in order to deliver high performance of transactions on the large-scale graphs. The enterprise version of

Neo4j provides *high availability* (HA) feature and balances the client requests among the neo4j HA servers by integrating its cluster infrastructure with HAproxy [36]. The *HAproxy* is an open source high performance and reliable *load balancer* for TCP- and HTTP-based applications involving high traffic rates. The HAproxy supports almost *nine load balancing algorithms*, such as, round robin, least connection, uri, and others, which can employed by the Neo4j cluster to balance the transactional workload across the HA servers. Furthermore, Neo4j employs *cache-based sharding* [37] while dealing with large dataset (represented as graph) that pushes the limits of resources (such as RAM) available. Therefore, if the graph is too big to fit in cache of single node, then Neo4j uses a consistent routing algorithm to distribute the graph load across the caches of different nodes of the cluster. Thus, the consistent request routing facilitated by cache-based sharding aids in achieving good performance in presence of resource constraints.

10.6.2 Fault Tolerance

Neo4j scalability package is equipped with mechanisms for handling diverse workloads and fault tolerance. Neo4j supports *master/slave architecture* and all the graph data is *replicated* on all the servers across the cluster to provide *redundancy* in case of instance failures (as shown in Fig. 10.6). The *multi-paxos consensus protocol* [38] is implemented in Neo4j infrastructure to provide *fault tolerance* against node failures. Neo4j uses paxos in order to perform cluster management, which involves choosing a master among the slaves, coordination and synchronization, and data replication and transaction propagation. Moreover, Neo4j maintain a *quorum* in order to execute write operations, where a minimum of servers have to be online in the cluster to allow propagation of write loads [39]. The quorum allows neo4j service to be

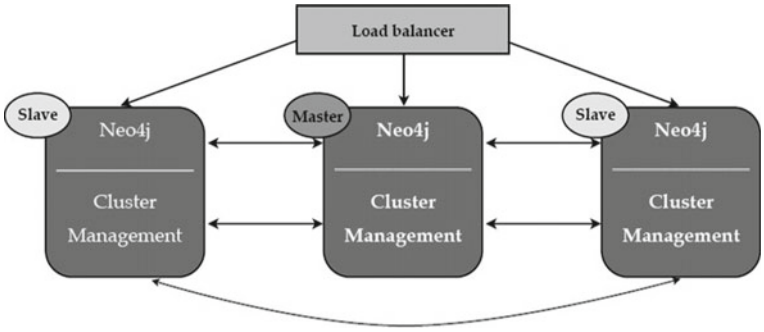


Fig. 10.6 Neo4j high availability (HA) master-slave cluster architecture enabling synchronization and data replication for a failure-free transaction propagation

reliable and available in the case server failures. Furthermore, an enterprise backup feature is integrated with Neo4j which enable full or *incremental online backup* of mission-critical data on the running clusters.

10.7 Conclusions

With rapidly increasing computing power and decreasing storage cost, many applications are now producing large volumes of data. This "fire hose" of data presents unique problems in analyzing the meaning of the data. The applications involving simulation, analysis, and visualization of big data support numerous challenges, such as the ability to spot business trends, determine quality of research, prevent diseases, link legal citations, combat crime, and determine real-time roadway traffic conditions. Graph-based data, e.g., social network, compute network, and power grid, is becoming increasingly more important to our society. Thus performance of the big data platforms enabling the processing of big data applications is often essential, even critical sometimes, to achieve the objectives proposed by the domain areas which are making use of them. Therefore, many research efforts have been attempted to enhance the performance and reliability of such computing platforms. These efforts include improving performance (per core) under stressful workloads, increasing scalability of their execution in parallel and distributed environments, and dealing with dynamically changing large data sets. Moreover, given the increase in the complexity of the computational systems, the resource failures are becoming a norm than exception, thus preserving the reliability of big data analytics executing on such computing systems is becoming extremely challenging for the big data platforms. Consequently, gaining deep insight about the state-of-the-art load balancing and fault tolerance approaches is of paramount importance, especially for researchers from both academic and industrial domains dealing with issues pertaining to the scalability and resiliency of applications involving big data.

References

1. Kambatla, K., Kollias, G., Kumar, V., Grama, A.: Trends in big data analytics. J. Parallel Distrib. Comput. **74**(7), 2561–2573 (2014)
2. Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A., Hellerstein, J.M.: Distributed graphlab: a framework for machine learning and data mining in the cloud. Proc. VLDB Endowment **5**(8), 716–727 (2012)
3. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: Cluster computing with working sets. HotCloud **10**, 10–10 (2010)
4. Cattell, R.: Scalable sql and nosql data stores. ACM SIGMOD Rec. **39**(4), 12–27 (2011)
5. Zikopoulos, P., Eaton, C., et al.: Understanding big data: Analytics for enterprise class hadoop and streaming data. McGraw-Hill Osborne Media (2011)
6. Tanenbaum, A.S., Van Steen, M.: Distributed systems: principles and paradigms, Vol. 2. Prentice hall Englewood Cliffs (2002)

7. Wang, P., Zhang, K., Chen, R., Chen, H., Guan, H.: Replication-based fault-tolerance for large-scale graph processing. In: 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 562–573. IEEE (2014)
8. Power, R., Li, J.: Piccolo: Building fast, distributed programs with partitioned tables. OSDI **10**, 1–14 (2010)
9. Leavitt, N.: Will nosql databases live up to their promise? Computer **43**(2), 12–14 (2010)
10. Gilbert, S., Lynch, N.: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. ACM SIGACT News **33**(2), 51–59 (2002)
11. Titan: Titan graph database. http://thinkaurelius.github.io/titan/
12. OrientDB: Orientdb graph database. http://orientdb.com/orientdb
13. ArangoDB: Arangodb nosql database. https://www.arangodb.com/
14. Giraph: Apache giraph. http://giraph.apache.org/
15. Neo4j: Neo4j graph database. http://neo4j.com/
16. Cassandra, A.: Apache Cassandra (2013)
17. Borthakur, D., Gray, J., Sarma, J.S., Muthukkaruppan, K., Spiegelberg, N., Kuang, H., Ranganathan, K., Molkov, D., Menon, A., Rash, S., et al.: Apache hadoop goes realtime at facebook. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pp. 1071–1080. ACM (2011)
18. Oracle Berkeley, D.: Java edition (2008)
19. Dede, E., Sendir, B., Kuzlu, P., Hartog, J., Govindaraju, M.: An evaluation of cassandra for hadoop. In: 2013 IEEE Sixth International Conference on Cloud Computing, pp. 494–501. IEEE (2013)
20. Lakshman, A., Malik, P.: Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review **44**(2), 35–40 (2010)
21. Mohan, C., Haderle, D., Lindsay, B., Pirahesh, H., Schwarz, P.: Aries: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. ACM Transactions on Database Systems (TODS) **17**(1), 94–162 (1992)
22. Tesoriero, C.: Getting Started with OrientDB. Packt Publishing Ltd (2013)
23. Gray, J., Reuter, A.: Transaction Processing: Soncepts and Techniques. Elsevier (1992)
24. Malewicz, G., Austern, M.H., Bik, A.J., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: a system for large-scale graph processing. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 135–146. ACM (2010)
25. Valiant, L.G.: A bridging model for parallel computation. Commun. ACM **33**(8), 103–111 (1990)
26. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1–10. IEEE (2010)
27. Hunt, P., Konar, M., Junqueira, F.P., Reed, B.: Zookeeper: Wait-free coordination for internet-scale systems. In: USENIX Annual Technical Conference, Vol. 8, p. 9 (2010)
28. Khayyat, Z., Awara, K., Alonazi, A., Jamjoom, H., Williams, D., Kalnis, P.: Mizan: a system for dynamic load balancing in large-scale graph processing. In: Proceedings of the 8th ACM European Conference on Computer Systems, pp. 169–182. ACM (2013)
29. Sakr, S.: Processing large-scale graph data: A guide to current technology. IBM Developerworks, p. 15 (2013)
30. Schelter, S.: Large scale graph processing with apache giraph. Invited talk at GameDuell Berlin 29th May (2012)
31. ArangoDB: Arangodb white paper sharding. https://www.arangodb.com/documents/
32. Store, R.K.V.: Reliable key-value store, etcd. https://coreos.com/etcd/docs/latest/
33. Ongaro, D., Ousterhout, J.: In search of an understandable consensus algorithm. In: 2014 USENIX Annual Technical Conference (USENIX ATC 14), pp. 305–319 (2014)
34. Lamport, L., et al.: Paxos made simple. ACM Sigact News **32**(4), 18–25 (2001)
35. Webber, J.: A programmatic introduction to neo4j. In: Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity, pp. 217–218. ACM (2012)

36. Tarreau, W.: Haproxy-the reliable, high-performance tcp/http load balancer (2012)
37. Montag, D.: Understanding neo4j Scalability. White Paper, Neotechnology (2013)
38. Rao, J., Shekita, E.J., Tata, S.: Using paxos to build a scalable, consistent, and highly available datastore. Proc. VLDB Endowment **4**(4), 243–254 (2011)
39. Partner, J., Vukotic, A., Watt, N., Abedrabbo, T., Fox, D.: Neo4j in Action. Manning Publications Company (2014)

Chapter 11
Fault Tolerance in MapReduce: A Survey

Bunjamin Memishi, Shadi Ibrahim, María S. Pérez and Gabriel Antoniu

11.1 Introduction

Data-intensive computing has become one of the most popular forms of parallel computing. This is due to the explosion of digital data we are living. This data expansion has mainly come from three sources: (i) scientific experiments from fields such as astronomy, particle physics, or genomics; (ii) data from sensors; and (iii) citizens publications in channels such as social networks.

Data-intensive computing systems, such as Hadoop MapReduce, have as main goal the processing of an enormous amount of data in a short time, by transmitting the computation where the data resides. In failure-free scenarios, these frameworks usually achieve good results. Given that failures are common at large scale, these frameworks exhibit some fault tolerance and dependability techniques as built-in features. In particular, MapReduce frameworks tolerate machine failures (crash failures) by re-executing all the tasks of the failed machine by the virtue of data replication. Furthermore, in order to mask temporary failures caused by network or machine overload (timing failure) where some tasks are performing relatively slower than other tasks, Hadoop relaunches other copies of these tasks on other machines.

B. Memishi (✉) · M.S. Pérez
OEG, E.T.S. Ingenieros Informáticos, Universidad Politécnica de Madrid, Campus de Montegancedo s/n, 28660 Boadilla del Monte, Madrid, Spain
e-mail: bmemishi@fi.upm.es

M.S. Pérez
e-mail: mperez@fi.upm.es

S. Ibrahim · G. Antoniu
Inria Campus Universitaire de Beaulieu, Rennes, 35042 Brittany, France
e-mail: shadi.ibrahim@inria.fr

G. Antoniu
e-mail: gabriel.antoniu@inria.fr

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_11

Foreseeing MapReduce usage in the next generation Internet [46], a particular concern is the aim of improving the MapReduce's reliability by providing better fault-tolerant mechanisms. As far as we know, there is not a complete review of the research in MapReduce fault tolerance, in such a way that it represents an overall picture of what has been done and what is missing. This survey addresses this gap, by means of the following contributions:

- An exhaustive study on the MapReduce framework, and its default fault-tolerant mechanisms. As one may assume, the leading MapReduce-based systems which are open source, with particular emphasis on Hadoop MapReduce, have evolved significantly since their first appearance. This study has taken into account all of these releases, by considering the most important advances in the fault-tolerant area.
- A systematic literature review on contributions with extensive analysis and proposals of new fault-tolerant mechanisms in MapReduce-based systems.
- A discussion about the open issues and key challenges for providing efficient fault tolerance in MapReduce-based systems.

This book chapter is organized as follows. Section 11.2 introduces the methodology which has guided the survey. Section 11.3 represents the MapReduce fundamentals, with particular emphasis on its fault-tolerant mechanisms. Section 11.4 gives an extensive analysis of the literature review. Section 11.5 describes some of the most popular data-intensive computing systems, mentioning their fault tolerance mechanisms. Section 11.6 discusses the opportunities and challenges to design efficient fault-tolerant mechanisms in MapReduce. Finally, Sect. 11.7 concludes the book chapter.

11.2 Methodology

In order to analyze the existing work, a previous identification of the main areas related to the topic addressed in this survey was performed. Concretely, four groups were defined:

- *Context* Contributions related to the general context of dependability.
- *MapReduce* Contributions that introduce the fundamentals of the MapReduce framework.
- *Optimizations* Contributions related to direct fault-tolerant solutions on MapReduce-based systems.
- *Others* Contributions proposing different solutions for other systems, which could be considered as added values in the context of MapReduce-based systems.

Additionally, this study has made a general analysis of the different types of failures in computing systems, and their connection to the MapReduce framework. This has enabled a solid construction of boundaries between different failures types

within MapReduce and other distributed systems. In the computer science literature, it is also very common to have a distinction between faults, errors, and failures [7], considering faults and errors as implications of failures. According to the existing literature [8, 11, 50], the most common failure-type division in MapReduce is: crash, omission, arbitrary, network and security failures. More specifically:

- *Crash failure* The process crashes at a specific point of time and never recovers after that time.
- *Omission failure* It is a more general kind of failures. A process does not send (or receive) a message that it is supposed to send (or receive).
- *Arbitrary (Byzantine) failure* The process fails in an arbitrary manner if it can deviate in any conceivable way from the algorithm assigned to it. It is the most expensive failure to tolerate. Even though it is assumed mostly as intentional and malicious, the arbitrary failure can simply be a bug in the implementation, the programming language, or even the compiler.
- *Network failure* The processes cannot communicate with each other. There are two kind of failures: (i) *One-way link.* There is difficulty in communication between two processes (e.g., one communication party can send, but the other party cannot receive); and (ii) *Network partition.* A line connecting two larger sections of a network fails.
- *Security failure* The messages between processes are inspected, modified, or prevented from being delivered. This group also considers *eavesdropping failures*, which are those failures related to leaking information obtained in an algorithm to an outside entity, possibly threatening the confidentiality of the data handled by the algorithm.

Section 11.4 will further clarify these concepts within MapReduce-based systems.

11.3 MapReduce Framework

The MapReduce framework is one of the most widespread approaches of data-intensive computing. It represents a programming model for processing large datasets [19, 33]. MapReduce has been discussed by researchers for more than a decade, including the database community. Even though its benefits have been questioned when compared to parallel databases, some authors suggest that both approaches have their own advantages, and there is not a risk that one could become *obsolete* [54]. MapReduce's advantages over parallel databases include storage-system independence and fine-grain fault tolerance for large jobs. Other advantages are simplicity, automatic parallelism, and scalability. These features make MapReduce an appropriate option for data-intensive applications, being more and more popular in this context. Indeed, it is used for different large-scale computing environments, such as Facebook Inc. [23], Yahoo! Inc. [65], and Microsoft Corporation [45].

By default, every MapReduce execution needs a special node, called *master*; the other nodes are called *workers*. The master keeps several data structures, like

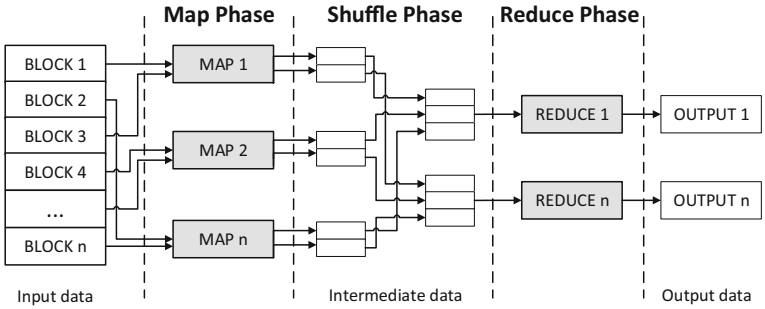


Fig. 11.1 MapReduce logical workflow

the state and the identity of the worker machines. Different tasks are assigned to the worker nodes by the master. Depending on the phase, tasks may execute two different functions: Map or Reduce. As explained in [19], users have to specify a Map function that processes a $key/value$ pair to generate a set of intermediate $key/value$ pairs, and a Reduce function that merges all intermediate values associated with the same intermediate key. In this way, many real-world problems can be expressed by means of the MapReduce model.

A simple MapReduce data workflow is shown in Fig. 11.1. This figure represents a MapReduce workflow scenario, from the input data to the output data. The most common implementations keep the input and output data in a reliable distributed file system, while the intermediate data is kept in the local file system at the worker nodes.

11.3.1 MapReduce 1.0 Versus MapReduce 2.0

The most common implementation of MapReduce is part of the Apache Hadoop open-source framework [56]. Hadoop uses the Hadoop Distributed File System (HDFS) as the underlying storage backend, but it was designed to work on many other distributed file systems as well.

The main components of Apache Hadoop are MapReduce and HDFS. Hadoop MapReduce consists of a JobTracker and many TaskTrackers, which constitute the processing master and workers, respectively. TaskTrackers consist of a limited number of slots for running map or reduce tasks. The MapReduce workflow is managed by the JobTracker, whose responsibility goes beyond the MapReduce process. For instance, the JobTracker is also in charge of the resource management. HDFS consists of a NameNode and many DataNodes, that is, the storage master and workers, respectively, whereas the NameNode manages the file system metadata, DataNodes hold a portion of data in blocks.

The traditional version of Hadoop has faced several shortcomings on large-scale systems, concerning scalability, reliability, and availability. The YARN (*Yet Another Resource Negotiator*) project has recently been developed with the aim of addressing these problems [57].

In the classic version of Hadoop, the JobTracker handles both resource management and job scheduling. The key idea behind YARN is to separate concerns, by splitting up the major functionalities of the JobTracker, resource management, and job scheduling/monitoring, into separate entities. In the new architecture, there is a global ResourceManager (RM) and per-application ApplicationMaster (AM). The ResourceManager and a per-node slave, the NodeManager (NM) compose the data-computation framework. The per-application ApplicationMaster is in charge of negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the progress of the tasks. The ResourceManager includes two components: a scheduler and application manager. Whereas the scheduler is in charge of resource allocation, the application manager accepts job submissions, and initiates the first job container for the job master (ApplicationMaster). This architectural change has as main goals to provide scalability and remove the single point of failure presented by the JobTracker. However, the resource scheduler, the application manager, and the application master now become single points of failure in the YARN architecture.

11.3.2 MapReduce Fault Tolerance

In Fig. 11.2 we show a big picture of the default fault-tolerant concepts and their mechanisms in MapReduce.

At the core of failure detection mechanism is the concept of heartbeat. Any kind of failure that is detected in MapReduce has to fulfill some preconditions, in this case to miss a certain number of heartbeats, so that the other entities in the system detect the failure. The classic implementation of MapReduce has no mechanism for dealing with the failure of the master, since the heartbeat mechanism is not used to detect this kind of failure. Workers send a heartbeat to the master, but the master's

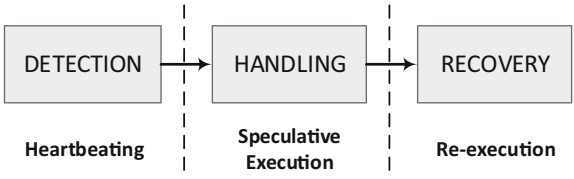


Fig. 11.2 Fault tolerance in MapReduce: The basic fault tolerance definitions (detection, handling, and recovery) with their corresponding implementations

health is monitored by the cluster administrator. This person must first detect this situation, and then manually restart the master.

Because the worker sends heartbeats to the master, its eventual failure will stop this notification mechanism. From the worker side, there is a simple loop that periodically sends heartbeat method calls to the master; by default, this period has been adjusted to 3 s in most of the implementations. The master makes a checkpoint every 200 s, in order to detect if it has missed any heartbeats from a worker for a period of 600 s, that is, 10 min. If this condition is fulfilled, then a worker is declared as dead and removed from the master's pool of workers upon which can schedule tasks on. After the master declares the worker as dead, the tasks running on a failed worker are restarted on other workers. Since the map tasks that completed its work, kept their output on the dead worker, they have to be restarted as well. On the other hand, reduce tasks that were not completed need to be executed in different workers, but since completed reduce tasks saved its output in HDFS, their re-execution is not necessary.

Apart from telling to the master that a worker is alive, heartbeats also are used as a channel for messages. As a part of the heartbeat, a worker states whether it is ready to run a new task, and in affirmative case, the master will use the heartbeat return value for communicating the actual task to the worker. Additionally, if a worker notices that it did not receive a progress update for a task in a period of time (by default, 600 s), it proceeds to mark the task as failed. After this, the worker's duty is to notify the master that a task attempt has failed; with this, the master reschedules a different execution of the task, trying to avoid rescheduling the task on the same worker where it has previously failed.

The master's duty is to manage both, the completed and ongoing tasks on the worker to be re-executed or speculated, respectively. In the case of a worker failure, before the master decides to re-execute the completed and ongoing tasks so that may skip the default timeout of MapReduce (10 min), there is only one opportunity left, speculative execution.

The speculative execution is meant to be a method of launching another equivalent task as a backup, but only after all the normal tasks have been launched, and after the average running time of the other tasks. In other words, a speculative task is basically run for map and reduce tasks that have its completion rate below a certain percentage of the completion rate of the majority of running tasks.

An interesting dilemma is how to differentiate the handling and recovery mechanisms. A simple question arises: Does MapReduce differentiate between handling and recovery?

In some sense, both, speculative execution and re-execution try to complete a MapReduce job as soon as possible, with the least processing time, while executing its tasks on minimal resources (e.g., to avoid long occupation of resources by some tasks).[1] However, the sequence of performing the speculative execution and re-execution is what makes them different, therefore considering the former one be

[1]This is not particularly true in the case of speculative execution, since it has proven to exhaust a considerable amount of resources, when executed on heterogeneous environments [39, 72] or when the system is going under failures [22].

part of the handling process, and the latter one part of the recovery. An additional difference to this is that, while the re-execution mechanism tries to react after the heartbeat mechanism has declared that an entity has failed, the speculative execution does not need the same timeout condition in order to take place; it reacts sooner.

Regarding to the nomenclature related to failures and errors, we consider a job failure when the job does not complete successfully. In this case, the first task that fails can be considered as an error, because it will request its speculation or re-execution from the master. A task failure can happen because the network is overloaded (in this case, this is also an error, because the network fault is active and loses some deliveries). In order to simplify this, we assume that in MapReduce, a task or any other entity is facing a failure, whenever it does not fulfill its intended function.

From the point of view of Hadoop's MapReduce, failures can happen in the master and worker. When the master fails, this is a single point of failure. But in the case of the worker, it may have a task fail (map or reduce task, or shuffle phase) or the entire worker. During a map phase, if a map task crashes, Hadoop tries to recompute it in a different tasktracker. In order to make sure that this computation takes place, most of the reducers should complain for not receiving the map task output or the number of notifications is higher or equal to three [51]. The failed tasks have higher priority to be executed than the other ones; this is done to detect when a task fails repeatedly due to a bug and stop the job. In a reduce phase, a reduce task failure will have to be executed in a different tasktracker, having in mind that the three reduce phases should start from the beginning. The reduce task is considered as failed, if the majority of its shuffle attempts fails, the shuffle phase does not succeed to get five map outputs, or its progress stops for a long time. During the shuffle phase, a failure may also happen (in this case, a network failure), because two processes (in our case two daemons) can be in a working state, but a network failure may stop any data interchange between them. MapReduce implementations have been improved by means of the Kerberos authentication system, preventing a malicious reduce task from requesting another user's map output.

11.4 Analysis

Several projects have addressed different reliability issues in data-intensive frameworks, in particular for MapReduce. In this section, we have tried to collect those studies, adapting them to the most common failure type divisions in distributed systems [8, 11, 50]: crash, omission, arbitrary, network and security failures. Table 11.1 summarizes the main studies included in this review, listed in publication date order.

In [22], authors have evaluated Hadoop, demonstrating a large variation in Hadoop job completion time in the presence of failures. According to authors, this is because Hadoop uses the same functionality to recover from worker failure, regardless of the cause or failure type. Since Hadoop couples failure detection and recovery with overload handling into a conservative design with conservative parameter choices, it is often slow reacting to failures, exhibiting different response times under failure.

Table 11.1 The main studies included in this review, listed in publication date order

Year and study	Algorithms	Concepts
2004, [19]	Speculative execution	The foundations
2008, [72]	Longest Approximate Time to End (LATE)	Heterogeneity considerations
2009, [15]	UpRight library ($3f + 1$)	Byzantine fault tolerance
2009, [59]	Metadata replication	Crash, failure handling
2009, [10]	Re-execution evaluation	Crash, omission failure, failure recovery
2009, [58]	MapReduce setup simulation	Crash, network failure
2010, [39]	Dedicated compute resources, Hibernate state, Hybrid task scheduling	Volunteer computing systems
2010, [4]	Outlier culling, Cause- and resource-aware	Crash, omission failure, failure detection, failure handling, failure recovery, Data locality, Network hotspot
2010, [37]	Intermediate storage system—ISS (through asynchronous replication, rack-level replication, selective replication)	Crash, omission failure, failure handling, failure recovery
2010, [16]	Pipelined intermediate data, Online aggregation, Continuous queries	MapReduce and beyond (e.g., Interactive applications), Crash, omission failure, failure handling, failure recovery
2010, [48]	BlackBox approach (based on OS-level metrics)	Crash, omission failure, failure detection and diagnosis
2010, [55]	Massive fault tolerance, replica management, barriers free execution, latency-hiding optimization, distributed result checking	Network failure, failure handling, failure recovery
2010, [14]	Hybrid calculation model ($n - step$ probability, VM expected lifetime, cost of termination)	Network, failure handling, failure recovery
2010, [51]	Mandatory access control, differential privacy	Security failure, failure detection, failure handling
2011, [41]	Map and shuffle (phases) overlap, Task duplication, pull mechanism, queues	Cloud-based MapReduce, Crash, single point of failure, failure detection, failure recovery
2011, [73]	Adaptive interval, reputation-based detector	Crash, failure detection
2011, [9]	Metadata replication	Crash, failure handling
2011, [34]	Stochastic prediction model	MapReduce fault tolerance understanding
2011, [18]	Deferred execution, Tentative reduce execution, Digest outputs, Tight storage replication	Crash, and Byzantine fault tolerance, failure handling

(continued)

Table 11.1 (continued)

Year and study	Algorithms	Concepts
2011, [42]	Split message format modification, save the intermediate work, commit mechanism modification	Network, failure handling, failure recovery
2012, [52]	Kerberos protocol	Security, failure detection, failure handling
2012, [24]	JobTracker re-architecture	Crash, single point of failure, failure handling, failure recovery
2013, [57]	JobTracker re-architecture through Hadoop YARN	Crash, single point of failure, failure handling, failure recovery
2013, [29]	Encryption	Eavesdropping failure, failure detection, failure handling
2013, [35]	Analytical failure measurement	Crash, failure handling, failure recovery
2014, [3]	Greedy Speculative Scheduling (GS), Resource Aware Speculative Scheduling (RAS)	Approximation analytics
2014, [13]	Maximum Cost Performance (MCP)	Weighted moving average (EWMA)
2014, [62]	Accountability test	Byzantine fault tolerance, failure detection, failure handling
2014, [64]	Early cloning, Enhanced Speculative Execution (ESE)	Single job
2015, [63]	Smart Cloning Algorithm (SCA), Enhanced Speculative Execution (ESE)	Multiple jobs
2015, [44]	Diarchy algorithm	Single point of failure, failure handling
2015, [30]	Failure recovery evaluation	Crash failure, multiple jobs
2015, [66]	Failure-aware scheduling	Failure handling, failure recovery, multiple jobs
2015, [49]	Energy cost of speculation	Omission failure, resource heterogeneity, energy considerations
2016, [43]	MapReduce timeout analysis	Failure detection, handling

Authors conclude that Hadoop makes unrealistic assumptions about task progress rates, rediscovers failures individually by each task at the cost of great degradation in job running time, and does not consider the causes of connection failures between tasks, which leads to failure propagation to healthy tasks.

In [43], authors report that, in the presence of single machine failure the applications latencies vary not only in accordance to the occupancy time of the failure, similar to [22], but also vary with the job length (short or long).

In [10], authors have evaluated the performance and overhead of both the checkpointing-based fault tolerance and the re-execution based fault tolerance in MapReduce through event simulation driven by Los Alamos National Labs (LANL)

data. Regarding MapReduce, the fault tolerance mechanism which was explored is re-execution, where all map or reduce tasks from a failed core are reallocated dynamically to operational cores whether the tasks had completed or not (i.e., partial results are locally stored), and execution is repeated completely. In the evaluation of the performance of MapReduce in the context of real-world failure data, it was identified that there is pressure to decrease the size of individual map tasks as the cluster size increases.

In [35], authors have introduced an analytical study of MapReduce performance under failures, comparing it to MPI. This research is HPC oriented and proposes an analytical approach to measure the capabilities of the two programming models to tolerate failures. In the MapReduce case, they have started with the principle that any kind of failure is isolated in one process only (e.g., map task). Due to this, the performance modeling of MapReduce was built on the analysis of each single process. The model consists of introducing an upper bound of the MapReduce execution time when no migration/replica is utilized, followed by an algorithm to derive the best performance when replica-based balance is adopted. According to the evaluation results, MapReduce achieves better performance than MPI on less reliable commodity systems.

11.4.1 Crash Failure

During a crash failure, the process crashes at a specific point of time and never recovers after that time. Since a crash failure involves process failing to finish its function according to its general definition, this means that in MapReduce a crash failure can lead to a node (machine), daemon (JobTracker or TaskTracker), or task (map, reduce) failure. These failures surge when a node simply crashes, and affects all of its daemons. But this is not only the case for a crash failure; there are many other cases when particular daemons or tasks crash due to Java Virtual Machine (JVM) issues, high overloads, memory or CPU errors, etc.

At this section point, we will summarize the master crash failures first, and then continue with other crash failures in MapReduce. An important contribution to the high availability of JobTracker is the work of Wang et al. [59]. Their paper proposes a metadata replication-based solution to enable Hadoop high availability by removing single point of failure in Hadoop, regardless of whether it is NameNode or a JobTracker. Their solution involves three major phases:

- Initialization phase. Each standby/slave node is registered to active/primary node and its initial metadata (such as version file and file system image) are caught up with those of active/primary node.
- Replication phase. The runtime metadata (such as outstanding operations and lease states) for failover in future are replicated.
- Failover phase. Standby/new elected primary node takes over all communications.

A well-known implementation of this contribution has been done by Facebook, Inc. [9], by creating the active and standby AvatarNode. This node is simply wrapped

to the NameNode, and the standby AvatarNode takes the role of the active AvatarNode in less than a minute; this is because every DataNode speaks with both AvatarNodes all the time.

However, the above solution did not prove to give the optimum for the company requirements, since their database has grown by 2500× in the past 4 years. Therefore, another approach named Corona [24] was used. This time, for Facebook researchers it was obvious that they should separate the JobTracker responsibilities: resource management and job coordination. The cluster manager should look for cluster resources only, while a dedicated JobTracker is created per each job. As you can notice, at many points, the design decisions of Corona are similar to Hadoop YARN. Additionally, Corona has been designed to use push-based scheduling, as a major difference to the pull-based scheduling of the Hadoop MapReduce.

In the work [47], authors propose an automatic failover solution for the JobTracker to address the single point of failure. It is based on the Leader Election Framework [11], using Apache Zookeper [5]. This means that multiple JobTrackers (at least three) are started together, but only one of them is the leader at a particular time. The leader does not serve any client, but receives periodical checkpoints from the remaining JobTrackers. If one of the NameNodes fails, the leader recovers its availability from the most recent checkpointed data. However, this solution within Yarn has not been explored for job masters [57] and only addresses other single points of failure, such as the resource manager daemon.

In a recent study [44], the authors propose Diarchy, a novel approach for management of masters, whose aim is to increase the reliability of Hadoop YARN, based on the sharing and backup of responsibilities between two masters working as peers. Despite the fact that Diarchy seems only to improve the reliability of failure handling between masters, its functioning also puts a lower boundary in the worst-case assumption, with the number of Diarchy failed tasks not surpassing the half number of failed tasks of Hadoop YARN.

In case of a TaskTracker crash failure, its tasks are by default re-executed in the other TaskTrackers. This is valid for both, map and reduce tasks. Map tasks completed on the dead TaskTracker are restarted because the job is still in the progress phase and did not finish yet, and contains n number of reduce tasks, which need that particular map output. Reduce tasks are re-executed as well, except for those reduce tasks that have completed, because they have saved its output in a distributed file system, that is, in HDFS.

MapReduce philosophy is based on the fact that a TaskTracker failure does not represent a drastic damage to the overall job completion, especially long jobs. This is motivated by large companies [20], which use MapReduce on a daily basis, and argue that even with a loss of a big number of machines, they have finished in a moderate completion time.[2] Any failure would simply speculate/re-execute the task in a different TaskTracker.

[2]Jeff Dean, one of the leading engineers in Google, said: (we) "lost 1600 of 1800 machines once, but finished fine".

There are cases where TaskTrackers may be blacklisted by mistake from the JobTracker. In fact, this happens because the ratio of the number of the failed tasks in the respective TaskTracker is higher than the average failure rate on the overall cluster [61]. By default, the Hadoop's blacklist mechanism marks a TaskTracker as blacklisted if the number of tasks that have failed is more than four. After this, the JobTracker will stop assigning future tasks to that TaskTracker for a limited period of time. These blacklisted TaskTrackers can be brought to live, only by restarting them; in this way, they will be removed from the JobTracker's blacklist. The blacklisting issue could also go beyond this. This can be explained with one scenario. Let us assume that, at some point, reduce tasks that are running in the other TaskTrackers will try to connect to the failed TaskTracker. Some of the reduce tasks need the map output from the failed TaskTracker. However, as they cannot terminate the shuffle phase (because of the missing map output from the failed TaskTracker), they fail. Experiments in [22] show that reduce tasks die within seconds of their start (without having sent notifications) because all the conditions which declare the reduce task to be faulty become temporarily true when the failed node is chosen among the first nodes to connect to. In these cases, when most of the shuffles fail and there is little progress made, there is nothing left except re-execution, while wasting an additional amount of resources.

The idea behind the paper [18] is doubling each task in execution. This means that if one of the tasks fails, the second backup task will finish on time, reducing the job completion time using larger (intuitively, you may guess that doubling the tasks leads to approximately doubling the resources) amounts of resources.

In [73], authors have proposed two mechanisms to improve the failure detection in Hadoop via heartbeat, but only in the worker side, that is, the TaskTracker. While the adaptive interval mechanism adjusts the TaskTracker timeout according to the estimated job running time in a dynamic way, the reputation-based detector compares the number of fetch errors reported when copying intermediate data from the mapper and when any of the TaskTrackers reaches a specific threshold that TaskTracker will be announced as a failed one. As authors explain, the adaptive interval is advantageous to small jobs while the reputation-based detector is mainly intended to longer jobs.

In the early versions of Hadoop (including the Hadoop 0.20 version), a crash failure of the JobTracker involved that all active work was lost entirely when restarting the JobTracker. The next Hadoop version 0.21 gave a partial solution to this problem, making periodic checkpoints into the file system [56], so as to provide partial recovery.

In principle, it is very hard to recover any possible data after a TaskTracker's failure. That is why Hadoop's reaction is to simply re-execute the tasks in the other TaskTrackers. However, there are works which have tried to take the advantage of checkpointing [16], or saving the intermediate data in a distributed file system [37, 38].

Regarding [16], among the interesting aspects of the pipelined Hadoop implementation is that it is robust to the failure of both map and reduce tasks, introducing the "checkpoint" concept. It works on the principle that each map and reduce task notifies the JobTracker, which spreads or saves the progress, informing the other nodes about it. For achieving this, a modified MapReduce architecture is proposed that allows data to be pipelined between operators, preserving the programming interfaces and fault tolerance models of a full-featured MapReduce framework. This provides significant new functionality, including "early returns" on long-running jobs via online aggregation, and continuous queries over streaming data. The paper has also demonstrated the benefits for batch processing: by pipelining both within and across jobs, the proposed implementation can reduce the time to job completion. This study work can also be considered as an optional solution to an omission failure.

In [37], authors propose an intermediate storage system, with two features in mind: data availability and minimal interference. According to this paper, these issues are solved with ISS (intermediate storage system), which is based on three techniques:

- Asynchronous replication. This does not block the ongoing procedures of the writer. Moreover the strong consistency is not required when having in mind that in platforms like Hadoop and similar, there is a single writer and single reader for intermediate data.
- Rack-level replication. This technique is chosen, because of the higher bandwidth availability within a rack, taking into account that the rack switch is not heavily used as the core switch.
- Selective replication. It is used considering that the replication will be applied only to the locally consumed data; in case of failure, the other data may be fetched again without problems.

This work is important to be mentioned, because for every TaskTracker failure, every map task that has been completed, it has already saved its output in a reliable storage system different from the local file system. In this way, the amount of redundant work for re-executing the map task that has been completed on the failed TaskTracker is reduced again.

A recent study [30] has investigated the impact of failures in shared Hadoop clusters. Accordingly, the authors evaluated the performance of Hadoop under failure when applying several schedulers (i.e., Fifo, delay, and capacity schedulers). They observe that the current failure handling mechanism is entirely entrusted to the core of Hadoop and hidden from Hadoop schedulers. This in turn results in a significant increase of the execution time of jobs with failed tasks. The performance degradation is caused by: (i) the waiting time for free resources to re-execute failed tasks, and (ii) not considering locality when scheduling failed tasks. In [66], the authors have proposed Chronos, a failure-aware scheduling strategy in shared Hadoop cluster.

Chronos triggers a lightweight preemption technique to free resources as soon as failure is detected, thus eliminating the waiting time. Furthermore, Chronos takes into consideration data locality of recovery tasks when preempting a running task. As a result, Chronos is able to correct the operation of Hadoops schedulers while improving the performance of MapReduce applications under failures.

11.4.2 Omission Failure (Stragglers)

An omission failure is a more general kind of failures. This happens when a process does not send (or receive) a message that it is supposed to send (or receive). In MapReduce terminology, omission failures are synonym for stragglers. Indeed, the concept of stragglers is very important in the MapReduce community, especially task stragglers, which could jeopardize the job completion time. Typically, the main causes of a MapReduce straggler task are: (i) a slow node, (ii) network overload, and (iii) input data skew [4].

Most of the state of the art in this direction has intended to improve the job execution time, by means of doubling the overall small jobs [2], or just by doubling the suspected tasks (stragglers) through different speculative execution optimizations [4, 13, 19, 32, 64, 72].

In [72], authors have also proposed a new scheduling algorithm called Longest Approximate Time to End (LATE) to improve the performance of Hadoop in a heterogeneous environment, brought by the variation of VM consolidation amongst different physical machines, by preventing the incorrect execution of speculative tasks. In this work, authors try to solve the issue of finding the real stragglers[3] among the MapReduce tasks, so as to speculatively execute them, while giving them the deserved priority. As the node heterogeneity is common in the real-world infrastructures and particularly cloud infrastructures, the speculative execution in the default Hadoop's MapReduce implementation is facing difficulties to give a good performance. The paper proposes an algorithm which should in some way improve the MapReduce performance in heterogeneous environments. It starts giving some assumptions made by Hadoop, and how they are broken down in practice. Later on, it proposes the LATE algorithm, which is based on three principles: prioritizing tasks to speculate, selecting fast nodes to run on, and capping speculative tasks to prevent thrashing. The paper has an extensive experimental evaluation, which proves the valuable idea implemented in LATE.

Mantri [4] is another important contribution related to omission failures, which are called outliers in this paper. The main aim of the contribution is to monitor and cull or

[3]It is important to mention that, differently from [72] which considers tasks as stragglers, in the default paper of Google [19], a straggler is "a machine that takes an unusually long time to complete one of the last few map or reduce tasks in the computation".

relax the outliers, accordingly to their causes. Based on their research, outliers have many causes, but mainly are enforced by MapReduce data skew, crossrack traffic, and bad (or busy) machines. In order to detect these outliers, Mantri does not rely only on task duplication. Instead, its protocol enhances according to outlier causes. A real-time progress score is able to separate long tasks from real outliers. Whereas the former tasks are allowed to be run, the real outliers are only duplicated when new available resources arise. Since the state-of-the-art contributions were mostly duplicating tasks at the end of the job, Mantri is able to make smart decision even before this, in case the progress score of the task is heavily progressing. Apart from data locality, Mantri places task based on the current utilization of network links, in order to minimize the network load and avoid self-interference among loads. In addition, Mantri is also able to measure the importance of the task output, and according to a certain threshold, it decides whether to recompute task or replicate its output. In general, the real-time evaluations and trace-driven simulations show Mantri to improve the average completion time for about 32 %.

GRASS [3] is another novel optimization framework, which is oriented to trimming the stragglers for approximation jobs. Approximation jobs are very common in the last period, because many domains are willing to have partial data in a specific deadline or error margin, instead of processing the entire data in an unlimited time or with 0 % error margin. After the introduction of the MapReduce programming model, which came with a simple solution of speculative execution of slow tasks (stragglers), the research community proposed decent alternatives, such as LATE [72] or Mantri [4]. However, they were not meant to give near to optimal solution for the domain of approximation analytics. And this is the advantage of GRASS, which is basically formed of two algorithms:

1. Greedy Speculative Scheduling (GS). This algorithm is intended to greedily pick a task that will be scheduled next.
2. Resource Aware Speculative Scheduling (RAS). This algorithm is able to measure the cost of leaving an old task to run or schedule a new task, according to some important parameters (e.g., time, resources, etc.).

GRASS is a combination of GS and RAS.

Depending on the cluster infrastructure size, but also on other parameters, the scheduler could impose different limitations per user or workload. Among others, it is common to place a limit on the number of concurrent running tasks. The overall set of these simultaneous tasks per each user (or workload) is known as wave. If a GRASS job requires many waves, then it starts with RAS and finally, in the last two waves uses GS. If the jobs are short, it may use only GS. This switching is mostly dependent on:

- Deadline–error bound.
- Cluster utilization.
- Estimation accuracy for two parameters, t_{rem} (remaining time for and old job), and t_{new} (an estimated time for a new job).

Evaluations show that GRASS improves Hadoop and Spark, regardless of the usage of LATE or Mantri, by 47 and 38 %, respectively, in production workloads of Facebook and Microsoft Bing. Apart from approximation analytics, the speculative execution of GRASS also shows to be better for exact computations.

In [13], authors propose an optimized speculative execution algorithm called Maximum Cost Performance (MCP) that is characterized by:

- Apart from the progress rate, it takes into consideration the process bandwidth in a phase, in order to detect the slow tasks.
- It uses exponentially weighted moving average (EWMA), whose duty is to predict the process speed and also predict the task remaining time.
- It builds a cost-aware model that determines what task needs a backup based on the cluster load.

In addition, the MCP contribution is based on the disadvantages of previous contributions, which mainly rely on the task progress rate to predict stragglers, inappropriate reaction on input data skews scenarios, unstable cost comparison between the backup and ongoing straggler task, etc. Evaluation experiments on a small-cluster infrastructure show MCP to have 39 % faster completion time and 44 % improved throughput when compared to default Hadoop.

In [64], authors propose an optimized speculative execution algorithm that is oriented to solving a single-job problem in MapReduce. The advantage of this work is that it takes into account two cluster scenarios, heavy and lightly loaded case. For the lightly loaded cluster, authors introduce two different speculative execution policies, early cloning, and later speculative execution based on the task progress rate. During the stage of heavily loaded cluster, the intuition is to use a later backup task. In this case, an Enhanced Speculative Execution (ESE) algorithm is proposed, which basically extends the work of [4]. Same authors have also introduced an additional extended work that assumes to work for multiple MapReduce jobs [63].

An important project related to Hadoop's omission failures is presented in [15]. In this work, authors have tried to build separate fault tolerance thresholds in the UpRight library for omission and commission failures, because omission failures are likely to be more common than commission failures. As we have mentioned before, during omission failures, a process fails to send or receive messages specified by the protocol. Commission failures exclude omission failures, including the failures upon which a process sends a message not specified by the protocol. Therefore, in the case of omission failures, the library can be fine-tuned in order to provide the liveness property (meaning that the system is "up") despite any number of omission failures.

In [49], the authors have studied the implications of speculative execution on the performance and energy consumption in Hadoop clusters. They observed that

speculative execution may result in a reduction in the energy consumption of Hadoop cluster if and only if the execution time of MapReduce application is noticeably reduced to compensate the energy cost of speculative execution (i.e., the extra power consumption due to the extra used resources).

The TaskTracker omission failures have also been addressed in some of the previous works we have mentioned [16, 18].

11.4.3 Arbitrary (Byzantine) Failure

The work discussing the omission failures in [15], is actually a wider review that includes the byzantine failures in general. The main properties upon which the UpRight library is based are:

- An UpRight system is safe ("right") despite r commission failures and any number of omission failures.
- An UpRight system is safe and eventually live ("up") during sufficiently long synchronous intervals when there are at the most u failures of which at most r are commission failures and the rest are omission failures.

The contribution of this paper is to establish byzantine fault tolerance as a viable alternative to crash fault tolerance for at least some cluster services rather than any individual technique. As authors say, much of their work involved making existing ideas fit well together, rather than presenting something new. Additionally, the performance is a secondary concern, with no claim that all cluster services can get low-cost BFT (byzantine fault tolerance).

The main goal of the work presented in [18] is to represent a BFT MapReduce runtime system that tolerates faults that corrupt the results of computation of tasks, such as the cases of DRAM and CPU errors/faults. These last ones cannot be detected using checksums and often do not crash the task they affect, but can only silently corrupt the result of a task. Because of this, they have to be detected and their effects masked by executing each task more than once. This BFT MapReduce follows the approach of executing each task more than once, but in particular circumstances. However, as the state machine approach requires $3f + 1$ replicas to tolerate at the most f faulty replicas, which gives a minimum of four copies of each task, this implementation uses several mechanisms to minimize both the number of copies of tasks executed and the time needed to execute them. In case there is a fault, from the evaluation results, it is confirmed that the cost of this solution is close to the cost of executing the job twice, instead of three times as the naive solution. Authors argue that this cost is acceptable for critical applications that require high level of fault tolerance. They introduce an adaptable approach for multicloud environments in [17].

In [62], authors propose another solution for commission failures called Accountable MapReduce. This proposal forces each machine in the cluster to be responsible

for its behavior, by means of setting a group of auditors that perform an accountability test that checks the live nodes. This is done in real time, with the aim of detecting the malicious nodes.

11.4.4 Network Failure

During a network failure, many nodes leave the Hadoop cluster; this issue has been discussed in different publications [14, 39, 42, 55], although for particular environments.

The work presented in [39] introduces a new kind of implementation environment of MapReduce called MOON, which is MapReduce on Opportunistic eNvironments. This MapReduce implementation has most of the resources coming from volunteer computing systems that form a Desktop Grid. In order to solve the resource unavailability, which is vulnerable to network failure, MOON supplements a volunteer computing system with a small number of dedicated compute resources. These dedicated resources keep a replica in order to enhance high reliability, maintaining the most important daemons, including the JobTracker. To enforce its design architecture, MOON differentiates files into reliable and opportunistic. Reliable files should not be lost under any circumstances. In contrast, opportunistic files are transient data that can tolerate some level of unavailability. It is normal to assume that reliable files have priority for being kept in dedicated computers, while opportunistic files are saved in these resources only when possible. In a similar way, this separation is also managed for read and write requests. MOON is very flexible in adjusting these features, based on the Quality of Service (QoS) needs. A reason for this is the introduction of a hibernate state and hybrid task scheduling. The hibernate state is an intermediate state whose main duty is to avoid having an expiry interval that is too long or short, which can incorrectly consider a worker node as dead or alive. A worker node enters in this state earlier than its expiry interval, and as a consequence it will not be supplied with further requests from clients. MOON changes the speculative execution mechanism by differentiating straggler tasks in frozen and slow lists of tasks, adjusting their execution based on the suspension interval, which is significantly smaller than the expiry interval. An important change to speculating tasks is their progress score, which divides the job into normal or homestretch. During the normal phase, a task is speculatively executed according to the default Hadoop framework; in a homestretch phase, a job is considered to have advanced toward its completion, therefore MOON tries to maintain more running copies of straggler tasks.

A later project similar to MOON is presented in [55]. Here, authors try to present a complete runtime environment to execute MapReduce applications on a Desktop Grid. The MapReduce programming model is implemented on top of an open-source middleware, called BitDew [25], extending it with three main additional software components: the MapReduce Master, MapReduce worker programs, and the MapReduce library (and several functions written by the user for their particular MapReduce

application). Authors wanted to benefit from the BitDew basic services, in order to provide highly needed features in Internet Desktop Grid, such as "massive fault tolerance, replica management, barriers free execution, and latency-hiding optimization, as well as distributed result checking". The last point (distributed checking) is particularly interesting, knowing that result certification is very difficult for intermediate results which might be very large to send for verification on the server side. The introduced framework implements majority voting heuristics, even though it involves larger redundant computation.

A research paper presented in [41] describes Cloud MapReduce (CMR), a new fully distributed architecture to implement the MapReduce programming model on top of the Amazon cloud OS. The nodes are responsible for pulling job assignments and their global status in order to determine their individual actions. The proposed architecture also uses queues to shuffle results from map tasks to reduce tasks. Map tasks are meant to write results as soon as they are available and reduce tasks need to filter out results from failed nodes, as well as duplicate results. The preliminary results of the work indicate that CMR is a practical system and its performance is comparable to Hadoop. Additionally, from the experimental results it can be seen that the usage of queues that overlap the map and shuffle phase seems to be a promising approach to improve MapReduce performance.

The works presented in [14, 42] are related to cloud environments, with particular emphasis on Amazon cloud. They discuss the MapReduce implementation on environments consisting of Spot Instances (SIs).[4]

In [14], a simple model has been represented. This model calculates the n-step probability, the expected lifetime of a VM, and the cost of termination, that is, the amount of time lost compared to having the set of machines stay up until completion of the job. Using the spot instances, in cases when there is no fault, the completion time may be speeded up. Otherwise, if there are failures, the job completion time may be longer than without using spot instances.

Liu's contribution [42] is a more mature proposal than the previous work. Here authors have tried to prove that their implementation, called Spot Cloud MapReduce, can take full advantage of the spot market, proposed by Amazon WS. As the name suggests, this implementation has been built on top of Cloud MapReduce (CMR), with additional changes:

- Modifying the split message format in the input queues (adding a parameter which indicates the position in the file where the processing should start).
- Saving the intermediate work when a node is terminated.
- Changing the commit mechanism to perform a partial commit.

[4]Spot instances are virtual machines resources in Amazon Web Services (WS), for which a user defines a maximum biding price that he/she is willing to pay. If there is no concurrence, the prices are lower and the possibility of using them is higher. But when the demand is higher, then Amazon WS has the right to stop your spot instances. If the spot instances are stopped by Amazon, the user does not pay, otherwise if the user decides to stop them before completing the normal hour, the user is obliged to pay for that consumption.

- Changing the way CMR determines the successful commit for a map split (electing a set of commit messages that is one more than the last key–value pair's offset).

The experimental evaluation shows that Spot CMR can work well in the spot market environment, significantly reducing cost by leveraging spot pricing.

11.4.5 Security Failure

The security concept is basically the absence of unauthorized access to, or handling of, system state [7]. This means that, authentication, authorization, and auditing go hand in hand, in order to ensure a system security. Whereas authentication refers to the initial identification of the user, the authorization determines the user rights, after he or she has entered into the system. Finally, the audit process represents an official user inspection (monitoring) to check if the user behaves according to its role. In other words, we could equate these terms with the pronouns *who* (authentication), *what* (authorization), and *when* (audit).

MapReduce's security in Hadoop is strictly linked to the security of HDFS; as the overall Hadoop security is grounded in HDFS, this means that other services including MapReduce store their state in HDFS. While Google's MapReduce does not make any assumption on security [19], early versions of Hadoop assumed that HDFS and MapReduce clusters would be used by a group of cooperating users within a secure environment. Furthermore, any access restriction was designed to prevent unintended operations that could cause accidental data loss, rather than to prevent unauthorized data access [40, 61].

The basic security definitions that include authentication, authorization, and auditing, were not present in Hadoop from the beginning. The authorization (managing user permissions) had been partially implemented. The auditing took place in the version 0.20 of Hadoop. The authentication was the last one, which came with Kerberos, an open-source network authentication protocol.

A user needs to be authenticated by the JobTracker before submitting, modifying, or killing any job. Since Kerberos authentication is bidirectional, even the JobTracker authenticates itself to the user; in this way, the user will be assured that the JobTracker is reliable. Additionally, each task is seen as an individual user, due to the fact that tasks now are run from the client perspective, the one which submitted the job, and not from the TaskTracker owner. In addition, the JobTracker's directory is not readable and writable by everyone as it happens with the task's working directories. During the authentication process, each user is given a token (also called a ticket) to authenticate once and pass credentials to all the tasks of a job; the token's default lifetime is meant to be around 8 h. While the NameNode creates these tokens, the JobTracker manages a token's renewal; token expiration is reasonably JobTracker dependent, in order not to expire prematurely for long-running jobs.

At the same year when Kerberos was implemented in Hadoop, another proposal called Airavat [51] tried to ensure security and privacy for MapReduce computations

on sensitive data. This work is an integration of mandatory access control (MAC) and differential privacy. MAC's duty is to assign security attributes to system resources, to constrain the interaction of subject with objects (e.g., subject can be a process, object can be a simple file). On the other side, differential privacy is a methodology which ensures that the aggregated computations maintain the integrity of each individual input. The evaluation of Airavat on several case studies shows flexibility in the maintenance of both accurate and private-preserving answers on runtimes within 32 % of the default Hadoop's MapReduce.

Apart from the different improvements in Hadoop security [31, 61], the work for preventing the Hadoop cluster from eavesdropping failures, has been slow. The explanation from the Hadoop community was that encryption is expensive in terms of CPU and I/O speed [52].

At the beginning, the encryption over the wire was dedicated only to some socket connections. In the case of Remote Procedure Call (RPC), an important protocol for communication between daemons in MapReduce, its encryption was added only after the main security improvement in Hadoop (by integrating Kerberos [36]). Most of the other encryption improvements (for instance, the shuffle phase encryption) came in a very recent Hadoop version [29], taking into consideration that Hadoop clusters may also hold sensitive information.

11.4.6 Apache Hadoop Reliability

Since its appearance in 2006, Apache Hadoop has undergone many releases [27]. Each of them has tried to improve different features of previous version, including fault tolerance. Table 11.2 shows Apache Hadoop 1.0 fault tolerance patches in a tree-like form, from the first Apache Hadoop 1.0 release (0.1.0) until release 1.2.1 which is the latest stable release up to the time of writing. These upgrades have played an important role in the later Hadoop evolution. An example of this is the introduction of speculative execution for reduce tasks, which caused many bugs in the previous days of its implementation. Therefore, the overall speculative execution mechanism was turned off by default later on, due to bugs in the framework. Actually the speculative execution mechanism was removed for some period, and later on, placed once again in the default functioning of the Apache Hadoop.

The Hadoop community was very active at the beginning, but this changed drastically through the years. A crucial reason for this was the existence of parallel projects, which tested new proposed features, but that were in their early phases (alpha or beta). Finally, a new release, Apache Hadoop 2.0, widely known as Hadoop YARN, was created. Table 11.3 shows Apache Hadoop 2.0 fault tolerance patches in a tree-like form, from the first Apache Hadoop 2.0 release (0.23.0) until release 2.7.1, which is the latest stable release up to the time of writing.

Table 11.2 Apache Hadoop 1.0: timeline of its fault tolerance patches

Year	Release	Patch
2006	0.1.0	The first release
	0.2.0	Avoid task rerun where it has previously failed (142); Don't fail reduce for a map impossibility allocation (169, 182); Five client attempts to JT before aborting a job (174); Improved heartbeat (186)
	0.3.0	Retry a single fail read, to not cause a failure task (311)
	0.7.0	Keep-alive reports, changed to seconds [10] rather than records [100] (556); Introduced killed state, to distinguish from failure state (560); Improved failure reporting (568); Ignore heartbeats from stale TTs (506)
	0.8.0	Make DFS heartbeats configurable (514); Re-execute failed tasks first (578)
	0.9.0	Introducing speculative reduce (76)
	0.9.2	Turn off speculative execution (827)
2007	0.10.0	Fully remove killed tasks (782)
	0.11.0	Add support for backup NNs, to get snapshotting (227, 959); Rack awareness added in HDFS (692)
	0.12.0	Change mapreduce.task.timeout to be per-job based (491); Make replication computation as a separate thread, to improve heartbeat in HDFS's NN (923); Stop assigning tasks to a dead TT (654)
	0.13.0	Distinguish between failed and killed task (1050); If nr of reduce tasks is zero, map output is written directly in HDFS (1216); Improve blacklisting of TTs from JTs (1278); Make TT expiry interval configurable (1276)
	0.14.0	Re-enable speculation execution by default (1336); Timed-out tasks counted as failures rather than killed (1472)
	0.15.0	Add metrics for failed tasks (1610)
2008	0.16.0	File permissions improvements (2336, 1298, 1873, 2659, 2431); Fine-grain control over speculative execution for map and reduce phase (2131); Heartbeat and task even queries interval, dependent on cluster size (1900); NN performance degradation from large heartbeat interval (2576)
	0.18.0	Completed map tasks should not fail if nr of reduce tasks is zero (1318)
	0.19.0	Introducing job recovery when JT restarts (3245); Add FailMon for hardware monitoring and analysis (3585)
2009	0.20.0	Improved blacklisting strategy (4305); Add test for injecting random failures of a task or a TT (4399); Fix heartbeating (4785, 4869); Fix JT (5338, 5337, 5394)
2010	0.20.202.0 (unreleased)	Change blacklist strategy (1966, 1342, 682); Greedily schedule failed tasks to cause early job failure (339); Fix speculative execution (1682); Add metrics to track nr of heartbeats by the JT (1680, 1103); Kerberos

(continued)

Table 11.2 (continued)

Year	Release	Patch
2011	0.20.204.0	TT should handle disk failures by reinitializing itself (2413)
	0.20.205.0	Use a bidirectional heartbeat to detect stuck pipeline (724); Kerberos improvements
2012	1.0.2	A single failed name dir can cause the NN to exit (2702)
	1.1.0	Lower minimum heartbeat between TT and JT for smaller clusters (1906)
2013	1.2.0	Looking for speculative tasks is very expensive in 1× (4499)
	1.2.1	The last stable release

Table 11.3 Apache Hadoop 2.0: timeline of its fault tolerance patches

Year	Release	Patch
2011	0.23.0	The first release; Lower minimum heartbeat interval for TaskTracker (MR-1906); Recovery of MR AM from failures (MR-279); Improve checkpoint performance (HDFS-1458)
2012	0.23.1	NM disk-failures handling (MR-3121); MR AM improvements: job progress calculations (MR-3568), heartbeat interval (MR-3718), node blacklisting (MR-3339, MR-3460), speculative execution (MR-3404); Active nodes list versus unhealthy nodes on the webUI and metrics (MR-3760)
	0.23.3	Timeout for Hftp connections (HDFS-3166); Hung tasks timeout (MR-4089); AM Recovery improvement (MR-4128)
	0.23.5	Fetch failures versus map restart (MR-4772); Speculation + Fetch failures versus hung job (MR-4425); INFO messages quantity on AM to RM heartbeat (MR-4517)
	2.0.0-alpha	NN HA improvements: fencing framework (HDFS-2179), active and standy states (HDFS-1974), failover (HDFS-1973), standbyNode checkpoints (HDFS-2291, HDFS-2924), NN health check (HDFS-3027), HA Service Protocol Interface (HADOOP-7455), in standby mode, client failing back and forth with sleeps (HADOOP-7896); haadmin with configurable timeouts for failover commands (HADOOP-8236)
	2.0.2-alpha	Encrypted shuffle (MR-4417); MR AM action on node health status changes (MR-3921); Automatic failover support for NN HA (HDFS-3042)

(continued)

Table 11.3 (continued)

Year	Release	Patch
2013	0.23.6	AM timing out during job commit (MR-4813)
	2.0.3-alpha	Stale DNs for writes (HDFS-3912); Replication for appended block (HDFS-4022); QJM for HDFS HA for NN (HDFS-3901, HDFS-3915, HDFS-3906); Kerberos issues (HADOOP-9054, HADOOP-8883, HADOOP-9070)
	2.1.0-beta	Reliable heartbeats between NN and DNs with LDAP (HDFS-4222); Tight DN heartbeat loop (HDFS-4656); Snapshots replication (HDFS-4078); Flatten NodeHeartbeatResponse (YARN-439); NM heartbeat handling versus scheduler event cause (YARN-365); NMTokens improvements (YARN-714, YARN-692); Resource blacklisting for Fifo scheduler (YARN-877); NM heartbeat processing versus completed containers tracking (YARN-101); AMRMClientAsync heartbeating versus RM shutdown request (YARN-763); Optimize job monitoring and STRESS mode versus faster job submission. (MR-3787); Timeout for the job.end.notification.url (MR-5066)
	2.1.1-beta	RM failure if the expiry interval is less than node-heartbeat interval (YARN-1083); AMRMClient resource blacklisting (YARN-771); AMRMClientAsync heartbeat versus runtime exception (YARN-994); RM versus killed application tracking URL (YARN-337); MR AM recovery for map-only jobs (MR-5468)
	2.2.0	MR job hang versus node-blacklisting feature in RM requests (MR-5489); Improved MR speculation, with aggressive speculations (MR-5533); SASL-authenticated ZooKeeper in ActiveStandbyElector (HADOOP-8315)
2014	2.3.0	SecondaryNN versus cache pools checkpointing (HDFS-5845); Add admin support for HA operations (YARN-1068); Added embedded leader election in RM (YARN-1029); Support blacklisting in the Fair scheduler (YARN-1333); Configuration to support multiple RMs (YARN-1232)
	2.4.0	DN heartbeat stucks in tight loop (HDFS-5922); Standby checkpoints block concurrent readers (HDFS-5064); Make replication queue initialization asynchronous (HDFS-5496); Automatic failover support for NN HA (HDFS-3042)
	2.4.1	Killing task causes ERROR state job (MR-5835)
	2.5.0	NM Recovery. Auxiliary service support (YARN-1757); Wrong elapsed time for unstarted failed tasks (YARN-1845); S3 server-side encryption (HADOOP-10568); Kerberos integration for YARN's timeline store (YARN-2247, HADOOP-10683, HADOOP-10702)

(continued)

Table 11.3 (continued)

Year	Release	Patch
	2.6.0	Encryption for hftp. (HDFS-7138); Optimize HDFS Encrypted Transport performance (HDFS-6606); FS input streams do not timeout (HDFS-7005); Transparent data at rest encryption (HDFS-6134); Operating secure DN without requiring root access (HDFS-2856); Work-preserving restarts of RM (YARN-556); Container-preserving restart of NM (YARN-1336); Changed NM to not kill containers on NM resync if RM work-preserving restart is enabled (YARN-1367); Recover applications upon NM restart (YARN-1354); Recover containers upon NM restart (YARN-1337); Recover NMTokens and container tokens upon NM restart (YARN-1341, YARN-1342); Time threshold for RM to wait before starting container allocations after restart/failover (YARN-2001); Handle app-recovery failures gracefully (YARN-2010); Fixed RM to load HA configs correctly before Kerberos login (YARN-2805); RM causing apps to hang when the user kill request races with AM finish (YARN-2853)
	2.6.1 (unreleased)	Make MR AM resync with RM in case of work-preserving RM restart (MR-5910); Support for encrypting Intermediate data and spills in local filesystem. (MR-5890); Wrong reduce task progress if map output is compressed (MR-5958)
2015	2.7.0	Block reports process during checkpointing on standby NN (HDFS-7097); DN heartbeat to Active NN may be blocked and expire if connection to Standby NN continues to time out (HDFS-7704); Active NN and standby NN have different live nodes (HDFS-7009); Expose Container resource information from NM for monitoring (YARN-3022); AMRMClientAsync missing blacklist addition and removal functionality (YARN-1723); NM fail to start with NPE during container recovery (YARN-2816); Fixed potential deadlock in RMStateStore (YARN-2946); NodeStatusUpdater cannot send already-sent completed container statuses on heartbeat (YARN-2997); Connection timeouts to NMs are retried at multiple levels (YARN-3238); Add configuration for MR speculative execution in MR2 (MR-6143); Configurable timeout between YARNRunner terminate the application and forcefully kill (MR-6263); Make connection timeout configurable in s3a. (HADOOP-11521)
	2.7.1	The last stable release

11.5 Other Data-Intensive Computing Systems

As mentioned before, MapReduce framework represents the de facto standard in the data-intensive computing community. However, there are many other projects, whose design and functionality differ from the basic MapReduce framework. Next, we present a collection of projects with significant impact in data-intensive computing.

11.5.1 Dryad/DryadLINQ

Knowing the benefits of Google's MapReduce, Microsoft designed its own data processing engine. In this way, Dryad [32] was introduced in 2007. After 1 year, Microsoft introduced a high-level language system for Dryad, composed of LINQ expressions, and called it DryadLINQ [67].

Dryad represents a general-purpose distributed execution engine, whose main target is coarse-grain data-parallel applications. In order to form a dataflow graph, Dryad combines computational *vertices* with communication *channels*. An application is run in Dryad by executing the vertices of the graph on a set of available machines, communicating as appropriate through files, TCP pipes, and shared-memory FIFOs.

Whereas mainly inspired from the (i) graphic processing units (GPUs) languages, (ii) Google's MapReduce, and (iii) parallel databases, Dryad is built also having in mind their disadvantages. As a consequence, Dryad as a framework allows the developer to have fine control over the communication graph, as well as the subroutines that live at its vertices. In order to describe the application communication patterns, and express the data transport mechanisms (files, TCP pipes, and shared-memory FIFOs) between the computation vertices, a Dryad application developer can specify an arbitrary directed acyclic graph (DAG). By directly specifying this kind of graph, the developer has also greater flexibility to easily compose basic common operations, leading to a distributed analogue of "piping" together traditional Unix utilities such as grep, sort, and head.

Dryad graph vertices are enabled to use an arbitrary number of inputs and outputs. It is assumed that the communication flow determines each job structure. Consequently many other Dryad mechanisms (such as resource management, fault tolerance, etc.) follow this pattern. A Dryad job is a directed acyclic graph where each vertex is a program and edges represent data channels. It is a logical computation graph that is automatically mapped onto physical resources by the runtime. At runtime each channel is used to transport a finite sequence of structured items.

Every Dryad job is coordinated by a master called "job manager" that runs either within the cluster or on a user's workstation, by having network access to the cluster. The job manager contains (i) the application-specific code, that allows to construct the job's communication graph, and (ii) library code, that allows to schedule the work across the available resources. Vertices transfer the data between them, therefore the job manager is only responsible for control decisions.

In Dryad, a failure of job manager means that the entire job fails, although other mechanisms (for example, checkpointing or replication) could be considered. As mentioned before, the fault tolerance policy works on a common case that all the vertex executions are deterministic. Due to failures, every vertex may be executed multiple times in sequence, or its many instances at any given time. If a vertex program runs slower than its peers, it gets duplicate executions; otherwise, after each heartbeat timeout, it gets re-executed.

The authors admit that for the nondeterministic vertices, Dryad does not provide any fault tolerance. However, this issue was planned as future goal of the framework.

Indeed, the Dryad fault-tolerant policy could be implemented by means of an extensible mechanism that allows nonstandard applications the possibility to modify their own behavior.

DryadLINQ represents a very important extension of Dryad, since it is a set of language extensions and the corresponding system that can automatically and transparently compile SQL, MapReduce, Dryad, and similar programs in a general-purpose language into distributed computations that can run on large-scale infrastructures. DryadLINQ does this in two ways, by (i) adopting an expressive data model of .NET objects; and (ii) by supporting general-purpose imperative and declarative operations on datasets within a traditional high-level programming language. A DryadLINQ program is based on LINQ expressions that are sequentially run on top of datasets. The DryadLINQ main duty is to translate the parallelism portion of the program into a distributed execution, ready to be executed on the Dryad engine.

11.5.2 SCOPE

SCOPE is a scripting language for massive data analysis [12], also coming from Microsoft. Its design has a strong resemblance to SQL, which was intentionally decided. SCOPE is a declarative language. As in the case of MapReduce, it hides the complexity of the lower platform and its implementation.

A user SCOPE script runs the basic SCOPE modules, (i) compiler, (ii) runtime, and (iii) optimizer, before initiating the physical execution. In order to manipulate input and output, SCOPE provides respective customizable commands, which are, *extract* and *output*. The *select* command of SCOPE is similar to the SQL one, with the main difference that subqueries are not allowed. To solve this issue, a user should rewrite complex queries with outer joins.

Apart from the SQL functionalities, SCOPE provides MapReduce-alike commands, which manipulate rowsets: *process*, *reduce*, and *combine*. The process command takes a rowset as input, and after processing each row, it outputs a sequence of rows. The reduce command takes a rowset as input, which has been grouped on the grouping columns specified in the ON clause. Then, it processes each group, and returns as output zero, one or multiple rows per group. The combine command takes two rowsets as input. It combines them depending on the requirements, and outputs a sequence of rows. The combine command is a binary operator.

Every SCOPE script resembles SQL, but its expression is implemented with C#, which needs to pass through the SCOPE compiler and optimizer, in order to be ready to run on parallel execution plan, which gets executed on the cluster as a Cosmos (Dryad) job. According to different evaluation experiments, SCOPE demonstrates its powerful query execution performance, that scales in a linear manner with respect to the cluster and data sizes.

The SCOPE fault tolerance policy relies on the Dryad's one. In SCOPE, Dryad has been renamed as Cosmos. In order to ensure availability, Cosmos replicates the data and its metadata by a quorum group of $2f + 1$ replicas, so as to tolerate f number

of failures. In order to ensure reliability, Cosmos enforces end-to-end checksums, to detect crash faulty components, whereas the data on disks is periodically scrubbed, to detect any corrupted or bit rot data before usage.

11.5.3 Nephele

In [60], authors present the basic foundations of Nephele, a novel research project at the time, whose aim was parallel data processing in dynamic clouds.

According to authors, state-of-the-art frameworks like MapReduce and Dryad are cluster-oriented models, which assume that their resources are a static set of homogeneous nodes. Therefore, these frameworks are not prepared enough for production clouds, whose exploitation of the dynamic resource allocation is a must. Based on this, they propose Nephele, a project which shares many similarities with Dryad, but providing more flexibility.

Nephele's architecture has a master–worker design pattern, with one Job Manager and many Task Managers. Each instance (aka VM) has its own Task Manager. As in Dryad, every Nephele job is expressed as a directed acyclic graph (DAG), where the vertices are tasks, and graph edges define the communication flow.

After writing the code for particular tasks, the user should define a Job Graph, consisting of linked edges and vertices. In addition, a user could specify other details, such as the number of subtasks in total, the number of subtasks per instance, instance types, etc. Each user Job Graph is then transformed into an Execution Graph by the Job Manager. Every specified manual configuration is taken into account by the Job Manager. Otherwise, the Job Manager places the default configuration according to the type of the respective job.

Compared to the default Hadoop on a small cloud infrastructure, the evaluation metrics are impressive and in favor to Nephele, showing better performance and resource utilization.

The main drawback of Nephele is that, due to its academic origin, it was not embraced by the research and industry community. One of the reasons could be its similarity with Dryad. An additional drawback of Nephele was its complexity, mainly compared to Hadoop MapReduce.

Finally, the Nephele fault tolerance policy still remains an open question. The authors admit the fault tolerance as desired but uncertain feature, because, as they mention, the optimal strategy is dependent on many parameters, among others, the task, its operations, the data, and the environment.

11.5.4 Spark

Spark is a novel framework for in-memory data mining on large clusters, whose main focus are applications that reuse the same dataset across multiple operations [69].

In this domain we found basically applications that are based on machine learning algorithms, such as text search, logistic regression, alternating least squares, etc. Spark programs are executed on top of the Mesos environment [28], where each of them needs its own driver (master) program to manage the control flow of the operations. Currently, Spark can also be run on top of other resource management frameworks, such as Hadoop YARN.

The main abstractions of Spark are:

- Resilient Distributed Datasets (RDDs) [68]. These are read-only collections of objects that are spread on cluster nodes.
- Parallel operations. These operations can be performed on top of the RDDs. Examples of these operations are *reduce*, *collect*, *foreach*, etc.
- Shared variables. These variables may be twofold: (i) broadcast variables, that copy the data value once to each worker; and (ii) accumulators, that can only "add" for being used as an associative operation, whose purpose (value) is readable by the driver only.

The most important abstraction of Spark are RDDs. Its primary use is to enable efficient in-memory computations on large clusters. This abstraction evolves in order to solve the main issues of parallel applications, whose intermediate results are very important in future multiple computations.

The main advantage of RDDs is the efficient data reuse, which comes with a good fault tolerance support. Its interface is based on coarse-grained transformations, by applying the same operation in parallel to a large amount of data. Each RDD is represented through a common interface, consisting of:

- A set of partitions. These are atomic pieces of the dataset.
- Set of dependencies. These are dependencies on parent RDDs.
- A function for computing the dataset from its parent.
- Metadata about its (i) partitioning scheme, and (ii) data placement.

Examples of applications that can take advantage of this feature are iterative algorithms and interactive data mining tools. Spark shows great results on some of these applications, outperforming Hadoop by 10× [68, 69, 71].

The Spark fault tolerance policy as well the scalability property are mainly based on the fundamental properties of MapReduce. In the case of the RDDs, they deploy the fault tolerance by means of the lineage concept, that is, if a node fails and an RDD partition is lost, the failed RDD partition is rebuilt from its parent datasets and eventually cached on other nodes. In addition to this, the Spark authors are planning to extend the fault tolerance in order to support other levels of persistence by means of in-memory replication across multiple nodes.

As part of Spark, the research community has proposed different modules, such as D-Streams [70, 71], GraphX [26], Spark SQL [6], and many others.

D-Streams represents a stream processing engine, an alternative to live queries (or operators) maintained by distributed event processing engines. Authors argue that it is better to have small batch computations using the advantages of in-memory

RDDs, instead of using long-live queries which are more costly and complex, mainly in terms of fault tolerance.

GraphX is a graph processing framework, which is built on top of Spark. GraphX represents an alternative to the classical graph processing systems, because it can efficiently handle iterative processing requirements of graph algorithm, unlike the general-purpose frameworks, such as MapReduce. The advantage of GraphX with respect to the classical graph processing frameworks is that it enables wider range of computations, and preserves the advantages of general-purpose dataflow frameworks, mainly the fault tolerance.

Finally, Spark SQL is another Apache Spark module, which enables an efficient intersection between relational processing and Spark functional programming. It does this by introducing the (i) *DataFrame API*, which enables the execution of relational operations, and (ii) *Catalyst*, which is another module that optimizes queries, and in addition simplifies data sources additions, and optimization rules, among others.

The main idea behind Apache Spark is to use iterative queries that are main memory based, which is also its main drawback. If the user request is not related to the previous and recent RDDs, the query process should start from the beginning. In this scenario, if we have to go back to the first iteration, MapReduce usually performs better than Spark.

11.6 Discussion

MapReduce programming model and the above-mentioned state-of-the-art improvements have filled many gaps on data processing requirements. However, there are many fault-tolerant issues that have not been solved yet. Below we address some open challenges.

It has been more than a decade since Google introduced MapReduce [19], but there are no many projects that analyze MapReduce with real-life traces and real-time large-scale infrastructures. What we can observe from the current situation is that leading companies process sensitive data with MapReduce. This data processing is highly confidential for their business and is a sufficient reason to avoid giving any details related to these results. As a consequence, most of the today's contributions are based on simulating failures [10, 21, 22, 35], or simulating the overall environment [58].

In addition, we are not aware of any large-scale comparison of datasets. The most common comparison we have seen in the data-intensive community field is the sorting time of 1 PB of data [53]. It would be desirable to have benchmark competitions of companies' data processing engines in the Big Data field. Indeed, it would be really challenging for the research and industry community to formalize a competition to measure different metrics of data-intensive processing frameworks, such as performance, scalability, or dependability.

Fault-tolerant abstraction models are another issue which is indeed missing in data-intensive computing systems. There are some projects that have offered different

approaches for this issue. Jin et al. [34] have derived a stochastic model that in some way predicts the performance of MapReduce applications under failures (crash failures). Their goal was to have a better understanding of fault tolerance mechanisms in Hadoop. A partial contribution to a theoretical failure model can be found in [58], which gives a simulation environment, and the possibility of fine-tuning different kinds of parameters. Another interesting work is presented in [48], where the authors propose a black box approach in order to detect and diagnose faults in MapReduce systems. While this contribution is valuable, it fails to solve half of the injected failures. Moreover, the mechanism has been evaluated only offline.

On the other side, there are studies that have expressed the view that this model is unworkable. In [10], it was explicitly stated that there is neither equivalent mathematical analysis that starts with a failure distribution and derives expected run time in the presence of failures nor optimization of the parameters of the system to minimize expected run time of the parallel applications under execution.

We consider that a theoretical model has not been presented yet, accepted by the whole community. We believe that additional challenges are important as well as possible, starting from the basic fault tolerance definitions, such as the failure detection.

While the handling and recovery in MapReduce fault tolerance via data replication and task re-execution seem to work well even at large scale [1, 37, 72], there is relatively little work on detecting failures in MapReduce. Accurate detection of failures is as important as failures recovery, in order to improve applications latencies and minimize resource waste. A new methodology to adaptively tune the timeout detector can significantly improve the overall performance of the applications, regardless of their execution environment [43]. Every MapReduce job should have its proper timeout, because in this way it could be possible to efficiently detect failures.

When reliability is improved, it is reasonable to think that these improvements are made at the expense of additional resource consumption. For instance, the replication improves the reliability, but increases the cost. The same occurs with cloning or speculating a task. More work is needed in improving reliability, maintaining similar resource utilization.

11.7 Summary

Many research projects have studied the MapReduce framework in the last few years, including its fault tolerance concepts and mechanisms. However, as far as we know, there is not a complete review of the research in MapReduce fault tolerance as an overall picture of what has been done and what is not solved yet. This survey addresses this gap, providing a systematic literature review on many contributions and an extensive analysis of new fault-tolerant mechanisms in MapReduce-based systems. Since there are other data-intensive approaches that have tried to go beyond the fundamental MapReduce functionality, we have also listed a relevant selection of these systems.

Finally, we have outlined some opening issues and key challenges for building efficient fault tolerance mechanisms in the MapReduce context. We argue that it is worth having a joint project from the different communities in order to handle issues such as a large-scale study of failures, where major companies could have a crucial role, and innovative and optimized failure models, where research communities can provide a significant contribution.

Acknowledgments The research leading to these results has received funding from the H2020 project reference number 642963 in the call H2020-MSCA-ITN-2014.

References

1. Ananthanarayanan, G., Agarwal, S., Kandula, S., Greenberg, A., Stoica, I., Harlan, D., Harris, E.: Scarlett: coping with skewed content popularity in mapreduce clusters. In: Proceedings of the Sixth Conference on Computer Systems, ACM, New York, NY, USA, EuroSys '11, pp. 287–300, (2011). http://doi.acm.org/10.1145/1966445.1966472
2. Ananthanarayanan, G., Ghodsi, A., Shenker, S., Stoica, I.: Effective straggler mitigation: Attack of the clones. In: Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation, USENIX Association, Berkeley, CA, USA, NSDI'13, pp. 185–198, (2013). http://dl.acm.org/citation.cfm?id=2482626.2482645
3. Ananthanarayanan, G., Hung, M.C.C., Ren, X., Stoica, I., Wierman, A., Yu, M.: GRASS: trimming stragglers in approximation analytics. In: Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation, USENIX Association, Berkeley, CA, USA, NSDI'14, pp. 289–302, (2014). http://dl.acm.org/citation.cfm?id=2616448.2616475
4. Ananthanarayanan, G., Kandula, S., Greenberg, A., Stoica, I., Lu, Y., Saha, B., Harris, E.: Reining in the outliers in map-reduce clusters using Mantri. In: Proceedings of the 9th USENIX conference on Operating Systems Design and Implementation, USENIX Association, Berkeley, CA, USA, OSDI'10, pp. 1–16, (2010). http://dl.acm.org/citation.cfm?id=1924943.1924962
5. Apache Zookeeper: (2015). http://zookeeper.apache.org/
6. Armbrust, M., Xin, R.S., Lian, C., Huai, Y., Liu, D., Bradley, J.K., Meng, X., Kaftan, T., Franklin, M.J., Ghodsi, A., Zaharia, M.: Spark sql: Relational data processing in spark. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, SIGMOD '15, pp. 1383–1394 (2015). http://doi.acm.org/10.1145/2723372.2742797
7. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.E.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secure Comput. **1**(1), 11–33 (2004)
8. Barborak, M., Dahbura, A., Malek, M.: The consensus problem in fault-tolerant computing. ACM Comput. Surv. **25**(2), 171–220 (1993). http://doi.acm.org/10.1145/152610.152612
9. Borthakur, D., Gray, J., Sarma, J.S., Muthukkaruppan, K., Spiegelberg, N., Kuang, H., Ranganathan, K., Molkov, D., Menon, A., Rash, S., Schmidt, R., Aiyer, A.: Apache Hadoop goes realtime at Facebook. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, ACM, New York, NY, USA, SIGMOD '11, pp. 1071–1080 (2011). http://doi.acm.org/10.1145/1989323.1989438
10. Bressoud, T.C., Kozuch, M.A.: Cluster fault-tolerance: An experimental evaluation of checkpointing and MapReduce through simulation. In: Proceedings of the 2009 IEEE International Conference on Cluster Computing and Workshops, IEEE, pp. 1–10 (2009). http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5289185
11. Cachin, C., Guerraoui, R., Rodrigues, L.: Introduction to Reliable and Secure Distributed Programming (2. ed.). Springer (2011)

12. Chaiken, R., Jenkins, B., Larson, P., Ramsey, B., Shakib, D., Weaver, S., Zhou, J.: SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets. Proc. VLDB Endow **1**(2), 1265–1276 (2008). http://dl.acm.org/citation.cfm?id=1454159.1454166
13. Chen, Q., Liu, C., Xiao, Z.: Improving mapreduce performance using smart speculative execution strategy. IEEE Trans. Comput. **63**(4), 954–967 (2014). doi:10.1109/TC.2013.15
14. Chohan, N., Castillo, C., Spreitzer, M., Steinder, M., Tantawi, A., Krintz, C.: See spot run: using spot instances for MapReduce workflows. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, USENIX Association, Berkeley, CA, USA, HotCloud'10, pp. 7–7 (2010). http://dl.acm.org/citation.cfm?id=1863103.1863110
15. Clement, A., Kapritsos, M., Lee, S., Wang, Y., Alvisi, L., Dahlin, M., Riche, T.: Upright cluster services. In: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, ACM, New York, NY, USA, SOSP '09, pp. 277–290 (2009). http://doi.acm.org/10.1145/1629575.1629602
16. Condie, T., Conway, N., Alvaro, P., Hellerstein, J.M., Elmeleegy, K., Sears, R.: MapReduce online. In: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, USENIX Association, Berkeley, CA, USA, NSDI'10, pp. 21–21 (2010). http://dl.acm.org/citation.cfm?id=1855711.1855732
17. Correia, M., Costa, P., Pasin, M., Bessani, A., Ramos, F., Verissimo, P.: On the feasibility of byzantine fault-tolerant mapreduce in clouds-of-clouds. In: 2012 IEEE 31st Symposium on Reliable Distributed Systems (SRDS), pp. 448–453 (2012). doi:10.1109/SRDS.2012.46
18. Costa, P., Pasin, M., Bessani, A., Correia, M.: Byzantine Fault-Tolerant MapReduce: Faults are Not Just Crashes. In: Proceedings of the 3rd IEEE Second International Conference on Cloud Computing Technology and Science, IEEE Computer Society, Washington, DC, USA, CLOUDCOM '11, pp. 17–24 (2010). http://dx.doi.org/10.1109/CloudCom.2010.25
19. Dean, J., Ghemawat, S., Inc, G.: MapReduce: simplified data processing on large clusters. In: Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation, USENIX Association, OSDI'04 (2004)
20. Dean, J.: Building software systems at google and lessons learned. Stanford EE Computer Systems Colloquium (2010). http://www.stanford.edu/class/ee380/Abstracts/101110-slides.pdf
21. Dinu, F., Ng, T.S.E.: Hadoop's Overload Tolerant Design Exacerbates Failure Detection and Recovery. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, ACM, New York, NY, USA, NetDB'11, pp. 1–7 (2011)
22. Dinu, F., Ng, T.E.: Understanding the effects and implications of compute node related failures in Hadoop. In: HPDC '12: Proceedings of the 21st International Symposium on High-Performance Parallel and Distributed Computing, ACM, New York, NY, USA, pp. 187–198 (2012). http://doi.acm.org/10.1145/2287076.2287108
23. Facebook, Inc.: (2015). https://www.facebook.com/
24. Facebook, I.: Under the Hood: Scheduling MapReduce jobs more efficiently with Corona (2012). http://www.facebook.com/notes/facebook-engineering/under-the-hood-scheduling-mapreduce-jobs-more-efficiently-with-corona/10151142560538920
25. Fedak, G., He, H., Cappello, F.: BitDew: A data management and distribution service with multi-protocol file transfer and metadata abstraction. J Netw. Compu. Appl. **32**(5), 961–975 (2009)
26. Gonzalez, J.E., Xin, R.S., Dave, A., Crankshaw, D., Franklin, M.J., Stoica, I.: Graphx: Graph processing in a distributed dataflow framework. In: Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation, USENIX Association, Berkeley, CA, USA, OSDI'14, pp. 599–613 (2014). http://dl.acm.org/citation.cfm?id=2685048.2685096
27. Hadoop Releases: (2015). http://hadoop.apache.org/releases.html
28. Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A.D., Katz, R., Shenker, S., Stoica, I.: Mesos: A Platform for Fine-grained Resource Sharing in the Data Center. In: Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, USENIX Association, Berkeley, CA, USA, NSDI'11, pp. 22–22 (2011). http://dl.acm.org/citation.cfm?id=1972457.1972488

29. How-to: Set Up a Hadoop Cluster with Network Encryption: (2013). http://blog.cloudera.com/blog/2013/03/how-to-set-up-a-hadoop-cluster-with-network-encryption/
30. Ibrahim, S., Phuong, T.A., Antoniu, G.: An Eye on the Elephant in the Wild: A Performance Evaluation of Hadoop's Schedulers Under Failures. In: Workshop on Adaptive Resource Management and Scheduling for Cloud Computing (ARMS-CC-2015), held in conjunction with PODC'15 (2015)
31. Introduction to Hadoop Security: (2013). http://www.cloudera.com/content/cloudera/en/home.html
32. Isard, M., Budiu, M., Yu, Y., Birrell, A., Fetterly, D.: Dryad: distributed data-parallel programs from sequential building blocks. In: Proceedings of the 2nd ACM SIGOPS/EuroSys 2007, ACM, New York, NY, USA, EuroSys '07, pp. 59–72 (2007). http://doi.acm.org/10.1145/1272996.1273005
33. Jin, H., Ibrahim, S., Qi, L., Cao, H., Wu, S., Shi, X.: The MapReduce programming model and implementations. Cloud Computing: Principles and Paradigms pp. 373–390. doi:10.1002/9780470940105.ch14
34. Jin, H., Qiao, K., Sun, X.H., Li, Y.l.: Performance under Failures of MapReduce Applications. In: Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE Computer Society, Washington, DC, USA, CCGRID '11, pp. 608–609 (2011). http://dx.doi.org/10.1109/CCGrid.2011.84
35. Jin, H., Sun, X.H.: Performance comparison under failures of MPI and MapReduce: An Analytical Approach. Future Gener. Comput. Syst. **29**(7), 1808–1815 (2013). http://dx.doi.org/10.1016/j.future.2013.01.013
36. Kerberos: The Network Authentication Protocol: (2015). http://web.mit.edu/kerberos/
37. Ko, S.Y., Hoque, I., Cho, B., Gupta, I.: Making cloud intermediate data fault-tolerant. In: Proceedings of the 1st ACM Symposium on Cloud Computing, ACM, New York, NY, USA, SoCC '10, pp. 181–192 (2010). http://doi.acm.org/10.1145/1807128.1807160
38. Ko, S.Y., Hoque, I., Cho, B., Gupta, I.: On availability of intermediate data in cloud computations. In: Proceedings of the 12th conference on Hot topics in operating systems, USENIX Association, Berkeley, CA, USA, HotOS'09, pp. 6–6 (2009). http://dl.acm.org/citation.cfm?id=1855568.1855574
39. Lin, H., Ma, X., Archuleta, J., Feng, W.c., Gardner, M., Zhang, Z.: MOON: MapReduce On Opportunistic eNvironments. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, ACM, New York, NY, USA, HPDC '10, pp. 95–106 (2010). http://doi.acm.org/10.1145/1851476.1851489
40. Lin, J., Dyer, C.: Data-Intensive Text Processing with MapReduce. Tech. rep., University of Maryland, College Park (2010)
41. Liu, H., Orban, D.: Cloud MapReduce: A MapReduce implementation on top of a cloud operating system. In: 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 464–474 (2011). doi:10.1109/CCGrid.2011.25
42. Liu, H.: Cutting MapReduce Cost with Spot Market. In: Proceedings of the 3rd USENIX Conference on Hot topics in Cloud Computing, USENIX Association, Berkeley, CA, USA, HotCloud'11, pp. 5–5 (2011). https://www.usenix.org/conference/hotcloud11/cutting-mapreduce-cost-spot-market
43. Memishi, B., Ibrahim, S., Pérez, M.S., Antoniu, G.: On the Dynamic Shifting of the MapReduce Timeout. In: Kannan, R., Rasool, R.U., Jin, H., Balasundaram, S. (eds) Managing and Processing Big Data in Cloud Computing, IGI Global, Hershey, Pennsylvania (USA), pp. 1–22 (2016). doi:10.4018/978-1-4666-9767-6
44. Memishi, B., Pérez, M.S., Antoniu, G.: Diarchy: An Optimized Management Approach for MapReduce Masters. Procedia Comput. Sci. **51**, 9–18 (2015). http://www.sciencedirect.com/science/article/pii/S1877050915009874. International Conference On Computational Science, ICCS Computational Science at the Gates of Nature
45. Microsoft, Inc.: (2015). http://www.microsoft.com/
46. Mone, G.: Beyond Hadoop. Commun. ACM **56**(1), 22–24 (2013). http://doi.acm.org/10.1145/2398356.2398364

47. Okorafor, E., Patrick, M.K.: Availability of Jobtracker machine in Hadoop/MapReduce Zookeeper coordinated clusters. Adv. Comput.: An Int. J. **3**(3), 19–30 (2012). http://www.chinacloud.cn/upload/2012-07/12072600543782.pdf
48. Pan, X., Tan, J., Kavulya, S., Gandhi, R., Narasimhan, P.: Ganesha: blackBox diagnosis of MapReduce systems. SIGMETRICS Perform. Eval. Rev. **37**(3), 8–13 (2010). http://doi.acm.org/10.1145/1710115.1710118
49. Phan, T.D., Ibrahim, S., Antoniu, G., Bougé, L.: On Understanding the energy impact of speculative execution in Hadoop. In: IEEE International Conference on Green Computing and Communications (GreenCom 2015), Sydney, Australia (2015). https://hal.inria.fr/hal-01238055
50. RedHat: A guide for developers using the JBoss Enterprise SOA Platform (2008). http://www.redhat.com/docs/en-US/JBoss_SOA_Platform/4.3.GA/html/Programmers_Guide/index.html,programmersGuide
51. Roy, I., Setty, S.T.V., Kilzer, A., Shmatikov, V., Witchel, E.: Airavat: security and privacy for MapReduce. In: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, USENIX Association, Berkeley, CA, USA, NSDI'10, pp. 20–20 (2010). http://dl.acm.org/citation.cfm?id=1855711.1855731
52. Shih, J.: Hadoop security overview—from security infrastructure deployment to high-level services. Hadoop & BigData Technology Conference (2012). www.hbtc2012.hadooper.cn/subject/keynotep8shihongliang.pdf
53. Sorting 1PB with MapReduce: (2013). http://googleblog.blogspot.com/2008/11/sorting-1pb-with-mapreduce.html
54. Stonebraker, M., Abadi, D., DeWitt, D.J., Madden, S., Paulson, E., Pavlo, A., Rasin, A.: MapReduce and parallel DBMSs: friends or foes? Commun. ACM **53**:64–71 (2010). http://doi.acm.org/10.1145/1629175.1629197
55. Tang, B., Moca, M., Chevalier, S., He, H., Fedak, G.: Towards MapReduce for Desktop Grid Computing. In: Proceedings of the 2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, IEEE Computer Society, Washington, DC, USA, 3PGCIC '10, pp. 193–200 (2010). http://dx.doi.org/10.1109/3PGCIC.2010.33
56. The Apache Hadoop Project: (2015). http://hadoop.apache.org/
57. Vavilapalli, V.K., Murthy, A.C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., Saha, B., Curino, C., O'Malley, O., Radia, S., Reed, B., Baldeschwieler, E.: Apache Hadoop YARN: Yet Another Resource Negotiator. In: Proceedings of the 4th Annual Symposium on Cloud Computing, ACM, New York, NY, USA, SoCC '13, p. 5:1–5:16 (2013). http://doi.acm.org/10.1145/2523616.2523633
58. Wang, G., Butt, A.R., Pandey, P., Gupta, K.: A simulation approach to evaluating design decisions in MapReduce setups. In: 17th Annual Meeting of the IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, IEEE, MASCOTS 2009, pp. 1–11
59. Wang, F., Qiu, J., Yang, J., Dong, B., Li, X., Li, Y.: Hadoop high availability through metadata replication. In: Proceedings of the First International Workshop on Cloud Data Management, ACM, New York, NY, USA, CloudDB '09, pp. 37–44 (2009). http://doi.acm.org/10.1145/1651263.1651271
60. Warneke, D., Kao, O.: Nephele: Efficient parallel data processing in the cloud. In: Proceedings of the 2Nd Workshop on Many-Task Computing on Grids and Supercomputers, ACM, New York, NY, USA, MTAGS '09, pp. 8:1–8:10 (2009). http://doi.acm.org/10.1145/1646468.1646476
61. White, T.: Hadoop—The Definitive Guide: Storage and Analysis at Internet Scale (3. ed., revised and updated). O'Reilly (2012)
62. Xiao, Z., Xiao, Y.: Achieving accountable MapReduce in cloud computing. Future Gener. Comput. Syst. **30**, 1–13 (2014). http://dx.doi.org/10.1016/j.future.2013.07.001
63. Xu, H., Lau, W.C.: Optimization for speculative execution in a MapReduce-like cluster. In: 2015 IEEE Conference on Computer Communications, INFOCOM 2015, Kowloon, Hong Kong, April 26–1May 1, 2015, pp. 1071–1079. http://dx.doi.org/10.1109/INFOCOM.2015.7218480

64. Xu, H., Lau, W.C.: Speculative execution for a single job in a mapreduce-like system. In: 2014 IEEE 7th International Conference on Cloud Computing (CLOUD), pp. 586–593 (2014). doi:10.1109/CLOUD.2014.84
65. Yahoo! Inc: (2015). http://www.yahoo.com/
66. Yildiz, O., Ibrahim, S., Phuong, T.A., Antoniu, G.: Chronos: Failure-aware scheduling in shared Hadoop clusters. In: IEEE International Conference on Big Data (BigData 2015), pp 313–318 (2015). doi:10.1109/BigData.2015.7363770
67. Yu, Y., Isard, M., Fetterly, D., Budiu, M., Erlingsson, U., Gunda, P.K., Currey, J.: DryadLINQ: a system for general-purpose distributed data-parallel computing using a high-level language. In: Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation, USENIX Association, Berkeley, CA, USA, OSDI'08, pp. 1–14 (2008). http://dl.acm.org/citation.cfm?id=1855741.1855742
68. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M.J., Shenker, S., Stoica, I.: Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, USENIX Association, Berkeley, CA, USA, NSDI'12, pp. 2–2 (2012). http://dl.acm.org/citation.cfm?id=2228298.2228301
69. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: Cluster computing with working sets. In: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, USENIX Association, Berkeley, CA, USA, HotCloud'10, pp. 10–10 (2010). http://dl.acm.org/citation.cfm?id=1863103.1863113
70. Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., Stoica, I.: Discretized streams: Fault-tolerant streaming computation at scale. In: Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, ACM, New York, NY, USA, SOSP '13, pp. 423–438 (2013). http://doi.acm.org/10.1145/2517349.2522737
71. Zaharia, M., Das, T., Li, H., Shenker, S., Stoica, I.: Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters. In: Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Ccomputing, USENIX Association, Berkeley, CA, USA, HotCloud'12, pp. 10–10 (2012). http://dl.acm.org/citation.cfm?id=2342763.2342773
72. Zaharia, M., Konwinski, A., Joseph, A.D., Katz, R., Stoica, I.: Improving MapReduce performance in heterogeneous environments. In: Proceedings of the 8th USENIX conference on Operating Systems Design and Implementation, USENIX Association, Berkeley, CA, USA, OSDI'08, pp. 29–42 (2008). http://dl.acm.org/citation.cfm?id=1855741.1855744
73. Zhu, H., Haopeng, C.: Adaptive failure detection via heartbeat under Hadoop. In: Proceedings of the 2011 IEEE Asia-Pacific Services Computing Conference, IEEE, New York, NY, USA, ApSCC'11, pp. 231–238 (2011)

Chapter 12
Big Data Security

Agnieszka Jakóbik

12.1 Introduction

Big Data (BD) systems become essential element for scientists, business executives, healthcare systems, advertising, online sales systems, companies, and governments. It enables gathering and processing huge data sets in areas such as Internet search, media, finance, meteorology, genomics, biology, environmental research, social media. Big Data systems offer services like business intelligence, cloud computing and data storage, testing, machine learning and natural language processing, data visualization, data mining, distributed file systems, and many more.

Big Data systems support modern everyday life by being the engine behind www.eBay.com, www.Amazon.com, Google or Facebook. Big Data systems are also necessarily and very valuable part of many advanced projects dedicated mainly for researchers.

Regardless of the objective of data collecting and processing, a huge amount of critical information is gathered by such systems. This data is stored, processed, and used by the data owners, BD system vendors and third-party organizations. Data confidentiality, data provenience, and management of access for the data is very important. The data volume and velocity of data gathering and processing causes security problems that are specific to Big Data systems itself.

The paper is presenting short study about the most critical issues concerning security problems in Big Data Systems. First, main concepts and definitions are presented. Then, Big Data security is considered from several point of view: user/vendor security, data security, and security of computer systems. Chosen methods for security assurance in BD systems are presented. Traditional cryptography solutions are presented and methods dedicated to BD systems only. As far as trust management is considered international standards and institutions are presented. Secure

A. Jakóbik (✉)
Cracow University of Technology, ul. Warszawska 24, 31-155 Cracow, Poland
e-mail: agneskrok@gmail.com

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_12

communication protocols and infrastructure design for security are also described. Case studies of Hewlett Packard, IBM Microsoft, and Teradata solutions for BD systems are depicted.

The following survey is not a state of art concerning BD security, but an overview of problems and solutions for further research investigation. The aim of the article was to provide broad overview of issues and to be a good starting point for researchers from the field of Cryptography, Cloud systems, computer networks, complex systems modeling, and simulation as well as academic users who are employing such systems during their scientific research.

Furthermore, understanding BD systems and their aware and responsible usage is essential for any member of Internet of Things society.

12.2 Main Concepts and Definitions

The Big Data Systems (BD) are defined by three features: huge volume of data gathered and proceeded, promptness velocity of data flow, and large variety of data itself. Petabyte-scale data gathered by such systems are processed using dedicated techniques, [1]. The chosen services offered by vendors are: Virtual Servers, Containers, Web Deployment, Load Balancing, Database Migration, Caching, Data Warehousing, Object Storage, File System Storage, Archive Storage, Data Transport, Business Intelligence, Machine Learning, Streaming Data, Source Code Management, Code Deployment, Continuous Delivery, API Management, Application Streaming, Searching engines, Email servers, large variety of data Analysis itself, real-time stream processing. The data in BD systems is treated differently in comparison to the traditional systems. The splitting and conjunction of the data is important for data updating and processing because data is dispersed widely. It was stated also in [2] that 80 % of the effort involved in processing data is cleaning and changing the incoming stream of data into applicative form.

Big Data projects rarely involve single organization or company. A lot of communication between many participants is incorporated in the process. Moreover, a lot of analysis in the fields of engineering, computer science, physics, economics, and life sciences is made using machine learning and automatic methods. These approaches are beyond the direct control of human, [3]. During all this stages: reliance, integrity, confidentiality, genuineness, and availability have to be monitored and provided nearly in the real time and on the massive scale.

The systems and social media based on Big Data systems such as like LinkedIn, Netflix, Facebook, Twitter, Expedia, big sales companies, banking companies, and a lot of other organizations are generating enormous economic, social, and political impact into modern society. The decision made based on Big Data analytic has large consequences in the real world, [4].

Therefore, it is very important to assure the save acquisition, storage, and usage of such data. Especially, when the facts about peoples attributes, behavior, preferences,

Table 12.1 Data characteristics in BD systems: composition of each source, data type, and features is possible in single data set

Source	Type	Features
Single user	At rest	Variety
Corporation	In motion	Velocity
Social media	Being processed	Variability
Sensors	Distributed	Incomplete
Automated systems	Shared	Biased

relationships, and locations are being gathered, processed, and used in favor of many subjects.

The problem of assuring security in BD systems is very complicated and not very well recognized. The rapid growth of popularity of such systems imposes the necessity to formulation and formal description of this problem. The main responsibility relays on BD systems providers but improper, irresponsible or negligent and users action may cause a lot of damage (Table 12.1).

12.3 Big Data Security

This section presents security problems classification that concerns Big Data systems.

Information security

ISO/IEC 27002 norm introduces definition of *information security* as the the protection of information from a wide range of threats in order to ensure business continuity, minimize business risk, and maximize return on investments and business opportunities. Information security is accomplished in the form of policies, Service Level Agreements, processes, procedures, organizational structures, software and hardware functions. These components need to be established, implemented, monitored, reviewed, and improved constantly. *Security of computer systems* are specified as the following C-I-A model, [5]:

- the preservation of **confidentiality** (C)—information should be accessible only to those authorized to have access,
- **integrity** (I)—information should be accurate and complete and
- **availability** (A)—ensuring that authorized users have access to information when required).

Security of the system user

Security of the system user may be considered as far as the role of user is concerned:

- security from data uploaders point of view, e.g., the right to privacy,
- security from data users point of view, e.g., the right for not being deceived by having wrongly uploaded or manipulated data,

- security from society point of view, e.g., not being abused by those who obtained information from BD sources.

Computing security, security for computing, computing for security

Another classification proposed in [6] is the following:

- **BD computing security** as the group of problems concerning security of a computing systems infrastructure,
- **security for computing** deals with issues connected with the trust on the services that users are employing using Big Data Technology,
- *BD for security* involves the usage of BD technologies to develop and deliver security solutions for massive scale recipients.

SO/IEC 27002:2013 classification

Next classification was specified in the ISO/IEC 27002:2013, [7] standard for information security published by the International Organization for Standardization (ISO)

1. Organization of Information Security—responsibilities for information security assigned to individual persons and duties should be allocated across roles and to avoid conflicts of interest and prevent inappropriate information leaks.
2. Human Resource Security—security responsibilities monitored during recruiting employees, workers, and temporary staff.
3. Asset Management—data segregated according to the security protection that is necessary, and security policies diversified appropriately.
4. Access Control—data access restricted and controlled over roles and privileges.
5. Cryptography—cryptography protocols constantly monitored due to their validity.
6. Physical and environmental security—physical barriers, with physical entry controls and working procedures protecting staff offices, data centers storage places, delivery/loading areas against unauthorized access.
7. Operation Security–protection against harmful software, backup done regularly, logging and monitoring as far as users and administrators activities, incidents, faults and security violation, synchronizing clocks, operational software life cycle monitoring, operational systems updating.
8. Communication security—all networks and network services properly secured.
9. System acquisition, development, and maintenance—all software installed monitored, the development environment protected, and external resources controlled.
10. Information security incident management—incidents reported, responded as soon as possible to and learn from
11. Compliance with legal and contractual requirements enforced.

Variety, Volume, Velocity-related security issues

Aforementioned security fields have to be considered together with treats for security resulting from with three feathers of Big Data systems itself, [8].

Table 12.2 Data life cycle in BD systems and security problems connected to each stage of data processing

Gathering	Warehousing	BD system	BD application
Massive scale	Persistent storage	Raw form storage	Real-time processing
Authentication	Authorization	Confidentiality	Integrity
Key safety	Key management	Up to date procedures	Valid hash functions
Lean clients methods	Velocity tools	Volume solutions	Fast processing
Provenience	Trusted transferring	Safe storage	Correctness

- Variety: changing traditional relational database into nonrelational databases enforces different kind of methods as far as the need to disable reading the data by unauthorized persons, storing data in the ciphered form, and disable the identity detection from anonymized data sets by correlating with public databases.
- Volume: The volume of Big Data enforces the storage in multitiered storage media. That causes additional problem of securing data integrity and inviolability when data are segmented, moving between tiers and merged.
- Velocity: The velocity of data collection enforces usage of such a security methods, algorithms, and hardware that can proceed fast enough not to disrupt the flow of data.
- Variability: Security and privacy requirements have to be ensured also when the data are moved to another vendor or are no longer valid and should be erased (Table 12.2).

Big Data may be collected from variety of end points. The roles that are incorporated into authentication and authorization include more types than traditional system providers, consumers, data owners. In particular, mobile users and social network users have to be taken into consideration. Data aggregation and dissemination must be secured inside the storage system, because lean clients do not have enough computing power to perform necessary numerical operations involved in data ciphering or hashing. The secure availability of data on demand by a broad group of stakeholders have to be ensure using readily understood security frameworks, dedicated to users who do not have any knowledge in the subject. Data search and selection can also lead to privacy or security concerns. Legacy security solutions need to be retargeted to Big Data. They have to be used in High Performance Computing resources. Attention must be given particularly to systems that are collecting data from fully public domains. Methods to prevent adversarial manipulation and preserve integrity of data have to be incorporated. Its extreme scalability causes that Big Data systems are also sensitive to recovery methods and practices. Traditional backup tools are impractical. Big Data systems enforces that monitoring and prevention and protection against hacker attacks have to be scaled enormously. Security and privacy may be weakened by unintentional operations made by uneducated users. Therefore educational policies for end users have to be considered and the methods for detecting accidentally caused security threads, [9].

Data confidentiality provenance and access management

1. Data Confidentiality. Confidentiality of data in Big Data systems have to be assured during three stages of data processing: in transit, at rest, during processing. Each stage requires and enables different kind of protocols to compromise between effectiveness and computational cost. Moreover, methods of computing on encrypted data have to be used especially during searching and reporting. Aggregating data without compromising privacy, and data anonymization methods have to be incorporated.
2. Provenance. A mechanisms to validate if input data were uploaded from authenticated, trustful sources. For example, digital signatures may be incorporated. Moreover, validation at a syntactic level and semantic validation is obligatory.
3. Identity and Access Management. Key management methods have to take into consideration greater variety of user types and have to be suitable for volume, velocity, variety, and variability of data. In Big Data systems virtualization layer identity, application layer identity, end-user layer identity, and identity of provider is necessary. Key Management life cycle involves: generating Keys, assuring nonlinear Key spaces, transferring Keys, verifying Keys, using Keys, updating Keys, storing Keys, backuping Keys, compromised management Keys and destroying Keys, for all participants involved in the process and on the massive scale, [10].

12.4 Methods and Solutions for Security Assurance BD Aystems

Various cryptographic methods and protocols used in so called '**traditional cryptography**' may be transformed and successfully used also in BD approaches. The wide class of such methods includes: symmetric Cryptography methods, One-Way Hash Functions, Public-Key Cryptography, Digital Signatures with Encryption, Authentication and Key-Exchange Cartographic Protocols, Multiple-Key Public-Key Cryptography, Secret Splitting and Secret Sharing protocols, Undeniable Digital Signatures, Designated Confirmed Signatures, Blind Signatures, Identity-Based Public-Key Cryptography methods, [11, 12].

12.4.1 Authorization and Authentication Methods

Authentication methods dedicated to the Big Data systems requires scalability and instant high performance. Moreover machines on which such a systems are processing

are equipped with massively parallel software running on many commodity computers in distributed computing frameworks. For this reason, all protocols and schemes have to be optimized to such environment.

Master, scheduler, workers architecture implications.

Big Data (BD) systems are built from few components: Master System (MS) receives data from data sources, then task Distribution function is applied to distribute tasks to the workers/slaves of the system (Cooperated System—CS), after Data distribution function distributes data to the cooperated systems and after tasks are finished Result Collection function gathers information and sends it to users. These functions may be installed in different hosts and all stages needs authorization and authentication methods to obtain the proper flow of the data. Security techniques for Access Control for each component is necessary. The authorization is much more complicated than non-Big Data systems, because of the necessity of synchronizing access privileges between the MS and CSs.

Security Agreements and Service Level Agreement implications.

Security Agreement is made between data uploaders (BD source providers) and the MS. It helps to categorize security classes of data sources. The aim of SA is to make decisions about levels of security (or trust) that is required by the data uploaders (e.g., different security levels for emails, and free stock photographs).
Trust Workers List (TCSL) lists the trusted Workers and categorizes them by the security classes. **MS access Policy** is characterizing a set of access rules that are imposed by Master System into Workers. **Worker access Policy** is the list of rules for access to the distributed resources managed by the particular worker (e.g., disk space, Virtual Machines).

Master—workers secure data flow example

To apply proper data flow, coordination of data up loaders with Master System by Security Agreement have to be completed, then Master System is gathering information about available Workers. After matching, security needs of Master Systems with security offers by workers, the data are uploaded, task are formulated, and sending to the Workers. The proposed scheme can be formalized by the following rules: A set of security classes from c_1 to c_n combined as

$$C = \{c_1, \ldots, c_n\} \tag{12.1}$$

A set of BD source providers from bd_1 to bd_n combined as

$$BD = \{bd_1, \ldots, bd_n\} \tag{12.2}$$

A set of Cooperated System from cs_1 to cs_n in the form of the set

$$CS = \{cs_1, \ldots, cs_n\} \tag{12.3}$$

A set of security attributes from at_1 to at_n in the set

$$AT = at_1, \ldots, at_n$$

A set of (bd_x, c_x) pairs in the set SA in BD xi C. A set of (cs_x, c_x) pairs in the set TCSL in CS xi C. A set of MS policy rules from mp_1 to mp_n in the set

$$MSP = \{mp_1, \ldots, mp_n\} \tag{12.4}$$

where

$$mpi = (at_x, a_x, bd_x)$$

is a tuple where, at_x from AT is an attributes, a_x is an action, and bd_x from BD is a source provider that means subject with attribute at_x is permitted to perform action a_x on object from bd_x A set of AC policy rules for CS CSi collected in the set

$$CSPi = \{cspi_1, \ldots, cspi_n\} \tag{12.5}$$

where each

$$cspii = (bd_x, a_x, rs_x) \tag{12.6}$$

is a tuple where bd_x is the element from BD, a_x is an action, and rs_x is a local resource from CSi that means subject from bd_x is permitted to perform action a_x on object rsx

Let $BDU = (u, a_u, bd_u)$ be the user request; where u is an authenticated BD user by the MS, a_u is a requested action, and bd_u is a BD source provider of the data or service that u is request to perform a_u from. The algorithm to accept (u, a_u, bd_u) on the cs_l is, [13]

for $C_u = \{c_1, \ldots c_k\}$ **such that** (bd_u, c_i) is from SA;
if C_u = empty set then
{request = deny (for this cs_l)
else
$CS_u = cs_1, \ldots cs_k$ such that (cs_i, c_i) is from TCSL and c_i is from C_u;
if CS_u = empty set or cs_l is not in the set CS_u {
request = deny (for this cs_l)
else
if (there exist $(mp_x = (at_x, a_x, bd_x)$ in MSP set, such that
$a_u == a_x$ and $bd_u == bd_x$) and (there exist $(csp_x = (bd_x, a_x, rs_x)$ in set CSP_l
such that $a_u == a_x$ and $bd_u == bd_x$ and rs_x = = resource required))
{
request = granted for this cs_l

if perform a_u on rs_x = = success
return result to RC
else
return RC resource from cs_l unavailable?
}
else
request = deny for this cs_l
}
}
}

12.4.2 Data Privacy

Big Data analytics invades the personal privacy of individuals, [14].

The negative impact of the improper usage of BD

The negative impact of the improper usage of the analysis that is made based on the Data may be

- Discrimination. The usage of predictive analytics to make determinations about users intelligence, habits, education, health, ability obtain a job, finance. The use of users associations in predictive analytics to make decisions that have a negative impact on individuals. Such opinions are automated, and therefore errors are more difficult to detect or prove, and may influence for example employment, promotions, fair housing, and many more.
- An embarrassment. Online shops and restaurants, government agencies, universities, online media corporations may be the reason for the personal information leakage resulting in revealing the personal information about users or employees. Especially very private information that people would like to keep separated from their business life (health problems, sexual orientation, or an illnesses).
- The lost of anonymity. If data masking is not done effectively, analysis could easily match the individual whose data has been masked to this person.
- Government exemptions. Personally Identifiable Information (PII): name, any aliases, race, sex, date and place of birth, Social Security number, passport and drivers license numbers, address, telephone numbers, photographs, fingerprints, financial information like bank accounts, employment and business information, and more are collected by governments. The discredit of such a data bases is the great threat and might lead to the destabilization of the county, [15].

12.4.3 Technologies and Strategies for Privacy Protection

Encryption algorithms, anonymization or de identification, deletion and non-retention methods helps to protect privacy.

There are three basic types of cryptographic algorithms, [16], that were successfully adopted to BD systems:

1. Cryptographic hash functions. Hash function produces a short (the length is always known) representation of a any message. Hash function is a one-way function: it is easy to compute the hash value from a particular input but calculating the input from hash value is extremely computationally difficult or impossible. Hash function are also collision resistant: is extremely difficult to find two particular inputs that produce the same hash value. Because of these feathers, hash functions are used to determine whether or not data has changed and are the part of the digital signature schemes. In Big Data systems data integrity errors could appear because of splitting ad merging the data, hardware errors, software errors, intrusions, or user errors. Therefore data integrity needs to be checked after each stage of data processing. Approved safe hash functions are: from SHA-2 family: SHA-224, SHA-256, SHA-384 and SHA-512,and SHA-3, [17].
2. Symmetric algorithms (secret key algorithms) incorporating a single key to both ciphering and to remove or check the protection (deciphering). Additional methods of safe key exchange between sender and receiver have to be used. The Approved algorithms for symmetric encryption and decryption are: the Advanced Encryption Standard (AES) and the Triple Data Encryption Algorithm (TDEA), based on the Data Encryption Standard (DES). Using symmetric key block cipher algorithm, in case of the multiple blocks in a message (data stream) are encrypted they must not be processed separately. The Recommendation for Block Cipher Modes of Operation defines modes of operation that have to be used, citemodes.
3. Asymmetric algorithms (public key algorithms) incorporating two keys (i.e., a key pair): a public key (may be stored in public database) and a private key (must be kept secret) that cannot be calculated one from another. Approved safe asymmetric algorithm is RSA and the length of the key should be set properly, [18].
4. Digital Signatures and the Digital Signature Standard (DSS). A digital signature is an electronic analogue of a hand-written signature and it is used for proving to the recipient or a third party, the proof that the message was signed by the originator. Signature generation process incorporates a private key to generate a signature. Signature verification process uses the public key that matches to public key, to verify the signature. Hash functions are used to exclude manipulations with the sent data. Digital Signature Standard (DSS) includes three digital signature algorithms: the Digital Signature Algorithm (DSA), the Elliptic Curve Digital Signature Algorithm (ECDSA) and RSA. The DSS is used with Secure Hash Standard, [13].
5. Public Key Infrastructure (PKI) regulates generating and distributing methods for public key certificates and ways of maintaining and distributing certificate status information for unexpired certificates. PKI defines components

   - certification authorities to create certificates and certificate status information,
   - registration authorities to verify the information in the public key certificates and determine certificate status,
   - authorized repositories to distribute certificates and certificate revocation lists

- online Certificate Status Protocol servers to distribute certificate status information,
- key recovery services to backup private keys,
- credential servers to distribute private key material and the corresponding certificates.

The example of certificates used in PKI may be the X.509 Certificate, [19].

Solutions dedicated to the BD systems

Many cryptography solutions were proposed that are dedicated to the BD systems: quantum cryptography and privacy with authentication for mobile data center, group key transfer based on secret sharing over Big Data, [20], and ID-based generalized signcryption method to obtain confidentiality or/and authenticity, capability based authorization.

12.4.4 Quantum Cryptography and Privacy with Authentication for Mobile Data Center

Quantum cryptography was proposed with Grovers algorithm (GA), [21], and PairHand authentication protocol, [22], to asset secure communications between the mobile users and authentication servers.

Proposed model includes several layers, and supports secure Big Data sending by mobile user to the nearest mobile data center.

- Data center front end Layer: verification and identification of the mobile user and Big Data using Quantum cryptography and authentication protocols
- Data reading interface Layer: during each operation of the interface, provides the best performance to minimize the complexity
- Quantum key processing Layer: quantum key distribution (QKD) based on QC is taken into considerations, and the size of the Big Data and level of the security
- Key management Layer: the size of the Big Data and traffic load, the security key generations is performed, protocols based on QC are applied
- Application Layer: depending on the applications that are used by data upleader, division should be made according to organization policy with different level of the security and privacy.

It was stated that designing mobile data center with the PairHand protocol reduces the computational cost and increases the efficiency of the handover authentication.

12.4.5 Group Key Transfer Based on Secret Sharing Over Big Data

A key transfer protocol for secure group communications over Big Data was proposed and designed particularly for group-oriented applications over Big Data systems.

Linear secret sharing schemes are used, [23]. A secret is divided into shares and is shared among a set of shareholders by a trusted dealer in such a way that authorized subsets of shareholders can reconstruct the secret but unauthorized subsets of cannot. The Vandermonde Matrix is used as the share generation algorithm, [24]. Key transfer protocol consists of two phases: the secret establishment phase and the session key transfer phase.

12.4.6 ID-Based Generalized Signcryption Method to Obtain Confidentiality or/and Authenticity

Generalized signcryption (GSC) methods ware used to provide multi-receiver identity-based generalized signcryption (MID-GSC) method. Bilinear Diffie–Hellman (BDH) assumption and Computational Diffie–Hellman (CDH) assumption was used to ensure safety of the system. Either a single message or multiple messages can be signcrypted for one or multiple receivers and by one or multiple senders, [25].

12.4.7 Capability-Based Authorization Methods

The capability-based authorization models have many additional feathers in comparing to the traditional models, that is, [26]:

- delegation support: a user can obtain access rights to another user, and the permit ion to further delegate the rights. Moreover, the delegation depth is controllable at each level,
- capability revocation: the right to the resources may be canceled by authorized subjects,
- information granularity: dynamic adaptation of permissions and access privileges is assumed by the provider to react on changes in users needs,
- capability token is used and Role-Based Access Control (RBAC) systems or the Attribute-Based Access Control (ABAC) are incorporated.

These models were proposed for the systems where, at the same time, users needs could be highly dynamic and limited in time therefore no permanent link between a user and an service is beneficial.

A user that has specified a service for which he needs access submits a request to the Digital Ecosystem Clients Portal. This request is passed to the Policy Decision Point (PDP). The PDP decides if the users' request may be accepted or have to be denied. In the first case, it provides an access token (capability token) that enables access to the service. This token is given to the user. Each capability token has the following characteristics: the resource(s) that it gives the permission to use, the subject (user) to which the rights have been allowed, the granted privileges, and the

authorization chain, that user is required to fulfill in order to prove his identity. The model is using Zero Knowledge Proofs, to prove, without disclosing any personal information, the user has the right to receive an access capability for a resource.

The proposed CapBAC architecture consists several components:

- the resource in the form of information service or an application service that has to be identifiable and may serve user,
- authorization capability that is the characterization of rights to be given together with the specification which ones can be delegated further, and their delegation depths, in contests of the resources on which those permissions are executed,
- capability revocation rules together with the specification of users who may revoke a single capability, a complex capability and all its descendants,
- service/operation request center that is processing requests from users and sending them to one single vendor,
- Policy Decision Point in charge of managing resource access requests,
- the Monitoring Service that checks capability tokens and digitally signs the request to prove that it is the owner of the capability. It also is doing formal validity of all capabilities in the authorization chain and logical validity of the operation request,
- the particular resource manager that proceeds users' requests for the particular resource.

12.4.8 No QSL Data Basis Security

NoSQL databases such as CouchDB, MongoDB, and Cassandra were initially not designed with security issues as a main feature. Therefore, third-party tools and services have to be used. Sharding that is also done generates security risks, caused by geographic distribution of data, un-encrypted data storage, unauthorized exposure of backup and replicated data, insecure communication over the network. Security of NoSQL databases involves securing data-at-rest on a single node, data security during transmission between various nodes in a sharded environment that are made often between countries and inside international structures, [27]. Such methods as Consistent Hashing (Distributed Hash Tables), Flexible Partitioning, and High Availability monitoring, Inter-cluster communication, Denial of Service problem governing, Continuous Auditing, Potential for injection attacks monitoring methods, Intrusion Detection Systems, Kerberos Mechanism, Bull Eye Algorithm Approach supports ensuring the safety of data in such systems, [28, 29]. Action aware access control roadmap was proposed in, [30]. As far as platform selection and analysis is concerned, selection of existing MapReduce systems and NoSQL data stores should be considered at the first place. Then, identification of policy components have to be made. The next step is defining of enforcement mechanisms for chosen BD database

- MongoDB with Role-based access control (RBAC) at database and collection level access control,
- Cassandra RBAC at key-space and table level access control,

- Redis for which access control can only be achieved at application level,
- HBase incorporating control lists at column family and/or table level,
- CouchDB with no native access control,
- Hive equipped with fine grained access control and relational model
- Hadoop having Access control lists at resource level,
- Spark incorporating Access control lists at resource level, [31].

12.4.9 Trust Management

Trust according to the international standards and the international law management/monitoring have to assured with special concern about the fact that cryptography methods and protocols may become outdated. The listed below institution are publishing the updated guidance and regulation according to the Big Data security

- US National Security Agency (NSA), [32],
- US National Computer Security Center (NCSC), [33],
- US National Institute of Standards and Technology (NIST), [34],
- RSA Data Security, Inc, [35],
- International Association for Cryptologic Research (IACR), [36],
- International Organization for Standardization (ISO), [37],
- Federal Information Processing Standards (FIPS), [38],
- Cloud Security Alliance, [39],
- Electronic Privacy Information Center (EPIC), [40],
- Academic researchers centers (MIT Computer Science and Artificial Intelligence Laboratory, Lawrence Berkeley National Laboratory, Industry University cooperative research center for Intelligent Maintenance Systems at university of Cincinnati).

The massiveness of Big Data systems has a great impact of privacy, security, and consumer welfare; moreover it was stated that social and economic costs and potential negative detriment is very high, [41] (Table 12.3).

Table 12.3 Chosen security responsibilities

User	BD vendor	Third parity
Private key physical safety	Key validity	Own key physical safety
Data originality	Data integrity	Proper data usage
Authentication	Authorization	Authentication
Uniqueness	Anonimization	Proper generalization
Security awareness	Security extortion	Security submission
Desk computer security	Infrastructure security	Applications security
Security agreement	Audit	Certification

The policies may be different for particular regions in with the data are collected, for example

- Canada The Personal Information Protection and Electronic Documents Act (PIPEDA), [42], specifies the rules to govern collection, use or disclosure of personal information, and gives users the rights to understand the reasons why organizations collect, use, or disclose personal information, to expect organizations to protect the personal information in a reasonable and secure way.
- European Union: the 8th article of the European Convention on Human Rights (ECHR) gives the right to respect ones private and family life, his home and his correspondence is provided. EU members states adopted legislation pursuant Data Protection Directive, [43], adapted their existing laws.

Resolving conflicts of different security rule sets according to different laws in each country, and the territorial dissipation between data uploaders, data owners, and data users caused the need for international certification of systems. The examples of such certificates are

- TRUSTes APEC Privacy Certification program, checking among the others way in which collected information is used, types of Third Parties, with whom collected data is shared and for what purpose, methods for updating privacy settings, types of passive collection technologies used (e.g., cookies, web beacons, device recognition technologies). A statement that collected information is subject to disclosure pursuant to judicial or other governmental laws, warrants, orders to protect the rights of the Participant, or protect the safety of the Individual, [44].
- The ISO 27000 family of standards: ISO 27001 that covers security in the cloud, ISO 27002, ISO 27018 that is the description of policy for personally identifiable information to the scope of 27001. ISO 27018 compatibility means that owner of the data will not use customer data for their own independent purposes, such as advertising and marketing, without the customer's agreement and establishes clear and transparent parameters for the return, transfer, and secure disposal of personal information, [45].
- The SSAE16 Auditing Standard: If the system is protected against unauthorized access, if the system is available for operation and use as committed or agreed, processing data is complete, accurate, timely, and authorized, and if data designated as confidential is protected, [46].
- The Cloud Security Alliance Cloud Controls Matrix (CCM), [47] gathers 13 most critical security issues to enable the users to compare different BD systems. The components of the CMM are: as far as *Application and Interface Security* is considered: Application Security, Customer Access Requirements, Data Integrity. Regarding *Audit Assurance and Compliance*: Audit Planning, Independent Audits, and Information System Regulatory Mapping. Considering *Business Continuity Management and Operational Resilience* the issues are Business Continuity Planning, Business Continuity Testing, Datacenter Utilities and Environmental Conditions, Operational Resilience Documentation, Environmental Risks considerations, Equipment Location, Equipment Maintenance, Equipment Power Failures,

Impact Analysis, Policies and procedures, Retention Policy. As far as *Change Control and Configuration Management* is considered the matrix consists of New Development/Acquisition problem, Outsourced Development regulations, Quality testing requirements, Unauthorized Software Installations procedures, policies for Changes. Regarding *Data Security and Information Life cycle Management*: classification by the data owner based on data type, value, sensitivity, and criticality, Data Inventory/Flows procedures, E-commerce Transactions, Handling/Labeling Policy, separating Production Data from non-production environments, Ownership/Stewardship of the data. For obtaining *Data-center Security*: Asset Management, Controlled Access Points, Equipment Identification, secure working environment policies, Secure Area Authorization methods, User Access restrictions. Additionally: Sensitive Data Protection, Access Keys management, appropriate encryption, compliance with security requirements, Risk Assessments, support in the form of clearly-documented rules, requirements and help documents. The Matrix also specifies that Policy Enforcement is absolutely necessary and updates to security policies, procedures, standards, and controls should be done to ensure credibility and validity of all security procedures. A security awareness training program have to be obligatory for third-party users and employees.

12.4.10 Secure Communication

Big Data gathering, processing, and storage require cooperation of may system. Data uploaders may use different methods to connect to the storage servers: computers, tablets, mobile phones, lean stations equipped with web browser only.

Network layer security protocols ensures secure network communications at the layer where packets are routing across networks. Transport layer security methods describe security at the layer responsible for end-to-end communications. Transmission Control Protocol/Internet Protocol (TCP/IP) model, Secure Sockets Layer (SSL) and Transport Layer Security (TLS), [48] are recommended.

12.4.11 Infrastructure Design for Security

BD systems infrastructure may be designed in such a way that assuring of security is easier to implement and maintenance, [6]

1. **Separation Model** incorporates the architecture where separation of duty and privileges is made by the assumption that at least two or more sides (recipients/customers/vendors/users/data owners) are involved in any single operation. In this model, each participant is performing only part of the transaction. It results in sharing the duties among the sides in such a way that no single side may have excessive control over security processes and critical operations. The example

of such a technique might be to design two independent services responsible for data processing and data storage.
2. **Availability Model** To provide the fully availability of the BD services, infrastructure should be replicated. At least two independent data processing services, and two independent data storage services should be designed. Data should be replicated and synchronized by the Replication Service. The Availability Model incorporates redundancy into the Separation Model.
3. **Migration Model** where the migration of data is guaranteed by BD service provider. BD Data Migration Service should be designed as the separate part of the model and have to cooperate with the Storage Services. Second infrastructure also should be prepared for a second Storage Service that enables importing data and exporting data.
4. **Tunnel Model** incorporates a tunnel infrastructure between the Data Processing Center and the Data Storage Center. The tunnel is used as a communication channel between these tho parts of DB system. It serves as an interface, enabling interaction and provides hiding the unnecessary information. For example, it enables isolation between data processing information (the exact data source location) and data storage information (data stored after some post processing). It also serves as a center for enforcing security policies and both Data Processing Center and the Data Storage Center.

12.5 Case Studies

The most popular BD solutions are provided by big companies such as Software AG, Oracle Corporation, IBM, Microsoft, SAP, EMC, HP and Dell. The leading solutions are Alteryx, KNIME, Microsoft Revolution Analytics, Oracle Advanced Analytics, RapidMiner, SAP Predictive Analytics, SAS Enterprise Miner, The Teradata Aster Discovery Platform, Big Data platform from Amazon Web Services. The following section is the summary of solutions provided by chosen vendors for assuring different aspects of security.

Hewlett Packard Enterprise The company incorporated the OECD Guidelines on the Protection of Privacy and Transborder Flows, EU Directive 95/46/EC, APEC Privacy Framework, and the Madrid Resolution on International Privacy Standards and the Australian Privacy Principles under the Privacy Act 1988 (Cth), HPE complies with the U.S. E.U. Safe Harbor framework and the U.S. Swiss Safe Harbor framework as set forth by the U.S. Department of Commerce regarding the collection, use, and retention of personal data from European Union member countries and Switzerland. HPE was also certified according to the Safe Harbor Privacy Principles of notice, choice, onward transfer, security, data integrity, access, and enforcement. HPE has also established a set of binding corporate rules (BCR), which have been approved by the majority of Data Protection Regulators in the EEA and Switzerland, effective June 2011. The BCRs ensure that personal data from the EEA is adequately

protected while being processed by any of HPEs global entities. The company has received TRUSTe's APEC Privacy Seal signifying that this privacy statement and our practices have been reviewed for compliance with the TRUSTe program and the services are compatible with APEC Cross Border Privacy Rules System, including transparency, accountability, and choice regarding the collection and use of your personal information. The certification does not cover information that may be collected through downloadable software on third party platforms.

IBM IBM has been awarded TRUSTe's Privacy Seal for the practices connected with security issues. All services are according to the APEC Cross Border Privacy Rules System, U.S.–EU Safe Harbor framework and the U.S.-Swiss Safe Harbor framework are incorporated. IBM implements physical, administrative, and technical methods to protect information from unauthorized access, use, and disclosure. The encryption is made for sensitive personal information during transmitting. IMB require all user of such a data protect it.

Microsoft Data access control is performed using two layers: physical and logical. Physical access to facilities is permitted by perimeter fencing, security officers, locked server racks, multifactor access control, alarm systems, and video surveillance. Access to customer data is controlled by role-based access policy, two-factor authentication, minimizing access to production data, logging and auditing. Microsoft incorporated data isolation techniques to logically separate users. The systems was certified with HIPAA and HITECH (The Health Insurance Portability and Accountability Act and the Health Information Technology for Economic and Clinical Health Act). It also participate in the Cloud Security Alliance STAR Registry program. It also was certified with ISO 27001 and SOC1 and SOC2 (AICPAs SSAE16 Service Organization Control for SOC 1 Type 2 requirements). SOC1 Type 2 attest is checking the design and operating effectiveness of controls implemented by a service provider and SOC 2 Type 2 audit is checking security, availability, and confidentiality.

Teradata http://bigdata.teradata.com/ The following are strictly appliqued and was checked by independent audits: ISO 29100:2011 (Privacy Framework), ISO 27002:2013 (Information Technology Security Techniques Code of Practice for Information Security Controls), ISO 27018:2014 (Protection of customer PII/data privacy in public cloud environments), Online Privacy Alliance Guidelines Organisation for Economic Co-operation and Development (OECD), Guidelines on the Protection of Privacy and Transborder Flows of Personal Data, OECD Guidelines for Multinational Enterprises (Article VIII regarding Privacy), OECD Guidelines for the Security of Information Systems and Networks, United Nations (UN) Guidelines for the Regulation of Computerized Personal Data Files, International Standards on Privacy and Personal Data Protection (the Madrid Resolution on International Privacy Standards), Asia Pacific Economic Cooperation (APEC) Privacy Framework, European Data Protection Directive (EU Directive 95/46/EC), European Privacy and Electronic Communications Directive (EU Directive 2002/58/EC), Drafts of the proposed European General Data Protection Regulation (GDPR), Council of Europe

Convention for the Protection of Individuals with regard to Automatic Processing of Personal Data, and its Additional Protocol regarding Supervisory Authorities and Transborder Data Flows.

12.6 Summary

Main problems concerning security in Big Data systems were presented. Security from user side, data owner, and data uploader point of view was considered. Chosen methods for the preservation confidentiality, integrity, and availability were presented. Some of them were adopted from traditional systems and still have to be adjusted to be fully applicative. The methods particularly designed for BD systems and may not be still tested enough to provide security. The data life cycle in BD systems is complicated and involves a lot users and operation. Therefore, there are many week points as far as security is concerned. Data being sent, data at rest, data being processed and deleted from the system requires different kind of techniques to assure authenticity and provenance. The need for third party trust centers was emphasized. The necessity for external control as far as international low obedience was stated.

Assuring the security of the users rights, the data itself and the vendor is sometimes confluent. High level of security incorporated in the system increases the trust for the vendor but also generates expenses. Not every user needs top security level but for some users it is indispensable. Security roles and privileges according to the user requirement have to be balanced with BD vendor business goals. Furthermore, in every modern society privacy as one of the constitutional issues and some aspects of security are regulated by the law. That is the reason of external audits and trust centers certification. The security topics have to be also monitored and developed by independent organizations not related to BD commercial companies.

Designing fully secure system is impossible. The system that consists of so many elements will always be vulnerable to some kind of attack. Lack of knowledge from the users side or targeted action to the detriment are the main week points beyond the control.

Health care, education, media research, banking, online sales are supported by analysis, capturing, searching, sharing, storing, transferring, and visualization of large amount of data. Therefore, BD systems are predicted to be absolutely essential and growing even bigger in the nearest future. Further studies on the impact of variety, volume, and velocity on data security is very important. The preservation of confidentiality, integrity, and availability should be the main concern assuring the trust for BD services. The social security and scientific credibility.

References

1. Marz, N., Warren, J.: Big Data: Principles and Best Practices of Scalable Realtime Data Systems, Manning Publications (2015)
2. Big Data Now: 2012 Edn. OReilly Media, Inc. (2012)
3. Liu, H., Gegov, A., Cocea, A.: Rule Based Systems for Big Data A Machine Learning Approach, Springer (2016). ISBN:978-3-319-23696-4
4. Davis, K., Patterson D.: Ethics of Big Data, OReilly Media, Inc. (2012)
5. INTERNATIONAL STANDARD ISO/IEC 27002: Information technology Security techniques Code of practice for information security management, ISO/IEC FDIS 17799:2005(E) (2005)
6. Zhao, G., Rong, Ch., Gilje Jaatun, M., Sandnes, F.E.: Reference deployment models for eliminating user concerns on cloud security. J. Supercomputing **61**(2), 337–352 (2012). August
7. http://www.iso.org/iso/cataloguedetail?csnumber=54533
8. NIST Special Publication 1500-1, NIST Big Data Interoperability Framework: Vol. 1, Definitions, NIST Big Data Public Working Group (NBD-PWG). doi:10.6028/NIST.SP.1500-1
9. Top Ten Big Data Security and Privacy Challenges, Cloud Security Alliance. http://www.isaca.org/groups/professional-english/big-data/groupdocuments/big_data_top_ten_v1.pdf. Accessed 22 March 2016
10. NIST Special Publication 1500-4, NIST Big Data Interoperability, Security and Privacy, NIST Big Data Public Working Group. doi:10.6028/NIST.SP.1500-4
11. van Tilborg, H.C.A., Jajodia, S. (Eds.): Encyclopedia of Cryptography and Security, Springer. ISBN:978-1-4419-5905-8
12. Schneier,B.: Applied Cryptography Protocols, Algorithms, and Source Code in C, John Wiley and Sons (1996)
13. Hu,V.C., Grance, T., Ferraio D. F., Kuhn,D.: An Access Control Scheme for Big Data Processing, National Institute of Standards and Technology, USA. http://csrc.nist.gov/projects/ac-policy-igs/big_data_control_access_7-10-2014.pdf. Accessed 22 March 2016
14. Rotenberg M.: COMMENTS OF THE ELECTRONIC PRIVACY INFORMATION CENTER, THE OFFICE OF SCIENCE AND TECHNOLOGY POLICY Request for Information: Big Data and the Future of Privacy, Electronic Privacy Information Center (EPIC) (2014). https://epic.org/privacy/big-data/EPIC-OSTP-Big-Data.pdf. Accessed 22 March 2016
15. Armerding, T., The 5 worst Big Data privacy risks (and how to guard against them). http://www.csoonline.com/article/2855641/big-data-security/the-5-worst-big-data-privacy-risks-and-how-to-guard-against-them.html. Accessed 22 March 2016
16. Stallings, W.: Cryptography and Network Security: Principles and Practice, Pearson (2013)
17. http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf
18. NIST Special Publication 800-57 (SP 800-57), Recommendation for Key Management, provides guidance on the management of cryptographic keys. http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf. Accessed 22 March 2016
19. X.509: Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks. http://www.itu.int/rec/T-REC-X.509/en,cited. Accessed 22 March 2016
20. Zeng, B., Zhang, M.: A novel group key transfer for big data security. Appl. Math. Comput. **249**, 436443 (2014). doi:10.1016/j.amc.2014.10.051
21. Goorden, S.A., Horstmann M., Mosk, A.P., Kori, B., Pinkse, P. W. H.: Quantum-Secure Authentication of a Physical Unclonable Key. Optica **1**(6) (2014)
22. He, D., Jiajun B., Chan, S., Handauth, Ch.: Efficient Handover Authentication with Conditional Privacy for Wireless Networks. IEEE Trans. Comput. **62**(3) (2013)
23. Farras, O., Padr, C.: Ideal hierarchical secret sharing schemes. In: Theory of Cryptography, pp. 219236. Springer (2010)
24. Hsu, C.-F., Cheng, Q., Tang, X., Zeng, B.: An ideal multi-secret sharing scheme based on msp. Inf. Sci. **181**(7), 14031409 (2011)

25. Wang, H., Jiang, X., Kambourakis, G.: special issue on security, privacy and trust in network-based Big Data. Inf. Sci. **318**, 4850 (2015). doi:10.1016/j.ins.2015.05.040
26. Piccione, S., Rotondi, D.: A capability-based security approach to manage access control in the Internet of Things. Math. Comput. Model. **58**, 11891205 (2013)
27. Zahid, A., Masood, R., Awais Shibli M.: Security of Sharded NoSQL Databases: A Comparative Analysis, Conference on Information Assurance and Cyber Security (CIACS) (2014). doi: 978-1-4799-5852-8/14/
28. Okman, L., Gal-Oz, N., Gonen, Y., Gudes, Y.: Security Issues in NoSQL Databases. IEEE (2011). doi:10.1109/TrustCom.2011.70
29. Pazhanirajaa,N., Victer Paula,P., Saleem Bashab M.S., Dhavachelvanc P.: Big Data and Hadoop-A Study in Security Perspective. Procedia Computer Science, Vol. 50, Big Data, Cloud and Computing Challenges, (2015). doi:10.1016/j.procs.2015.04.091
30. Colombo, P., Ferrari, E.: Privacy Aware Access Control for Big Data: A Research Roadmap. Big Data Res. **2**, 145154 (2015). doi:10.1016/j.bdr.2015.08.001
31. http://craigchamberlain.com/library/security/NIST/NIST%20800-8%20-%20Security%20Issues%20%20the%20Database%20Language%20SQL.pdf
32. www.nsa.gov
33. www.ncsc.gov
34. www.nist.gov
35. www.rsa.com
36. www.iacr.org
37. www.iso.org
38. www.csrc.nist.gov/publications
39. https://cloudsecurityalliance.org
40. https://www.epic.org
41. Kshetri, N.B.: Big data's impact on privacy, security and consumer welfare. Telecommun. Policy **38**(11), 1134-1145. www.elsevier.com/locate/telpol. Accessed 22 March 2016
42. Personal Information Protection and Electronic Documents Act, Published by the Minister of Justice, Canada, (2016). http://laws-lois.justice.gc.ca/PDF/P-8.6.pdf
43. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, Official Journal L 281, 23/11/1995 P. 0031–0050
44. APEC Certification Standards. https://www.truste.com/privacy-certification-standards/apec/. Accessed 22 March 2016
45. The International Standard for Data Protection in the Cloud, ISO/IEC 27018 (2014). https://www.iso.org/obp/ui/iso:std:iso-iec:27018:ed-1:v1:en. Accessed 22 March 2016
46. The SSAE16 Auditing Standard: (2015). http://www.ssae-16.com/. Accessed 22 March 2016
47. https://cloudsecurityalliance.org/group/cloud-controls-matrix/
48. Guide to SSL VPNs, Special Publication 800-113, Recommendations of the National Institute of Standards and Technology (2008). http://csrc.nist.gov/publications/nistpubs/800-113/SP800-113.pdf. Accessed 22 March 2016
49. Barker, E.B., Barker, W.C., Lee A.: NIST Special Publication 800-21, Guideline for Implementing Cryptography In the Federal Government, U.S. Department of Commerce, (2005). http://csrc.nist.gov/publications/nistpubs/800-21-1/sp800-21-1_Dec2005.pdf. Accessed 22 March 2016
50. Thayananthan, V., Albeshri, A.: Big data security issues based on quantum cryptography and privacy, with authentication for mobile data center. Procedia Comput. Sci. **50**, 149–156 (2015); 2nd International Symposium on Big Data and Cloud Computing (ISBCC15), (2015). doi:10.1016/j.procs.2015.04.077
51. Kizza, J.: Computer Network Security. Springer (2005). ISBN-10:0387204733

Part III
Biological and Medical Big Data Applications

Chapter 13
Big Biological Data Management

Edvard Pedersen and Lars Ailo Bongo

13.1 Introduction

The cost of producing data in bioinformatics is rapidly decreasing [39]. This has resulted in several peta-scale omics data repositories [11]. In addition, there is a similar growth in reference databases that contain data analysis results [37]. However, with rapidly increasing dataset sizes, the analysis cost and resource usage are also rapidly increasing. The wealth of data therefore requires new approaches and technical solutions for biological data analysis. In this chapter, we will explore challenges and the use of state-of-the-art technical solutions for big biological data analysis with a focus on data management.

The current state-of-the-art in big data management [1] include systems such as Amazon RedShift [18], Google's Dremel [25], Apache Spark [40], and MapReduce [8]. Most of these were developed to analyze text-based data with few dimensions. Biological data differs in that it has more dimensions and noise, it is heterogeneous both with regard to biological content and data formats, and the statistical analysis methods are often more complex. It is therefore not straightforward to adapt these state-of-the-art systems for biological data analysis, nor to integrate these with the analysis framework. It is challenging to even know at which level of the data analysis stack to integrate them at.

In this chapter, we give an introduction to biological data analysis, with short descriptions of the workflows, pipelines, and execution environments used in the field. In addition, we provide a case study from our own lab. Then we provide a short review of big data storage and processing solutions, highlighting advantages

E. Pedersen (✉) · L.A. Bongo
University of Tromsø- The Arctic University of Norway, Tromsø, Norway
e-mail: edvard.pedersen@uit.no

L.A. Bongo
e-mail: larsab@cs.uit.no

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_13

and disadvantages of different approaches for biological data management. Finally, we summarize our own experiences in biological data management using big data systems.

13.2 Biological Data Analysis

In this section, we provide the necessary background required to understand the data management requirements for biological data analysis. We describe how biological data analysis is typically implemented, configured, and executed. We use our own META-pipe [31] pipeline from the metagenomics field as a case study.

13.2.1 Metagenomic Data Analysis

The typical analysis of metagenomic data involves the following steps:

1. Retrieve and prepare the sample. This includes sample cleaning and DNA isolation.
2. Analyze the sample using instruments such as next-generation sequencing machines.
3. Raw data processing. This is often done using vendor-specific software and operating procedures, and it is typically done at the instrument lab.
4. Quality control and data cleaning of the data received from the instrument lab. This is typically the first step done by the researcher, and it can often be done once for each dataset.
5. Run the data through a series of tools in a pipeline to produce the output data needed to answer a particular research question. The same data may be used in several data analysis pipelines, and a pipeline may be run periodically to update the results with new input data or with updated reference databases.
6. Analyze the output data. This is typically done using interactive visual tools that are often decoupled from the analysis pipeline.

In this chapter, we will focus on the data analysis pipeline (Step 5). This is the step where the researchers put most effort into development time and computation resources.

13.2.2 Data Analysis Pipelines

Biological data analysis is typically done through a collection of tools arranged in a pipeline where the output of one tool is the input to the next tool (Fig. 13.1). The

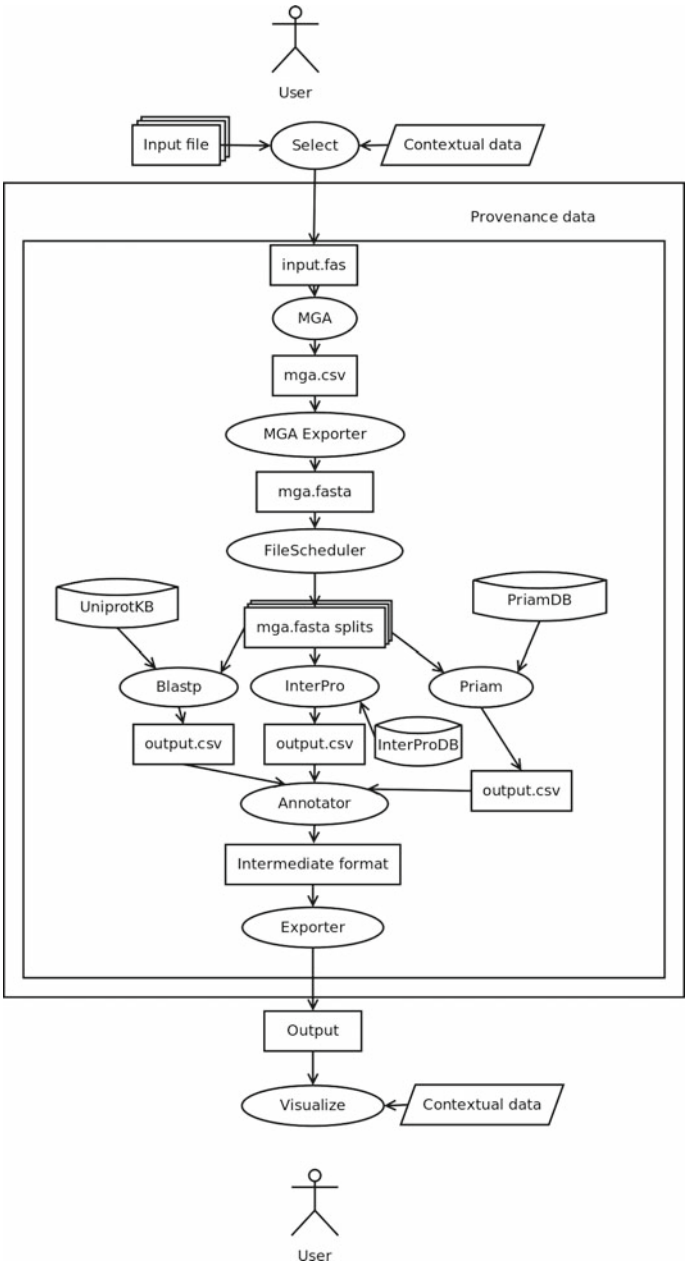


Fig. 13.1 The META-pipe pipeline tools, file formats, reference databases, and intermediate files. The items inside the "Provenance data" box must have provenance information recorded for analysis reproducability

data transformations include file conversion, data cleaning, normalization, and data integration. A specific biological data analysis project often requires a deep workflow that combines many tools [9]. There are many libraries [14, 16, 35] with hundreds of tools, ranging from small, user-created scripts to large, complex applications [23].

There are four types of data managed for such analysis pipelines:

1. The *input, intermediate, and output* data. These are typically structured files with many samples. The files may range in size from megabytes to terabytes. The pipeline input data is produced by biotechnology instruments such as next-generation sequencing machines, or downloaded from public repositories such as GEO [10] and ENA [22]. The pipeline output files are typically analyzed using standalone interactive visualization tools.
2. *Contextual data* contains information about the data samples required for data selection and interpretation. This includes information about how, where, and when the samples were collected. The contextual data may be used to select the datasets and records to process for a pipeline execution, and by visualization and data exploration tools.
3. *Reference databases* with human or machine curated metadata extracted from the published literature and from analysis of experimental data [12]. These are used to annotate the data to make it useful for scientists. The reference databases range in size from small collections of annotation data (Swiss-Prot), to peta-scale collections of analyzed samples (European Nucleotide Archive).
4. The *Provenance* required for experiment reproducibility. This includes data lineage information with descriptions of the pipeline tools, their parameters, databases and versions, and machines used in the analysis.

13.2.3 Pipeline Frameworks and Execution Environments

The analyst specifies, configures, and executes the pipeline using a pipeline framework (a review and taxonomy is provided in [23]). The pipeline framework provides a way of specifying the tools and their parameters, management of data and metadata, and execution of the tools. In addition, a pipeline framework may enable data analysis reproducibility by maintaining provenance data such as the version and parameters of the executed tools. It may also maintain the content of input data files, reference databases, output files, and possibly intermediate data.

A pipeline framework may comprise of a set of scripts run in a specific platform, or a system that maps high-level workflow configuration to executable jobs for many platforms. There are also frameworks that provide an interactive GUI for workflow configuration, and a backend that handles data management and tool execution.

Biological data analysis pipelines are typically run on a fat server, high performance computing clusters, or a data-intensive computing cluster. Using a single server has two main advantages. First, most biological analysis tools can be used unmodified. Second, it is not necessary to distribute and maintain tools and data on

Table 13.1 Data management approaches for selected pipeline frameworks

Pipeline	Data	Contextual data	Reference databases	Provenance
Galaxy	Local files	None	Varies	Workflow
GePan	Local files	None	Packaged with pipeline	Partial
EBI metagenomics	Remote files	Packaged with data	Packaged with tools	Report to user

a cluster. The main disadvantage is the lack of scalability, both concerning dataset size and the number of concurrent users.

Many biological data analysis tools can easily be run on high performance computing (HPC) clusters by splitting the input (or reference databases) into many files that can be computed in parallel. The main advantage of using an HPC cluster is their parallel compute performance. The main disadvantage is that the centralized storage system often becomes a bottleneck for large-scale datasets in I/O bound jobs.

Finally, data-intensive computing clusters [34] distribute storage distributed on the compute nodes, and provide data processing systems that utilize such distributed storage. The main advantage is improved performance and scalability for I/O bound jobs. The main disadvantage is that to fully utilize such a platform the analysis tools may need to be modified [7, 9, 30].

A comparison of how a selection of workflow managers handle different types of data is shown in Table 13.1.

13.2.4 Case Study: META-Pipe

To illuminate the data management issues in bioinformatics, we use as a case study, an in-house workflow manager and pipeline, which we have used extensively in research in this field. The pipeline and associated workflow managers are called META-pipe. We are currently developing version 2.0 of the pipeline and workflow manager that will be provided as a European Service in the ELIXIR e-infrastructure.

META-pipe is a successor to GePan, which was a workflow manager designed to do annotation of genomes. META-pipe extends GePan by integrating several new tools, as well as enabling pipelines to be run on supercomputer infrastructure.

META-pipe consists of a workflow manager that automatically generates a pipeline based on the input parameters, and runs this pipeline on the local supercomputer. The pipelines created are for marine metagenomics analysis, and integrates existing biological analysis frameworks with modern data management techniques and infrastructures.

The actual execution in META-pipe works by generating scripts based on the input of the user, these scripts and the input data are then submitted to an execution

site, where the reference databases and tools are housed. The tools are run in a data-parallel fashion, enabling the analysis of large samples. The user-facing frontend of META-pipe is Galaxy, which allows users to examine the output of the pipeline in several formats.

Data management in META-pipe is to a large degree up to the pipeline developer and administrator. The input, output, and intermediate data are stored as files managed by the META-pipe scripts. The META-pipe workflow manager handles the intermediate data in the pipeline, but the pipeline developer must maintain the input and output data. For parallel execution, the data must be split and distributed on a cluster by pipeline scripts. Contextual data is not handled by META-pipe, and neither are the reference databases. These must be manually maintained as files stored on a global file system. Finally, some data lineage is provided by the META-pipe job scripts that specify the tools, tool parameters, and file paths. There is however, no automated way of maintaining or specifying file and database versions unless these are encoded in the filenames. We provide additional details in the next section.

13.3 Big Data Management

Biological data analysis jobs have moved from personal computers to data centers and clusters, due to the increased need for storage and data processing resources. However, bioinformatics analysis tools still often rely on files stored locally. It is therefore not straightforward to start using state-of-the-art big data management and processing systems for such analyses, and the developer must consider the strengths and weaknesses of different big data approaches. In this section, we provide an overview of these approaches and systems with respect to biological data management. Throughout the chapter, we will use META-pipe as a case study for how the different approaches may be used.

13.3.1 Local Data Storage

The most common interface used by bioinformatics tools is the file system, where data is stored and managed as files in a directory structure. In addition to files, many tools use simple relational databases to store provenance and contextual data. This approach is sufficient for many cases, where the amount of data is limited. We will not go into detail about different databases and file systems in this chapter.

META-pipe version 1.0 uses the file system for data management. This worked well for small datasets. However, recent flagship metagenomics datasets are too large to be replicated on each compute node. Hence, the data must be stored on a global file system, which may then become a performance bottleneck.

13.3.2 Distributed Data Storage

In computing, it is often necessary to decrease locality to increase capacity. Examples include swapping to disk when a dataset does not fit in memory, or using a multilevel cache. To provide scalable storage and enable large-scale analysis without decreasing locality, the data must be distributed over multiple machines in a way that enables efficient distributed computing. Below we provide some approaches, but note that these approaches are examples of systems that make it easy to develop parallel programs that do computations on local data in a distributed fashion, and hence there are other approaches to distribute data while maintaining locality of data for computations.

One such approach are distributed file systems such as HDFS [34], GPFS [33], and the Google file system [15]. These have been shown to be extendable to large-scale datasets and they have been used to store biological datasets used by genomics analysis pipelines [7, 9, 21]. A distributed file system provides programmers with an abstraction of a single file system, while also enabling efficient parallel computations that maintain data locality, such that tasks are scheduled to run on the cluster nodes that contain the data to be processed.

Other big data fields, such as astronomy, distribute their data among multiple standalone relational databases and then use distributed queries in the analysis [38]. There are also distributed databases such as Cassandra [5] and HBase [3], which handle the distribution of queries and data automatically. These approaches both enable locality, either through user-defined coroutines, or through frameworks such as Spark or MapReduce. However, we are not aware of bioinformatics tools that extensively use distributed queries.

There are also many other big data management systems that are distributed RDBM systems or NoSQL databases. Systems such as MySQL Cluster [28] are relational databases that are deployed across multiple machines. NoSQL databases often trade off ACID properties for other features to improve, for example, performance. The NoSQL databases vary from simple in-memory distributed key–value stores such as memcached [13] to almost full-featured databases such as Cassandra [5]. There are also distributed databases that are even more connected than relational databases that are useful for biological data management. For example, graph databases such as Neo4j have been shown to be appropriate for some use cases in bioinformatics. Have et al. [19] used Neo4j to do calculations on protein interactions (which map well to graphs), and achieved a large speedup over PostgreSQL.

For META-pipe, we have used Storage Area Network for storage on the supercomputer, Network Addressed Storage for archiving on our smaller cluster, Network File System for storage on our smaller cluster, HDFS and HBase for expansions like Mario and GeStore, BerkleyDB for the internal representation of reference databases, and SQLite as the database for our frontend.

13.3.3 Generalized Distributed Data Processing

For biological data processing, we are primarily interested in distributed data processing frameworks that: (i) are easy to use and (ii) distribute computation efficiently.

Probably the easiest approach for data analysis is to use declarative queries. Many data storage frameworks provide the necessary support for queries for their users. The supported queries include complex joins supported by relational databases, and simpler operations such as scans and retrieval of a key–value pair supported by key–value stores and NoSQL databases. All of these queries enable data locality, provided that the underlying database system supports data locality scheduling of tasks. Many systems also provide support to extend queries with user-defined functions (stored procedures) that can be used to implement more complex processing. The combination enables the ease of use, flexibly, and power to solve many data processing requirements in a distributed fashion.

Many bioinformatics tools and pipelines can utilize graph-based processing frameworks such as MapReduce [8] and Spark [40] (note that these are systems for graph-based processing, and not graph processing systems such as Pregel [24]). These greatly simplifies embarrassingly parallel computations compared to earlier approaches such as MPI. These frameworks automate data distribution, manage data locality concerns, and handle load balancing. These are achieved by partitioning the work into many small data-parallel tasks, which are scheduled and executed by the compute engine of the framework. These frameworks model the computations as a set of shared-nothing operations. For example, the building blocks of the MapReduce framework are a map that transforms all values in a set, and a reduce that performs a summary operation. This allows more flexible processing than is available when using stored procedures or queries, as well as enabling the compute engine to perform optimizations such as efficient load balancing as well as strategies to minimize the effects of stragglers and failures on the parallel program.

The MPI programming model provides operations at an even lower level. The MPI operations are for passing messages between processes running on different nodes in the network, and they thereby allow the developer full control over the data and computational distribution, as well as the communication between nodes. This framework is widely used in bioinformatics tools that support distributed execution. The disadvantage of this approach is that it can be very complex to implement even relatively simple data processing, as all the minutia of data distribution and communication have to be explicitly defined by the developer. The large responsibility put on the developer renders this framework relatively impractical for many types of data processing, where something simpler can be used efficiently.

For META-pipe, the original pipeline was extended with analysis with tools that utilize MPI, as well as batch data processing using MapReduce. The latest version of META-pipe uses Spark extensively (Table 13.2).

Table 13.2 Distributed data processing framework abstractions

Approach	Load balance	Data-independent	Complexity	Power
Queries	Yes	No	Low	Low
Stored procedure	Yes	No	Low	Medium
Graph-based processing	Yes	Yes	Medium	High
Message passing	Yes	Yes	High	High

13.3.4 Specialized Data Processing

The general-purpose frameworks in the previous section have been used to implement specialized data processing systems for uses such as incremental updates, iterative computation, and interactive analysis. These systems provide additional features, besides batch processing and provenance, that are useful for biological data analysis.

Incremental systems, such as the Incoop [6] and Marimba [32] MapReduce extensions, provide iterative computation. These systems reduce the pipeline execution time for updated datasets, by only processing the new data that is appended to a dataset. The results are then combined with previously computed results. For METApipe, we provide this functionality using GeStore [29], which allows incremental updates for unmodified biological data analysis tools.

Interactive analysis systems, such as Cloudera Impala [20] and Apache PigPen [27], are designed to execute interactive data analysis jobs with very short execution time. Our Mario system, which uses HBase [3] as a backend, is built to interactively tune the parameters of pipeline tools. We have found parameter tuning especially useful for finding the best cutoff value for sample quality in a filtering step. This is a step done early in the pipeline, and parameter changes without Mario would require manually executing the full pipeline for each parameter.

Graph processing systems, such as GraphX [17], are designed for large-scale graph processing. These are used in biological data analysis tools such as in Spaler [2] a de novo graph-based genome assembler.

13.4 Big Data Systems for Biological Data Management

To integrate the data management and processing systems described in the previous section with the workflow managers and pipelines in use in bioinformatics, several approaches can be taken:

- Direct integration, where the distributed data management systems are used directly by the workflow manager or pipeline.

Table 13.3 Data-intensive computing systems we have used in our research

System	What we used it for
HDFS	Intermediate files, caching, unstructured data
HBase	Structured and semi-structured data
MapReduce	Computing delta files, parsing, and exporting data

- File system integration, where the workflow manager or pipeline is not changed, but instead the file system operations are replaced with use of the data management system through an interface or wrapper.
- Extra tool, where the data management is implemented as a tool in the pipeline, which is used between steps.

We have used all three approaches for integrating our data analysis pipelines with our data processing systems [30].

We have primarily used the Hadoop stack (HDFS, HBase, and MapReduce) as the data management and processing backend. We have also researched and tested other systems. We chose these as we had need for several layers of data storage and a simple processing framework. The Hadoop stack provides a simple integrated and mature framework. Using a mix of HBase and HDFS as well as MapReduce enables us to minimize the overhead large-scale data processing expansions that we added to the existing META-pipe workflow manager.

For example, we have been able to achieve a speedup of up to 14× for incremental updates with minimal changes to the workflow system [29]. This speedup comes from generating a small reference database for the tools from a tera-scale collection of reference database versions. In a similar vein, Mario uses HBase to generate tiny workloads, which in turn enables interactive parameter tuning on real data [7].

Other projects that have combined big data systems with biological data analysis includes ADAM [26], which implements an entire variant calling pipeline using Apache Avro [4], Parquet [36], and Spark. They achieve large speedups when compared to traditional tools. They also examine using the same approach with an astronomy workload, where the Spark-based implementation achieved an 8.9× speedup compared to a MPI-based approach (Table 13.3).

Diao et al. [9] have used large-scale data processing frameworks to run unmodified bioinformatics tools in parallel. They provide a generalized approach for using these tools and discuss some of the advantages and challenges of their approach.

For the next version of META-pipe, we plan to use a hybrid of these two approaches, where many of the in-house tools will be implemented in Spark, and the remaining tools will be run unmodified in a data-parallel fashion.

13.5 Summary

The field of bioinformatics has just recently started to experience the effects of exponential data growth. However, bioinformatics pipelines and tools are typically not designed to scale these large data volumes. There has been large influx of large-scale data processing frameworks that enable efficient distributed data processing in fields such as web-scale data mining and astronomy. By leveraging the knowledge gained in these fields to increase data analysis scalability, the exponential growth of data can be used to increase the quality of bioinformatics analyses.

We have found that integration with legacy pipelines is challenging but it offers great potential to improve analysis performance. Our experiences show that the integration of legacy systems with large-scale data processing can be done without disrupting or supplanting existing pipelines. In addition, it enables features such as better provenance management, data versioning, fault tolerance, and interactive parameter tuning. We believe our approaches are general and that they can be applied to other data analysis pipelines in bioinformatics and other fields.

References

1. Abadi, D., Agrawal, R., Ailamaki, A., Balazinska, M., Bernstein, P.A., Carey, M.J., Chaudhuri, S., Chaudhuri, S., Dean, J., Doan, A., Franklin, M.J., Gehrke, J., Haas, L.M., Halevy, A.Y., Hellerstein, J.M., Ioannidis, Y.E., Jagadish, H.V., Kossmann, D., Madden, S., Mehrotra, S., Milo, T., Naughton, J.F., Ramakrishnan, R., Markl, V., Olston, C., Ooi, B.C., Ré, C., Suciu, D., Stonebraker, M., Walter, T., Widom, J.: The beckman report on database research. Commun. ACM **59**(2), 92–99 (2016)
2. Abu-Doleh, A., Atalyrek, V.: Spaler: Spark and graphx based de novo genome assembler. In: 2015 IEEE International Conference on Big Data (Big Data), pp. 1013–1018 (2015)
3. Apache: Apache HBase. http://hbase.apache.org. Cited 18 April 2016
4. Apache: Avro. http://avro.apache.org. Cited 18 April 2016
5. Apache: Cassandra. http://cassandra.apache.org. Cited 18-April-2016
6. Bhatotia, P., Wieder, A., Rodrigues, R., Acar, U.A., Pasquini, R.: Incoop: MapReduce for Incremental Computations. In: Proceedings of the 2nd ACM Symposium on Cloud Computing, p. 7. ACM Press (2011)
7. Bongo, L.A., Pedersen, E., Ernstsen, M.: Data-intensive computing infrastructure systems for unmodified biological data analysis pipelines. In: Computational Intelligence Methods for Bioinformatics and Biostatistics, *LNBI*, vol. 8623 (2014)
8. Dean, J., Ghemawat, S.: MapReduce. Commun. ACM **51**(1), 107 (2008)
9. Diao, Y., Roy, A., Bloom, T.: Building highly-optimized, low-latency pipelines for genomic data analysis. In: Proceedings of 7th Biennial Conference on Innovative Data Systems Research (2015)
10. Edgar, R., Domrachev, M., Lash, A.E.: Gene expression omnibus: NCBI gene expression and hybridization array data repository. Nucleic Acids Res. **30**(1), 207–210 (2002)
11. EMBL-European Bioinformatics Institute: EMBL-EBI Annual Scientific Report 2014. http://www.ebi.ac.uk/about/brochures. Cited 18 April 2016
12. Fernández-Suárez, X.M., Rigden, D.J., Galperin, M.Y.: The 2014 nucleic acids research database issue and an updated NAR online molecular biology database collection. Nucleic Acids Res. **42**(Database issue), D1–6 (2014)

13. Fitzpatrick, B.: Distributed caching with memcached. Linux J. **2004**(124), 5 (2004)
14. Gentleman, R.C., Carey, V.J., Bates, D.M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M., Rossini, A.J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, J.Y.H., Zhang, J.: Bioconductor: open software development for computational biology and bioinformatics. Genome Biol. **5**(10), R80 (2004)
15. Ghemawat, S., Gobioff, H., Leung, S.T.: The google file system. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles. SOSP '03, pp. 29–43. ACM, New York, NY, USA (2003)
16. Goecks, J., Nekrutenko, A., Taylor, J.: Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol. **11**(8), R86 (2010)
17. Gonzalez, J.E., Xin, R.S., Dave, A., Crankshaw, D., Franklin, M.J., Stoica, I.: Graphx: Graph processing in a distributed dataflow framework. In: 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), pp. 599–613. USENIX Association, Broomfield, CO (2014)
18. Gupta, A., Agarwal, D., Tan, D., Kulesza, J., Pathak, R., Stefani, S., Srinivasan, V.: Amazon redshift and the case for simpler data warehouses. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. SIGMOD '15, pp. 1917–1923. ACM, New York, NY, USA (2015)
19. Have, C.T., Jensen, L.J.: Are graph databases ready for bioinformatics? Bioinformatics **29**(24), 3107–3108 (2013)
20. Kornacker, M., Behm, A., Bittorf, V., Bobrovytsky, T., Ching, C., Choi, A., Erickson, J., Grund, M., Hecht, D., Jacobs, M., Joshi, I., Kuff, L., Kumar, D., Leblang, A., Li, N., Pandis, I., Robinson, H., Rorke, D., Rus, S., Russell, J., Tsirogiannis, D., Wanderman-Milne, S., Yoder, M.: Impala: A modern, open-source sql engine for hadoop. In: CIDR. www.cidrdb.org (2015)
21. Kovatch, P., Costa, A., Giles, Z., Fluder, E., Cho, H.M., Mazurkova, S.: Big omics data experience. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '15, pp. 39:1–39:12. ACM, New York, NY, USA (2015)
22. Leinonen, R., Akhtar, R., Birney, E., Bower, L., Cerdeno-Tárraga, A., Cheng, Y., Cleland, I., Faruque, N., Goodgame, N., Gibson, R., Hoad, G., Jang, M., Pakseresht, N., Plaister, S., Radhakrishnan, R., Reddy, K., Sobhany, S., Hoopen, P.T., Vaughan, R., Zalunin, V., Cochrane, G.: The European nucleotide archive. Nucleic Acids Res. **39**(SUPPL. 1) (2011)
23. Leipzig, J.: A review of bioinformatic pipeline frameworks. Briefings in Bioinformatics (2016)
24. Malewicz, G., Austern, M.H., Bik, A.J., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: A system for large-scale graph processing. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. SIGMOD '10, pp. 135–146. ACM, New York, NY, USA (2010)
25. Melnik, S., Gubarev, A., Long, J.J., Romer, G., Shivakumar, S., Tolton, M., Vassilakis, T.: Dremel: interactive analysis of web-scale datasets. Proc. VLDB Endowment **3**(1–2), 330–339 (2010)
26. Nothaft, F.A., Massie, M., Danford, T., Zhang, Z., Laserson, U., Yeksigian, C., Kottalam, J., Ahuja, A., Hammerbacher, J., Linderman, M., Franklin, M.J., Joseph, A.D., Patterson, D.A.: Rethinking data-intensive science using scalable analytics systems. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. SIGMOD '15, pp. 631–646. ACM, New York, NY, USA (2015)
27. Olston, C., Chopra, S., Srivastava, U.: Generating example data for dataflow programs. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data. SIGMOD '09, pp. 245–256. ACM, New York, NY, USA (2009)
28. Oracle: MySQL. http://www.mysql.com. Cited 18 April 2016
29. Pedersen, E., Bongo, L.A.: Large-scale biological meta-database management. In: Future Generation Computer Systems (2016)

30. Pedersen, E., Raknes, I.A., Ernstsen, M., Bongo, L.A.: Integrating data-intensive computing systems with biological data analysis frameworks. In: Proceedings of 23rd Euromicro International Conference on Parallel, Distributed and Network-based Processing, pp. 733–740. IEEE (2015)
31. Robertsen, E.M., Kahlke, T., Raknes, I.A., Pedersen, E., Semb, E.K., Ernstsen, M., Bongo, L.A., Willassen, N.P.: Meta-pipe - pipeline annotation, analysis and visualization of marine metagenomic sequence data. arXiv:1604.04103 (2016)
32. Schildgen, J., Jorg, T., Hoffmann, M., Dessloch, S.: Marimba: A framework for making mapreduce jobs incremental. In: 2014 IEEE International Congress on Big Data, pp. 128–135. IEEE (2014)
33. Schmuck, F., Haskin, R.: Gpfs: A shared-disk file system for large computing clusters. In: Proceedings of the 1st USENIX Conference on File and Storage Technologies, FAST '02. USENIX Association, Berkeley, CA, USA (2002)
34. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies **0**(5), 1–10 (2010)
35. Stajich, J.E., Block, D., Boulez, K., Brenner, S.E., Chervitz, S.A., Dagdigian, C., Fuellen, G., Gilbert, J.G.R., Korf, I., Lapp, H., Lehväslaiho, H., Matsalla, C., Mungall, C.J., Osborne, B.I., Pocock, M.R., Schattner, P., Senger, M., Stein, L.D., Stupka, E., Wilkinson, M.D., Birney, E.: The Bioperl toolkit: Perl modules for the life sciences. Genome Res. **12**(10), 1611–1618 (2002)
36. Twitter, and Cloudera: Parquet. http://www.parquet.io. Cited 18 April 2016
37. UniProt Consortium: UniProt release 201504. http://www.uniprot.org/help/2015/04/01/release. Cited 18-April-2016
38. Wang, D.L., Monkewitz, S.M., Lim, K.T., Becla, J.: Qserv: A distributed shared-nothing database for the lsst catalog. In: State of the Practice Reports, SC '11, pp. 12:1–12:11. ACM, New York, NY, USA (2011)
39. Wetterstrand, K.: DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP). http://www.genome.gov/sequencingcosts. Cited 18-April-2016
40. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: Cluster Computing with Working Sets. In: Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, p. 10 (2010)

Chapter 14
Optimal Worksharing of DNA Sequence Analysis on Accelerated Platforms

Suejb Memeti, Sabri Pllana and Joanna Kołodziej

14.1 Introduction

A deoxyribonucleic acid (DNA) sequence contains specific genetic information that is used in the development, functioning, and reproduction of living organisms. The DNA sequence consists of two biopolymer strands that form the so-called *double helix*. These two strands are composed of simpler units called *nucleotides* (also known as *bases*). The four possible nucleotides of a DNA sequence are: *Adenine* (A), *Cytosine* (C), *Guanine* (G), and *Thymine* (T).

Modern genetic sequencing instruments can generate a huge amount of DNA sequences, and therefore high performance data analytic (HPDA) solutions are required. According to [44] the rate of growth of genomic data over the last decade is exponential, doubling approximately every seven months. If the growth continues at the current rate, in the next five years we should reach more than one Exa-bases (4 bases correspond to 1 byte) per year, and approach one Zetta-bases per year by 2025.

Fast analysis of DNA sequences is important in many contexts, such as, detecting the evolution of different bacteria and viruses during an early phase [11], diagnosing genetic predisposition to various diseases, such as cancer or cardiovascular diseases [29], discovery of evolutionary relationship of different organisms, or in DNA forensics for criminal investigation or parentage testing [27].

Recently different software-based approaches for DNA sequence analysis that are designed for multi-core systems have been proposed. [18] presented an implemen-

S. Memeti (✉) · S. Pllana
Department of Computer Science, Linnaeus University, 351 95 Vaxjo, Sweden
e-mail: suejb.memeti@lnu.se

S. Pllana
e-mail: sabri.pallana@lnu.se

J. Kołodziej
Cracow University of Technology, 31 155 Cracow, Poland
e-mail: jokoldziej@pk.edu.pl

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_14

tation of the Aho-Corasick algorithm based on pattern partitioning. A prefix-based input partitioning approach is presented by [13]. [26] use a speculation approach based on the most visited states to determine the starting state on a thread's sub-input. Related work has addressed the problem of pattern matching algorithms on heterogeneous systems that are accelerated with GPU devices. [25] proposed the PFAC algorithm for pattern matching on GPUs. [6] presents a parallel implementation of the Knuth–Morris–Pratt pattern matching algorithm for GPU. [46] proposed Aho-Corasick for DNA analysis on clusters of GPUs.

So far not much research was focused on DNA sequence analysis using pattern matching algorithm that is designed for heterogeneous systems that are accelerated with the Intel Xeon Phi coprocessor. Such platforms deserve our attention because of the high performance, programmability, and portability [7, 10, 19, 42, 47]. Furthermore, determining the optimal number of threads, thread affinities, and worksharing for multi-core processors of the host and the accelerating devices using enumeration of all possibilities may be prohibitively time consuming. Let us consider that the optimal system configuration is determined by a set of parameters $C = \{c_1, c_2, .., c_m\}$, where each parameter c_i has a value range R_{c_i}. The number of possible system configurations is a product of parameter value ranges,

$$\prod_{i=1}^{m} R_{c_i} = R_{c_1} \times R_{c_2} \times .. \times R_{c_m} \tag{14.1}$$

In this chapter, we describe an optimized HPDA solution for DNA sequence analysis on heterogeneous systems that are accelerated with the Intel Xeon Phi coprocessor. Our parallel DNA sequence analysis algorithm is based on finite automata and finds patterns in large-scale DNA sequences. The algorithm implementation targets heterogeneous systems that are accelerated with the Intel Xeon Phi coprocessor and exploits both the thread-level and the SIMD parallelism.

Our optimization approach combines combinatorial optimization and machine learning to determine the optimal system configuration parameters (such as, number of threads, thread affinities, DNA sequence fractions for the host and accelerating device) of a *heterogeneous* system such that the overall execution time of the DNA sequence analysis is minimized. We propose to use a combinatorial optimization method, such as the *Simulated Annealing*, for searching for the optimal system configuration in the given parameter space. For each system configuration suggested by simulated annealing, we use machine learning (in our case, the Boosted Decision Tree Regression) for evaluation of system performance. The objective function that we aim to minimize is the execution time of the DNA sequence analysis. We evaluate our approach experimentally with real-world DNA sequences (of human, mouse, cat, dog) using a heterogeneous platform that comprises two 12-core Intel Xeon E5 CPUs and an Intel Xeon Phi 7120P coprocessor with 61 cores. Using the near optimal system configuration determined by our optimization approach, we achieved a speedup of 1.74× compared to the case when all the cores of the host are used, and up to 2.21× speedup compared to the fastest execution time on the device.

The major contributions of this chapter include

1. an algorithm for parallel DNA sequence analysis that allows efficient use of the computational resources on the host and accelerating device,
2. an optimization approach that determines the optimal worksharing between the host and the accelerating device,
3. an empirical evaluation of our approach for optimized DNA sequence analysis using real-world DNA sequences of human (3.2 GB), mouse (2.7 GB), cat (2.4 GB), dog (2.4 GB).

The rest of the chapter is organized as follows. Section 14.2 provides background information with respect to pattern matching with finite automata and accelerated computing platforms. Our methodology for optimized DNA sequence analysis on accelerated systems is described in Sect. 14.3. Section 14.4 presents the experimentation environment and discusses the experimental evaluation results. The work described in this chapter is compared and contrasted to the state-of-the-art related work in Sect. 14.5. Section 14.6 concludes the chapter and discusses the future work.

14.2 Background

In this section we provide background information on the Aho-Corasick algorithm for pattern matching. Furthermore we describe a heterogeneous computing platform that is accelerated with the Intel Xeon Phi coprocessor.

14.2.1 Pattern Matching with Finite Automata

The process of pattern matching verifies whether a pattern is present in a string. Pattern matching is commonly used for determining the locations of a pattern within a sequence of tokens, in search and replace functions, or to highlight important information out of a huge data set. In the context of computational biology, pattern matching is used for analyzing and processing biological information in order to extract the useful parts of the data and make them evident.

Formally, in DNA sequence analysis, the process of pattern matching can be defined as follows: the input text (DNA sequence) is an array $T[1..n]$ where n is the length of the input, and pattern $P[1..m]$ where the length of the pattern $m \leq n$. The finite alphabet $\sum$ defines the possible characters of the input string, in this case $\sum = \{A, C, G, T\}$, each item of $\sum$ corresponds to one of the four nucleotide bases.

A finite automaton (FA) is a machine for processing information by scanning the input text T in order to find the occurrences of the pattern P. Formally, a FA can be defined as follows: FA is a quintuple of $(Q, \sum, \delta, q_0, F)$, where Q is the finite set of states, $\sum$ is the finite alphabet, δ is the transition function $Q \times \sum \longrightarrow Q$, q_0 is the

start state and F is the distinguished set of final states. FA is an efficient technique for pattern matching, because it examines each character from T exactly once.

A well-known algorithm for multiple pattern matching is the Aho-Corasick algorithm. It is able to match any occurrences (including the overlapping ones) of multiple patterns linearly to the size of the input string. It examines each character of the input string only once. It builds an automaton by creating states and transitions corresponding to these states. It adds failure transitions when there is no regular transition leaving from the current state on a particular character, which makes it possible to match multiple and overlapping occurrences of the patterns. Furthermore, this algorithm is capable of delivering input-independent performance if implemented efficiently in parallel systems, which is a reason why we use this algorithm as basis of our work.

14.2.2 Systems Accelerated with the Intel Xeon Phi

A typical heterogeneous platform that is accelerated with the Intel Xeon Phi is diagrammed in Fig. 14.1. Such platforms may consist of one or two CPUs on the host (left-hand side of the figure), and one to eight accelerators (right-hand side of the figure). The host CPUs shown in Fig. 14.1 are of type Xeon E5 2694 v2, which comprise three columns of four Ivy Bridge Cores (IVB-C) that amount to a total of 12 cores. These cores are connected using a three ring design, which features low latency and high throughput. The L3 cache is split in two parts, in total it features a 30MB L3 cache. The CPUs are connected to the memory using the quad-channel memory controller. The two host CPUs are linked through the QuickPath Interconnect, which offers speed of up to 8.0 GT/s.

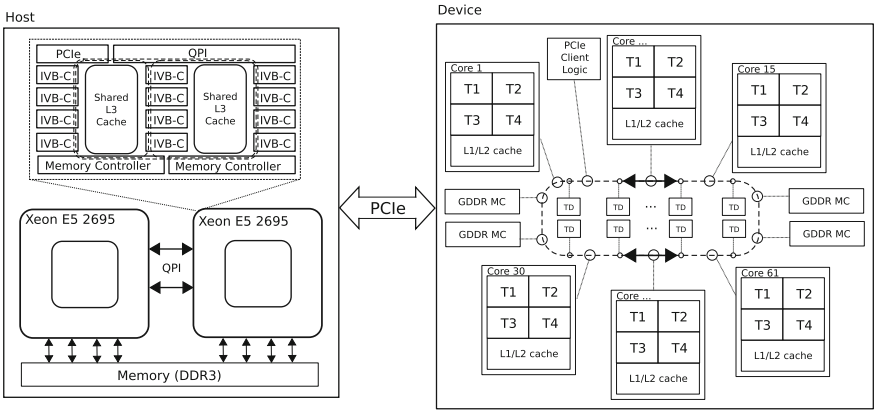


Fig. 14.1 Our target accelerated system comprises a host with two CPUs and an Intel Xeon Phi device. The host CPUs consists of 12 Intel Ivy Bridge cores. The Intel Xeon Phi consists of 61×86 cores, each core with 4-way symmetric multi-threading and a 512 kb private slice of the unified L2 cache

The Intel Xeon Phi (codenamed Knights Corner) device is a many-core shared memory coprocessor, which runs a lightweight Linux Operating System that offers the ability to connect over *ssh*. In our system we use the Intel Xeon Phi coprocessor 7120P. The Intel Xeon Phi offers two programming models

1. *offload*—parts of the applications running on the host are offloaded to the coprocessor
2. *native*—the code is compiled specifically for running natively on the coprocessor. The code and all the required libraries have to be transferred on the device.

The Intel Xeon Phi comprises 61×86 cores, each core runs at 1.2 GHz base frequency, and up to 1.3 GHz on max turbo frequency [10]. Each core has four hardware threads (T1, T2, T3, T4), which amounts to a total of 244 threads per coprocessor. Each core has its own L1 (32 KB) and L2 (512 KB) cache. The L2 cache is kept fully coherent by a global distributed tag-directory (TD). The cores are connected through a bidirectional ring bus interconnect, which forms a unified shared L2 cache of 30.5 MB. In addition to the cores, there are 16 memory channels that in theory offer a maximum memory bandwidth of 352 GB/s. The GDDR memory controllers provide direct interface to the GDDR5 memory, and the PCIe Client Logic provides direct interface to the PCIe bus. The Intel Xeon Phi, theoretically, is capable of delivering up to one teraFLOP/s of double-precision performance, or two teraFLOP/s of single-precision performance. One of the key features of the Intel Xeon Phi is its vector processing units that are essential to fully utilize the coprocessor [45]. Through the 512-bit wide SIMD registers it can perform 16 (16 wide $\times$ 32 bit) single-precision or 8 (8 wide $\times$ 64 bit) double-precision operations per cycle.

The host communicates with the coprocessor through the PCIe bus which is limited to 8GB/s transfer bandwidth. The PCIe bus is a bottleneck for the offload programming model, where data has to be transferred from the host to the coprocessor and vice versa. To minimize this overhead, it is recommended that the data is transferred to the coprocessor and is kept there (reused).

14.3 Methodology

In this section we describe our parallel algorithm for DNA sequence analysis on accelerated systems. Thereafter, we describe our approach for DNA sequence worksharing among computational resources of the host and accelerating device such that the overall execution time is reduced.

Algorithm 1 DFA-based Parallel DNA Sequence Analysis

Input: STT, final state f, input I, number of threads p, vector length v, pattern length m
Output: Count of pattern matches and their location

```
1: procedure AC(STT, f, T, p, m)
2:     n = I.length
3:     chunkLength = n/p
4:     for i = 1 to p do
5:         q = 0
6:         chunkStart = i * chunkLength
7:         for j = 1 to chunkLength do
8:             c = j + chunkStart
9:             q = δ(q, I[c])   ▷ load the next node from STT, by following the transition from the
                   current node q labeled by the symbol T[c]
10:            if q ≥ f then                    ▷ check if the transition to next node is final
11:                print matching pattern at position c
12:            end if
13:        end for
14:    end for
15: end procedure
```

14.3.1 Design and Implementation Aspects of our Algorithm for DNA Sequence Analysis

The key features of our deterministic finite automata (DFA) based algorithm for parallel DNA sequence analysis are: (1) exploiting the thread-level parallelism by using a decomposition approach to split the input among the available threads, (2) exploiting the SIMD-level parallelism, and (3) reducing the memory references using a suitable representation for the state transition table (STT).

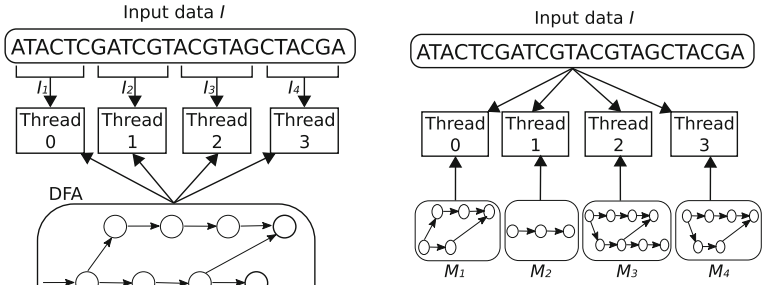
14.3.1.1 Exploiting the Thread- and SIMD-Level Parallelism

Figure 14.2a depicts two possible ways for parallel execution of regular expression matching for bio-computing applications:

- *input-based partitioning* (see Fig. 14.2a) that splits the input string into smaller chunks and processes them in separate threads, and
- *pattern-based partitioning* (see Fig. 14.2b) that splits the patterns in sub-patterns, such that for each sub-pattern a separate DFA is built and each thread examines the given DNA sequence with one of the sub-patterns [18].

The *input-based partitioning* approach is suitable for applications where the execution time is dominated by the input length, whereas the *pattern-based partitioning* approach is suitable for application where the execution time is dominated by the size of the pattern data set.

We use the *input-based partitioning* strategy to exploit both thread and SIMD-level parallelism, which is based on splitting the input by the number of available threads



(a) *input-based partitioning approach* - splits the input into smaller chunk sand processes them in separate threads.

(b) *pattern-based partitioning approach* - splits the patterns in sub-patterns. For each of the sub-patterns a separate finite automaton is created. Each thread examines the input with each different automaton.

Fig. 14.2 Load balancing strategy among the available threads. The input-based partitioning is suitable when the execution time is dominated by the length of the input text, whereas the pattern-based partitioning approach is suitable when the execution time is dominated by the size of the pattern set

and/or length of vector registers. This process is simple and straightforward, however determining the exact initial state for each sub-input is nontrivial task. Finding the exact initial state is important to match the occurrences of patterns that appear in the cross boundaries. Researchers have proposed different solutions to this problem, for example [26] uses a speculation approach based on the most visited states, [9] uses an approach based on suffix arrays.

Whether the length of patterns within a pattern set is the same or not, we present two methods of matching occurrences of patterns that appear in the crossing border

- a pattern length independent approach, which uses a converging point to merge the matches starting from different possible initial states, and
- a pattern length dependent approach, which overlaps the chunks by $m - 1$ characters, where m represents the pattern length.

The first approach includes the following steps: (1) find the set of source states (L) for the first element of the sub-input mapped to the running thread (T_n); (2) find the set of destination states (S) for the last character of the sub-input mapped to the previous thread (T_{n-1}); (3) find the intersection of S and L ($S \cap L$), which is the set of possible initial states [30, 32]. The first thread (T_0) always starts from the initial state q_0. Each thread is responsible for finding the set of possible initial states, and for each state of this set a regular expression matching is performed. However, only one regular expression matching is required to be fully completed, the others will converge after a very small number of characters is examined. When all threads have finished their job, the results are joined by a binary reduction, which connects the last visited state of T_n to the first visited state of T_{n+1}. In comparison to

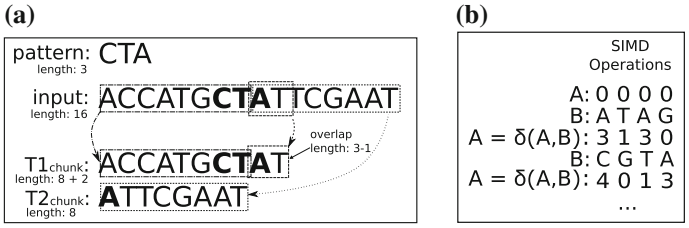


Fig. 14.3 Thread-level and SIMD parallelism; **a** splitting the DNA sequence into chunks; **b** vectorization of the transition function

the first approach, the second one is simpler to implement, however it can be used only to match multiple patterns of the same length [31]. For our experiments we use pattern set that comprises multiple patterns of the same length, therefore we use the second approach. The process of splitting the input among the available threads and overlapping them to match the cross boundary occurrences is depicted in Fig. 14.3a.

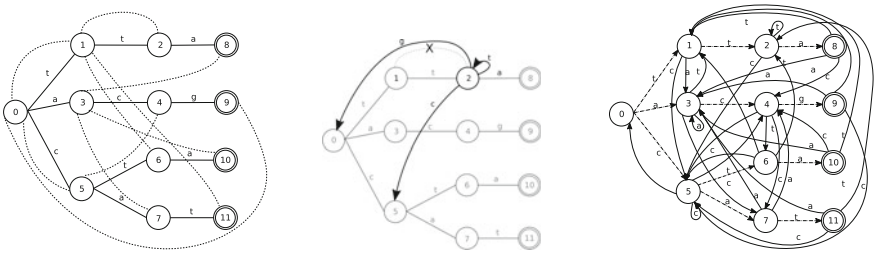
We exploit the SIMD parallelism of the Intel Xeon Phi using a similar strategy. Multiple transitions per cycle are performed using the SIMD-δ function. Figure 14.3b illustrates the SIMD operations assuming that the input assigned to a thread is the same one as in Fig. 14.3a. The first SIMD δ operations are performed on the characters at positions 0, 4, 8, and 12, the second SIMD operations are performed on characters at position 1, 5, 9, and 13, and so on. This function allows for efficient use of the vector processing unit by performing simultaneously multiple transitions. An extract from the vectorization report for the SIMD-δ function, which is shown in Listing 14.1, depicts an estimated potential speedup of 2.6×.

Listing 14.1 Vectorization Report for the SIMD-δ function

```
remark #15475: --- begin vector loop cost summary ---
remark #15476: scalar loop cost: 36
remark #15477: vector loop cost: 12.930
remark #15478: estimated potential speedup: 2.600
remark #15479: lightweight vector operations: 42
remark #15480: type converts: 3
remark #15475: --- end vector loop cost summary ---
```

14.3.1.2 Implementation Aspects

As a basis for our DNA Sequence analysis approach we use the Aho-Corasick algorithm. It is capable of matching any occurrences of multiple patterns, and promises the capability to deliver input-independent performance for parallel execution. However, the nondeterministic automaton that is generated by this algorithm is considered as a drawback with respect to the performance. We find the right transition for each state and each character and build a deterministic automaton with only valid transitions. Having a valid transition for each symbol guarantees that for each character the same number of operations will be performed.



(a) AC NFA automaton (b) Replacing failure transitions (c) AC DFA automaton

Fig. 14.4 The process of converting an Aho-Corasick NFA into a DFA by replacing the failure transitions with valid ones. The automaton in the *left-hand side* shows the NFA generated using the Aho-Corasick algorithm, which is able to match overlapping occurrences of the following patterns: *tta, acg, cta, cat*. The *middle figure* shows the process of replacing a failure transition with valid transitions; and in the *right-hand side* is depicted the result of this process, a DFA that corresponds to the same NFA

This process is depicted in Fig. 14.4. Figure 14.4a represents the AC-NFA that is able to match overlapping occurrences of the following patterns: *tta, acg, cta, cat*. The process of finding valid transitions for a selected state is depicted in Fig. 14.4b, where the corresponding valid transitions from state q_2 are added and the failure transition is deleted. Figure 14.4c depicts the corresponding DFA to the NFA automaton shown in Fig. 14.4a.

Regarding the process of checking if a match (represented as final state in DFA) appeared, we use a simplified approach, which includes reordering of the number of states. Reordering of the states is done in such a way that the regular states are numbered from 0 to $a - b$, and the final states are numbered from a to b, where a is the length of the set of states, and b is the length of the set of final states. Determining whether a match has been found is done by checking if the state number is greater or equal to $a - b$. For example, finding whether a state is a final one or not in the automaton in Fig. 14.4c can be done by simply checking if the current states id is greater or equal to seven (see algorithm 1 line 10).

An equivalent representation of the DFA is the state transition table (STT), which forms a two-dimensional matrix M of size $x \times y$, where x is the total number of states, and y is the total number of elements in $\sum$. Thus for a given state i and an input character j, an entry $M[i][j]$ denotes the corresponding next state. This form of representation, requires a symbol pointer to the columns of the STT (such as, $A \rightarrow 0$, $C \rightarrow 1$, $G \rightarrow 2$ and $T \rightarrow 3$). We consider creating a (ASCII) STT with 256 columns, where only columns that correspond to the elements in $\sum$ represent valid transitions, whereas the unused symbols represent failure transitions to the initial state. In such a way, when $M[i][j]$ entry is needed, instead of asking for j-s pointer, we use j-s ASCII code to get the entry. Such representation of the STT is more memory expensive, however it guarantees less operations, so we consider it a reasonable trade-off between memory space and access speed.

14.3.1.3 Accelerated DNA Sequence Analysis

One of the most compelling features of the Intel Xeon Phi device is the so-called double advantage of transforming-and-tuning, which means that applications designed for the Intel Xeon Phi will most probably perform well on the Intel Xeon CPUs. In other words, tuning an application on Intel Xeon Phi device for scaling (more cores and threads), vectorization and memory usage, benefits an application when running on the Intel Xeon processors.

Our approach for parallel DNA sequence analysis using resources of the host and the device also benefits from the double advantage of transforming-and-tuning feature. This is why with not much additional programming effort we can use the same algorithm for both host and device. We use *offload* programming model which allows us to overlap the execution of the application on the host and device. However when hybrid execution is performed, the workload is partitioned such that both the load of host processors and the load of Xeon Phi device is balanced.

Our approach is depicted in Fig. 14.5. The preprocessing phase includes two steps: (1) Splitting the DNA sequence I into the part I' assigned to the host CPUs and the part I'' assigned to the Xeon Phi device, and (2) Building the STT.

The DNA sequence is split based on a given fraction ratio F, which determines the percentage of the fraction that needs to be processed on the host I'. The remaining part I'' is assigned to be processed on the device.

$$I' = (F/100) * I \tag{14.2}$$

$$I'' = I - I' \tag{14.3}$$

An asynchronous host-to-device transfer is initiated immediately, which transfers I'' to the coprocessor. The *STT Builder* generates a deterministic finite automaton for the given set of patterns. The process of transferring I'' and the building of the STT is overlapped. Beside I'', once the STT is constructed it is also copied to the devices memory. Both host and device use a similar algorithm implementation to match the assigned DNA sequence fraction against the given set of patterns. The device transfers the result (the total number and the location of matched patterns) to the host, and a simple join is performed.

14.3.2 Determining Optimal System Configuration using Combinatorial Optimization and Machine Learning

In this section we describe our approach for determining the optimal system configuration parameters (such as, number of threads, thread affinities, DNA sequence fractions for the host and accelerating device) of a heterogeneous system such that

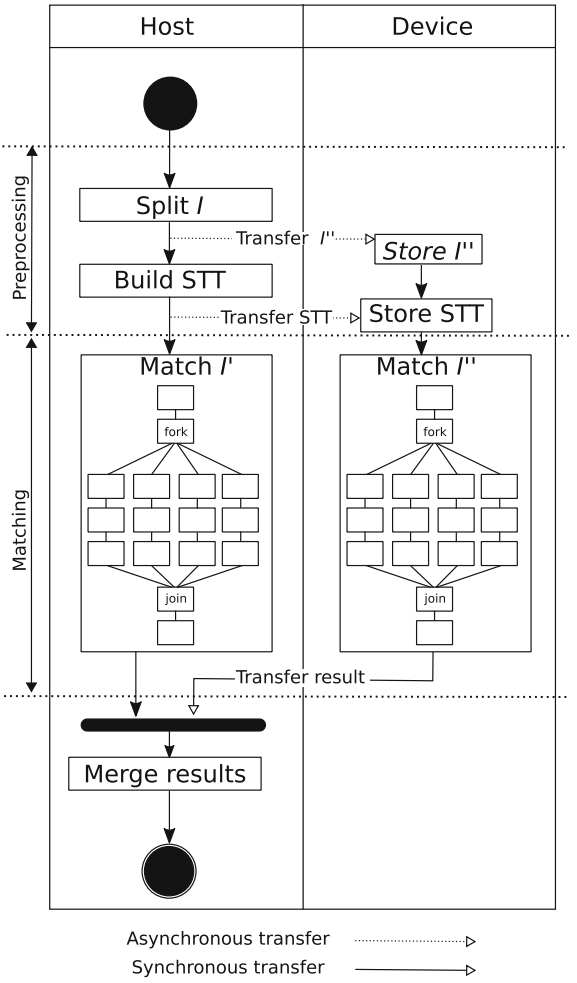


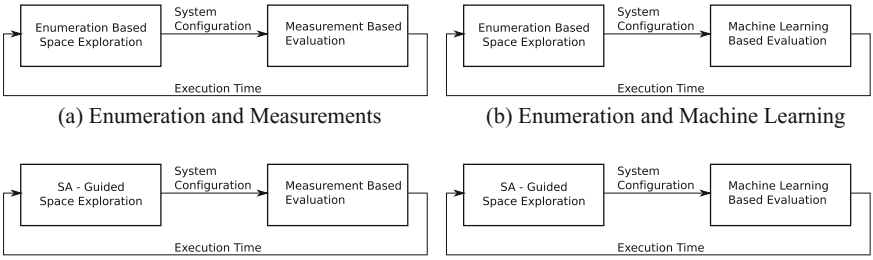
Fig. 14.5 Parallel DNA sequence analysis using resources of the host and the device. The preprocessing phase includes two steps: (1) Splitting the DNA sequence I into the part I' assigned to the host CPUs and the part I'' assigned to the Xeon Phi device, and (2) Building the STT

the overall execution time of the DNA sequence analysis is minimized. Table 14.1 lists the parameters of our target system.

For determining the optimal system configuration one could try out all possible parameter values. We refer to this method as *enumeration*. For evaluation of system performance for each system configuration we may use *measurements* or *model-based prediction*. For development of the performance model we may use *machine learning*. Using enumeration for design space exploration in a real-world context may be prohibitively time consuming. Therefore, we propose to use a combinatorial optimization method, such as the *Simulated Annealing*, for searching for the optimal system configuration in the given parameter space. For instance, we may use the Simulated Annealing to guide parameter space exploration, and use measurements or

Table 14.1 The set of considered parameters and their values for our target system

	Host	Device
Threads	{2, 4, 6, 12, 24, 36, 48}	{2, 4, 8, 16, 30, 60, 120, 180, 240}
Affinity	{none, scatter, compact}	{balanced, scatter, compact}
DNA sequence fraction	{1..100}	{100—Host Sequence Fraction}
DNA sequences	{Human.fna, Mouse.fna, Cat.fna, Dog.fna}	



(a) Enumeration and Measurements (b) Enumeration and Machine Learning

(c) Simulated Annealing and Measurements (d) Simulated Annealing and Machine Learning

Fig. 14.6 System optimization approaches

machine learning for evaluation of system performance. We have considered various optimization methods (see Fig. 14.6),

(a) Enumeration and Measurements (EM)
(b) Enumeration and Machine Learning (EML)
(c) Simulated Annealing and Measurements (SAM)
(d) Simulated Annealing and Machine Learning (SAML)

Table 14.2 lists major properties of the considered optimization methods. While EM determines certainly the optimal system configuration, it involves a very large number of performance experiments and the expected optimization effort is high.

Table 14.2 Properties of optimization methods

Method	Space exploration	Sys. conf. evaluation	Effort	Accuracy	Prediction
EM	Enumeration	Measurements	High	Optimal	No
EML	Enumeration	Machine learning	High	Near-optimal	Yes
SAM	Simulated annealing	Measurements	Medium	Near-optimal	No
SAML	Simulated annealing	Machine learning	Medium	Near-optimal	Yes

Since EM has no performance prediction capabilities, for each new DNA sequence we need to repeat the whole optimization process. EML uses machine learning to infer about the system performance in case of a new DNA sequence. However, the effort for parameter space exploration is high. SAM and SAML reduce significantly the effort for parameter space exploration. Compared to SAM, SAML provides the possibility to predict the system performance for new DNA sequences.

In what follows in this section, we describe our approach for parameter space exploration using Simulated Annealing and our approach for performance prediction using Machine Learning.

14.3.2.1 Using Simulated Annealing for Space Exploration

There are several heuristic methods [39] for solving optimization problems, including: Genetic Algorithms, Ant Colony Optimization, Simulated Annealing, Local Search, Tabu Search. Choosing the most convenient heuristic depends on various factors such as, the type of optimization problem and search space, available computational time, or demanded solution quality. We have decided to use simulated annealing because of its ability to cope with a very large discrete configuration space, and its global optimization features that promise that it will not get stuck into local optima.

The simulated annealing method is inspired by the process of material cooling and annealing. At a high temperature particles of the material have more freedom of movement, and as the temperature decreases the movement of particles is decreased as well. When the material is cooled slowly the particles are ordered in the form of a crystal, which represents its minimal energy state.

Similarly in the simulated annealing, there is a temperature T variable that simulates the cooling process. While T is higher, it is more likely to accept new solutions. Therefore, there is a corresponding chance to get out of a local minimum, in favor of searching for a global optimum. The lower the temperature, less likely it accepts new solutions [39].

The method of simulated annealing is a suitable technique for optimization of large-scale problems, especially the ones where the global optimum is hidden among many local optima. Examples like the traveling salesman problem (TSP) or designing complex integrated circuits are just some of many problems that can be solved using the simulated annealing. The space over which the objective function is defined is discrete and very large (factorial) configuration space, for example, in the TSP the set of possible orders of cities.

In the context of optimizing the DNA sequence analysis on heterogeneous systems, the configuration space is as follows:

- sequence partitioning is a discrete value from 0–100, which means that if 40 % of the sequence is assigned to the host, the remaining 60 % is assigned to the device;
- number of threads for the host and device;
- the thread allocation strategy for the host and device.

The objective function E (analog of energy) of our approach is defined as the total execution time of the application running on the host and device, which basically is determined by the maximum of the T_{host} and T_{device}:

$$E = max(T_{host}, T_{device}) \tag{14.4}$$

We have an initial value of T and a cooling rate $coolingRate$, which define the annealing schedule

$$T = T * (1 - coolingRate); \tag{14.5}$$

We use the T value in our acceptance function to decide which solution to accept. When a new solution is generated, we first check whether its energy E is lower than the energy of the current solution. If it is, we accept it unconditionally, otherwise we consider how much worse is the time of the new solution compared to the current one, and what is the temperature of the system. If the temperature is high, the system is more likely to accept solutions that are worse. The so-called Boltzmann probability distribution (p) [39] is expressed as follows

$$p = exp((E - E')/T) \tag{14.6}$$

where E' determines the energy of the newly generated solution.

The acceptance probability distribution p allows the system to get out of the local optima, and find a new better global one.

14.3.2.2 Using Machine Learning for Performance Evaluation

We use machine learning to predict the execution time of DNA sequence analysis on the host T_{host} and device T_{device}. We use the predicted execution times to determine the optimal system configuration. The aim is to achieve a good load balancing between the host and the device, such that the overall execution time is reduced.

For the development of the performance prediction model we have considered various supervised machine learning approaches [41]: Linear Regression, Poisson Regression, and the Boosted Decision Tree Regression. In our performance prediction context, we achieved more accurate prediction results using the Boosted Decision Tree Regression. The Boosted Decision Tree Regression is a supervised learning algorithm that uses boosting to generate a group of regression trees and determine the optimal tree based on a loss function.

The execution time for a parallel regular expression matching is mainly influenced by the length of the DNA sequence and the available resources of the computing platform. Furthermore thread affinities may affect significantly the execution time of a parallel DNA sequence analysis.

We have generated the training data for the performance prediction model by executing our DNA sequence analysis algorithm for different numbers of threads,

thread affinities, and DNA sequences. Table 14.1 lists the features and their possible values used to train and evaluate our prediction model.

We trained our model by running experiments on two different environments (host and device), with 2, 4, 6, 12, 24, 36, and 48 threads for the host, and 2, 4, 8, 16, 30, 60, 120, 180 and 240 threads on the Xeon Phi. Furthermore we used *none, scatter, compact* thread affinity modes for the host, and *balanced, scatter, compact* thread affinity for the Xeon Phi. We trained our performance prediction model with different input fractions of four DNA sequences of different organisms (Human, Mouse, Cat, and Dog). In total, the data of about 7200 experiments were used to train and evaluate the performance prediction model using the Boosted Decision Tree Regression. Half of the experiments were used to train the prediction model and the other half were used for evaluation.

14.4 Evaluation

In this section, we evaluate experimentally our combinatorial optimization approach for DNA sequence analysis on heterogeneous platforms. We describe the following,

- the experimentation environment,
- scalability of our parallel algorithm implementation for DNA sequence analysis,
- evaluation of our prediction model,
- comparison of the SAML with SAM and EM,
- achieved performance improvement.

14.4.1 Experimentation Environment

In this section we describe the experimentation environment used for the evaluation of our approach including the system configuration, the used data set (that is DNA sequences), the set of DNA patterns, and the parameters that define the system configuration.

The experiments were performed on a platform that consists of two Intel Xeon E5 processors that are accelerated with one Intel Xeon Phi of type 7120P coprocessor. The major features of our *Emil* system at the Linnaeus University are listed in Table 14.3. An abstract view of the corresponding architecture for the host CPUs and the device accelerator is depicted in Fig. 14.1. Each of the host CPUs has 12 cores (each core supports two hardware threads known as logical cores) that amount to a total of 24 cores and 48 threads. The host CPUs have 30 MB L3 cache each, and they are connected to the main memory using the Quick Path Interconnect that offers speed of 8.0 GT/s.

The Intel Xeon Phi coprocessor has 61 cores (each of the cores support four hardware threads), in total 244 threads per coprocessor. The μOS, which is a lightweights

Table 14.3 *Emil*: hardware architecture

Specification	Intel Xeon	Intel Xeon Phi
Type	E5-2695v2	7120P
Core frequency	2.4–3.2 GHz	1.238–1.333 GHz
# of Cores	12	61
# of Threads	24	244
Cache	30 MB	30.5 MB
Max Mem. Bandwidth	59.7 GB/s	352 GB/s
Memory	8 × 16 GB	16 GB
TDP	115 W	300 W

Table 14.4 DNA data set

	Gneome reference	Size (MB)
Human	GRCh38	3250
Mouse	GRCm38.p2	2830
Cat	Felis_catus-6.2	2490
Dog	CanFam3.1	2440

version of Linux Operating System, runs in one of these cores, so 60 of the cores are available for our experiments. The coprocessor runs the Intel Manycore Platform Software Stack version 3.1.1. The Xeon Phi cores have a unified L2 cache of 30 MB.

For experimental evaluation of our approach we have used real-world DNA sequences of human, mouse, cat, and dog. These DNA sequences are extracted from the GenBank sequence database of the National Center for Biological Information [34]. Table 14.4 lists the information about the genome references and the corresponding length of the DNA sequences. The data sizes vary from 2.38 GB up to 3.17 GB, which allows us to evaluate our algorithm with different sizes.

We use a set of patterns from the *regex-dna* benchmark that matches and extracts specific k-mers [16] from a DNA sequence. The set of patterns is given as input to our algorithm in form of a regular expression, which are used to construct a DFA of 137 states that corresponds to a sparse transition table of 137 rows and 256 columns.

The parameters that define the system configuration for our combinatorial optimization approach are shown in Table 14.1. All the parameters are discrete. The considered values for the number of threads for host are {2, 6, 12, 24, 36, 48}, whereas for device are {2, 4, 8, 16, 30, 60, 120, 180, 240}. The thread affinity can vary between $\{none, compact, scatter\}$ for the host, and $\{balanced, compact, scatter\}$ for the device. The DNA Sequence Fraction parameter can have any number in the range {0, .., 100}, such that if 60 % of the DNA sequence is assigned for processing to the host, the remaining $100 - 60 = 40$ % is assigned to the device.

14.4.2 Evaluation of our Parallel Algorithm for DNA Sequence Analysis

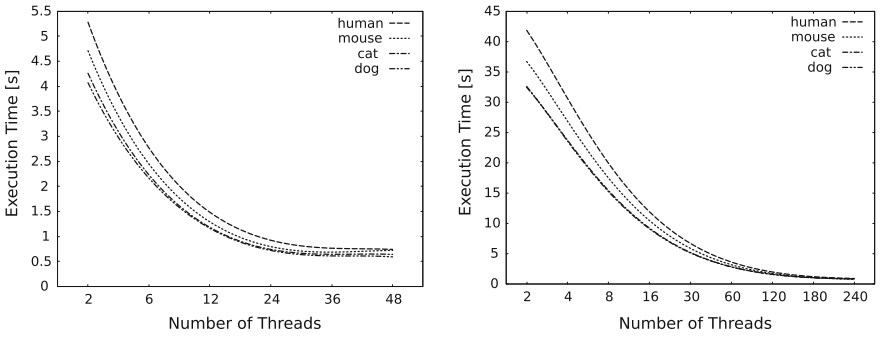
Execution time of our parallel DNA sequence analysis algorithm for various DNA sequences and various number of threads is depicted in Fig. 14.7. We may observe a good scalability of our algorithm when running on host (Fig. 14.7a) and device (Fig. 14.7b). The host has 24 physical cores, each supporting two hyper-threads leading to a total of 48 hyper-threads (or logical cores). Figure 14.7a shows that there is no significant performance gain when 48 threads are used compared to the case with 24 threads.

With respect to the speedup, we first compare the performance of our parallel algorithm for n threads and one 1 thread (Fig. 14.8), and thereafter the performance of our parallel algorithm is compared with a sequential implementation (Fig. 14.9).

We may observe in Fig. 14.8 that our algorithm achieves a maximum speedup of 18.5 for 24 threads of the host processors while analyzing the DNA sequence of dog. We achieve a maximum speedup of 78.73 for 240 threads on the coprocessing device while analyzing the DNA sequence of mouse.

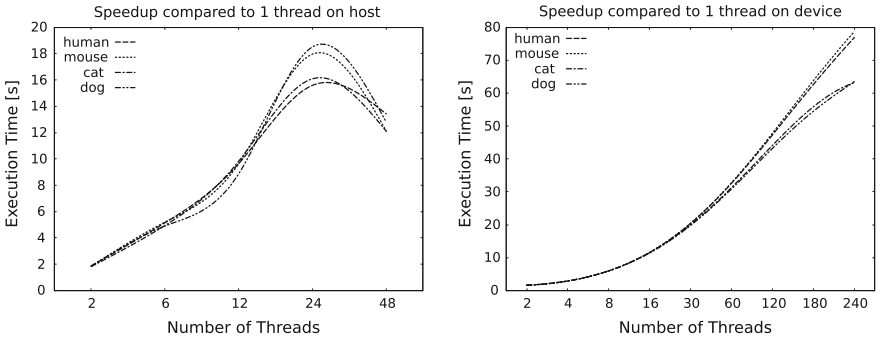
In Fig. 14.9 we show the speedup of our parallel algorithm compared to the sequential implementation of the Aho-Corasick algorithm that does not consider our algorithmic optimizations described in Sect. 14.3.1. As depicted in Fig. 14.9a, our approach running on host is up to 51.98× better when using 24 threads to analyze the DNA sequence of dog. For 240 threads in the coprocessing device, we achieve a maximal speedup of 31.50 when analyzing the DNA sequence of mouse.

The performance results discussed in this section are achieved for *compact* thread affinity on the host and *balanced* thread affinity on the device.



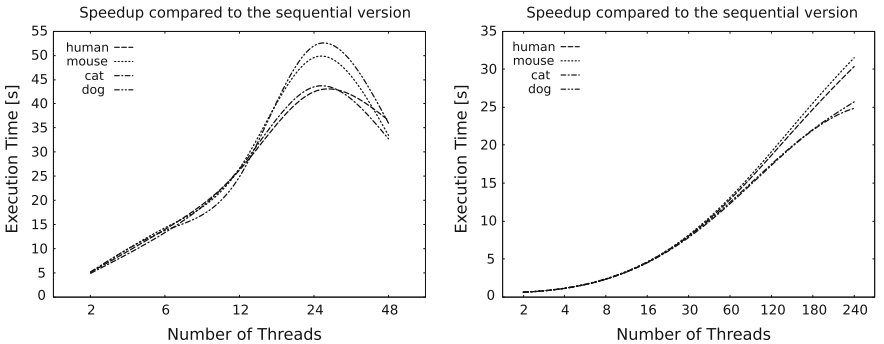
(a) Host (b) Device

Fig. 14.7 Performance of our parallel DNA sequence analysis algorithm on host and device



(a) Host (b) Device

Fig. 14.8 The speedup of our DNA sequence analysis algorithm compared to the execution running in one thread of host and device



(a) Host (b) Device

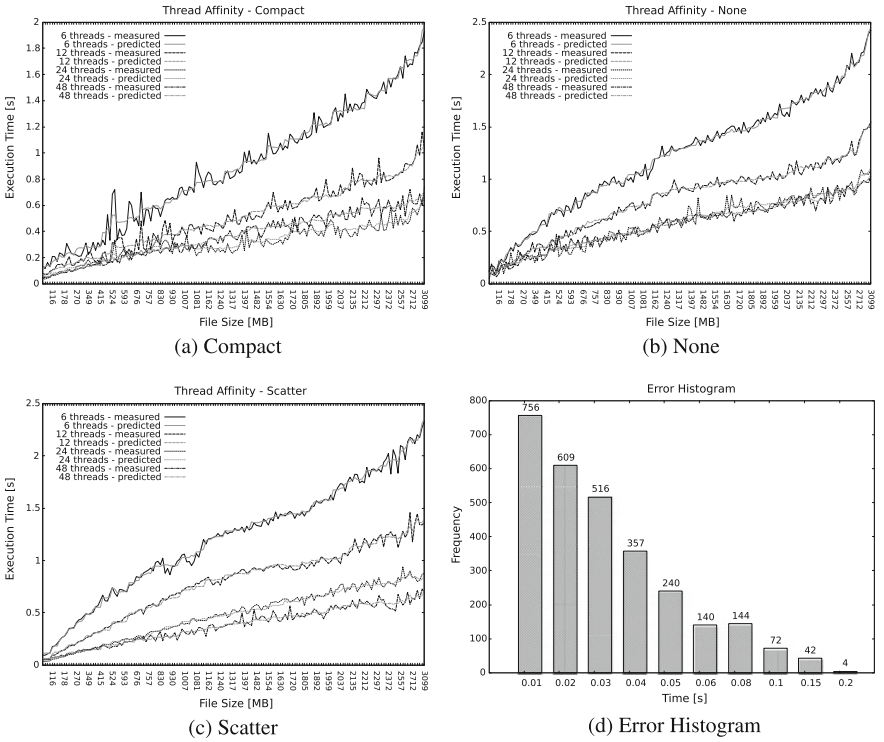
Fig. 14.9 The speedup of our DNA sequence analysis algorithm with respect to a sequential implementation of the Aho-Corasick algorithm

14.4.3 Evaluation of our Performance Prediction Model

We have trained the performance prediction model for a set of four DNA Sequences of different organisms (Human, Mouse, Cat, and Dog). A total of 7200 experiments (2880 on host and 4320 on device) were performed, half of which were used to train the prediction model, and the other half is used to evaluate the model. We use the absolute error and the percent error to express the prediction accuracy,

$$absolute\_error = |T_{measured} - T_{predicted}| \tag{14.7}$$

$$percent\_error = 100 \cdot absolute\_error / T_{measured} \tag{14.8}$$



(a) Compact (b) None

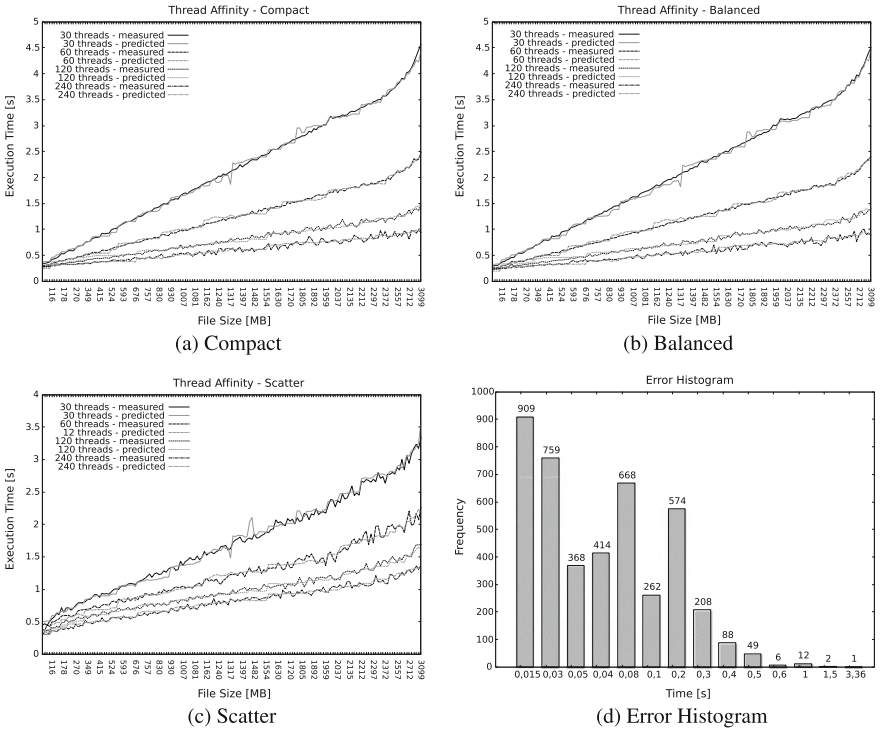
(c) Scatter (d) Error Histogram

Fig. 14.10 Performance prediction accuracy for the device. A total of 4320 experiments with DNA sequences of human, mouse, cat, and dog were needed. Half of the experiments are used to train the model, and the other half to evaluate it

Figure 14.10a–c show the measured and predicted execution time of DNA sequence analysis on the host CPUs. We perform the experiments for various number of threads, thread affinities, and fractions of the selected DNA sequences. The fractions include 2.5–100 % of the DNA sequence size. We may observe that predicted values match well the measured values execution times for most configurations.

Figure 14.10d depicts a histogram of the frequency of performance prediction absolute error for the host. It shows that most of the absolute error values are low. For instance, 756 predictions have an absolute error less than 0.01 s, 609 predictions have an absolute error in the range 0.01–0.02 s, and the rest of the predictions have an absolute error in the range of 0.02–0.2.

Figures 14.11a–c depict the measurement and prediction results of the execution time on the Intel Xeon Phi device for different number of threads, thread affinities and fractions of the selected DNA sequences. For most of the test cases the predicted execution time values match well the measured values.



(a) Compact (b) Balanced

(c) Scatter (d) Error Histogram

Fig. 14.11 Performance prediction accuracy for the host. A total of 2880 experiments with DNA sequences of human, mouse, cat and dog were needed. Half of the experiments are used to train the model, and the other half to evaluate it

Figure 14.11d depicts a histogram of the frequency of performance prediction absolute errors for the device. Most of the predictions have an absolute error less than 0.3 s.

The average percent error that considers all the tested configurations for different number of threads is shown in Table 14.5. The average percent error for the experiments on the host is 5.239 %, whereas the average percent error on the device is 3.132 %. In other words, the average absolute error on the host is 0.027 s, and 0.074 on the device.

14.4.4 Comparison of SAML with EM

Enumeration (also known as *brute force*) finds the system parameter values that result with the best performance by trying out all values of the parameters of the system under study. While this approach determines certainly the best system configuration, for the large search space of real-world problems enumeration may be prohibitively

Table 14.5 Performance prediction accuracy expressed via the absolute error (s) and percent error (%) for the host and device

Threads	Absolute (s)	Percent (%)
Host		
2	0.032	1.756
6	0.032	4.102
12	0.027	5.678
24	0.026	7.141
36	0.023	6.555
48	0.023	6.201
Average	0.027	5.239
Device		
2	0.159	1.206
4	0.163	1.976
8	0.112	2.684
16	0.061	2.565
30	0.047	2.925
60	0.037	3.543
120	0.032	4.379
180	0.029	4.224
240	0.029	4.683
Average	0.074	3.132

expensive. Results presented in this chapter required 19,926 experiments when we used enumeration, despite the fact that we tested only what we considered reasonable parameter values (see Table 14.1 in Sect. 14.3.2). Our heuristic-guided approach SAML that is based on simulated annealing and machine learning leads to comparatively good performance results, while performing only a relatively small set of experiments compared to enumeration.

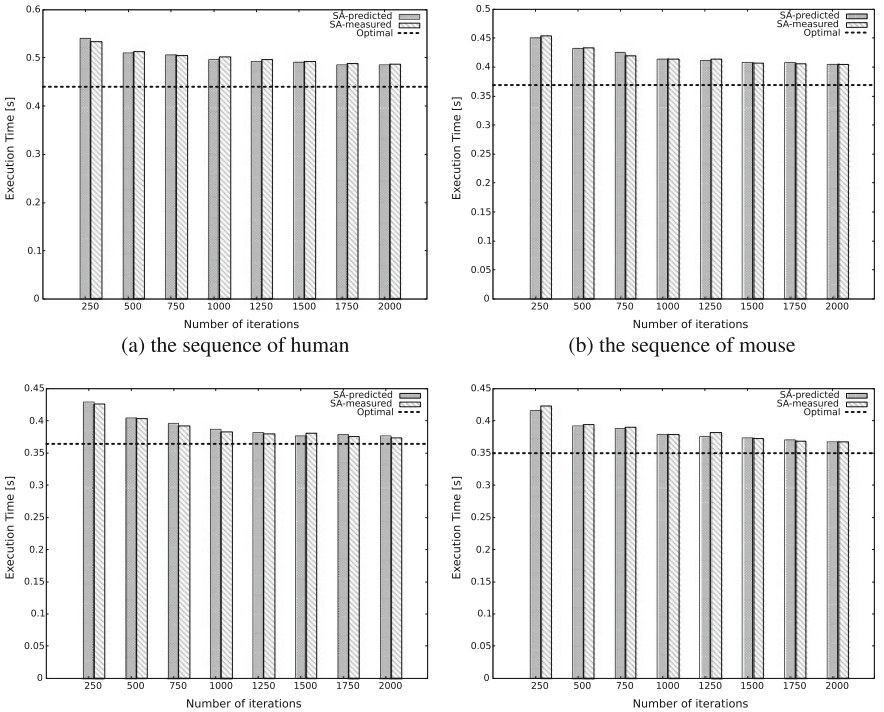
For performance comparison, we use the absolute difference and the percent difference (EM),

$$absolute\_difference = |T_{EM} - T_{SAML}| \tag{14.9}$$

$$percent\_difference = 100 \cdot absolute\_difference / T_{EM} \tag{14.10}$$

where T_{EM} indicates the best execution time determined using EM, and T_{SAML} indicates the execution time of our algorithm with a system configuration suggested by the SAML approach.

Figure 14.12 depicts the execution time of our DNA sequence analysis implementation using the system configuration suggested by the simulated annealing for



(a) the sequence of human

(b) the sequence of mouse

(c) the sequence of cat

(d) the sequence of dog

Fig. 14.12 Performance comparison between the best system configuration determined by the Enumeration and Measurements (EM) and the near-to-optimal one determined by the Simulated Annealing and Measurements (SAM) and Simulated Annealing and Machine Learning (SAML)

various types of DNA sequences. The horizontal line indicates the execution time of the system configuration determined with EM.

Simulated annealing suggests at each *iteration* parameter values for the system configuration. We may observe that after 1000 iterations, we determine a system configuration that results with a performance that is close to the performance of the system configuration determined with 19,926 experiments of EM. By running only about 5 % (that is $100 \times 1000/19{,}926$) of experiments we find a near-optimal system configuration. Please note that simulated annealing is a global optimization approach and to avoid ending at a local optima during the search sometimes may accept a worse system configuration that results with a higher execution time compared to previous iterations.

The percent difference is shown in Table 14.6, whereas the absolute difference is shown in Table 14.7. The percent difference for 250 iterations is 20.49 %. By increasing the number of iterations to 500, 750, and 1000 the percent difference decreases significantly, into 14.13, 12.57, and 9.88 % respectively. Further increase

Table 14.6 Percent difference (%). The performance of system configuration suggested by SAML after 250, 500, 750, 1000, 1250, 1500, 1750, and 2000 iterations is compared with the best one determined by EM

	System configuration							
	250	500	750	1000	1250	1500	1750	2000
Human sequence	22.905	15.887	15.051	12.749	11.799	11.707	10.251	10.28
Mouse sequence	22.126	17.231	15.343	12.172	11.578	10.551	10.611	9.775
Cat sequence	17.861	11.212	8.704	6.161	4.966	3.425	4.018	3.354
Dog sequence	19.067	12.191	11.191	8.444	7.554	6.939	6.182	5.023
Average difference	20.489	14.13	12.572	9.881	8.974	8.155	7.765	7.108

Table 14.7 Absolute difference (s). The performance of system configuration suggested by SAML after 250, 500, 750, 1000, 1250, 1500, 1750, and 2000 iterations is compared with the best one determined by EM

	System configuration							
	250	500	750	1000	1250	1500	1750	2000
Human sequence	0.101	0.069	0.066	0.056	0.052	0.051	0.045	0.045
Mouse sequence	0.082	0.063	0.056	0.045	0.043	0.039	0.039	0.036
Cat sequence	0.065	0.041	0.032	0.022	0.018	0.012	0.014	0.012
Dog sequence	0.066	0.043	0.039	0.029	0.026	0.024	0.022	0.017
Average difference	0.078	0.054	0.048	0.038	0.035	0.032	0.03	0.027

of the number of iterations (1250, 1500, 1750, and 2000) results with a modest decrease of the percent difference (8.97, 8.15, 7.76, 7.11).

With respect to the absolute difference, the determined system configuration using SAML with 250 iteration is 0.078 s slower than EM approach. Increasing the number of iterations into 500, 750, and 1000 decreases the difference between the execution time into 0.054, 0.048, and 0.038 s, respectively. Using 2000 iterations to determine the near-to-optimal solution using SAML is slower by 0.027 s compared to the solution determined with EM.

14.4.5 Performance Improvement

In this section, we present the performance improvement obtained when all available resources of the host and device are utilized for DNA sequence analysis.

The results in Table 14.8 demonstrate the performance improvement that is achieved when the system configuration determined by the SAML and EM is used for DNA sequence analysis compared to the case when all the available cores on the host are used. We achieve a maximal speedup of 1.74 after 1000 system con-

Table 14.8 Speedup achieved when host and device are used for the DNA sequence analysis compared with the host only. We consider system configurations determined by EM and SAML after 250, 500, 750, 1000, 1250, 1500, 1750, and 2000 iterations

	System configuration								
	250	500	750	1000	1250	1500	1750	2000	Enumeration
Human sequence	1.37	1.45	1.46	1.49	1.5	1.51	1.52	1.53	1.68
Mouse sequence	1.6	1.66	1.7	1.74	1.75	1.77	1.77	1.78	1.95
Cat sequence	1.5	1.58	1.62	1.66	1.68	1.7	1.7	1.7	1.76
Dog sequence	1.42	1.51	1.52	1.56	1.57	1.58	1.6	1.6	1.69

Table 14.9 Speedup achieved when host and device are used for the DNA sequence analysis compared with the device only. We consider system configurations determined by EM and SAML after 250, 500, 750, 1000, 1250, 1500, 1750, and 2000 iterations

	System configuration								
	250	500	750	1000	1250	1500	1750	2000	Enumeration
Human sequence	1.64	1.74	1.76	1.79	1.81	1.81	1.83	1.84	2.02
Mouse sequence	1.7	1.77	1.80	1.85	1.86	1.88	1.88	1.89	2.07
Cat sequence	1.96	2.08	2.13	2.18	2.21	2.24	2.23	2.24	2.31
Dog sequence	1.99	2.1	2.13	2.18	2.19	2.21	2.23	2.25	2.36

figurations have been tried with SAML, whereas the maximal speedup that can be achieved using EM is 1.95.

Table 14.9 shows the performance improvement that is achieved when the system configuration determined by the SAML and EM is used for DNA sequence analysis compared to the case when all the available cores on the device are used. The maximal speedup that can be achieved using EM is 2.36. We achieve a close to maximal speedup (2.21) using only 1000 iterations.

14.5 Related Work

In this section, we discuss related software-based approaches for DNA sequence analysis that target both multi-core and many-core architectures. Thereafter we discuss the intelligent methods for resource management that utilize the combined computation power of multi-core CPUs and many-core accelerators in heterogeneous systems.

14.5.1 Approaches Targeting Multi-core Architectures

[18] presented an implementation of the Aho-Corasick string matching algorithm using pthreads (posix threads), which is based on the pattern partitioning approach. A replication of the Herath's study with the intention to improve the software implementation of the Aho-Corasick algorithm was conducted by [3].

[13] achieved significant speedup by partitioning the input string among the threads in such a way that each thread processes only sequences starting with a specified prefix used to divide the radix tree among the threads. They achieved up to 6.9× speedup on a shared memory system with 8 cores.

A method for searching arbitrary regular expressions using speculation is proposed by [26]. The drawback of this approach is that if an REM performed by a thread does not converge on its sub-input, then the next thread has to start from a new state that breaks the serialization and limits the scalability.

An approach based on the Aho-Corasick string matching algorithm designed for the Cray XMT architecture is proposed by [48]. They split the input among the available threads, and overlap the input by the pattern length. Their approach is applicable for multiple patterns as long as they are of the same length, otherwise, the occurrences of the shortest patterns occurring on the crossing border may be counted by both threads.

14.5.2 Approaches Targeting Many-Core Architectures

An acceleration of exact string matching Knuth–Morris–Pratt algorithm on GPU is conducted by [6]. They achieve nearly a 29× speedup compared to the sequential version of the KMP algorithm. Similarly, [23] conducted an experiment on the Naive, KMP, Boyer–Moore–Horspool and Quick-Search string matching algorithms in the context of DNA sequencing using the CUDA toolkit.

[46] implemented the algorithm presented by [48] for GPU clusters. Their implementation is based on splitting the input into chunks, and then processing each chunk in a separate thread. In contrast to our approach, their algorithm for pattern matching relies on the features of the GPU architecture.

[24] implemented in CUDA the Wu-Manber algorithm, which is used for approximate matching of nucleotides in DNA sequences on GPU. In contrast our algorithm performs exact pattern matching.

[25] evaluated their Parallel Failure-less AC algorithm on GPU and showed improvement of 14.74×. This algorithm allocates a new thread to each character of the input to identify any pattern starting from that character, which means that it creates n number of threads, where n is the input length. In their experiments the length of input string is up to 256 MB.

14.5.3 Intelligent Methods for Resource Management

Utilizing the combined computation power of multi-core CPUs and many-core accelerators in heterogeneous systems is important to achieve high performance. Various approaches to distribute the workload across different devices in heterogeneous systems have been proposed.

[43] proposed an adaptive worksharing library to schedule computational load across devices. They extend the accelerated OpenMP by introducing a cross-device worksharing construct/directive, which allows the programmer to specify the association between the computation and data. Such extensions allows their library to automatically handle the workload distribution across devices. They evaluate the speed of each device statically, then use these indicators to split the workload across different devices.

Similarly [5] investigated the extension of OpenMP to allow workload distribution on future iterations based on the results of first static ones.

[35] proposed an approach that combines the pragma-based XcalableMP (XMP) [33] programming language with the runtime system by StarPU to utilize both GPU and CPU resources on each node for work distribution of the loop executions. They use the XMP for data distribution and synchronization purposes, whereas the StarPU is used for scheduling the tasks among host CPUs and accelerating devices.

While [5, 43] tend to offer solutions that require minimal changes to the original source code, task block models, such as StarPU [4] and OmpSs [14] require the user to determine workload distribution manually and may require significant structural changes to an original serial code. These models provide a powerful platform for scheduling on heterogeneous systems, which are based on splitting the workload into smaller tasks and queuing these tasks across the available resources.

Qilin [28] is a programming system that is based on a regression model to predict the execution time of kernels. Similarly to our approach, it uses off-line learning that is thereafter used in compile time to predict the execution time for different input size and system configuration.

[40] proposed their dynamic scheduling framework that divides tasks into smaller ones that later on are distributed across different processing elements in a task-farm way. They consider architectural trade-offs, computation, and communication patterns while making scheduling decisions. In comparison to our approach, we consider only system runtime configuration and the input size, which enables us a more general use of our approach with different applications and architecture.

[12] proposed a C++ framework for dynamic distribution of the work among the host CPUs and coprocessor devices. The workload is distributed using a priority queue technique, where one core of the host is responsible for the queue management.

[17] proposed a static partitioning approach to distribute OpenCL programs on heterogeneous systems. Their approach is based on static analysis to extract code features from OpenCL programs. These features are then used to determine the best partitioning across the different devices. Their approach relies on the architectural characteristics of a system.

However, in comparison to the aforementioned approaches, we use combinatorial optimization to determine the near-optimal system configuration.

Our work is closely related to the work presented by [2]. They propose a profiling-based approach for mapping kernel computations to heterogeneous platforms. Their approach extracts profiling information by running each application of every device (including host CPUs and accelerators), to collect information such as execution time and data transfer time. These information are then passed to solvers such as Greedy Algorithm to select the optimal mapping for a specific kernel. In comparison to Albayrak et al. we use machine learning to evaluate the performance of a selected application, and we use simulated annealing to minimize the overall execution cost, whereas they use an improved version of the Greedy Algorithm. Furthermore, they target applications that are designed as sequence of kernels, whereas we target OpenMP applications with divisible workload (data parallel).

The use of intelligent methods (such as, meta-heuristics) for efficient scheduling in the context of Grid computing environments has been addressed by [21, 22]. The use of intelligent software agents in the context of multi-core computing systems was proposed by [36] to address the programmer productivity and performance optimization. [20] aim at identifying which parameters of workflow activities affect more the overall result of the workflow, and propose to use the ant colony optimization heuristic to search for a subset of most significant parameters.

The of use modeling and simulation techniques to study properties of large-scale parallel and distributed computing systems has been studied in [1, 8, 15, 37, 38].

14.5.4 Our Approach

In contrast to related work, our approach targets heterogeneous systems that use as accelerator the Intel Xeon Phi coprocessor. Our approach enables using all available resources on the host CPUs and Intel Xeon Phi coprocessor for DNA sequence analysis. Furthermore, we use combinatorial optimization and machine learning to determine the optimal partitioning of the DNA sequence between the host and accelerating device such that the load is balanced between the host and device and the overall execution time is reduced. In addition, we use combinatorial optimization to determine the optimal number of threads and thread affinities for the host and device.

14.6 Summary

We have proposed a new approach for DNA sequence analysis designed for heterogeneous platforms that comprise a host with multi-core processors and one or more many-core accelerating devices. Our algorithm uses efficiently all the available resources of the host and device. Furthermore, we have proposed a combinatorial optimization approach that uses machine learning to determine the system

configuration (that is, the number of threads, thread affinity, and the DNA sequence fraction for the host and device) such that the overall execution time is minimized.

Our parallel algorithm scales well when the number of threads is increased. The speedup of our DNA sequence analysis algorithm compared to the sequential version is up to 51.98× when using the available resources of the host only, and 31.5× when using the resources of the device only.

We have observed that searching for the best system configuration using enumeration is time consuming, since it required many experiments. Using simulated annealing to suggest at each *iteration* parameter values for the system configuration after 1000 iterations, we determined a system configuration that results with a performance that is close to the performance of the system configuration determined with 19,926 experiments of enumeration. By running only about 5 % of experiments we were able to find a near-optimal system configuration. Furthermore, we have proposed a machine learning approach that is able to predict the execution time for a system configuration. We have observed in our experiments that the average percent error of 4.2 % of the performance prediction enables us to satisfactory suggest near-to-optimal system configurations. Using the near optimal system configuration determined by the simulated annealing and machine learning, we achieved a maximal speedup of 1.74× compared to the case when all the cores of the host are used, and up to 2.21× faster compared to the fastest execution time on the device.

Future work will study DNA sequence analysis on the heterogeneous computing platforms that use as accelerator the second generation of the Intel Xeon Phi coprocessor, codenamed *Knights Landing*.

References

1. Abraham, E., Bekas, C., Brandic, I., Genaim, S., Johnsen, E.B., Kondov, I., Pllana, S., Streit, A.: Preparing HPC applications for exascale: challenges and recommendations. In: 2015 International Conference on Network-Based Information Systems (NBiS), IEEE (2015)
2. Albayrak, O.E., Akturk, I., Ozturk, O.: Improving application behavior on heterogeneous manycore systems through kernel mapping. Parallel Comput. **39**(12), 867–878 (2013). doi:10.1016/j.parco.2013.08.011
3. Arudchutha, S., Nishanthy, T., Ragel, R.G.: String matching with multicore CPUs: performing better with the Aho-Corasick algorithm. arXiv preprint arXiv:14031305 (2014)
4. Augonnet, C., Thibault, S., Namyst, R., Wacrenier, P.A.: StarPU: a unified platform for task scheduling on heterogeneous multicore architectures. Concurrency Comput.: Pract. Experience **23**(2), 187–198 (2011)
5. Ayguadé, E., Blainey, B., Duran, A., Labarta, J., Martínez, F., Martorell, X., Silvera, R.: Is the schedule clause really necessary in OpenMP? In: OpenMP Shared Memory Parallel Programming, pp. 147–159. Springer (2003)
6. Bellekens, X., Andonovic, I., Atkinson, R., Renfrew, C., Kirkham, T.: Investigation of GPU-based pattern matching. In: The 14th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNet2013) (PGNet2013) (2013)

7. Benkner, S., Pllana, S., Traff, J., Tsigas, P., Dolinsky, U., Augonnet, C., Bachmayer, B., Kessler, C., Moloney, D., Osipov, V.: PEPPHER: efficient and productive usage of hybrid computing systems. Micro IEEE **31**(5), 28–41 (2011)
8. Brandic, I., Pllana, S., Benkner, S.: An approach for the high-level specification of QoS-aware grid workflows considering location affinity. Sci. Program. **14**(3–4), 231–250 (2006)
9. Chacón, A., Moure, J.C., Espinosa, A., Hernndez, P.: In-step FM-Index for faster pattern matching. In: Alexandrov V.N., Lees M., Krzhizhanovskaya V.V., Dongarra J., Sloot P.M.A. (eds.) ICCS, Elsevier, Procedia Computer Science, vol. 18, pp. 70–79 (2013)
10. Chrysos, G.: Intel Xeon Phi Coprocessor-the Architecture. Intel Whitepaper (2014)
11. Collins, F.S., Green, E.D., Guttmacher, A.E., Guyer, M.S.: A vision for the future of genomics research. Nature **422**(6934), 835–847 (2003)
12. Dokulil, J., Bajrovic, E., Benkner, S., Pllana, S., Sandrieser, M., Bachmayer, B.: High-level support for hybrid parallel execution of C++ applications targeting Intel Xeon Phi coprocessors. In: ICCS, Elsevier, Procedia Computer Science, vol. 18, pp. 2508–2511 (2013)
13. Drews, F., Lichtenberg, J., Welch, L.R.: Scalable parallel word search in multicore/multiprocessor systems. J. Supercomput. **51**(1), 58–75 (2010)
14. Duran, A., Ayguadé, E., Badia, R.M., Labarta, J., Martinell, L., Martorell, X., Planas, J.: Ompss: a proposal for programming heterogeneous multi-core architectures. Parallel Process. Lett. **21**(02), 173–193 (2011)
15. Fahringer, T., Pllana, S., Testori, J.: Teuta: tool support for performance modeling of distributed and parallel applications. Computational Science - ICCS 2004. Lecture Notes in Computer Science, vol. 3038, pp. 456–463. Springer, Berlin (2004)
16. Farkaš, T., Kubán, P., Lucká, M.: Effective parallel multicore-optimized k-mers counting algorithm. In: SOFSEM 2016: Theory and Practice of Computer Science: 42nd International Conference on Current Trends in Theory and Practice of Computer Science, Harrachov, Czech Republic, January 23–28, 2016, pp. 469–477. Springer, Berlin (2016)
17. Grewe, D., OBoyle, M.F.: A static task partitioning approach for heterogeneous systems using OpenCL. In: Compiler Construction, pp. 286–305. Springer (2011)
18. Herath, D., Lakmali, C., Ragel, R.: Accelerating string matching for bio-computing applications on multi-core CPUs. In: 2012 7th IEEE International Conference on Industrial and Information Systems (ICIIS), pp. 1–6 (2012)
19. Kessler, C.W., Dastgeer, U., Thibault, S., Namyst, R., Richards, A., Dolinsky, U., Benkner, S., Trff, J.L., Pllana, S.: Programmability and performance portability aspects of heterogeneous multi-/manycore systems. IEEE, pp. 1403–1408 (2012)
20. Khan, F.A., Han, Y., Pllana, S., Brezany, P.: An ant-colony-optimization based approach for determination of parameter significance of scientific workflows. In: 24th IEEE International Conference on Advanced Information Networking and Applications. Perth, WA, 2010, pp. 1241–1248 (2010). doi:10.1109/AINA.2010.24
21. Kołodziej, J., Khan, S.: Data scheduling in data grids and data centers: a short taxonomy of problems and intelligent resolution techniques. In: Nguyen, N.T., Kolodziej, J., Burczyski, T., Biba, M. (eds.) Transactions on Computational Collective Intelligence X. Lecture Notes in Computer Science, vol. 7776, pp. 103–119. Springer, Berlin (2013)
22. Kołodziej, J., Khan, S.U., Wang, L., Zomaya, A.Y.: Energy efficient genetic-based schedulers in computational grids. Concurrency Comput.: Pract. Experience **27**(4), 809–829 (2015)
23. Kouzinopoulos, C., Margaritis, K.: String matching on a multicore GPU using CUDA. In: 13th Panhellenic Conference on Informatics, 2009. PCI '09, pp. 14–18 (2009)
24. Li, H., Ni, B., Wong, M.H., Leung, K.S.: A fast CUDA implementation of agrep algorithm for approximate nucleotide sequence matching. In: SASP, pp. 74–77. IEEE Computer Society (2011)
25. Lin, C.H., Liu, C.H., Chien, L.S., Chang, S.C.: Accelerating pattern matching using a novel parallel algorithm on GPUs. IEEE Trans. Comput. **62**(10), 1906–1916 (2013)
26. Luchaup, D., Smith, R., Estan, C., Jha, S.: Speculative parallel pattern matching. IEEE Trans. Inf. Forensics Secur. **6**(2), 438–451 (2011)
27. Luftig, M.A., Richey, S.: DNA and forensic science. New Eng. L Rev. **35**, 609 (2000)

28. Luk, C.K., Hong, S., Kim, H.: Qilin: exploiting parallelism on heterogeneous multiprocessors with adaptive mapping. In: 42nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-42, 2009, pp. 45–55. IEEE (2009)
29. Mellmann, A., Harmsen, D., Cummings, C.A., Zentz, E.B., Leopold, S.R., Rico, A., Prior, K., Szczepanowski, R., Ji, Y., Zhang, W., McLaughlin, S.F., Henkhaus, J.K., Leopold, B., Bielaszewska, M., Prager, R., Brzoska, P.M., Moore, R.L., Guenther, S., Rothberg, J.M., Karch, H.: Prospective genomic characterization of the german enterohemorrhagic escherichia coli O104:H4 outbreak by rapid next generation sequencing technology. PLoS ONE **6**(7):e22, 751 (2011)
30. Memeti, S., Pllana, S.: PaREM: a novel approach for parallel regular expression matching. In: 17th International Conference on Computational Science and Engineering (CSE-2014), pp. 690–697 (2014). doi:10.1109/CSE.2014.146
31. Memeti, S., Pllana, S.: Accelerating DNA sequence analysis using Intel Xeon Phi. In: PBio at the 2015 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA). IEEE (2015a)
32. Memeti, S., Pllana, S.: Analyzing large-scale DNA sequences on multi-core architectures. In: 18th IEEE International Conference on Computational Science and Engineering (CSE-2015). IEEE (2015b)
33. Nakao, M., Lee, J., Boku, T., Sato, M.: XcalableMP implementation and performance of NAS parallel benchmarks. In: Proceedings of the Fourth Conference on Partitioned Global Address Space Programming Model, p. 11. ACM (2010)
34. NCBI: National center for biotechnology information U.S. National Library of Medicine. http://www.ncbi.nlm.nih.gov/genbank (2015). Accessed Dec 2015
35. Odajima, T., Boku, T., Hanawa, T., Lee, J., Sato, M.: GPU/CPU work sharing with parallel language XcalableMP-dev for parallelized accelerated computing. In: 2012 41st International Conference on Parallel Processing Workshops (ICPPW), pp. 97–106. IEEE (2012)
36. Pllana, S., Benkner, S., Mehofer, E., Natvig, L., Xhafa, F.: Towards an intelligent environment for programming multi-core computing systems. In: Euro-Par Workshops, Lecture Notes in Computer Science, vol. 5415, pp. 141–151. Springer (2008a)
37. Pllana, S., Benkner, S., Xhafa, F., Barolli, L.: Hybrid performance modeling and prediction of large-scale computing systems. In: CISIS 2008. International Conference on Complex, Intelligent and Software Intensive Systems, 2008, pp. 132–138 (2008b)
38. Pllana, S., Brandic, I., Benkner, S.: A survey of the state of the art in performance modeling and prediction of parallel and distributed computing systems. Int. J. Comput. Intell. Res. (IJCIR) **4**(1), 17–26 (2008c)
39. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes, 3rd edn. In: The Art of Scientific Computing, 3rd edn. Cambridge University Press (2007)
40. Ravi, V.T., Agrawal, G.: A dynamic scheduling framework for emerging heterogeneous systems. In: 2011 18th International Conference on High Performance Computing (HiPC), pp. 1–10. IEEE (2011)
41. Rohrer, B.: How to choose algorithms for Microsoft Azure Machine Learning. https://azure.microsoft.com/en-us/documentation/articles/machine-learning-algorithm-choice/ (2015). Accessed Oct 2015
42. Sandrieser, M., Benkner, S., Pllana, S.: Using explicit platform descriptions to support programming of heterogeneous many-core systems. Parallel Comput. **38**(1–2), 52–56 (2012)
43. Scogland, T.R., Feng, Wc., Rountree, B., de Supinski, B.R.: CoreTSAR: adaptive worksharing for heterogeneous systems. In: Supercomputing, pp. 172–186. Springer (2014)
44. Stephens, Z.D., Lee, S.Y., Faghri, F., Campbell, R.H., Zhai, C., Efron, M.J., Iyer, R., Schatz, M.C., Sinha, S., Robinson, G.E.: Big data: astronomical or genomical? PLoS Biol **13**(7):e1002, 195 (2015)
45. Tian, X., Saito, H., Preis, S., Garcia, E.N., Kozhukhov, S., Masten, M., Cherkasov, A.G., Panchenko, N.: Practical SIMD vectorization techniques for Intel Xeon Phi Coprocessors. In: IPDPS Workshops, pp. 1149–1158. IEEE (2013)

46. Tumeo, A., Villa, O.: Accelerating DNA analysis applications on GPU clusters. In: 2010 IEEE 8th Symposium on Application Specific Processors (SASP), pp. 71–76 (2010)
47. Viebke, A., Pllana, S.: The potential of the Intel (R) Xeon Phi for supervised deep learning. In: 2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC). pp. 758–765. IEEE (2015)
48. Villa, O., Chavarra-Miranda, D.G., Maschhoff, K.J.: Input-independent, scalable and fast string matching on the Cray XMT. In: IPDPS, IEEE, pp. 1–12 (2009)

Chapter 15
Feature Dimensionality Reduction for Mammographic Report Classification

Luca Agnello, Albert Comelli and Salvatore Vitabile

15.1 Introduction

Nowadays, the exponential increase in the amount and variety of data generated from multiple heterogeneous sources, such as networks, sensors, mobile devices, archives, Internet of Things, software records, health informations, etc., has led to the scientific community to study how to manage, analyze, and extract information from large amounts of data.

The Big Data, defined as a collection of very large and complex datasets, inhibits analysis, manual interpretation, and use of simple data management applications due to the large amount of informations. They became a common element in many fields of application in the real world, such as commercial, computer engineering, medicine, and e-health.

Especially in the healthcare domain, the development of techniques that permit the management, analysis, mining, and pattern recognition of health data for clinical decision support systems, genomics, processing of medical images, medical information retrieval, health data mining has become worldwide important.

The healthcare sector has very large datasets of different nature that play an essential role in health information systems (HIS) and clinical decision support systems (CDSS). Early data insertion, retrieval, and analysis of information from various health records, such as public health data, drug-to-drug, drug-to-disease, disease-to-disease, and many others are the challenge of healthcare practitioners that must

L. Agnello · A. Comelli · S. Vitabile (✉)
Department of Biopathology and Medical Biotechnologies,
University of Palermo, Palermo, Italy
e-mail: salvatore.vitabile@unipa.it

L. Agnello
e-mail: luca.agnello@unipa.it

A. Comelli
e-mail: albert.comelli@unipa.it

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_15

understand and process the RAW health datasets and make an exact final decision. All this restricts the benefits of large datasets and HIS/CDSS frameworks for medical decision-making processes.

Thousands of heterogeneous data (images, text, video, sensor data, etc.) are steadily generated and must be efficiently handled (stored, distributed, and indexed) in order to not compromise the quality of service of final users in terms of data availability, delay in data search, delay in data analysis, etc. Many of the existing ICT systems that store, process, distribute, and index hundreds of heterogeneous data are not sufficiently adequate or still need to be developed.

Data mining and information retrieval solutions acting on traditional clinical databases [1–5], medical images [6], natural language [7], and artificial intelligence-based Decision Support System (DDS) [8] are applied to healthcare domain, but they are not able to manipulate heterogeneous health data distributed in hospital information systems.

Such automation requires the use of algorithms capable of treating different types of reports, to evaluate the weight according to what they contain, imitating the cognitive processes of a human being in the analysis of reports, with the potential of being able to carry out very large quantity data. In other words, the solution of the problem requires the creation of a form of "artificial intelligence" in the broad sense, which reproduces the typically human analysis by a computer; therefore, it becomes essential to build and implement quantitative methods to recreate and produce strictly qualitative results. In this sense, the reports must be "deconstructed", reduced to a sequence of meaningless symbols from which to deduce properties subsequently interpretable. In this article, a methodology that improves mammographic reports memorization, search speed, analysis, and retrieval is proposed. The reports are converted in widely used Term Frequency-Inverse Document Frequency (TF-IDF) representation, that has the disadvantage creating large and sparse matrices with a low percentage of useful information. Data simplification and dimensionality reduction issues are then addressed. Decomposition methods such as LSI, PCA, and SVD have been applied to the TF-IDF matrix, obtaining results comparable to the results achieved using the raw unprocessed matrix, where the processed matrix contains less than 13 % of the raw TF-IDF data using PCI-LSI technique and less than 6 % of the raw TF-IDF data using SVD technique, obtaining computational complexity and processing time reduction.

In the second section, the related works and theoretical remarks for this work are presented. In the third section, the proposed methodology is described. In the fourth section, the experimental results are reported, while in fifth section the final conclusions are reported.

15.2 Related Works and Theoretical Remarks

Most of the techniques that allow for dimensionality reduction are based on statistical methods. One of the most common methodology, called Latent Semantic Analysis (LSA) is useful when we are dealing with a corpus formed by numerous

texts, and allows the detection of the so-called "latent semantics": in other words, a set of variables that can be interpreted as abstract concepts and it allows greater efficiency in the analysis of the documents in question, through the use of algebraic technique called Single Value Decomposition (SVD) as key tool for the detection of latent semantic, in order to apply the link-oriented research methods to document collections without its own link structure (such as a document collection inside a corporate intranet). Typically, the easiest way to extract information from a large set of documents is using text-matching techniques, where a literal matching between query data terms and those present in the documents themselves is performed. As well known, however, this method has many limitations, not even negligible faced with the evidence of its speed and ease of use. Assuming that there are different ways to express the same concept (synonymies) and that the same word can have different meanings, the simple text matching may return documents that have no relevance. A better approach might be to search for information not on the basis of the signifier, but on that of meaning.

The authors of [9] have used tools such as ontologies and semantic processing, calculating the distances between the documents and implementing a retrieval system using an ad hoc ontology with the addition of "is-a" and "equal to" relationships. The method, although showing the best results, is slow due to the computationally expensive algorithm for the processing of documents through the ontology.

The Latent Semantic Indexing (LSI) [10] tries to do some research looking for a latent semantic structure in the various documents that is partially hidden by the intrinsic variability of the words. LSI is an indexing technique based on SVD concepts [11], which allows estimation of the semantic structure of the analyzed documents. This structure is placed at a higher level compared to that occupied by the meaning of words used in the document.

The authors of [12] describe a new method that generates features in LSI method by meaning of SVD that reduces the dimensionality and removes the noise in the raw data matrix. The LSI technique can lead to a great reduction of computation complexity and CPU time because of the reduction of matrix dimensionality.

The work in [13] proposes to integrate the information retrieval method and document clustering as concept space approach. This method used LSI which used SVD or Principal Component Analysis (PCA) to reduce the matrix dimension by finding the pattern in document collection with refers to concurrent of the terms. Each method is implemented to weight term-document in vector space model (VSM) for document clustering using fuzzy c-means algorithm. This research also uses the cosine similarity measurement as replacement of Euclidean distance. The performance of the proposed method is better than document clustering without LSI which uses Euclidean distance, with a significant improvement when applied in huge data volumes. In [14] authors developed a data dimensionality reduction approach using Sparsified Singular Value Decomposition (SSVD) technique, in order to identify and remove trivial features before applying any advanced feature selection algorithm. A set of experiments were conducted and the results show that applying feature selection techniques on the data where the nonessential features are removed by data

dimensionality reduction approach generally results in better performances with significantly reduced computing time.

In the research of [15] the authors use large training sets that make computation complex and expensive. Latent Semantic Indexing Subspace Signature Model (LSISSM) is applied to labeling for active learning of unstructured text. Based on SVD, the LSISSM methodology represents terms and documents as semantic signatures by the distribution of their local statistical contribution across the top-ranking LSI latent dimensions after dimension reduction. Tests demonstrate that the sample subsets with the optimized term subsets substantially improve the learning accuracy.

In what follows, a theoretical disquisition on main dimensionality reduction techniques is reported.

15.2.1 Latent Semantic Analysis (LSA)

The Latent Semantic Analysis [16] comes from the need to improve the process by which search words are associated with the contents of searched papers in question. The main problem is that usually users search documents relying on concepts, while the words entered for search do not provide an evidence of the document text in which they are contained. Usually, there are several ways to express a given concept, and then the search terms can not be used at all in the searched document. In addition, many words have multiple meanings so the search terms can appear in a document that has nothing to do with the searched document.

The LSA analysis operates under the premise that there is some latent semantic structure that is partially hidden by the randomness of the selected words. In order to highlight this structure, algebraic and statistics techniques are used to eliminate noise. The result is a continuous parametric description of terms and documents based on the underlying structure. In practice, the latent structure is transmitted through correlation diagrams, arising from the way in which the words appear in the documents; this leads to a primary language model described by a limited set of words. In addition, consistent with an approach of quantitative nature, semantics simply means that the words of a document can be regarded as an indicator of a document or of its argument. Despite these simplifications, the description that is obtained is undoubtedly advantageous, improving the recognition of the most relevant documents on the basis of terms that appear in the search.

Suppose M is an M-unit inventory (for example the words of a vocabulary) and N a collection of N combinations of those units (a corpus of documents or literary texts). The purpose of the LSA is to define a mapping between M and N discrete sets and a continuous vector space L, in which each word r_i of M is represented by a vector u_i in L, and each text c_j in N is represented by a vector v_j in L.

Firstly, a matrix W with co-occurrences between words and the texts of the corpus is constructed. The $w_{i,j}$ element is obtained by the standardization with respect to the text length and the entropy of words:

$$w_{i,j} = (1 - \varepsilon_i) \frac{k_{i,j}}{\lambda_j} \tag{15.1}$$

where $k_i, _j$ is the number of times that r_i appeared in c_j, λj is the total number of words that appear in c_j (length normalization) and ε_i is the normalized entropy r_i within the entire corpus (entropy normalization). By definition of entropy, if $\tau_i = \Sigma_j$ $k_i, _j$ is the total number of occurrences of the *i*th word in the corpus, it results:

$$\varepsilon_i = -\frac{1}{\log N} \sum_{j=1}^{N} \frac{k_{i,j}}{\tau_i} \log \frac{k_{i,j}}{\tau_i} \tag{15.2}$$

The overall weight given by the quantity 1-ε_i indicates the fact that two units that appear the same number of times c_j does not necessarily make the same amount of information. A value ε_i close to 1 indicates that the word is evenly distributed in the body, and then it carries less information than another with ε_i closer to 0. It can be summarized by saying that the overall weight 1-ε_i is a measure of the "power index" of the word r_i.

15.2.2 Principal Component Analysis (PCA)

The analysis of the Principal Component Analysis [17] is directly derived from the factor analysis theory, which consists of a set of statistical techniques that allow obtaining a reduction of the complexity of the number of factors that explain a phenomenon. It is therefore proposed to determine a number of "latent variables" (factors not directly measurable in reality) more restricted and summarizing with respect to the number of starting variables.

Let us consider, for example, the collection of the votes of a class of students in a school. The votes regard student achievement in various subjects (math, science, geography, history, etc.). Students can therefore be considered as "variables" that assume different values (votes) on the different matters. To simplify the problem, it must be assumed that the learning ability of students can stand in two factors: the humanities matters and science matters. Thanks to factorial analysis it is possible to measure these two skills through the construction of two latent synthetic variables (as a linear combination) of the original variables, each designed on the basis of the importance in discriminating people on the basis of their scientific and humanistic skills. The principal component analysis is therefore a factorial method for the synthesis of p quantitative variables, related to each other, through the identification of $h < p$ latent variables (not observed) called "main components" that have two properties:

- are mutually uncorrelated (orthogonal) and linearly related to the original variables;
- are determined in ascending order of percentage of variability.

In order to simplify, suppose to represent on the Cartesian plane the points unit whose coordinates are the standardized values of the two variables. With the PCA identifies the factorial axis in the direction of maximum variability of the cloud of points units, in order to deform as little as possible the mutual distance between the points projected on that axis: namely, it minimizes the sum of distances points from the axis AB, which is equivalent to the Pythagoras's theorem to maximize the sum of the projections of the points on the axis OB, i.e., the distance of the points projected on the axis from the origin.

Suppose that a data matrix X($n \times p$):

$$X = \begin{pmatrix} x_{11} & \dots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix} \tag{15.3}$$

The row vectors X are points units in the R^p space generated by variables, instead column vectors are variable points in R^n space generated by the units. We wish to project the n vectors (called x_i, with $1 \leq \mathrm{i} \leq \mathrm{n}$) in the R^p space on a straight line r_1, which will be the first principal component identified with u_1: call θ the angle formed by x_i and u_i, and OH_i the ih projection x_i over r_1. We obtain:

$$OH_i = x_i^{\mathrm{T}} \cdot u_1 \tag{15.4}$$

This reasoning can be made for each of the n points, thus defining a matrix of projections given by:

$$X \cdot u_1 \tag{15.5}$$

It remains to understand how to choose the straight line r_1. To do this the least squares method is used, and then that line that minimizes the squares of the distances of the points from the line is taken; this is equivalent to the Pythagorean theorem, to maximize the amount:

$$\sum_{i=1}^{n} OH_i^2 = (Xu_1)^{\mathrm{T}} \cdot Xu_1 = u_1^{\mathrm{T}} X^{\mathrm{T}} X u_1 \tag{15.6}$$

Practically we maximize $u_1^T\ X^T X u_1$ with the constraint that u_1 has unit norm: we solve everything with Lagrange multipliers, defining the function:

$$F(u_1) = u_1^{\mathrm{T}} X^{\mathrm{T}} X u_1 - \lambda_1 (u_1^{\mathrm{T}} u_1 - 1) \tag{15.7}$$

Differentiating with respect to u_1 we obtain:

$$2X^{\mathrm{T}} X u_1 - 2\lambda_1 u_1 = 0 \tag{15.8}$$

that is

$$X^{\mathrm{T}} X u_1 = \lambda_1 u_1 \tag{15.9}$$

This is a simple eigenvalue problem, and maximizing we will have at the same time that u_1 is the eigenvector relative to the largest eigenvalue of the matrix $X^T X$. Defined u_1 we defined the first principal component:

$$r_1 = X u_1 = v_1 u_{1_1} + \cdots + v_p u_{1_p} \tag{15.10}$$

where v_ii are the initial variables (to be not confused with the units x_i), that is the column vectors of the matrix X, and $u_1 i$ (with $1 \leq \mathrm{i} \leq \mathrm{n}$) are the elements of the ith unit vector u_1. In addition, the square norm of r_1 is:

$$\|r_1\| = u_1^{\mathrm{T}} X^{\mathrm{T}} X u_1 = \lambda_1 \tag{15.11}$$

that is the norm of the first main direction coincides with the eigenvalue corresponding to the direction of the unit vector u_1 (which is the corresponding eigenvector). The following result is obtained: the first principal component r_1 is a linear combination of the p columns of the matrix X with coefficients equal to the components of u_1 eigenvector, associated with the maximum eigenvalue of the matrix $X^T X$. The subsequent principal components are obtained in the same way, with the constraint that they are orthogonal to the previous, or that the random variable corresponding to the main obtained direction is uncorrelated from the previous.

Remarks: The matrix S = $X^T X$ is a symmetric matrix:

- tr(S) = $\Sigma \lambda$i
- det(S) = $\Pi \lambda$i
- the eigenvalues of a symmetric matrix are real;
- the eigenvectors of a symmetric matrix are orthogonal two by two;
- the rank of a symmetric matrix is equal to its number of nonzero eigenvalues.

15.2.2.1 Interpretation of the Main Components

As mentioned above, the PCA is often used to try to study "latent variables". In this sense, the generated "artificial variables" are measurements of "hidden variables", not directly observable. Other times, the PCA is used as a method to "sum" the available data. However, the meaning to be attributed to the main components it is not unique but it is based on an interpretation that can be different on the basis of external environmental elements; then a central role is the experience and the sensitivity of those who have the task of giving them meaning.

Said that, some observations can be made. First, the *j*th main component r_j is defined as a linear combination of the variables $v_1, v_2, \ldots, v_p$ with coefficients $u_{i1}, u_{i2}, \ldots, u_{ip}$. That is:

$$r_{ij} = u_{j1}v_{i1}, u_{j2}v_{i2}, \ldots, u_{jp}v_{ip}, \tag{15.12}$$

Then:

$$y_j = Xu_j \tag{15.13}$$

Therefore, the generic coefficient u_{jh} represents the weight that the u_h variable has in the determination of the main component r_j (h = 1, …, p); the greater u_{jh} in absolute value, the greater the weight that v_{ih} values (i = 1, …, n) have in determining r_{ij}. This means that the main component r_j will be further characterized by v_h variables which correspond to the greatest u_{jh} coefficients in absolute value. Thus the u_{jh}. coefficients give a meaning to the main component r_j.

15.2.3 Singular Value Decomposition (SVD)

Singular Value Decomposition [18, 19] is generally used to rank estimation of a matrix and analysis of canonical correlation. For a rectangular array we can not define the eigenvalues. A generalization of the concept of eigenvalue can be obtained with the singular values.

Given a A_{mxn} matrix, the matrix A^TA has no negative n eigenvalues that, ranked in a decreasing manner $\lambda_1 \geq \lambda_2 \geq K \geq \lambda_n \geq 0$, can be expressed in the form:

$$\lambda_i = \sigma_i^2 \qquad \sigma_i \geq 0. \tag{15.14}$$

Scalar values $\sigma_1 \geq \sigma_2 \geq K \geq \sigma_n \geq 0$ are called singular values of the matrix A. The singular value decomposition (SVD) of the matrix A is given by:

$$A = U' \cdot \Sigma' \cdot V'^T \tag{15.15}$$

where U is an orthogonal matrix (m x m):

$$U' = [u_1 \quad u_2 \quad \text{K} \quad u_m] \tag{15.16}$$

V' is an orthogonal matrix (n x n):

$$V' = [v_1 \quad v_2 \quad \text{K} \quad v_n] \tag{15.17}$$

and Σ' is a matrix (m x n):

$$\Sigma' = \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \quad \Sigma = diag(\sigma_1, \sigma_2, \mathrm{K}, \sigma_n) \tag{15.18}$$

where $\sigma_1 \geq \sigma_2 \geq K \geq \sigma_n > 0$. The number of singular value different than zero is equal to the rank r of the matrix A. The columns of U' are the eigenvalues of the matrix $\mathrm{A}A^T$ and the columns of V' are eigenvalues of matrix $A^T\mathrm{A}$. Taking into account the structure of Σ', we obtain:

$$A = U_r \cdot \Sigma \cdot V_r^T \tag{15.19}$$

where U_r and V_r are the matrices obtained respectively from the first r columns of U_r and V_r.

The next step in the construction of the matrix W (M x N) is its decomposition by the method of SVD, a very similar technique to finding the eigenvectors and eigenvalues for the square matrices, and the analysis of resulting factors. The decomposition (of order R) is given by:

$$W \approx \hat{W} = USV^{\mathrm{T}} \tag{15.20}$$

where U is the left singular matrix (M x R) with ui as column vectors ($1 \leq \mathrm{i} \leq \mathrm{M}$), S is the diagonal matrix (R x R) of the singular values (the analog of the eigenvalues of the square matrices) $s_1 \geq s_2 \geq \ldots \geq s_r$ n > 0, V is the right singular matrix (N x R) with vj as row vectors ($1 \leq \mathrm{j} \leq$N). Finally, R < min (M, N) is the order of the decomposition. Both right and left matrices U and V are orthonormal columns, that is $U^TU = V^TV = I_R$. This is why the column vectors of U and V define an orthonormal basis for the vector space of dimension R (what we defined as L or the LSA space) generated by u_i and v_j carriers. The matrix W is defined as the rank R of the matrix which best approximates W in norm, for each norm invariant for unitary transformations (such that $||A|| = ||UAV||$ for each matrix A with U and V unitary). This means, for each matrix A of rank R:

$$\min_{\{A:rank(A)=R\}} \|W - A\| = \|W - \hat{W}\| = s_{R+1} \tag{15.21}$$

where $|| * ||$ it is the L2 norm, and s_{R+1} is the smallest singular value decomposition remained in the order (R + 1) of W.

15.2.4 Stochastic Singular Value Decomposition (SSVD)

The Stochastic Singular Value Decomposition method [20] produces reduced rank Singular Value Decomposition output in its mathematical definition:

$$\mathbf{A} \approx \mathbf{U}\Sigma\mathbf{V}^{\top} \tag{15.22}$$

i.e., it creates outputs for matrices U, V and Σ, each of which may be requested individually. The desired rank of decomposition, henceforth denoted as kεN1, is a parameter of the algorithm. The singular values inside diagonal matrix Σ satisfy $\sigma_{i+1} \leq \Sigma_i \forall_i \varepsilon [1, \mathrm{k}-1]$, i.e., sorted from biggest to smallest. Cases of rank deficiency rank (A) < k are handled by producing 0 s in singular value positions once deficiency takes place.

Single space for comparing row items and column items: There is an option to present decomposition output in a form of:

$$\mathbf{A} \approx \mathbf{U}_{\sigma}\mathbf{V}_{\sigma}^{\top} \tag{15.23}$$

where one can request $U_{\sigma} = \mathrm{U}\Sigma^{0.5}$ instead of U (but not both), $V_{\sigma} = \mathrm{V}\Sigma^{0.5}$ instead of V (but not both). Here, notation $\Sigma^{0.5}$ implies diagonal matrix containing square roots of the singular values:

$$\Sigma^{0.5} = \begin{pmatrix} \sqrt{\sigma_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sqrt{\sigma_k} \end{pmatrix} \tag{15.24}$$

Original singular values Σ are still produced and saved regardless.

This option is a sign to a common need of comparing actors represented by both input rows and input columns in a common space. For Example, if LSI is performed such that rows are documents and columns are terms, then it is possible to compare documents and terms (either existing or fold in new ones) in one common space and perform similarity measurement between a document and a term, rather than computing just a term-2-term or a document-2-document similarity.

Common applications for SVD include Latent Semantic Analysis (LSA), Principal Component Analysis (PCA), dimensionality reduction, and others.

15.2.5 Relationship Between SVD and PCA

Intuitively, the Principal Component Analysis requires to calculate eigenvectors and eigenvalues of the covariance matrix X^T where X denote the initial data matrix. Since the covariance matrix is symmetric by construction, it is diagonalizable, and the eigenvectors may be normalized so as to be orthonormal:

$$XX^{\mathrm{T}} = WDW^{\mathrm{T}} \tag{15.25}$$

On the other hand, by applying the Singular Value Decomposition of the data matrix X we can write:

$$X = USV^{\mathrm{T}} \tag{15.26}$$

and building the covariance matrix using the previous decomposition we have:

$$XX^{\mathrm{T}} = (USV^{\mathrm{T}})(USV^{\mathrm{T}})^{\mathrm{T}} \tag{15.27}$$

that by using the fact that V is orthonormal, gives:

$$XX^{\mathrm{T}} = US^2U^{\mathrm{T}} \tag{15.28}$$

The correspondence between the two methods is evident: the square roots of the eigenvalues of XX^T are the singular values of X, with all that implies. In fact, the use of SVD to perform the PCA is numerically more convenient, since is not required to calculate the covariance matrix XX^T at the beginning of the procedure, a problem that cannot be well place in some cases.

15.2.6 LSA Space

The decomposition of Eq. 15.20 can be interpreted as a representation of each word and each concept as a linear combination of abstract (hidden) concepts, which generate the linear space of W. Then, the resulting mapping corresponds to an efficient representation of empirical data. The idea that is at the basis of the procedure is that W contains in itself the most relevant structural associations of W and ignores the higher order effects (i.e., the noise). The closeness between the vectors in L is then determined from the general scheme of the compositions of the language in N. The mapping (M, N) $->$ L finally allows applying known algorithmic techniques, in the continuous vector space L. To do this, a suitable metric of L that is also adapted to the SVD formalism must first defined. We observe that the measure by which r_i and r_j units have a similar scheme inside the whole corpus can be deduced from the (i, j) cell of the WW^T matrix; in the same way, the measure to which two texts c_i and c_j contain similar patterns of words in them can be observed at the position (i, j) of the W^TW matrix; finally, the extent to which the word r_i is globally linked to c_i text relative to the entire corpus can be observed from the (i, j) place of the matrix W itself. All this leads to the following analysis concerning words and texts.

15.2.6.1 Comparison of Two Words

Because $WW^T = US^2U^T$ and S is diagonal, the (i, j) cell of WW^T can be obtained considering the inner product between the *i*th and *j*th row of the matrix US; let define $u_i = u_i$S and $u_j = u_j$S. To measure the proximity between two words is natural to consider the metric defined by the cosine of the angle between two vectors:

$$K(r_i, r_j) = \cos(u_i S, u_j S) = \frac{u_i S u_j^{\mathrm{T}}}{\|u_i S\| \|u_j S\|} \tag{15.29}$$

for $1 \leq \mathrm{i}, \mathrm{j} \leq \mathrm{M}$. $\mathrm{K}(r_i, r_j) = 1$ if two words always appear in the same type of texts, while $\mathrm{K}(r_i, r_j) \leq 1$ if they are used in different contexts. Despite does not constitute a well-defined distance, it is possible to define a new one. For example, in the interval $[\pi, 2\pi]$, $\mathrm{D}(r_i, r_j) = \cos -1\mathrm{K}(r_i, r_j)$ satisfies the properties of a distance on the space L.

15.2.6.2 Comparison of Two Texts

For the same reasons just explained consider the function:

$$K(c_i, c_j) = \cos(v_i S, v_j S) = \frac{v_i S v_j^{\mathrm{T}}}{\|v_i S\| \|v_j S\|} \tag{15.30}$$

for $1 \leq \mathrm{i}, \mathrm{j} \leq \mathrm{N}$; in the same way, $\mathrm{K(i, j)} = 1$ if two texts contain the same words while $\mathrm{K(i, j)} \leq 1$ in the decreasing of the words in common.

Comparison of words and texts: Finally, being W = USVT

$$K(r_i, c_j) = \cos(u_i S^{1/2}, v_j S^{1/2}) = \frac{u_i S v_j^{\mathrm{T}}}{\|u_i S^{1/2}\| \|v_j S^{1/2}\|} \tag{15.31}$$

for $1 \leq \mathrm{i} \leq \mathrm{M}$ and $1 \leq \mathrm{j} \leq \mathrm{N}$. In this case, $\mathrm{K}(r_i, c_j) < 1$ as the correlation between the word r_i and c_j text within the body decreases.

15.2.7 Cholesky Factorization (Decomposition)

For the Cholesky theorem [21], a symmetric matrix is positive definite if and only if there is one and only one R lower triangular matrix with positive diagonal entries such that $\mathrm{A} = R^T R$.

Let the matrix A symmetric positive definite for the Cholesky theorem. The factorization for a matrix of this type is simplified for two reasons: it is not necessary any pivoting and it is only possible to operate on the lower triangle of the matrix, being the upper one the transposed of the lower one, being A a symmetric matrix. Therefore, the factorization of the matrix consist in the product of two matrices R and R^T, the first lower triangular, the second transpose of the first. The determination of the elements of the $\mathrm{R} = \{R_{ij}\}$ matrix occurs in the following way:

$$\mathrm{r_{ij}} = \frac{1}{\mathrm{r_{ii}}}(\mathrm{a_{ij}} - \sum_{\mathrm{k}=1}^{\mathrm{i}-1} \mathrm{r_{ki} r_{kj}}), \quad \mathrm{i} < \mathrm{j} \tag{15.32}$$

$$r_{jj} = (a_{jj} - \sum_{k=1}^{j-1} r_{kj}^2)^{\frac{1}{2}}, \quad i = j. \tag{15.33}$$

The matrix must be positive, otherwise it would give rise to complex r_{ij} roots which would imply the impossibility of implementation of the algorithm. Once R is calculated, let proceed to the resolution of the two triangular systems:

$$Ry = b \quad e \quad R^T x = y; \tag{15.34}$$

15.2.8 QR Factorization (Decomposition)

An alternative to the Cholesky factorization is the QR factorization [22]. The QR factorization procedure consists in the decomposition of the matrix A into two matrices Q and R, where Q is an orthogonal matrix (i.e., such that $QQ^T = Q^T Q = I$) and R is an upper triangular matrix.

The relations that bind the elements of A to those of the QR matrix product are the following:

$$a_{ij} = \sum_{j=1}^{n} q_{ij} r_{jk} \sum_{j=1}^{k} q_{ij} r_{jk} \tag{15.35}$$

Infact, $r_{jk} = 0$ for $y = k + 1, \ldots, n$ being R upper triangular. There are several methods for the decomposition of the matrix in the QR form, here the Householder and Givens factorization will be explained, putting in relation the similarities and key differences. Once the QR factorization has been performed with the methods mentioned above it is possible to switch to the linear system:

$$\mathbf{A}\underline{\mathbf{x}} = \underline{\mathbf{b}}, \tag{15.36}$$

proceeding in the same way of the LU factorization. More precisely, two linear systems have to been solved:

$$\begin{aligned} \mathbf{Qc} &= \mathbf{b}, \\ \mathbf{Rx} &= \mathbf{c}. \end{aligned} \tag{15.37}$$

The advantage of the introduction of the Q matrix instead of the original matrix A, consists in the fact that, being Q orthogonal, for the inverse uniqueness, Q-1 = Qt, so that the vector c = Q-1b is simply obtained by taking the product matrix vector c = QTb (and not the linear system). As regards the system Rx = c, R being an upper triangular matrix, it is solved with the resolution of backward steps, as in the case of LU factorization.

$$\mathbf{H} = \mathbf{I} - 2\mathbf{w}\,\mathbf{w}^{\mathbf{T}}, \qquad \mathbf{w} \in \mathbb{R}^{n}, \|\mathbf{w}\| = 1, \tag{15.38}$$

The Householder factorization consists in the decomposition of the matrix A into two matrices Q and R from elementary Householder matrices [22]. An elementary Householder real matrix has the following form:

15.2.8.1 Householder Factorization (Decomposition)

H is an orthogonal and symmetric matrix. This method has some practical advantages; in particular it does not require the calculation of the inverse of an upper triangular matrix. Since each Householder transformation is orthogonal, it is enough to show that there is a product of the type matrix $H(x_i, e_j)$ that transforms A into an upper triangular matrix with all positive entries on the main diagonal. At this point, the uniqueness of the factorization ensures that what we have obtained is indeed the searched QR decomposition, which is called Householder matrix (alternately Householder reflection, transformation Householder) associated with w.

$$H_1 a_1^{(1)} = k_1 e_1 \tag{15.39}$$

We need to create a sequence of matrices $A^{(i)}$, with i = 1, … n so that $A^{(n)}$ is upper triangular. Multiplying the Householder matrix H1 to the left of the $A^{(1)}$ matrix we obtain:

$$A^{(2)} = H_1 A^{(1)} = [a_1^{(2)} \;\; a_2^{(2)} \;\; \ldots \;\; a_n^{(2)}] \tag{15.40}$$

where:

$$a1(2) = k1e1 \tag{15.41}$$

while the remaining elements:

$$aj(2) = H_1 aj(1), \qquad j = 2, \ldots, n \tag{15.42}$$

The $A^{(2)}$ matrix has the following structure:

$$A^{(2)} = \begin{pmatrix} k_1 & a_{12}^{(2)} & \cdots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix} = \begin{pmatrix} k_1 & v_1^{T} \\ \mathbf{0} & \hat{A}^{(2)} \end{pmatrix} \tag{15.43}$$

with 0 column vector and $\hat{A}^{(2)}$ a submatrix of appropriate size that we are going to submit to the same operations performed to H1, after having "edged" with one row and one column of an identity matrix to make them reach the size n x n and after

repeat the procedure as above, until arriving at the generic step where the matrix will have the form:

$$A^{(i)} = \begin{pmatrix} k_1 & \bullet & \cdots & \cdots & \cdots & \bullet \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & k_{i-1} & \bullet & \cdots & \bullet \\ \vdots & & 0 & & & \\ \vdots & & \vdots & & \hat{A}^{(i)} & \\ 0 & \cdots & 0 & & & \end{pmatrix} = \begin{pmatrix} A_{11}^{(i)} & A_{12}^{(i)} \\ 0 & \hat{A}(i) \end{pmatrix} \tag{15.44}$$

in which elements denoted by $*$ will not be changed and the sub-matrix $\mathring{A}^{(i)}$ has the form:

$$\hat{A}^{(i)} = \begin{bmatrix} \hat{a}_i^{(i)} & \hat{a}_{i+1}^{(i)} & \cdots & \hat{a}_n^{(i)} \end{bmatrix} \tag{15.45}$$

Now let us say:

$$\hat{H}_i \hat{a}_i^{(i)} = k_i e_1 \tag{15.46}$$

then Hi with i-1 rows and columns of the identity matrix until obtaining:

$$A^{(i+1)} = \begin{pmatrix} k_1 & \bullet & \cdots & \cdots & \cdots & \bullet \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & k_i & \bullet & \cdots & \bullet \\ \vdots & & 0 & & & \\ \vdots & & \vdots & & \hat{A}^{(i+1)} & \\ 0 & \cdots & 0 & & & \end{pmatrix} \tag{15.47}$$

We denote by:

$$(H_n H_{n-1} H_1) A = R_n \tag{15.48}$$

where $H = (H_n H_{n-1} H_1)$ is an orthogonal matrix, while R_n is an upper triangular matrix with positive values on the main diagonal. Multiplying by Q = HT = H-1 equality above we arrive now at the factorization:

$$A = QR \tag{15.49}$$

The description above is valid for square matrices. For rectangular matrices is necessary to make an additional step to reset the elements of the *n*th column of A matrix.

15.2.9 Latent Semantic Indexing (LSI)

A necessary step in the implementation of LSI [23] is the construction of the matrix terms–documents size, where m is the number of distinct terms present in the n documents. The generic element a_{ij} of the matrix terms–documents represent the occurrences of the *i*th word in the *j*th document; given that not all the words appear in all the documents, the terms matrix documents can be considered, without loss of generality, a sparse matrix.

After calculating the occurrences of each word in each document, it is possible, according to the chosen LSI implementation type, apply weighting function in such a way as to consider each word is in the context of local document, and more general of the whole collection of documents; as said, the generic element a_{ij} of A can takes the following form:

$$a_{ij} = L(i, j) \cdot G(i) \tag{15.50}$$

where L(i, j) is the local weighting function of the term i for the document j, and G(i) is the global weighting function for the term.

The terms-documents A matrix is then decomposed in three matrices using SVD, that somehow allows to derive the semantic structure of the document collection using: orthogonal matrices U_r and V_r (containing, respectively, the singular left and right vectors of A), and the diagonal matrix Σ of singular values of A.

After creating the matrices U_r, V_r and Σ taking out the first $k \leq r$ singular triple, it is possible to approximate the original matrix terms-documents A, with the k-rank matrix A_k, as shown in Fig. 15.1.

The V_r and U_r matrices are respectively defined as a matrix of the vector terms and vector documents, while $\Sigma$is said matrix of singular values. The product between the gray regions of matrices U_r, V_r and the diagonal matrix Σ, represents the A_k. Considering the decomposition of the A_k matrix in the three matrices U_k, Σ_k and V_k it is possible to obtain representation of the documents in the k-dimensional space reduced collection using the following equation:

$$\hat{A} = \sum_{k}^{-1} (U_k)^T \; A \tag{15.51}$$

Using this formula, it is possible that any two documents, originally different in the m space of all distinct terms, can be mapped into the same vector of limited space; the set of basis vectors of the dimensional k-space represents the set of concepts or the

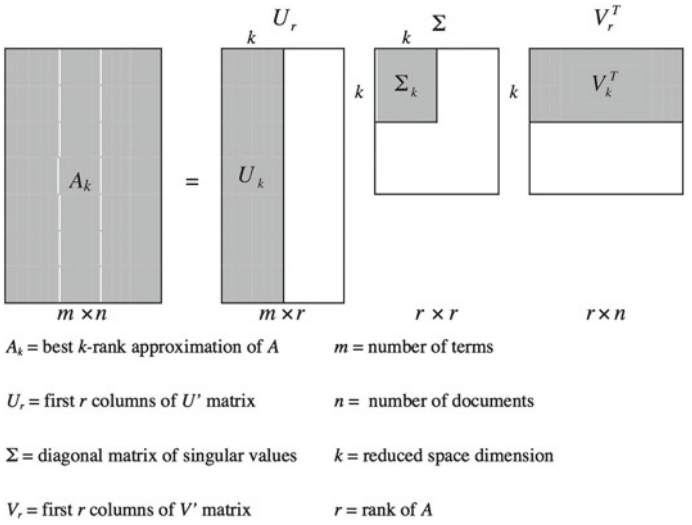


Fig. 15.1 Terms-documents matrix decomposition using the SVD technique. SVD reduction will be used in LSI

different meanings that different documents can assume; then a generic document in the k-dimensional space is represented as a linear combination of concepts or equivalently the basis vectors of the space itself.

Every document, term or query has therefore its representation in k-dimensional space. It has gone from a representation in a m-dimensional space (the size is equal to the number of terms found in the analysis of all documents) to a compressed form of the same vectors in k < m-dimensional space.

It is important to note that with A_k do not wish deliberately reconstruct in a perfect manner the matrix A, because it contains a certain level of noise introduced by uncertainty with which the terms can be chosen to deal with a certain speech or a certain argument.

15.3 The Proposed Method

In this article, a methodology that improves mammographic reports representation, analysis, and retrieval is proposed. The simplification of data and the dimensionality reduction issue is addressed.

A dataset composed of 4461 mammographic reports have been generated randomly extracting the medical diagnosis from the Radiology Information System (RIS) of the Radiological Department of University of Palermo Policlinico Hospital, Italy. The reports are created from breast physicians during the daily workflow and they are written in the italian language.

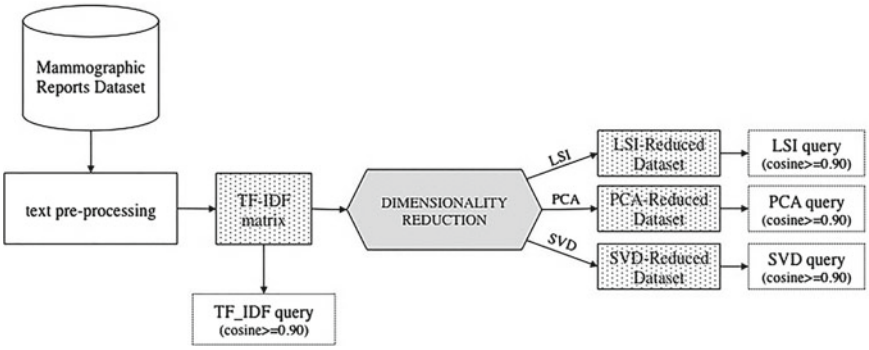


Fig. 15.2 Workflow of the proposed dimensionality reduction methodology

In Fig. 15.2, the workflow of the proposed method is depicted.

Firstly, a sequence of vectors where each of them represents a report is created. Vectors are generated from a directory of text documents (mammographic reports), storing them in the key, value pairs. The key is the report ID, and the value is text content in UTF-8 format.

In this phase, a procedure of cleaning of documents is performed: the preprocessing removes all punctuation characters (. , : ; ! + ?) and unnecessary stop-words (such as the Italian counterparts of a, in, of, this, that, the, and so on).

All the vectors are then processed in order to create the Term Frequency-Inverse Document Frequency (TF-IDF) matrix. The TF-IDF is a function used in Information Retrieval to measure the importance of a term with respect to a document or to a collection of documents. This function increases proportionally to the number of times that the word is contained in the document, but grows in inverse proportion to the frequency of the term in the collection. The idea at the basis of this behavior is to give more importance to the terms that appear in the document, but which in general are not frequent. The weight of the *i*th term of the *j*th document is:

$$\mathrm{tf}_{\mathrm{i,j}} = \frac{\mathrm{n}_{\mathrm{i,j}}}{|\mathrm{d}_{\mathrm{j}}|} \tag{15.52}$$

where n_{ij} is the number of occurrences of the ti term in d_j document, while the denominator is simply the size, expressed as number of terms, of the d_j document. The other factor of the function indicates the overall importance of the term in the collection:

$$\mathrm{idf}_{\mathrm{i}} = \log \frac{|\mathrm{D}|}{|\{\mathrm{d} : t_i \in \mathrm{d}\}|} \tag{15.53}$$

where |D| is the number of document of the collection, while the denominator is the number of documents that contains the ti term. In this way, the TF-IDF score is calculated as:

$$(\mathrm{tf} - \mathrm{idf})_{i,j} = \mathrm{tf}_{i,j} \times \mathrm{idf}_i \tag{15.54}$$

After the TF-IDF matrix calculation, a series of queries have been performed both on raw data and on the matrix after the application of dimensionality reduction methods.

15.4 Experimental Results

The obtained TF-IDF matrix is composed of 4461 × 1154 values: that is, in 4461 mammographic reports a series of 1154 different words have been found (dictionary). Each row (report) is then composed with the word number 0, 1,…, 1153 of the dictionary: if the *i*th word is present, the *i*th value of the *j*th vector (report) will contain the corresponding TF-IDF value of the word, or 0 if the word is missing.

Each report is composed of about 15 words on average (as depicted in Fig. 15.3): in fact, the matrix is composed of 5,147,944 terms, where only 68,741 of them are TF-IDF values, with a percentage of useful information of just 1.33 %, as depicted in Fig. 15.4.

A data dimensionality reduction is then almost necessary. Some terms with more occurrences are depicted in Table 15.1.

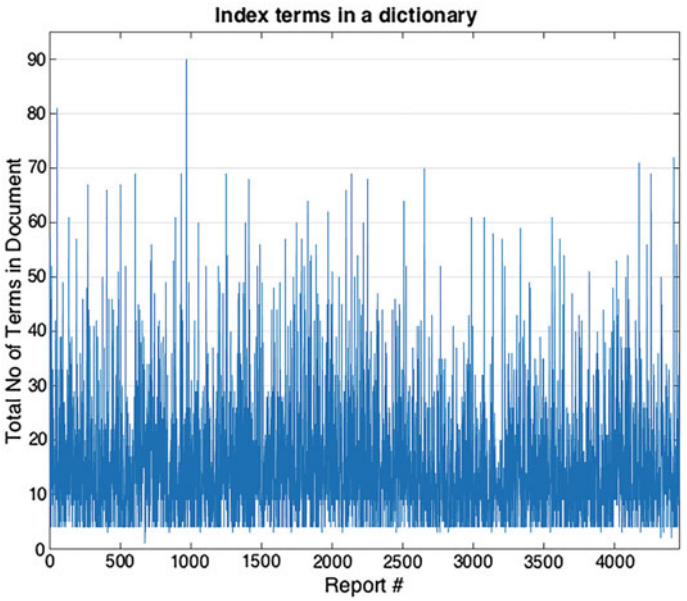


Fig. 15.3 Number of different words in each of the 4461 reports. On average, 15 words are present in each reports

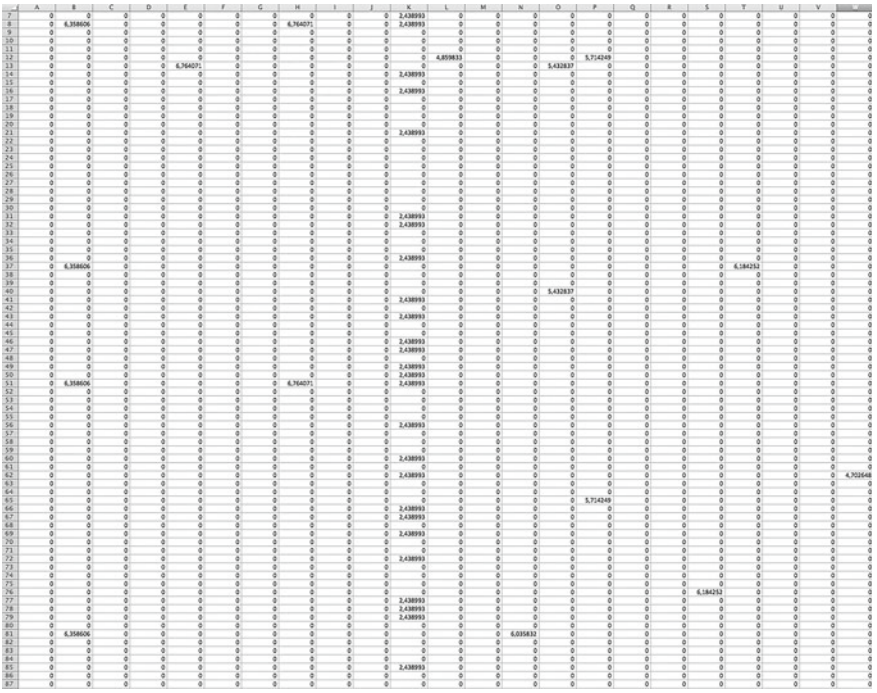


Fig. 15.4 A very small piece of the sparse TF-IDF matrix. The information (numbers) are extremely lower than the noninformation values (zeros)

Table 15.1 Most frequent words in mammographic reports

Word (Italian)	Word (English)	# Occurrences
Immagini	Images	4217
Patologich	Pathological	4215
Assenza	Absence	4095
x-graficamente	x-graphically	4086
Mammelle	Breast	3940
Controllo	Control	3869
Eteroformativo	Heteroformative	2876
Fibroadiposa	Fibrous fatty	2547
Presenza	Presence	1870
Calcificazioni	Calcifications	1598
Fibroghiandolare	Fibrous glandular	1527

In order to execute a simple query, and to check if dimensionality reduction brings to the same results, a series of queries have been performed.

A query is a simple comparison between a query report and all the other reports, where the cosine distance has been used in order to obtain a numerical degree of confidence (distance between vectors).

Given two vectors of attributes, R and S, their cosine similarity is calculated using a dot product and magnitude:

$$\text{similarity} = \cos(\theta) = \frac{R \cdot S}{\|R\| \|S\|} = \\ = \frac{\sum_{i=1}^{n} R_i S_i}{\sqrt{\sum_{i=1}^{n} R_i^2} \sqrt{\sum_{i=1}^{n} S_i^2}} \tag{15.55}$$

where R_i and S_i are components of vector R and S, respectively. The resulting similarity ranges from 1 (meaning exactly opposite) to 1 (meaning exactly the same), with 0 indicating orthogonality (decorrelation), and in between values indicating intermediate similarity or dissimilarity.

For text matching, the attribute vectors R and S are usually the term frequency vectors of the documents. In the case of information retrieval, the cosine similarity of two documents will range from 0 to 1, since the term frequencies (TF-IDF weights) cannot be negative. The angle between two-term frequency vectors cannot be greater than 90°.

The query #1 calculated all the cosine distances between the report n.1 with all the other 4460 reports. All the distances are included in the range from 0 (no similarity) to 1 (total similarity). Applying a threshold $t \geq 90$, the query retrieves 5 similar reports (reports with cosine similarity major or equal than 0.90 with respect to query report).

In Table 15.2, the retrieved reports (translated in English language) using the cosine distance.

As depicted in the previous table, all reports are quite similar: changes are related only to few (1–2) words, so the cosine similarity applied to the vectors does the job. At this point, the main goal of the research is to apply directly to TF-IDF matrix the dimensionality reduction techniques and to find if the same results with the RAW matrix are obtained using features reduction methodologies. In this context, three dimensionality reduction techniques have been compared: Principal Component Analysis (PCA), Singular Value Decomposition (SVD), and Latent Semantic Indexing (LSI). As said before, the PCA convert a set of observations of possibly correlated variables into a set of linearly uncorrelated variables where the number of principal components is less than or equal to the number of original variables while the SVD is a factorization of a real or complex matrix. The LSI is an indexing and retrieval method that identifies patterns and relationships among data.

Table 15.2 An example of the performed queries. The #PG793706 report has been used as the query report. The #PG693094, #PG893700, #PG806610, #PG903846, and #PG890481 reports have been selected by the system. Each selected report has a cosine distance ≥ 0.90 from the query report

PG793706	Left breast fibro-adipose structure no pathological images hetero formative x-sense graphically appreciable recommends annual inspection
PG693094	Left breast structure results fibro-fatty no pathological images hetero x-sense formative graphically appreciable recommends annual periodic inspection
PG893700	Left breast mainly structure fibro-fatty no pathological images hetero x-sense formative graphically appreciable recommends annual periodic inspection
PG806610	Results left breast structure fibro-adipose no pathological images hetero x-sense formative graphically appreciable recommends annual periodic inspection
PG903846	Left breast structure fibro-fatty results no pathological images hetero x-sense formative graphically appreciable recommends annual periodic inspection
PG890481	Left breast structure results mainly fibro-fatty no way pathological images hetero formative x-graphically appreciable recommends annual periodic inspection

Resulting dimensionality-reduced matrix have been calculated applying:

- Equations 15.3–15.13 for PCA reduction;
- Equations 15.14–15.24 for SVD reduction;
- Equations 15.50 and 15.51 for LSI reduction.

Moreover, the number of corrected retrieved documents (distance ≥ 0.90) has been evaluated for different range of number of features.

The graphs depicted in Figs. 15.5, 15.6, and 15.7 show the performance of the LSI, SVD, and PCA techniques in three testing retrieval tasks.

As depicted in Fig. 15.5, various ranks (number of features) have been tested in order to plot the number of retrieved reports varying the rank of the reduced matrices. The dotted line is the number of reports retrieved by means of TF-IDF query, in this case 10 reports. The LSI and PCA reduction techniques (red and blue line, respectively) obtains similar results when the number of features becomes greater than 170 (i.e., the reduced matrix is composed of 170 columns). The result

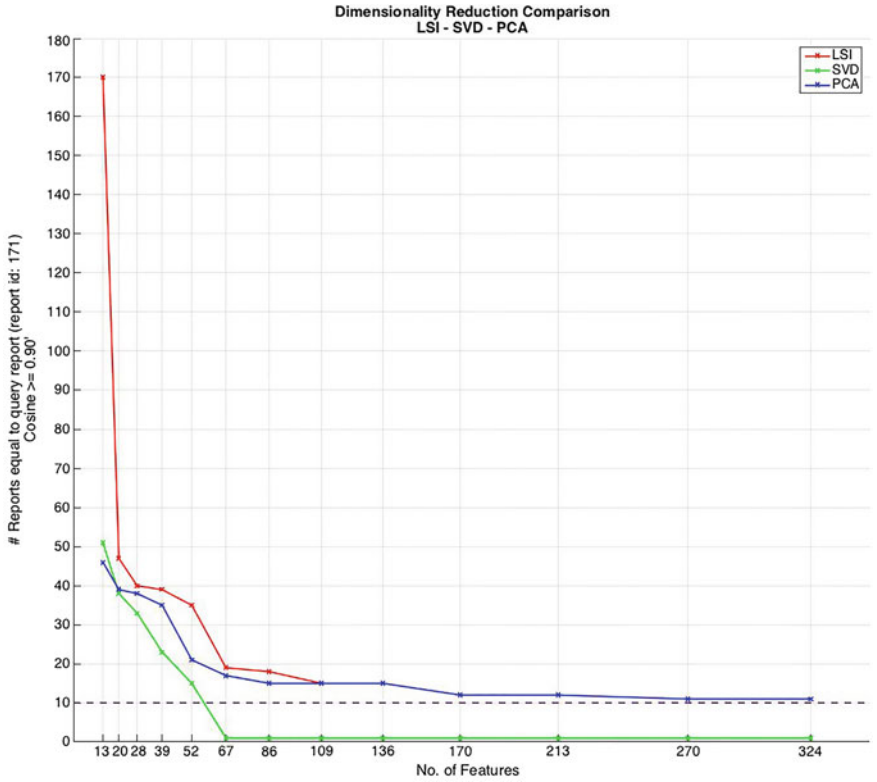


Fig. 15.5 Number of retrieved reports using the different reduction techniques. The plot has been built varying the number of used features. The *dotted horizontal* is the number of reports (10) obtained with the query report #171 using the raw TF-IDF matrix

persists even if the number of features grows. The SVD reduction technique (green line) achieve the best results, retrieving the correct number of documents when the number of features is approximately between 55 and 60, but decaying after this limit.

As depicted in Fig. 15.6, the dimensionality reduction brings to results similar result also using the query report #60. The TF-IDF query retrieves 27 reports: the LSI and PCA techniques achieve the result when the number of features is over about 150, while the SVD reduction technique achieve the best results using about 50 features. Also in this case, the LSI and PCA limit of retrieved reports is the TF-IDF threshold, while the SVD degenerates retrieving a very low number of reports.

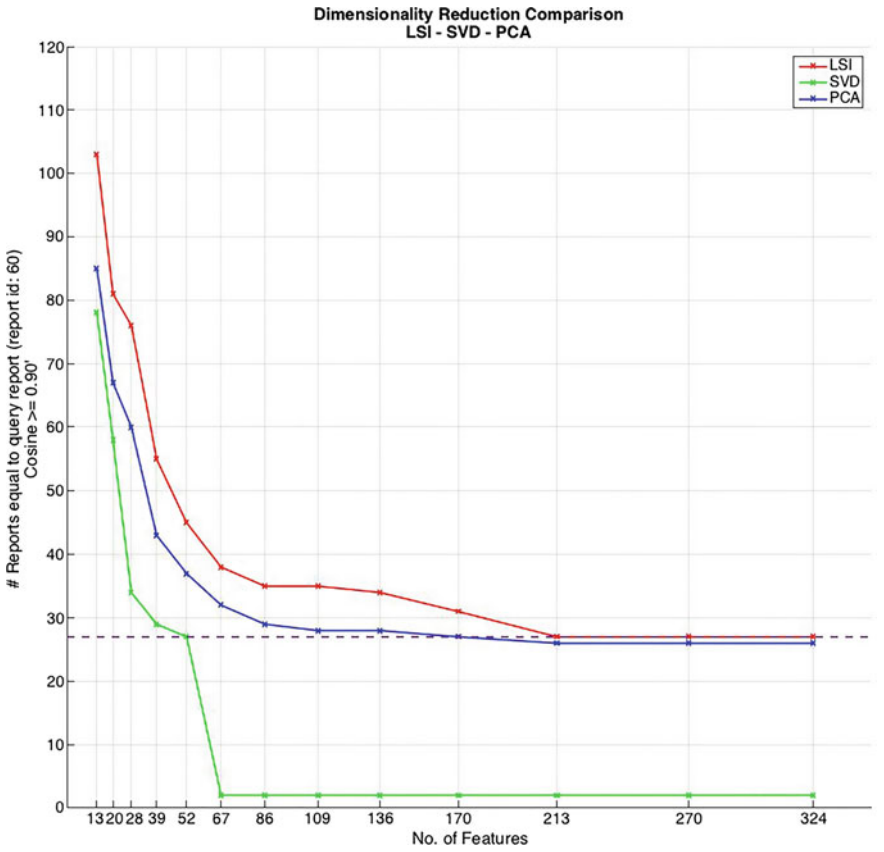


Fig. 15.6 Number of retrieved reports using the different reduction techniques. The plot has been built varying the number of used features. The *dotted horizontal* is the number of reports (27) obtained with the query report #60 using the raw TF-IDF matrix

Previous results are also confirmed by the third experiment (Fig. 15.7). The only difference is that the SVD do not degenerates as before. The number of optimal features is similar for LSI and PCA techniques, while SVD requests a lower number of features.

Table 15.3 summarizes the maximum number of features required by LSI, PCA, SVD performing the query tasks on the 4461 mammographic reports.

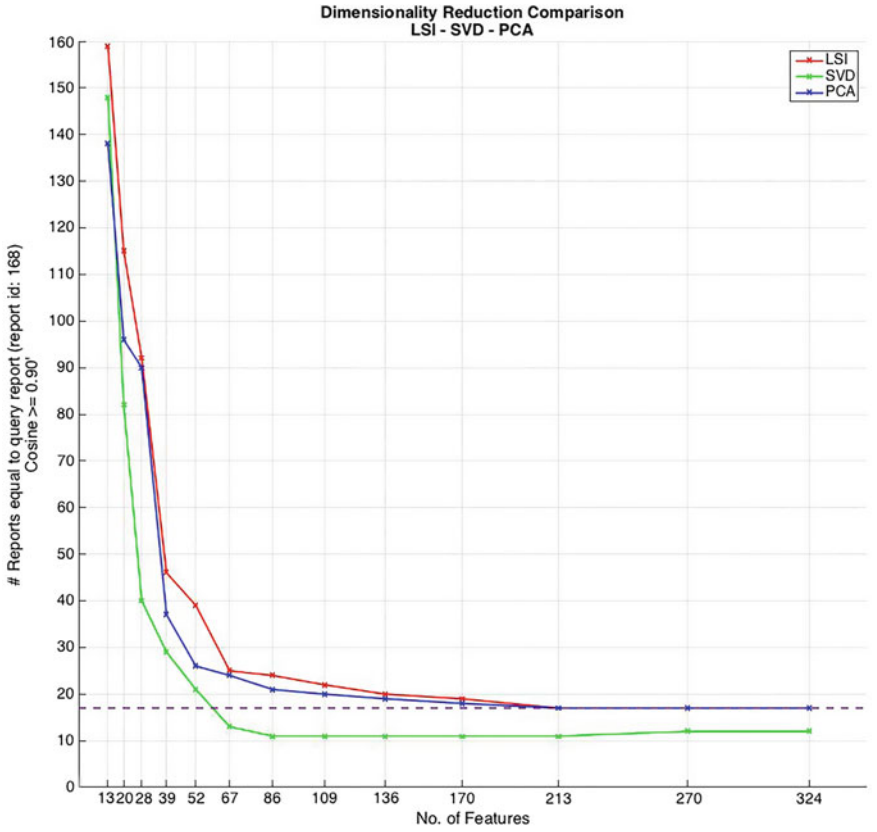


Fig. 15.7 Number of retrieved reports using the different reduction techniques. The plot has been built varying the number of used features. The *dotted horizontal* is the number of reports (17) obtained with the query report #168 using the raw TF-IDF matrix

Table 15.3 Summary of the maximum number of features required by LSI, PCA, SVD to achieve the performance of the method based on the raw TF-IDF matrix. The results has been extracted performing the query tasks on the 4461 mammographic reports

PCA	LSI	SVD
200	200	60

15.5 Conclusions

A comparison of the most known dimensionality reduction techniques has been performed on a dataset composed of 4461 mammographic reports. The dataset is represented using the TF-IDF weights, resulting in a big and sparse matrix of 4461×1154 elements, with extremely low information content (1.33 %).

A series of queries have been performed on the RAW matrix, simulating a mammographic report retrieval system, and the retrieved number of similar documents has been used as gold standard.

Techniques such as LSI, PCA, and SVD decomposition have been applied to the TF-IDF matrix, obtaining comparable results with respect to the ones achieved using the raw unprocessed matrix, when the processed reduced matrix contains less than 13 % of the raw TF-IDF data using the PCI-LSI technique and less than 6 % of the raw TF-IDF data using the SVD technique. The reduction techniques have successfully compressed and highlighted the most significant information, reaching the optimal results using up to 200 features with LSI and PCA reduction techniques, and just up to 60 features for SVD technique, instead of the 1154 features (words) used with the raw TF-IDF data matrix.

Due to the reliability of LSI and PCA results, their use is highly recommended, even if they requires a major number of features if compared to SVD technique, which they are still fewer than the RAW uncompressed original matrix, increasing dramatically any computation performed on reduced feature data.

References

1. Fayyad, U.M., Smyth, P., Uthurusamy, R.: Advances in knowledge discovery and data mining, vol. 21. AAAI Press Menlo Park (1996)
2. Koh, H.C., Tan, G.: Data mining applications in healthcare. J. Healthc. Inform. Manage. **19**(2), 65 (2011)
3. Farruggia, A., Magro, R., Vitabile, S.: Bayesian network based classification of mammography structured reports. In: 2013 International Conference on Computer Medical Applications (ICCMA), pp. 1–5. IEEE (2013)
4. Duan, L., Street, W.N., Xu, E.: Healthcare information systems: data mining methods in the creation of a clinical recommender system. Enterp. Inform. Syst. **5**(2), 169–181 (2011)
5. Farruggia, A., Magro, R., Vitabile, S.: A text based indexing system for mammographic image retrieval and classification. Future Gener. Comput. Syst. **37**, 243–251 (2014)
6. Agnello, L., Comelli, A., Ardizzone, E., Vitabile, S.: Unsupervised tissue classification of brain MR images for voxel-based morphometry analysis. Int. J. Imaging Syst. Technol. **26**(2), 136–150 (2016)
7. Farruggia, A., Magro, R., Vitabile, S.: A novel web service for mammography images indexing. In: 2013 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 225–230. IEEE (2013)
8. Anchala, R., Pant, H., Prabhakaran, D., Franco, O. H.: Decision support system (DSS) for prevention of cardiovascular disease (CVD) among hypertensive (HTN) patients in Andhra Pradesh, India—a cluster randomised community intervention trial. BMC Public Health **12**(1), 1 (2012)
9. Comelli, A., Agnello, L., Vitabile, S.: An ontology-based retrieval system for mammographic reports. In: 2015 IEEE Symposium on Computers and Communication (ISCC), pp. 1001–1006. IEEE (2015)
10. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. J. Am. Soc. Inform. Sci. **41**(6), 391 (1990)
11. Golub, G.H., Van Loan, C.F.: Matrix computations, vol. 3. JHU Press (2012)

12. Yang, Q., Li, F.: Support vector machine for intrusion detection based on LSI feature selection. In: 2006 6th World Congress on Intelligent Control and Automation, vol. 1, pp. 4113–4117. IEEE (2006)
13. Muflikhah, L., Baharudin, B.: Document clustering using concept space and cosine similarity measurement. In: International Conference on Computer Technology and Development, 2009. ICCTD'09, vol. 1, pp. 58–62. IEEE (2009)
14. Lin, P., Zhang, J., An, R.: Data dimensionality reduction approach to improve feature selection performance using sparsified SVD. In: 2014 International Joint Conference on Neural Networks (IJCNN), pp. 1393–1400. IEEE (2014)
15. Zhu, W., Allen, R.B.: Active learning for text classification: Using the LSI subspace signature model. In: 2014 International Conference on Data Science and Advanced Analytics (DSAA), pp. 149–155. IEEE (2014)
16. Landauer, T.K., Foltz, P.W., Laham, D.: An introduction to latent semantic analysis. Discourse process. **25**(2–3), 259–284 (1998)
17. Jolliffe, I.: Principal component analysis. John Wiley & Sons, Ltd (2002)
18. Wall, M.E., Rechtsteiner, A., Rocha, L.M.: Singular value decomposition and principal component analysis. In: A Practical Approach to Microarray Data Analysis, pp. 91–109. Springer, US (2003)
19. Golub, G.H., Reinsch, C.: Singular value decomposition and least squares solutions. Numer. Math. **14**(5), 403–420 (1970)
20. Gorrell, G.: Generalized hebbian algorithm for incremental singular value decomposition in natural language processing. In: EACL, vol. 6, pp. 97–104 (2006)
21. Saunders, M.A.: Large-scale linear programming using the Cholesky factorization (1972)
22. O'Leary, D.P., Whitman, P.: Parallel QR factorization by Householder and modified Gram-Schmidt algorithms. Parallel Comput. **16**(1), 99–112 (1990)
23. Hofmann, T.: Probabilistic latent semantic indexing. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 50–57. ACM (1999)

Chapter 16
Parallel Algorithms for Multirelational Data Mining: Application to Life Science Problems

Rui Camacho, Jorge G. Barbosa, Altino Sampaio, João Ladeiras, Nuno A. Fonseca and Vítor S. Costa

16.1 Introduction

The amount of data stored nowadays in databases is huge and increases every year at a very fast pace. The analysis of such data can be very useful for both business and research. However, in order to analyze large amounts of data or address highly complex problems, computational-based tools are required. *Knowledge Discovery in Databases* (KDD) [15] aims at the discovery of patterns that are both novel and potentially useful. In some applications the comprehensibility of the pattern is also a requirement. The KDD process encompasses a series of steps, one of them being the DM step. In the DM step, algorithms based on Machine Learning (ML) and Statistics, among others, are used to construct models from the data. One may classify the ML algorithms into two groups. Those that require the input data to be contained in a single table of a relational database, and those that can handle directly all the tables in a database. For simplicity, let us denominate the former ones as *propositional*

R. Camacho · J.G. Barbosa (✉) · J. Ladeiras
DEI & Faculty of Engineering of University of Porto, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
e-mail: jbarbosa@fe.up.pt

R. Camacho
e-mail: rcamacho@fe.up.pt

A. Sampaio
IPP, Escola Superior de Tecnologia e Gestão de Felgueiras, CIICESI, Portugal
e-mail: ams@estgf.ipp.pt

N.A. Fonseca
EMBL-European Bioinformatics Institute, Welcome Trust Genome Campus, Hinxton, CB10 1SD, UK
e-mail: nunofonseca@acm.org

V.S. Costa
DCC & Faculty of Sciences of University of Porto, Porto, Portugal
e-mail: vsc@dcc.fc.up.pt

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_16

learners and the latter ones as *multirelational learners*. Multirelational learners are the focus of this chapter and, for simplicity sake, we will use, from now on, a shorter name, that is *relational learners*.

One of the most well-known flavors of relational learning is Inductive Logic Programming [37, 42] (ILP). ILP has been used to construct highly sophisticated models that address very diverse tasks. Examples include Structure–activity prediction [26, 59], a major challenge in rational drug design; Natural language understanding with Grammar acquisition [62]; Protein secondary structure prediction [27]; Qualitative model identification in naive qualitative physics [6]; Workload prediction in computer networks [1]; and Expected survival time of kidney transplanted patients [52].

The success of ILP in the above-mentioned applications is due to the following three major features. First, ILP can very naturally accept background knowledge that can be integrated into the constructed models. Second, ILP learning can harmoniously combine numerical and symbolic computations, while being able to handle structured data. And finally, and very important, it has the capacity of building highly comprehensible models even for complex tasks.

In the remaining part of this chapter, we introduce the basic concepts of ILP that are necessary to understand the rest of the text (Sect. 16.2). We then survey the main approaches to take advantage of parallel execution to speedup ILP systems (Sect. 16.3). Scheduling approaches are discussed in Sect. 16.4 to improve execution of current ILP implementations. In Sect. 16.5, we present applications where the scheduling techniques discussed here will most benefit the parallel execution. Finally, we present a summary of this chapter and draw some conclusions in Sect. 16.6.

16.2 ILP Basic Concepts

We shall address ILP within the broader area of Machine Learning (ML) called *supervised learning*. This core learning task may be stated in a set-theoretic perspective. It aims at learning an intentional description of a certain set in a universe of elements (U), given that we are aware that some elements belong in this set (positive examples), and that others do not (negative examples). The intentional description we strive to learn is called the **concept**. In an ILP setting the concept is usually referred to as the **hypothesis**. Elements known to be in the target set are called instances of the concept. Let Q^+ denote the target set. Elements in $Q^- = \{\neg x \mid x \in U \setminus Q^+\}$ are called **negative instances** or **counter-examples** and are elements of the universe that are not instances of the target concept.

Note that both or either Q^+ and Q^- may be infinite. Learning systems usually consider finite subsets of Q^+ and Q^-. These finite subsets will be denoted by E^+ ($\subseteq Q^+$) and E^- ($\subseteq Q^-$). If not stated otherwise E^+ will be referred as the *positive examples* and E^- will be referred as the *negative examples*.

16.2.1 ILP Framework

ILP is concerned with the generation and justification of hypotheses from a set of examples making use of prior knowledge. The representation most often used is Horn clauses, a subset of First-Order Predicate Calculus.[1] The induced hypotheses are represented as a finite set or conjunction of clauses denoted by H. H is of the form $h_1 \wedge \cdots \wedge h_l$ where each h_i is a nonredundant clause. The prior knowledge, also called *background knowledge*, will be denoted by B. B is described in the same language, that is, it is a finite set or conjunction of clauses, $B = C_1 \wedge \cdots \wedge C_m$, typically definite clauses. In the ILP setting a positive example is often represented by a positive unit ground clause, also known as a *ground atom*; this largely corresponds to a database tuple. E^+ is a conjunction of ground atoms. $E^+ = e_1^+ \wedge e_2^+ \wedge \ldots \wedge e_n^+$, where e_i^+ is an individual positive example.

Negative examples are typically negative unit ground clauses. E^- represents the conjunction of negated ground atoms. If we denote a negative example by $\overline{f_i}$ then $E^- = \overline{f_1} \wedge \overline{f_2} \wedge \ldots \wedge \overline{f_r}$. $E^+ \wedge E^-$ is the training set.

Notice that ILP systems can use nonground examples and not all of them need negative examples to arrive at a concept description [40].

In the ILP framework to learn corresponds to induction, and we must meet the following conditions. First, we must ensure **consistency** conditions, that is, the background B and the training set must be logically consistent. The first two conditions are required for consistency:

$$B \not\models \square$$

$$E^+ \wedge E^- \not\models \square$$

The background knowledge should be consistent with the negative examples. In other words, B should not logically imply any of the negative examples. This condition is called **prior satisfiability** [38]:

$$B \wedge E^- \not\models \square.$$

To justify the need for the induction process it is necessary to satisfy the so called **prior necessity** condition [38]:

$$B \not\models E^+.$$

The induced hypotheses should satisfy the **posterior satisfiability** condition [38]:

$$B \wedge H \wedge E^- \not\models \square.$$

That means the hypotheses found should be consistent with the negative examples.

[1] We refer to John Loyd's book [30] for basic concepts and definitions of Logic Programming.

The set of hypotheses should not be vacuous and explain the positive examples, as stated by the **posterior sufficiency** condition [38]:

$$B \wedge H \models E^+$$

The last condition states that each hypothesis $h_i \in H$ should not be vacuous. This condition is called **posterior necessity** condition.

$$B \wedge h_i \models e_1^+ \vee e_2^+ \vee \cdots \vee e_n^+ \quad (\forall h_i, h_i \in H)$$

As an example imagine that the system is learning the concept of a virtuoso player. Consider that the information given to the system is the following:

$$\mathbf{B} \begin{cases} plays\_instrument(glenn\_gould, piano) \leftarrow \\ plays\_instrument(david\_oistrach, violin) \leftarrow \\ plays\_instrument(fisher, piano) \leftarrow \\ plays\_instrument(john, violin) \leftarrow \\ performance(glenn\_gould, piano, superb) \leftarrow \\ performance(david\_oistrach, violin, superb) \leftarrow \\ performance(fisher, piano, lousy) \leftarrow \\ performance(fisher, chess, superb) \leftarrow \\ performance(john, violin, lousy) \leftarrow \end{cases}$$

$$\mathbf{E}^+ \begin{cases} virtuoso(glenn\_gould) \leftarrow \\ virtuoso(david\_oistrach) \leftarrow \end{cases}$$

$$\mathbf{E}^- \begin{cases} \leftarrow virtuoso(fisher) \\ \leftarrow virtuoso(john) \end{cases}$$

The previously stated prior conditions are met. B is trivially consistent. $E^+ \wedge E^-$ are also trivially consistent. B does not logically entail the negative examples or any of the positive examples since the predicate symbol *virtuoso* does not appear in B.

A possible hypothesis generated by an ILP system could be the following single clause:

virtuoso(Player) ←
    plays_instrument(Player, Instrument) ∧
    performance(Player, Instrument, superb).

The hypothesis found satisfies the posterior conditions. $B \wedge H$ does not logically entail E^- since neither *fisher* nor *john* are capable of a superb performance when playing an instrument.

16.2.2 A Generalization Ordering

The number of hypotheses satisfying the previously stated conditions is, in general, very large and even infinite in most cases. However, it is possible to constrain the hypothesis space by imposing an ordering on the set of clauses. The learning algorithm may then take enumerate hypotheses according to that ordering. The search space may be systematically searched and some parts of the space may be justifiably ignored during the search.

One such ordering over the set of clauses was mentioned originally in [53] and is called **subsumption**.

Definition 1 If C and D are two distinct nonempty clauses, then C subsumes D and we write C $\preceq$ D iff there is a substitution θ such that Cθ $\subseteq$ D.

Definition 2 A clause C is **subsumption equivalent** to a clause D and we write C $\equiv_s$ D iff C$\preceq$ D and D $\preceq$ C. A clause is **reduced** if it is not subsumption equivalent to any proper subset of itself.

Subsumption is the most common ordering over the set of clauses used in ILP systems. θ-subsumption [46] is usually the name used in ILP to refer to the concept of subsumption. The ordering over the set of clauses is sometimes called a **generalization model** [7]. If not stated otherwise, the generalization ordering assumed in the definitions of the rest of the chapter is subsumption.

The concept of redundancy follows naturally from the idea of an ordering over the set of clauses and the concept of equivalence between clauses or sets of clauses. Note, that there are two kinds of redundancy. A literal may be redundant within a clause and a clause may be redundant within a set of clauses.

Definition 3 A literal l is **redundant** in clause C $\vee$ l relative to background theory B iff

$$B \wedge (C \vee l) \equiv B \wedge C.$$

Definition 4 A clause C is redundant in the theory B $\wedge$ C iff

$$B \wedge C \equiv B.$$

The subsumption ordering imposes a lattice over the set of clauses.

Definition 5 A **lattice** is a partially ordered set in which every pair of elements a,b has a greatest lower bound (glb) (represented by a $\sqcap$ b) and least upper bound (lub) (represented by a $\sqcup$ b).

Definition 6 A **generalization ordering** (or **generalization model**) is a partial order[2] over the set of clauses. The lattice imposed by a generalization ordering is called a **generalization lattice**.

[2] A partial order is a reflexive, antisymmetric, and transitive binary relation.

1. $i = 0$
2. $E_i^+ = E^+, H_i = \emptyset$
3. if $E_i^+ = \emptyset$ return H_i otherwise continue
4. increment i
5. $Train_i = E_{i-1}^+ \cup E^-$
6. $D_i = search(B, H_{i-1}, Train_i, \mathscr{L}, \rho, f)$
7. $H_i = H_{i-1} \cup \{D_i\}$
8. $E_p = \{e_p : e_p \in E_{i-1}^+ \; s.t. \; B \cup H_i \models \{e_p\}\}$.
9. $E_i^+ = E_{i-1}^+ \backslash E_p$
10. Go to Step 3

Fig. 16.1 An ILP implementation using a greedy cover set procedure. The final set H is constructed by progressively finding the next best clause in Step 6 (this is the clause with the highest utility). The search for this clause is some generic search procedure that returns the best clause that meets the requirements (previously stated in this section)

The top element of the subsumption lattice is $\square$, the empty clause. The **glb** of two clauses C and D is called the most general instance (**mgi**) and is the union of the two clauses mgi(C,D) = C ∪ D. The **lub** of two clauses C and D is called the **least general generalization** (lgg) [46] of C and D. Under subsumption the glb and lub of clauses are unique up to renaming of variables.

As pointed out by Mitchell [36] the task of concept learning can be mapped into a search through a space of hypothesis. A generalization ordering is a crucial concept in ILP for it is the basis of an organized search of the hypothesis space. The search for an hypothesis is mapped into the traversal of the generalization lattice. The traversal of the generalization lattice is, in general, what leads to the computationally expensive nature of the learning task.

Figure 16.1 shows how these principles are applied in the popular greedy set cover procedure. We repeatedly enumerate hypotheses until finding the best one, and then drop all examples entailed by the hypothesis. Opportunities for parallelism arise within the search process, or by relaxing the coverage algorithm so that searchhes can run in parallel, as we discuss next.

16.3 Parallel Algorithms for ILP

Based on the principal performance bottlenecks for ILP systems, we classify three main sources of parallelism in ILP systems [16].

Notice that other classification criteria can be used. For example, as for LP systems, we can divide strategies into those that expect to use shared memory and those that expect to use distributed memory. Clare and King's Polyfarm [9] is an example of a system designed for distributed environments. Fonseca et al.'s survey of parallel ILP systems [16] reports that most of the best results for parallel ILP were obtained

on shared-memory architecture, but argues that there is scope for experimenting with distributed-memory "clusters".

Search. We can distinguish here between parallel execution of multiple searches, and the parallel execution within a search. The granularity of the latter is substantially finer than the former.

Data. In this, individual processors are provided with subsets of the examples prior to invoking the search procedure in Fig. 16.1. We distinguish between two forms of parallel execution, with different communication requirements. In the first, each processor completes its search and returns the best clause. The set of all clauses are then reexamined in conjunction and a final result constructed by recomputing the utility of each clause using all the data. In the second approach, as each processor finds a good clause, its utility in the final set is recomputed using all the data. The granularity of the second approach is finer than the first, but it has the advantage that all clauses found will also be in the final set of clauses (in both cases, recursive clauses cannot be identified reliably).

Evaluation. The search procedure invoked in Fig. 16.1 evaluates the utility of a clause. This usually requires computing its "coverage", that is, determining the subset of E entailed by the D_i given B and H_{i-1}. An "example-based" strategy involves partitioning E into blocks. The blocks are then provided to individual processors, which compute the examples covered in the block. The final coverage is obtained by the union of examples entailed in each block. There is a similarity to the coarse data parallelism strategy described above. These two processors are provided with subsets of the data. There, the subsets are used to identify different clauses. Here, the subsets are used to evaluate a given set of clauses. An "hypothesis-based" strategy would involve determining subsets of literals in each D_i that can be evaluated independently (this could be identified, for example, using the "cut" transformation described in [54]). Each such independent subset is then evaluated on a separate processor and the final result obtained by the intersection of examples is entailed by the subsets.

The three strategies above are not mutually exclusive. In fact, a parallel algorithm may exploit several. Furthermore, we again observe that the parallel algorithms can be classified in many different ways. For instance, we could also classify the parallel algorithms regarding their *correct*, i.e., do they produce the same solution (correct) as the corresponding sequential algorithm. Next, we will focus our classification of previous work on parallel ILP systems on the three strategies mentioned above, along with the hardware architecture. Figure 16.2 shows a brief summary of the entries that follow.

Dehaspe and De Raedt [13] developed the first parallel ILP system that we are aware of. The system is a parallel implementation of Claudien, an ILP system capable of discovering general clausal constraints. The strategy is based on the parallel exploration of the search space where each processor keeps a pool of clauses to specialize, and shares part of them to idle processors (processors with an empty pool). In the end, the sets of clauses found in each processor are combined. The system was

Fig. 16.2 Parallel ILP Systems Reported in the Literature

Parallelism	Architecture	
	Shared-Memory	Distributed-Memory
Search	Dehaspe & De Raedt [13] Ohwada & Mizoguchi [43] Ohwada *et al.* [44] Wielemaker [61]	No reports
Data	Wang & Skillicorn [55] Graham *et. al* [22]	Matsui *et al.* [32] Clare & King [9] Blaták & Popelínský [4]
Evaluation	Ohwada & Mizoguchi [43] Graham *et al.* [22]	Matsui *et al.* [32] Konstantopoulos [28]

evaluated on a shared-memory computer with two data sets and exhibited a linear speedup up to 16 processors.

Ohwada and Mizoguchi [43] have implemented an algorithm based on inverse entailment [39]. The implementation uses a parallel logic programming language and explored the parallel evaluation of clause coverage, and two strategies for search parallelisation (parallel exploration of independent hypotheses and parallel exploration of each refinement branch of a search space arising from each hypothesis). The system was applied to three variants of an email classification data set and the experiments performed evaluated each strategy. The results on a shared-memory parallel computer showed a sublinear speedup in all strategies, although parallel coverage testing appeared to yield the best results.

An algorithm that explores the search space in parallel was first implemented by Ohwada et al. [44]. The set of nodes to be explored is dynamic and implemented using contract net communication [56]. Their paper investigated two types of interprocess communication, with results showing near-linear speedups on a 10-processor machine.

Wielemaker [61] implemented a parallel version of a randomized search found in the Aleph system. The parallel implementation executes concurrently several randomized local searches using a multithreaded version of the SWI Prolog engine. Experiments examined performance as the number of processors was progressively increased. Near-linear speedups were observed up to 4 processors, however the speedup was not sustained as the number of processors increased to 16.

Wang and Skillicorn [55] have implemented a parallel version of the Progol algorithm [41] by partitioning the data and applying a sequential search algorithm to each partition. Data are partitioned by dividing the positive examples among all processors and by replicating the negative examples at each processor. Each processor then performs a search using its local data to find the (locally) best clause. The true utility of each clause found is then recomputed by sharing it among all processors. Note that this algorithm exploits the parallelization strategies identified (parallel search, data and evaluation) mentioned above, thus being an example that the strategies are not mutually exclusive. Experiments with three data sets suggest linear speedups on machines with four and six processors.

A study by Matsui et al. [32] evaluates and compares two algorithms based on data parallelism and parallel evaluation of refinements of a clause (the paper calls this parallel exploration of the search space, although it really is a parallelisation of the

clause evaluation process). The two strategies are used to examine the performance of a parallel implementation of the FOIL [47] system. Experiments are restricted to a small synthetic data set (the "trains" problem [35]) and the results show poor speedups from parallelisation of clause evaluation. Data parallelism showed initial promise, with near-linear speedups up to four processors. Above four processors, speedup was found to be sublinear due to increased communication costs.

PolyFarm was a parallel ILP system specifically designed for the discovery of first-order association rules on distributed-memory machines developed by Clare and King [9]. Data are partitioned amongst multiple processors and the system follows a master–worker strategy. The master generates the rules and reports the results and workers perform the coverage tests of the set of rules received from the master on the local data. Counts are aggregated by a special type of worker that reports the final counts to the master. No performance evaluation of the system is available.

An implementation of a parallel ILP system, using the PVM message passing library, was done by Graham et al. [22]. Parallelisation is achieved by partitioning the data and by parallel coverage testing of sets of clauses (corresponding to different parts of the search space) on each processor. Near-linear speedups are reported up to 16 processors on a shared-memory machine.

Konstantopoulos [28] has investigated a data parallel version of a deterministic top-down search implemented within the Aleph ILP system [57]. The parallel implementation uses the MPI library and performs coverage tests in parallel on multiple machines. This strategy is quite similar to that reported in Graham et al., with the caveat that testing is restricted to one clause at a time (Graham et al. look at sets of clauses). Results are not promising, probably due to the overfine granularity of testing one clause at a time.

dRap was developed by Blaták and Popelínský [4]. This was a parallel ILP system specifically designed for the discovery of first-order association rules on distributed-memory machines. Data are partitioned amongst multiple processors and the system follows a master–worker strategy. The master generates the partitions and each worker then executes a sequential first-order association rules learner. The master collects the rules found by the workers and then redistributes the rules by all the workers to compute the support on the whole data set. No performance evaluation of the system is reported.

Angel-Martinez et al. [31] describe the use of GPUs to perform parallel evaluation of hypotheses. The authors extended the widely used Aleph system, parallelism is implemented by evaluating clauses as Datalog queries, and then taking advantage of prior work in implementing the main database primitives in GPUs. Results show a one or two order of magnitude in large data sets, where the overhead of sending a clause to a GPU is significantly less than the benefits of parallel execution.
Results reported by these papers are summarized in Fig. 16.3. The principal points that emerge are these:

1. Most of the effort has been focused on shared-memory machines where the communication costs are lower than for distributed-memory machines.

Fig. 16.3 Summary of speedups reported of parallel ILP systems. The numbers in parentheses refer to the number of processors. Neither Clare and King nor Blaták and Popelínský report any speedups

Parallelism	Speedup	
	Shared-Memory	Distributed-Memory
Search	Dehaspe & De Raedt: Linear (16) Ohwada & Mizoguchi: 2–3 (6) Ohwada *et al.*: 8 (10) Wielemaker: 7 (16)†	No reports
Data	Wang & Skillicorn: Linear or better (6) Graham *et. al*:linear (16)	Matsui *et al.*: 4 (15)‡ Clare & King: – Blaták & Popelínský: –
Evaluation	Ohwada & Mizoguchi: 4 (6) Graham *et al.*: 5 (8)	Matsui *et al.*: 1 (15) Konstantopoulos: none

† linear up to 4 processors
‡ linear up to 5 processors

2. Speedups observed on shared-memory machines are higher than those observed on distributed-memory ones. Maximum disparity is observed with parallel execution of the coverage tests: this is undoubtedly due to the fact that communication costs are high for distributed-memory machines, and the granularity of the task is finer than other forms of parallelism.

Despite the apparently discouraging results observed to date on distributed-memory machines, we believe that a further investigation is warranted for several reasons. First, the results are not obtained from a systematic effort to investigate the effect of the different kinds of parallelism. That is, results that are available are obtained from a mix of fine and coarse-grained parallelization, on differently configured networks and with different communication protocols. Second, the availability and-parallelism of shared-memory architecture machines continues to be substantially lower than distributed-memory ones (for example, distributed-memory "clusters" comprised of 10s or 100s of machines are relatively easy, and cheap, to construct). There is, therefore, practical interest in examining if significant speedups are achievable in distributed-memory architectures. In this paper, we present a systematic empirical evaluation of coarse-grained search, data and evaluation parallelization for such architectures using a well-established network of machines (a Beowulf cluster) and a widely accepted protocol for communication (an implementation of the Message Passing Interface, or MPI [17], that can be used by applications running in heterogeneous distributed-memory architectures).

16.3.1 The APIS ILP System

There is a strong connection between parallelism in the context of ILP and-parallelism in the context of logic programming (LP). Parallelism has been widely studied in LP [23], where it can be exploited implicitly, by parallelising the LP inference mechanism, or explicitly, by extending logic programs with primitives that create and manage tasks and allow for task communication.

Two major sources of implicit parallelism have been recognized within LP. In *or-parallelism*, the search in the LP system is run in parallel. Or-parallelism is known to achieve scalable speedups on current hardware [10] but it works best when we want to perform complete search, which may be expensive in the context of ILP.

And-Parallelism corresponds to running conjunctions of goals, or/and tasks, in parallel. If the goals communicate during the parallel computation, it is called *dependent and-parallelism*. Dependent and-parallelism may be used for concurrent languages or to implement pipelines [5]. On the other hand, *independent and-parallelism* (IAP) is useful in divide-and-conquer applications and often corresponds to coarse-grained tasks. Our approach is based on *independent and-parallelism* (IAP).

The APIS system introduces a new approach to the parallel execution of ILP systems. APIS partitions the hypothesis space so that each subspace can be executed in parallel. We define two types of subspaces: standard subspaces requiring theorem proving for clause evaluation; and subspaces that efficiently compute clause evaluation without the need of theorem proving. Not only the partition enables the parallel search but also achieves additional speedups resulting from the fact that some of the subspaces do not use theorem proving to evaluate the hypotheses. Unfortunately, although a partition is established on the hypothesis space the resulting subspaces are not completely independent as we discuss later.

It is well known in LP that if a clause has subsets of literals with literals in each subset not sharing variables with any literal of the other subsets, then each subset can be executed in parallel. When traversing the hypothesis space an MDIE-based ILP system constructs and evaluates clauses. Traditionally, clause evaluation is done using a theorem prover.[3] Among the clauses constructed during the search, there are clauses that satisfy the LP IAP constraint: clauses with sets of literals that do not share variables. In this case, we can then apply bottom-up techniques. We generate in parallel each subset of literals in the "traditional" way (using theorem proving for evaluation) and then combine each subset to form a new clause and make the evaluation of the combined clause in a more efficient way. The coverage of the combined clause is computed by the intersection of the coverage lists of the clauses being combined. This result cannot, however, be efficiently applied in a traditional ILP system since it is computationally expensive to determine if the partition of the clause's literals into subsets that do not share variables exists. The key point of the APIS system approach is to establish the partition of the hypothesis space based on the usage mode of the predicates and verify independence at compile time, thus avoiding the analysis of each clause for independent sets of literals at induction time. Such partition can be computed as a preprocessing step in an efficient way. The overall process is therefore divided into two steps: a preprocessing step where user-provided mode of usage information used to establish the partition of the hypothesis space; and the execution in parallel of the subspaces resulting from the previous step. We now explain each step in detail.

An *island* is a set of mode declarations satisfying the following two conditions. Each mode declaration shares at least one type with other modes in the same *island*.

[3] Counting the number of examples derivable from the hypothesis and the background knowledge.

Each mode declaration does not share any type with any other mode declaration outside the *island*. Types of the head literal are excluded from the above-mentioned "type checking".

The core of the APIS system is the identification of the *islands* since they will be used in the partition of the hypothesis space. The algorithm for the automatic identification of the *islands* is described by Algorithm 1. The use of the islands in the parallel search of the hypothesis space is described by Algorithm 2.

Algorithm 1 *Islands* computation from the mode declarations

```
1: function COMPUTEISLANDS(AllModes)
2:     IslandsSet ← ∅
3:     Modes ← removeHeadInputArguments(AllModes)          ▷ preprocessing step
4:     while Modes ≠ ∅ do                                  ▷ process all modes
5:         Mode = withoutInputArguments(Modes)
6:         Modes = Modes \ { Mode }
7:         Island = ExtendIsland({Mode}, Modes)
8:         IslandsSet ← IslandsSet ∪ { Island }
9:     end while
10:    return IslandsSet
11: end function
12:
13: function EXTENDISLAND(Island, Modes)
14:    repeat
15:        Mode = LinkedToTheIsland(Modes)                 ▷ returns ∅ if no mode was found
16:        Modes = Modes \ { Mode }
17:        Island ← Island ∪ { Mode }
18:    until Mode = ∅
19:    return Island                                       ▷ Island as a set of modes
20: end function
```

The algorithm accepts as input a set of mode declarations and returns a set of *islands*. First, a preprocessing step removes the types appearing in the head mode declaration and the mode arguments that are constants. After the preprocessing the algorithm enters a cycle where each island is determined and terminates whenever there are no more mode declarations to process. In the main cycle a seed mode is chosen to start a new *island* and then the island is "expanded". Expanding an island consists in adding any mode declaration not yet in the island sharing a type with any mode already in the island. The expansion stops as soon as there is no mode outside the island sharing a type with the modes inside the island.

The APIS execution algorithm is schematized as Algorithm 2. Algorithm 2 starts by computing the *islands*: each client node is instructed to upload the data set without the mode declarations. In the line of MDIE greedy cover ILP algorithms the main cycle generates hypotheses, adds the best discovered hypothesis to the final theory, and removes the examples covered by the added hypothesis. The cycle repeats until no uncovered positive examples are left. The specificity of APIS is evident in (Steps 8 through 19). In this part of the algorithm APIS uses a pool of client nodes and a pool of subspaces of the hypothesis space to search (determined by the partition made on the mode declarations). Each node searches a subspace. There are two kinds of subspaces: "saturation-based" subspaces; and "combination-based" subspaces. A saturation-based subspace is generated as in a typical ILP general-to-specific search

Algorithm 2 The APIS parallel execution algorithm

```
1: function INDUCETHEORY(DataSet, Clients)
2:     Islands ← COMPUTEISLANDS(GetModes(DataSet))
3:     Theory ← ∅
4:     Examples ← PositiveExamples(DataSet)                    ▷ initial positive examples
5:     broadCast(Clients, loadIslandsDataSets)
6:     while Examples ≠ ∅ do                                   ▷ while not covering all positives
7:         Samples = getSample(Examples)
8:         Jobs ← getJobs(Islands, Samples)
9:         while Jobs ≠ ∅ do                                   ▷ all islands processed in the cycle
10:            if Clients ≠ ∅ then
11:                W ← client(Clients)                         ▷ get next available client
12:                Clients ← Clients \ { W }
13:                J ← nextJob(Jobs)                           ▷ select a nonprocessed job
14:                Jobs ← Jobs \ { J }
15:                sendMsg(W, J)                               ▷ client W processes job J
16:            end if
17:            if FinishedClient(C) ≠ ∅ then Clients ← Clients ∪ { C }
18:            end if
19:        end while
20:        h = IslandsResults()                                ▷ returns the best hupothesis
21:        Covered = Cover(h, Examples)                        ▷ compute h coverage
22:        Examples = Examples \ Covered
23:        if Examples ≠ ∅ then broadcast(Clients, removeExamples(Covered))
24:        end if
25:        Theory ← Theory ∪ { h }
26:    end while
27:    return Theory
28: end function
```

followed by reduction steps that characterize MDIE systems [39]. The difference is that to generate the subspace a subset of user-provided mode declarations (an *island*) is used. All clauses constructed in this kind of subspace are evaluated by proving the examples from background knowledge and the hypothesis under evaluation. On the other hand in "combination-based" subspaces no further theorem proving is required. Each clause constructed in a combination-based subspace merges pairs of clauses each one coming from previously searched spaces that do not share islands. This restriction allows the evaluation of the new clauses by intersection of the parent's coverage lists. We can see that there is a dependency among combination-based subspaces. The saturation-based subspaces are the only ones completely independent. Let us further remark that in the main cycle of the algorithm we search several hypothesis spaces at the same time.[4] We have an hypothesis space for each example of the seed. All of the jobs to execute (subspaces to be searched) are in a common pool but only subspaces belonging to the same example are combined. The number of jobs associated with each example is equal to the number of all possible combinations of the islands up to the clause length. First, the saturation-based subspaces are generated, then these subspaces are combined in pairs, in groups of three, and so on up to the "clause length" value. The combinations are all computed once before execution of the algorithm and each subspace is scheduled to run as soon as the two "parents" finish.

[4] As many as the size of the sample

16.4 Scheduling and Load Balancing

Parallel implementations have been employed to significantly enhance and speed up solution search, allowing to reach high-quality results with reasonable execution times even for hard-to-solve optimization problems. In this section, we first present the state of the art of computing platforms for parallel computing, and then several approaches for accelerating ILP algorithms are discussed.

16.4.1 Parallel Computing Platforms

Parallel applications demand for substantial number of computing resources, which were traditionally deployed by dedicated high-performance computing (HPC) infrastructures such as clusters, and later by Grid computing [18]. Even though Grids introduce new capabilities such as larger number of resources belonging to different administrative domains and the ability to select the best set of machines meeting the requirements of applications, there are limitations related to runtime environments for applications and accomplishment of applications' needs.

Rather than owning physical, fixed-capacity clusters, organizations have recently shifted onto the Cloud computing [19] paradigm. Compared to aforementioned traditional networked computing environments, Cloud computing offers to end users a variety of services covering the entire computing stack. Clouds represent a new kind of computational model, providing better use of distributed resources, while offering dynamic flexible infrastructures and quality of service (QoS) guaranteed services. They support various configurations (e.g., CPU, memory, I/O networking, storage) and scaling capacities while abstracting resource management. Cloud computing has recently gained popularity as a resource platform for on-demand, high-availability, and high-scalability access to resources, using a pay-as-you-go model [3]. Some of today's major commercial Cloud providers are Amazon EC2 [2], and Google Cloud Platform [21]. Clouds rely on virtualisation technology for the management of traditional data center resource provisioning. Virtualisation has several benefits for scientific computing, such as provisioning of isolated computing environments on shared multicore machines. Huang et al. [24] have conducted a performance evaluation regarding the use of virtualisation and have concluded that HPC applications (which are performance oriented) can achieve almost the same performance as those running in a native, nonvirtualized environment.

By means of virtualisation, a collection of virtual machines (VMs) run on top of physical machines (PMs) to create virtual clusters [33]. Also, a VM can be suspended and later resumed on either the same or on a different PM, which is useful for fault tolerance and load balancing. A virtual cluster is created so a user has exclusive access to a customized virtual execution environment. The provisioning of elastic virtual clusters as on-demand pay-as-you-go resources is an essential characteristic of Clouds, endowing great flexibility and scalability for end users and their applications.

In this context, resources can be dynamically allocated, expanded, shrunk, or moved, according to applications demand.

Parallel and distributed applications, as is the case of parallel implementation of ILP systems, exploit the processing power of a cluster of processors. As such, these applications can utilize the Cloud to rapidly deploy an application-specific virtual cluster infrastructure to achieve new levels of availability and scalability. Although Clouds were built primarily with business computing needs in mind, their value has been already recognized within the scientific community as an easy way to store, process, and retrieve huge data without worrying about the hardware needed. For example, Delgado et al. [14] have explored the use of Cloud computing to execute scientific applications, with a specific focus on medical image processing and computational fluid dynamics applications. Also Juve and Deelman [25] discussed many possible ways to deploy scientific applications on a Cloud, ranging from astronomy to earthquake science. The Magellan project, which takes place at the Argonne Leadership Computing Facility and the National Energy Research Scientific Computing Facility, aimed at investigating the use of cloud computing for science [48]. A diverse set of scientific data parallel applications, with no tight coupling between tasks, was running on the Magellan resources, and the results allowed to conclude that current cloud software can be used for science clouds.

16.4.2 Load Balancing for ILP Algorithms

In the specific case of parallel ILP systems considered in this chapter, islands are characterized by having diverse sizes and requiring different processing capacity needs. A task parallel approach is adequate to this problem and homogeneously scheduling the same amount of CPU resource to jobs in charge of processing diverse-sized islands will result in jobs' finish time imbalances, with consequent nonoptimized makespan. Furthermore, if we take into account that Cloud resources are usually billed by hour [2], an inefficient schedule of parallel ILP jobs will result in increased costs to solve the parallel ILP search problem. Therefore, it is important to produce schedules of jobs that consider their diverse needs, and the heterogeneity of resources, in order to achieve the objectives of reducing the makespan (i.e., the finishing time of the last job in the system) and maximizing load balancing. To minimize makespan it is essential to allocate jobs correctly so that computer loads and communication overheads will be well balanced. In turn, load balancing aims to distribute workload between available machines to obtain as good throughput and resource utilization as possible. Despite load balancing and makespan concepts being related, a good load balancing does not always lead to minimal makespan and vice versa.

Several papers address the problem of static and dynamic minimization of makespan and maximization of load balancing in parallel and distributed systems. A good review and classification of state-of-the-art load balancing methods for jobs dispatched to run independently on multiple computers can be found in [45, 64, 67]. This section points out some relevant scheduling algorithms and

strategies that we believe can improve load balancing and optimize makespan of independent and-parallelism ILP approach. When managing resources in a cloud computing environment, scheduling can be made in different layers, as described next.

16.4.2.1 Scheduling at the Application Level

Concerning the scheduling of jobs onto VMs, Dhinesh and Krishna [29] proposed to minimize the makespan and maximize load balancing of applications in the Cloud. They use Swarm Intelligence (SI) to propose an Artificial Bee Colony (ABC) algorithm, named Honey Bee Behavior inspired Load Balancing (HBB-LB), which aims to achieve well-balanced load across VMs for maximizing the throughput and minimization of makespan in a Cloud infrastructure. Cumulatively, HBB-LB algorithm not only dynamically balances the load but also considers the priorities of tasks in the waiting queues of VMs in order to minimize the time spent in the queues. The scheduling problem is solved considering that a job is a honey bee and VMs are the food sources. First, VMs are grouped into three sets: (i) overloaded VMs; (ii) underloaded VMs; and (iii) balanced VMs. Processing time of a job varies from one VM to another based on VM's capacity. Then, jobs removed from overloaded VMs have to find suitable underloaded VMs to get placed in. If there are several suitable VMs to allocate the task in, the task chooses the VM which as a less number of tasks with the same kind of priority. The winning task is allocated to the selected VM and state information is updated, which includes the workload on all VMs, number of various jobs in each VM, jobs priority in each VM, and the number of VMs in each set. These details will be helpful for other jobs, i.e., whenever a high priority job is submitted, it will consider the VM that has less number of high priority jobs to execute earlier. Once the jobs switching process is over, the balanced VMs are included into the balanced VM set. The load balancing process ends when this set contains all the VMs. This solution was successfully tested with Cloudsim [8] by means of simulation. The results have showed a good performance for heterogeneous Cloud computing systems, in terms of average execution time and waiting time of jobs on queue.

Ramezani et al. [50] contributes with a Task-based System Load Balancing method using Particle Swarm Optimization (TBSLB-PSO) that achieves system load balancing in Cloud environments by only transferring extra tasks from an overloaded VM instead of migrating the entire overloaded VM (which is known to be time- and cost-consuming). The problem of finding an optimal solution for allocating these extra tasks from an overloaded VM to appropriate host VMs is solved using Particle Swarm Optimization (PSO) heuristic. PSO is a population-based search algorithm based on the simulation of the social behavior of birds. The scheduling algorithm is multi-objective, aiming at minimizing task execution time and task transfer time. TBSLB-PSO method considers both computing-intensive and data-intensive tasks. Computing-intensive tasks demand extensive computation (e.g., scientific applications) while data-intensive tasks are characterized by high volumes of data to be published and maintained over time. The scheduling optimization model takes into

account the number of CPUs on host VMs to schedule computing-intensive tasks, and the bandwidth as a variable to minimize the tasks transferring time for data-intensive applications. To solve this multi-objective optimization problem, the PSO algorithm is applied to find an optimal way to allocate extra tasks to the new (underloaded) VMs with less task execution and task transfer time. The TBSLB-PSO algorithm works in six steps, ranging from monitoring and analysis of PMs, VMs, and tasks, determining overloaded VMs, finding optimal homogeneous VMs to transfer the tasks to and optimal task migration schema, and transferring tasks and updating the scheduler information. Authors simulated their proposal with Cloudsim toolkit, which was able to schedule a set of 10 tasks onto 5 VMs over 3 PMs, in 0.224 s. The authors have also solved this problem with Multi-Objective Genetic Algorithm (MOGA) [49] as an alternative to PSO, although no comparison information of the two alternatives was provided.

16.4.2.2 Scheduling at the Virtualisation Level

In the case of scheduling VMs onto PMs, Dasgupta et al. [11] proposed a novel load balancing strategy using Genetic Algorithms (GA) aiming at balancing the load of the cloud infrastructure while trying to minimizing the makespan of a given tasks set. GAs is a stochastic searching algorithm based on the mechanisms of natural selection and genetics, widely used in complex and vast search space and known to be very efficient in searching out global optimum solutions. The results showed that the proposed algorithm outperformed the existing approaches like First Come First Serve (FCFS), Round Robing (RR), and the local search algorithm Stochastic Hill Climbing (SHC).

16.4.2.3 Scheduling at the Programming Level

Aiming at optimizing load balancing, Xu et al. [65] proposed a novel model to balance data distribution to improve Cloud computing performance in data-intensive applications, such as distributed data mining. Their work consists in extending the classic MapReduce model [12] in order to fight its computational imbalance problem. MapReduce provides a set of frameworks that aim to enable productive programming of computer clusters. The frameworks are efficient in the processing of huge data sets with flexible job decomposition and subtasks allocation. Using distributed programming frameworks such as Hadoop [60], the application programmers can construct parallel dataflows using map and reduce functions to achieve portability and scalability of their applications. However, these frameworks are not suitable for all scientific workloads since they are designed for data-intensive workloads. In fact, the original load balance in Hadoop considers only static storage space balance, without considering the workloads attached to the data blocks, which is of paramount importance for the unbalanced ILP jobs. To tackle this issue, authors have proposed a completely distributed approach for adjusting load balancing based on agent–workers running

on nodes, unlike original Hadoop which relies on an agent–master for centralized task processing such as task allocation. By representing each node, agents can communicate with each other to cooperatively manage and adjust the balance of working loads. These series of agents monitor the data nodes (e.g., hardware performance, working load in real time), and jointly make decisions on how to move data blocks to maximize load balancing. In the process of load balancing, the overload nodes request computing resources from other nodes, and copy their data to the new nodes who have more resources. Initially, agents have no knowledge of their neighbors' working load distribution, and so a token-based heuristic algorithm is proposed as well. The goal of balancing working load is to reduce calculation and free storage variance. In the optimal case, each node holds at most one computation task and has almost the same size of the free storage. Authors have evaluated their proposal through simulation and concluded that agents can improve load balancing efficiency with limited communication costs among agents.

16.5 Life Science Applications

The Life Sciences have a lot of scientific problems where Machine Learning technique may be very helpful. However, some of the important problems have to deal with data from quite different sources, encoded in difference representation schemes and with structure. To analyze data sets in those kinds of problems/domains using algorithm such Decision Trees, SVMs or Artificial Neural Nets,[5] the data has to be "enclosed" into a single table of a Relational Database.[6] Most often this reduction procedure leads to information loss or an extensive amount of preprocessing. The advantages of such algorithm's analysis are that, usually, they are much faster than ILP's analysis.

To analyze data with structure, encoded in different encoding schemes, ILP have been recognized to have advantages over propositional learners. ILP can handle "naturally" several relations, data with structure, encoded in different representation schemes, can combine harmoniously symbolic and numerical computations and, most importantly the constructed models and most often comprehensible to the domain experts. This last feature may be of capital importance to scientific applications where understating the phenomena that produced the data is required. ILP models may provide clues for such explanations.

In this section, we visit two applications where ILP have given very good results, produced comprehensible models, and showed several advantages over propositional learners. We discuss also the advantage of the application of the parallel execution of ILP in these types of applications.

[5] Propositional level algorithms

[6] or a sheet of a spreadsheet

16.5.1 Structure–Activity Relationship Experiments

ILP have been extensively used in Structure–Activity Relationship[7] (SAR) problems. Published studies include [26, 51, 59] where ILP systems predicted mutagenicity and [58] caricnogenicity activity. In order to assess the impact of the parallel approach to ILP implemented in the APIS system (See Sect. 16.3) we have used four data sets originated from SAR problems.[8] Two of them are the ones just mentioned for predicting mutagenicity and carcinogenicity. The other two are toxicity data sets (DBPCAN and CPDBAS). DBPCAN is part of the water disinfection by-products database and contains predicted estimates of carcinogenic potential for 178 chemicals. The goal is to provide informed estimates of carcinogenic potential to be used as one factor in ranking and prioritizing future monitoring, testing, and research needs in the drinking water area [63]. The second data set is CPDBAS, the Carcinogenic Potency Database (CPDB) that contains detailed results and analyzes of 6540 chronic, long-term carcinogenesis bio assays.[9] The other two data sets used in this study were the carcinogenesis and mutagenesis mentioned above.[10]

The data sets are characterized in Table 16.1 together with the associated Aleph's parameters used in the experiments. The nodes limit parameter indicated in the table concern the sequential execution value. When running APIS same node limit was used. The background provided for the data sets were as follows. Far all data sets, the structure of each molecule (atoms and bonds) was available in the background knowledge. For the toxicity data sets (DBPCAN and CPDBAS) a set of molecular descriptors for each molecule was also available in the background knowledge.

Table 16.1 Characterization of the data sets used in the study. In the cells of the second column P/N represents the number of positive examples (P) and negative examples (N). The five right most columns are the values for Aleph's parameters

Data set name	Number of examples	Number of *islands*	Clause length	Nodes limit (Millions)	Noise	Minimum positives	Sample size
carcinogenesis	162/136	4	5	0.5	10	12	30
mutagenesis	125/63	5	6	1	4	9	25
dbpcan	80/98	37	7	1	2	5	30
cpdbas	843/966	37	6	0.1	150	150	5

[7] An approach where the activity of a compound, for example, is predicted only based on its structure features.

[8] A detailed description of this study can be found in [66]

[9] Source data for both data sets is available from the Distributed Structure-Searchable Toxicity (DSSTox) Public Database Network from the US Environmental Protection Agency http://www.epa.gov/ncct/dsstox/index.html, accessed Dec 2008.

[10] Available from the Oxford University Machine Learning repository http://www.cs.ox.ac.uk/activities/machlearn/applications.html

Table 16.2 Speedups (a) and accuracy (b) obtained in the experiments numbers in each cell correspond to average and standard deviation (in parenthesis). There is no statistical difference ($\alpha \leq 0.05$) between the sequential execution accuracy values and the parallel execution for each data set.

(a)

Data set	Number of worker nodes				
	2	4	6	7	
carcinogenesis	4.8(2.1)	5.6(2.7)	6.7(2.6)	6.1(2.6)	
mutagenesis	76.5(32.9)	138.9(82.7)	188.4(119.3)	231.3(148.6)	
dbpcan	13.8(2.5)	26.7(4.3)	36.5(5.5)	41.1 (5.9)	
cpdbas	18.3(7.0)	31.4(16.1)	36.2(26.9)	28.5(11.8)	

(b)

Data set name	Sequential execution	Number of worker nodes			
		2	4	6	7
carcinogenesis	53.7(3.8)	58.9(5.5)	57.8(3.8)	57.8(4.8)	58.0(7.6)
mutagenesis	84.1(6.9)	80.7(5.4)	82.0(4.8)	80.9(5.2)	81.3(4.7)
dbpcan	87.9(5.0)	89.8(4.1)	89.3(5.1)	89.3(5.1)	89.3(5.1)
cpdbas	54.0(1.8)	51.2(1.4)	53.6(1.2)	53.5(1.2)	53.4(1.0)

Overall, the results show that significant speedups were achieved by APIS, well beyond the number of processors (Table 16.2 (a))[11] without affecting accuracy (no statistical significant difference for $\alpha \leq 0.05$), Table 16.2 (b).

The major contribution for the speedups is, however, from the parallel search of the subspaces. We identified two sources of the parallel execution on the speedups. With enough CPUs (number of workers larger than the number of the islands) the execution time would be broadly determined by the slower subspace search. For example, in mutagenesis data set, if we have more than five CPU workers we can search the five saturation-based subspaces in parallel. The overall time is determined by the slower search. With this effect alone we would expect the speedups to be close to the speedup of the search in the slower subspace.

However, we have also noticed that the speedup of the slowest subspace search alone does not explain the global speedups obtained. In a deeper analysis, we can see that the number of "slow" subspaces (1 in mutagenesis and 3 in dbpcan, for example) than the other subspaces use less than 10 % of the time of the slower ones. That is, there are one or few "slow" subspaces and their run time is much larger than the others. This means that we can start processing the next example much earlier than the finish time of the slower subspace. In practice we can run several examples in parallel. This is also a significant contribution for the global speedup.

Another contribution, although weaker, for the speedup results is the use of intersection of coverage lists instead of theorem proving. The number of clauses evaluated

[11] Except for the carcinogenesis data set

using intersection of coverage lists is rather small (when compared with the theorem proving case) but represent also a faster method to evaluate clauses.

As seen from the above results, the APIS approach provides several "opportunities" for the parallel execution to produce very good speedups. In the next application, we describe an example of a chemoinformatics application where the number of islands that can be identified is quite large making that kind of applications adequate for the use of the APIS approach.

16.5.2 Predicting Drug Efficiency in Cancer Cells Treatment

The data used in the experiments is the result of the work done in the "Genomics of Drug Sensitivity in Cancer" project [20], and its preprocessing follows the same approach used in [34]. The original data set, publicly available in the "Genomics of Drug Sensitivity in Cancer" project website, consists of measured IC50 values for various cell lines and compound pairs. It contains 639 different cell lines, each with 77 gene mutation properties. Each cell line also has information about its microsatellite instability status (MIS), cancer type, and correspondent tissue. The IC50 value is available in its natural logarithmic form, ranging from −18.92 (6.07E-9 raw form) to 15.27 (4.28E-6 raw form). Each gene mutation is described by its sequence variation and copy number variation.

The data set contains 131 drugs that were applied to the cell lines leading to 83709 potential IC50 values. Each example was characterized with the cell line feature together with the features of the drug that was used in that cell line and the IC50 value obtained. Cell features are the cell mutation properties and drug features are molecular descriptors and fingerprints generated with the same version of PaDEL used in [34]. The final amount of cell line features was 142, and the final amount of drug features was 790, resulting in a total of 932 features plus the IC50 value. The final data set resulted in 40,691 instances.

An experiment was done using the Aleph ILP system in the classification task of predicting "good" drugs in the cell line data set described above. We have transformed the original regression problem into a binary classification problem. We have sorted the examples by their IC50 value and established a lower threshold below which examples are in class "good" and a upper threshold above which example where in class "bad". Examples in the "gray zone" between the lower and upper thresholds were discarded. The discretisation resulted in a total of 27,120 examples. The background knowledge included the molecular descriptors, fingerprints as well as the cell lines features.

Some simple rules found by Aleph include:

body of Rule 1: pubchemfp567(Compound), pubchemfp516(Compound), pubchemfp692(Compound); Positive cover = 3571, Negative cover = 77.

"If the compound molecule has the substructure O-C-C-O, the substructure [#1]-C=C-[#1], and the substructure O=C-C-C-C-C-C, then the IC50 is considered as good (98 % of the covered examples)."

body of Rule 2: pubchemfp188(Compound); Positive cover = 4069, Negative cover = 2915.

"If the compound molecule has two or more saturated or aromatic heteroatom-containing ring of size 6, then the IC50 is considered good (58 % of the covered examples)."

Aleph was able to construct very simple rules that are easily understood by the experts. The accuracy was 91 %.

The background knowledge used in the just described experiment is typical in chemoinformatics data analysis. There is a large number of molecular descriptors and fingerprints. When encoding such information in the background knowledge, a large number of [APIS] islands will be produced. These types of chemoinformatic data analysis can profit a lot from the type of parallelisation available in the APIS system.

16.6 Conclusions

In this chapter, we have discussed how ILP systems can profit from parallel execution. We have surveyed parallel and distributed executions of ILP systems. We have paid special attention to the parallel approach implemented in the APIS ILP system. A survey on parallel and distributed computation was also presented. To link ILP to parallel execution we have also discussed how a distributed system scheduler framework could be used to improve the execution of ILP systems by running in parallel several of the tasks involved in the execution of an ILP system. We have presented several applications where ILP have been successfully used and discussed how those applications can profit from a parallel approach like the one of the APIS system.

As a conclusion we may state that ILP systems have several advantages when applied to data analysis in Life Sciences domains. Those applications can profit even more if a parallel execution is used. A parallel execution can substantially improve execution time or improve the quality of the models by searching larger regions of the hypothesis space in the same time as the sequential execution. Parallel execution can be used in a very large number of parts of an ILP algorithm. We can use it at the theory-level search (coarse level) to the hypothesis space search to even use ILP with a parallel execution of the Prolog engine, for highly nondeterministic background knowledge.

References

1. Alves, A., Camacho, R., Oliveira, E.: Discovery of functional relationships in multi-relational data using inductive logic programming. In: Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 Nov 2004, Brighton, UK, pp. 319–322 (2004)
2. EC Amazon: Amazon elastic compute cloud (amazon ec2), 2010. https://aws.amazon.com/ec2/

3. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
4. Blaták J., Popelínský, L.: dRAP: A framework for distributed mining firts-order frequent patterns. In: Proceedings of the 16th Conference on Inductive Logic Programming, pp. 25–27. Springer, (2006)
5. Bone, P., Somogyi, Z., Schachte, P.: Estimating the overlap between dependent computations for automatic parallelization. TPLP **11**(4–5), 575–591 (2011)
6. Bratko, I., Muggleton, S., Varsek, A.: Learning qualitative models of dynamic systems. In: Proceedings of the Eighth International Machine Learning Workshop, San Mateo, Ca, 1991. Morgan-Kaufmann
7. Buntine, W.: Generalised subsumption and its applications to induction and redundancy. Artif. Intell. J. **36**(2):149–176 (1988). revised version of the paper that won the A.I. Best Paper Award at ECAI-86
8. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw.: Pract. Experience **41**(1):23–50 (2011)
9. Clare, A., King, R.D.: Data mining the yeast genome in a lazy functional language. In: Proceedings of the Fifth International Symposium on Practical Aspects of Declarative Languages, pp. 19–36 (2003)
10. Costa, V.S., de Castro Dutra, I., Rocha, R.: Threads and or-parallelism unified. TPLP **10**(4–6), 417–432 (2010)
11. Dasgupta, K., Mandal, B., Dutta, P., Kumar Mandal, J., Dam, S.: A genetic algorithm (ga) based load balancing strategy for cloud computing. Proc. Technol. **10**, 340–347 (2013)
12. Jeffrey, D., Sanjay, G.: Mapreduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
13. Dehaspe, L., De Raedt, L.: Parallel inductive logic programming. In: Proceedings of the MLnet Familiarization Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases (1995)
14. Delgado, J., Salah Eddin, A., Adjouadi, M., Sadjadi, S.M.: Paravirtualization for scientific computing: performance analysis and prediction. In: 2011 IEEE 13th International Conference on High Performance Computing and Communications (HPCC), pp. 536–543. IEEE (2011)
15. Fayyad, U.M., Uthurusamy, R., (eds.) In: Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95), Montreal, Canada, August 20-21, 1995. AAAI Press (1995)
16. Nuno, A., Ashwin Srinivasan, F., Silva, F.M.A., Camacho, R.: Parallel ilp for distributed-memory architectures. Mach. Learn. **74**(3), 257–279 (2009)
17. Message Passing Interface Forum: MPI: A message-passing interface standard. Technical Report UT-CS-94-230, University of Tennessee, Knoxville, TN, USA (1994)
18. Ian Foster and Carl Kesselman: The Grid 2: Blueprint for a new computing infrastructure. Elsevier (2003)
19. Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I.: Above the clouds: A berkeley view of cloud computing. Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS 28:13 (2009)
20. Garnett, M.J., Edelman, E.J., Heidorn, S.J., Greenman, C.D., Dastur, A., Lau, K.W., Patricia Greninger, I., Thompson, R., Luo, X., Soares, J., et al.: Systematic identification of genomic markers of drug sensitivity in cancer cells. Nature **483**(7391), 570–575 (2012)
21. Cloud Google: Google cloud platform. https://cloud.google.com/
22. Graham, J., Page, D., Kamal, A.: Accelerating the drug design process through parallel inductive logic programming data mining. In: Proceeding of the Computational Systems Bioinformatics (CSB'03). IEEE (2003)
23. Gupta, G., Pontelli, E., Ali, K.A.M., Carlsson, M., Hermenegildo, M.V.: Parallel execution of prolog programs: a survey. ACM Trans. Program. Lang. Syst. **23**(4), 472–602 (2001)

24. Huang, W., Liu, J., Abali, B., Panda, D.K.: A case for high performance computing with virtual machines. In: Proceedings of the 20th annual international conference on Supercomputing, pp. 125–134. ACM (2006)
25. Juve, G., Deelman, E.: Scientific workflows and clouds. Crossroads **16**(3), 14–18 (2010)
26. King, R., Muggleton S., Lewis, R., Sternberg, M.: Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. Proc. National Acad. Sci. **89**(23) (1992)
27. King, R., Sternberg, M.J.E.: A machine learning approach for the prediction of protein secondary structure. J. Mol. Biol. **216**, 441–457 (1990)
28. Konstantopoulos, S.K.: A data-parallel version of Aleph. In: Proceedings of the Workshop on Parallel and Distributed Computing for Machine Learning, co-located with ECML/PKDD'2003, Dubrovnik, Croatia, 2003
29. Krishna, P.V.: Honey bee behavior inspired load balancing of tasks in cloud computing environments. Appl. Soft Comput. **13**(5), 2292–2303 (2013)
30. Lloyd, J.W.: Foundations of Logic Programming. Springer, New York (1984)
31. Martinez-Angeles, C.A., de Castro Dutra, I., Costa, V.S., Buenabad-Chavez, J.: A datalog engine for gpus. In: Declarative Programming and Knowledge Management - Declarative Programming Days, KDPD 2013, Unifying INAP, WFLP, and WLP, Kiel, Germany, September 11-13, 2013, Revised Selected Papers, vol. 8439 of Lecture Notes in Computer Science, pp. 152–168. Springer (2013)
32. Matsui, T., Inuzuka, N., Seki, H., Itoh, H.: Comparison of three parallel implementations of an induction algorithm. In: 8th International Parallel Computing Workshop, pp. 181–188. Singapore (1998)
33. Mauch, Viktor, Kunze, Marcel, Hillenbrand, Marius: High performance cloud computing. Future Gen. Comput. Syst. **29**(6), 1408–1416 (2013)
34. Menden, M.P., Iorio, F., Garnett, M., McDermott, U., Benes, C.H., Ballester, P.J., Saez-Rodriguez, J.: Machine learning prediction of cancer cell sensitivity to drugs based on genomic and chemical properties. PloS one **8**(4), e61318 (2013)
35. Michalski, R.S.: Pattern recognition as rule-guided inductive inference. In: Proceedings of IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 349–361 (1980)
36. Mitchell, T.M.: Generalization as search. Artificial intell. **18**(2), 203–226 (1982)
37. Muggleton, S.: Inductive logic programming. New Gener. Comput. **8**(4), 295–317 (1991)
38. Muggleton, S.: Inductive logic programming: derivations, successes and shortcomings. In: Proceedings of the European Conference on Machine Learning: ECML-93, pp. 21–37, Vienna, Austria, April 1993
39. Muggleton, S.: Inverse entailment and progol. New Gener. Comput. Special issue on Inductive Logic Programming **13**(3–4), 245–286 (1995)
40. Muggleton, S.: Learning from positive data. In: Inductive Logic Programming, 6th International Workshop, ILP-96, Stockholm, Sweden, August 26-28, 1996, Selected Papers, pp. 358–376 (1996)
41. Muggleton, S., Firth, J.: Relational rule induction with CProgol4.4: a tutorial introduction. In: Džeroski, S., Lavrač, N. (eds.) Relational Data Mining, pp. 160–188. Springer (2001)
42. Muggleton, S., De Raedt, L.: Inductive logic programming: theory and methods. J. Logic Program. **19**(20), 629–679 (1994)
43. Ohwada, H., Mizoguchi, F.: Parallel execution for speeding up inductive logic programming systems. In: Proceedings of the 9th International Workshop on Inductive Logic Programming, number 1721 in LNAI, pp. 277–286. Springer (1999)
44. Ohwada, H., Nishiyama, H., Mizoguchi, F.: Concurrent execution of optimal hypothesis search for inverse entailment. In: Cussens, J., Frisch, A. (eds.) Proceedings of the 10th International Conference on Inductive Logic Programming, vol. 1866 of LNAI, pp. 165–173. Springer (2000)
45. Pacini, Elina, Mateos, Cristian, Garino, Carlos García: Distributed job scheduling based on swarm intelligence: a survey. Comput. Electr. Eng. **40**(1), 252–269 (2014)
46. Plotkin, G.D.: A note on inductive generalisation, pp. 153–163. In: Meltzer, B., Michie, D. (eds.) Edinburgh University Press, Edinburgh (1969)

47. Quinlan, J.R., Cameron-Jones, R.M.: FOIL: A midterm report. In: Brazdil, P. (ed.) Proceedings of the 6th European Conference on Machine Learning, vol. 667, pp. 3–20. Springer (1993)
48. Ramakrishnan, L., Zbiegel, P.T., Campbell, S., Bradshaw, R., Canon, R.S., Coghlan, S., Sakrejda, I., Desai, N., Declerck, T., Liu, A.: Magellan: experiences from a science cloud. In: Proceedings of the 2nd International Workshop on Scientific Cloud Computing, pp. 49–58. ACM (2011)
49. Ramezani, F., Lu, J., Hussain, F.: Task based system load balancing approach in cloud environments. In: Knowledge Engineering and Management, pp. 31–42. Springer (2014)
50. Ramezani, F., Jie, L., Hussain, F.K.: Task-based system load balancing in cloud computing using particle swarm optimization. Int. J. Parallel Programm. **42**(5), 739–754 (2014)
51. RD1, K., Muggleton, S.H., Srinivasan, A., Sternberg, M.J.: Structure-activity relationships derived by machine learning: the use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. Proc Natl Acad Sci USA, **9**(93(1)):438–42 (1996)
52. Reinaldo, F., Fernandes, C., Rahman, A., Malucelli, A., Camacho, R.: Assessing the eligibility of kidney transplant donors. In: Machine Learning and Data Mining in Pattern Recognition, 6th International Conference, MLDM 2009, Leipzig, Germany, July 23–25, 2009. Proceedings, pp. 802–809 (2009)
53. Robinson, J.A.: A machine-oriented logic based on the resolution principle. J. ACM **12**(1), 23–41 (1965)
54. Vítor, S.C., Ashwin, S., Rui, C., Hendrik, B., Bart, D., Gerda, J., Jan, S., Henk, V., Wim, V.L: Query transformations for improving the efficiency of ILP systems. J. Mach. Learn. Res. **4**, 465–491 (2003)
55. Skillicorn, David B., Wang, Yu.: Parallel and sequential algorithms for data mining using inductive logic. Knowl. Inf. Syst. **3**(4), 405–421 (2001)
56. Smith, R.G.: The contract net protocol: high-level communication and control in a distributed problem solver. IEEE Trans. Comput. **29**(12), 1104–1113 (1980)
57. Srinivasan, A.: The Aleph Manual, 2003. http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph
58. Srinivasan, A., King, R.D., Muggleton, S., Sternberg, M.J.E.: Carcinogenesis predictions using ILP. In: Inductive Logic Programming, 7th International Workshop, ILP-97, Prague, Czech Republic, Sept. 17–20, 1997, Proceedings, pp. 273–287 (1997)
59. Fonseca, N.A., Pereira, M., Santos Costa, V., Camacho, R.: Interactive discriminative mining of chemical fragments. In: Proceedings of the 2010 International Conference on Inductive Logic Programming (ILP 2010), number 6489 in Lecture Notes in Artificial Intelligence, pp. 59–66. Springer (2011)
60. White, T.: Hadoop: The Definitive Guide. O'Reilly Media, Inc. (2012)
61. Wielemaker, J.: Native preemptive threads in SWI-Prolog. In: Palamidessi, C. (ed.) Proceedings of the 19th International Conference on Logic Programming, vol. 2916 of *LNAI*, pp. 331–345. Springer (2003)
62. Wirth., R.: Learning by failure to prove. In: Proceedings Third European Working Session on Learning, pp. 237–251. London (1988) Pitman
63. Woo, Y.T., Lai, D., McLain, J.L., Manibusan, M.K., Dellarco, V.: Use of mechanism-based structure-activity relationships analysis in carcinogenic potential ranking for drinking water disinfection by-products. Environ. Health Perspect **110**((Suppl 1)), 75–87 (2002)
64. Wu, F., Wu, Q., Tan, Y.: Workflow scheduling in cloud: a survey. J. Supercomput. 1–46 (2015)
65. Xu, Y., Wu, L., Guo, L., Chen, Z., Yang, L., Shi, Z.: An intelligent load balancing algorithm towards efficient cloud computing. In: Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence (2011)
66. Zaverucha, G., Santos Costa, V., Paes, A. (eds.) Inductive logic programming—23rd International Conference, ILP 2013, Rio de Janeiro, Brazil, August 28-30, 2013, Revised Selected Papers, volume 8812 of Lecture Notes in Computer Science. Springer (2014)
67. Zhan, Z.H., Fang Liu, X., Jiao Gong, Y., Zhang, J., Shu-Hung Chung, H., Li. Y.: Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Computing Surveys (CSUR) **47**(4), 63 (2015)

Part IV
Social Media Applications

Chapter 17
Parallelization of Sparse Matrix Kernels for Big Data Applications

Oguz Selvitopi, Kadir Akbudak and Cevdet Aykanat

17.1 Introduction

There is a growing interest in scientific computing community for big data analytics. Recent approaches aim to benefit the big data analytics with the methods and techniques that are very common in the mature field of optimization for high performance computing (HPC) [20]. These efforts rely on the observation that graphs often constitute the spine of the data structures used in analyzing big data (as data is almost always sparse), and the adjacency list representation of a graph actually corresponds to a sparse matrix. Hence, analysis operations on big data can be expressed in terms of basic sparse matrix kernels. For example, the popular graph mining library PEGASUS (A Peta-Scale Graph Mining System) [17] uses an optimized sparse matrix vector multiplication kernel, called GIM-V, as the basic operation in several graph mining algorithms such as PageRank, spectral clustering, finding connected components, etc. This work focuses on efficient parallelization of two other important sparse kernels on distributed systems: sparse matrix–matrix multiplication (SpGEMM) of the form $C = AB$ and sparse matrix–dense matrix multiplication (SpMM) of the form $Y = AX$.

SpGEMM kernel finds its application in a wide range of domains such as finding all-pair shortest paths (APSP) [11], finite element simulations based on domain decomposition (e.g., finite element tearing and interconnect (FETI) [13]), molecular dynamics (e.g., CP2K [10]), computational fluid dynamics [19], climate

O. Selvitopi · K. Akbudak · C. Aykanat (✉)
Department of Computer Engineering, Bilkent University, 06800, Cankaya, Ankara, Turkey
e-mail: aykanat@cs.bilkent.edu.tr
URL: http://www.cs.bilkent.edu.tr

O. Selvitopi
e-mail: reha@cs.bilkent.edu.tr

K. Akbudak
e-mail: kadir@cs.bilkent.edu.tr

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_17

simulation [25], and interior point methods [6]. These applications necessitate efficient large-scale parallelization in order to obtain shorter running times for processing today's rapidly growing "Big Data". There exist several software packages that provide SpGEMM computation for distributed-memory architectures such as Trilinos [15] and Combinatorial BLAS [7]. Trilinos uses one-dimensional (1D) partitioning of input matrices. Matrices A and C are stationary, whereas matrix B is communicated in K stages for a parallel system with K processors. This algorithm corresponds to replicating B matrix among K processors in K stages. Combinatorial BLAS uses the parallel matrix multiplication algorithm (SUMMA [30]) based on dense matrices. The motivation of Combinatorial BLAS is large-scale graph analytic for "Big Data". It also contains scalable implementations of kernel operations such as sparse matrix–vector multiplication (SpMV) and subgraph extraction. Recently, a matrix-partitioning method based on two-constraint hypergraph partitioning is proposed in [3] for reducing total message volume during outer-product–parallel SpGEMM. [3] is known to be the first work that proposes to preprocess the sparsity patterns of the matrices in order to reduce parallelization overheads. In [3], it is also proposed that the input and output matrices can be simultaneously partitioned.

SpMM is also an important kernel and many graph analysis techniques such as centrality measures use it as a building block. Apart from its popularity in block methods in linear algebra [14, 22, 23], it is also a very basic kernel in graph analysis as several works pointed out its relation to graph algorithms [2, 8, 17, 24, 28]. In these methods, the dimensions of the dense matrices X and Y are usually very small compared to the dimensions of the sparse matrix A. The importance of this kernel is also acknowledged by vendors Intel MKL [1] and Nvidia cuSPARSE [21], being realized respectively for multi-core/many-core and GPU architectures.

Our contributions in this work are centered around partitioning models to efficiently parallelize these two kernels on distributed systems. In order to do so we aim at reducing communication overheads. The proposed model for the SpGEMM kernel aims to reduce total message volume while the proposed model for the SpMM kernel consists of two phases and it strives for reducing both total and maximum message volume. The experiments on up to 1024 processes show that scalability can be drastically improved using the proposed models.

The rest of the paper is organized as follows: Section 17.2 describes the SpGEMM kernel and the proposed model for parallelization. Section 17.3 describes the SpMM kernel and the two-phase methodology to achieve parallelization. Section 17.4 presents our experiments separately for these two kernels and Sect. 17.5 concludes.

17.2 Parallelization of the SpGEMM Kernel

We investigate efficient parallelization of the SpGEMM operation on distributed-memory architectures. The communication overhead and imbalance on computational loads of processors become significant bottleneck in large-scale parallelization. Thus, we propose an intelligent matrix-partitioning method that achieve reducing

total message volume while maintaining balance on computational loads of processors in outer-product-based parallelization of the SpGEMM operation.

In Sect. 17.2.1, the outer-product-based parallelization of the SpGEMM operation is presented. We present the hypergraph model for this outer-product-based parallelization in Sect. 17.2.2. Section 17.2.3 describes how to decode a hypergraph partition as a matrix partition.

17.2.1 Outer-Product–Parallel SpGEMM Algorithm

We consider the SpGEMM computation of the form $C = AB$. Here, A and B denote the input matrices, and C denotes the output matrix, where all of these matrices are sparse.

Outer-product–parallel SpGEMM operation uses 1D columnwise and 1D rowwise partitioning of the input A and B matrices, respectively, as follows:

$$\hat{A} = AP = [A_1 A_2 \cdots A_K] \text{ and } \hat{B} = PB = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_K \end{bmatrix}. \tag{17.1}$$

In Eq. (17.1), K denotes the number of parts, which is in turn equal to the number of processors of the parallel system, P denotes the permutation matrix obtained from partitioning. The same permutation matrix is used for reordering columns of matrix A and rows of matrix B so that outer products are performed without any computation.

According to the input matrix partitioning given in Eq. (17.1), the SpGEMM computation is performed in two steps. The first step consists of local outer-product computations performed as follows by each processor P_k:

$$C^k = A_k B_k \text{ where } k = 1, 2, \ldots, K.$$

The second step consists of summing low-rank C^k matrices, which incur communication. The following operation is performed by all processors as follows:

$$C = C^1 + C^2 + \cdots + C^K \text{ where } k = 1, 2, \ldots, K.$$

17.2.2 Hypergraph Model

We propose a hypergraph partitioning (HP) based method to reduce the total message volume that occur in the second step of the outer-product–parallel SpGEMM algorithm (Sect. 17.2.1) while maintaining balance on computation loads of outer

products performed by processors in the first step of the parallel algorithm. The objective in this partitioning is to cluster columns of matrix A and rows of matrix B that contribute to the same nonzeros of matrix C into the same parts as much as possible. In other words, the outer-product computations that contribute to the same C-matrix nonzeros are likely to be performed by the same processor without any communication.

We model an SpGEMM instance $C = AB$ as a hypergraph $\mathscr{H}(C, A, B) = (\mathscr{V}, \mathscr{N})$. $\mathscr{V}$ contains a vertex v_x for each outer product of xth column of A with xth row of B. v_x represents the task of computing this outer product without any communication in the first step of the parallel SpGEMM algorithm given in Sect. 17.2.1. $\mathscr{N}$ contains a net (hyperedge) $n_{i,j}$ for each nonzero $c_{i,j}$ of C. Net $n_{i,j}$ connects the vertices representing the outer products producing scalar partial results for $c_{i,j}$. That is,

$$Pins(n_{i,j}) = \{v_x : x \in cols(a_{i,*}) \wedge x \in rows(b_{*,j})\} \cup \{v_{i,j}\}.$$

Here, $cols(a_{i,*})$ denotes the column indices of nonzeros in row i ($a_{i,*}$), whereas $rows(b_{*,j})$ denotes the row indices of nonzeros in column j ($b_{*,j}$). Hence, $n_{i,j}$ represents the summation operation of scalar partial results to obtain final result of $c_{i,j}$ in the second step of the SpGEMM algorithm. Each vertex $v_x \in \mathscr{V}$ is associated with a weight $w(v_x)$ as follows:

$$w(v_x) = |Nets(v_x)|,$$

where $Nets(v_x)$ denotes the set of nets that connect vertex v_x. This vertex weight definition encodes the amount of computation performed for the outer products. Each net $n_{i,j} \in \mathscr{N}$ is associated with a unit weight, i.e.,

$$w(n_{i,j}) = 1.$$

This net weight definition encodes the multi-way relation between the outer products regarding a single nonzero $c_{i,j}$.

17.2.3 Decoding Hypergraph Partitioning as Matrix Partitioning

A vertex partition $\Pi(\mathscr{V}) = \{\mathscr{V}_1, \mathscr{V}_2, \ldots, \mathscr{V}_K\}$ can be used to obtain a conformal columnwise and rowwise partition of A and B. That is, $v_x \in \mathscr{V}_k$ is decoded as assigning the outer product of xth column of A with xth row of B to processor P_k. If all the pins of $n_{i,j}$ reside in the same part $\mathscr{V}_k$ (i.e., the net is uncut), the summation operation regarding $c_{i,j}$ is performed locally by P_k. Otherwise, it means that the net is cut and this summation operation is assigned to one of those processors that yield

a scalar partial result for $c_{i,j}$. Hence, Π induces a 1D partition of A and B, and a 2D nonzero-based partition of C.

In the proposed HP-based method, the partitioning constraint used while obtaining Π corresponds to maintaining balance on computational loads of processors. Each cut net incurs communication of scalar partial results. The amount of communication due to a nonzero $c_{i,j}$ is equal to one less of the number of parts in which $n_{i,j}$ has pins. This in turn corresponds to one less of the number of processors that send scalar partial results to the processor responsible for $c_{i,j}$. Thus the partitioning objective of minimizing the cutsize according to "connectivity-1" metric [9] corresponds to minimizing the total message volume.

17.3 Parallelization of the SpMM Kernel

We consider the SpMM operation of the form $Y = AX$, where A is an $n \times n$ sparse matrix and X and Y are $n \times s$ dense matrices. Whatever the context, SpMM often reveals itself as an expensive operation and hence parallelization of this kernel must be handled with care in order to squeeze the best performance out of it. In this regard, communication metrics centered around volume play a crucial role. Assuming $Y = AX$ is performed in a repeated/iterative manner, where the elements of X change in each iteration and the elements of A remain the same, the partitions on X and Y matrices should be conformable in order to avoid unnecessary communication during the parallel operations.

In a system with K processors, we consider the problem of obtaining a rowwise partition of A, where processor P_k stores the submatrix blocks $A_{k\ell}$, for $1 \leq \ell \leq K$, where size of $A_{k\ell}$ is $n_k \times n_\ell$ These submatrix blocks form the row stripe R_k and P_k is held responsible for computing $Y_k = R_k X$. Since P_k only stores X_k, it needs to receive the corresponding elements of X from other processors to compute Y_k. This necessitates point-to-point communication between processors. This scheme is called *one-dimensional row-parallel algorithm* and it consists of the following steps for any processor P_k with $1 \leq k \leq K$:

1. For each off-diagonal block $A_{\ell k}$, for $1 \leq \ell \leq K$, with at least one nonzero in it, P_k sends the respective elements of X_k to processor P_ℓ. If a_{ij} is a nonzero in this off-diagonal block, then jth row of X need to be communicated.
2. Perform computations on local submatrix A_{kk} using X_k. Local computations do not necessitate communication. Y_k is first set with the result of this computation.
3. For each off-diagonal block $A_{k\ell}$, for $1 \leq \ell \leq K$, with at least one nonzero in it, P_k receives the respective elements of X from P_ℓ in order to perform computations on the respective nonlocal submatrix block. Y_k is updated with the results of nonlocal computations and its final value is computed.

Then, with some possible dense matrix operations that involve Y, new X is computed and used in the upcoming iteration. A local submatrix block is simply the diagonal block owned by the respective processor (A_{kk}), whereas nonlocal submatrix blocks are the off-diagonal ones ($A_{k\ell}$, with $k \neq \ell$). As hinted above, computations involving local submatrix blocks can be carried out without communication, whereas the computations involving nonlocal ones may necessitate communication if they are nonempty. In the following sections, we describe how to distribute these three matrices to processors via a hypergraph partitioning model that minimizes total communication volume and another model that reduces maximum volume applied on top of the former, hence able to address two important communication cost metrics that contain message volume.

In order to parallelize the SpMM kernel, we utilize the concept of an atomic task, which signifies the smallest computational granularity that cannot further be divided, hence, an atomic task shall be executed by a single processor. In SpMM, the atomic task is defined to be the multiplication of row $a_{i,*}$ with whole X. The result of this multiplication are the elements of row $y_{i,*}$. In the hypergraph model, the atomic tasks are represented by vertices.

17.3.1 Hypergraph Model

In the hypergraph model $\mathscr{H} = (\mathscr{V}, \mathscr{N})$, the vertices represent the atomic tasks of computing the multiplication of rows of A with X, i.e., $v_i \in V$ represents $a_{i,*}X$. Note that v_i also represents the row $a_{i,*}$ as well as the computations associated with this row. The computational load of this task is the number of multiply-and-add operations. The weight of v_i is assigned the computational load associated with the corresponding multiplication, i.e.,

$$w(v_i) = nnz(a_{i,*}) \cdot nnz(X).$$

The dependencies among the computational tasks are captured by the nets. For each row of X, there exists a net n_j in the hypergraph and it captures the dependency of the computational tasks to the row j of X. In other words, n_j connects the vertices corresponding to the tasks that need row j of X in their computations. Hence, the vertices connected by n_j are given by

$$Pins(n_j) = \{v_i : a_{ij} \neq 0\}.$$

n_j effectively represents column j of A as well. Note that $|Pins(n_j)| = nnz(a_{*,j})$, where $nnz(\cdot)$ denotes the number of nonzeros in a row or column of matrix. Since the number of elements in a row of X is s, n_j is associated with a cost $c(n_j)$ proportional to s. In other words,

$$c(n_j) = s.$$

This quantity signifies the number of elements required by the computational tasks corresponding to the vertices in $Pins(n_j)$. As a result, there are m vertices, n nets and $nnz(A)$ pins in $\mathscr{H}$. This model is a simple extension of the model used for sparse matrix vector multiplication [9].

17.3.2 Partitioning and Decoding

The formed hypergraph $\mathscr{H} = (\mathscr{V}, \mathscr{N})$ is then partitioned into K vertex parts to obtain $\Pi = \{\mathscr{V}_1, \ldots, \mathscr{V}_K\}$. Without loss of generality, the set of rows corresponding to the vertices in $\mathscr{V}_k$ and the respective computations involving these rows are assigned to processor P_k. A net n_j in the cut necessitates communication of the elements of row j of X and this communication operation involves the processors corresponding to the parts in the connectivity set of this net. Specifically, if $\mathscr{V}_k \in \Lambda(n_j)$ is the owner of row j of X, then it needs to send this row to the remaining processors, i.e., to each processor P_ℓ such that $\mathscr{V}_\ell \in \Lambda(n_j) - \{\mathscr{V}_k\}$, amounting to a volume of $s \cdot (|\Lambda(n_j)| - 1)$, s being the number of elements in row j. An internal net does not necessitate communication as all the rows corresponding to the vertices connected by this net belong to the same processor. The objective of minimizing cutsize according to the connectivity metric [9] in the partitioning hence encodes the total volume of communication. The constraint of maintaining balance on the vertex part weights corresponds to maintaining balance on the computational loads of the processors.

Aforementioned formulation strives for reducing total communication volume. However, the high volume overhead of SpMM kernel makes another related volume-related metric maximum communication volume also important, which we address next.

17.3.3 A Volume-Balancing Extension for SpMM

The formulation used to balance the volume loads of processors is based on a hypergraph model. This model was used to address multiple communication cost metrics regarding one- and two-dimensional sparse matrix vector multiplication successfully [26, 27, 29]. Although the main objective of this model is to reduce the latency overhead by aiming to minimize total message count, another important aspect of it is to maintain a balance on communication volume. In our case, i.e., for SpMM, the latter proves to be more crucial. Note that maintaining a balance on volume loads of processors corresponds to providing an upper bound on the same metric. We extend this model for SpMM. For this model, it is assumed a partition information is inherent, such as the one obtained in the previous section—which is also utilized for this model.

Using the partitioning information $\Pi = \{\mathscr{V}_1, \ldots, \mathscr{V}_k\}$ obtained on $\mathscr{H} = (\mathscr{V}, \mathscr{N})$, we form another hypergraph $\mathscr{H}^C = (\mathscr{V}^C, \mathscr{N}^C)$ to summarize the communication

operations due to this partitioning. Recall that a net n_j necessitates communication if it is cut in Π. This communication operation is represented by a vertex in $\mathscr{H}^C$. In other words, there exists a vertex $v_j^C \in \mathscr{V}^C$ for each cut net $n_j \in \mathscr{N}$

$$\mathscr{V}^C = \{v_j^C : |\Lambda(n_j)| > 1\}.$$

There exists a net $n_k^C \in \mathscr{N}^C$ for each processor corresponding to the parts in Π. Hence, there are K nets in $\mathscr{H}^C$. v_j^C is connected to each net corresponding to the processor that participate in the communication operation represented by v_j^C. Hence,

$$Nets(v_j^C) = \{n_k^C : \mathscr{V}_k \in \Lambda(n_j)\}.$$

The vertices connected by n_k^C correspond to the communication operations that P_k participates in

$$Pins(n_k^C) = \{v_j^C : Pins(n_j) \cap \mathscr{V}_k \neq \emptyset\} \cup v_f^C.$$

Net n_k^C connects another vertex v_f^C, which is fixed to $\mathscr{V}_k^C$ in partitioning and later used to decode the assignment of communication operations to processors. Here, the important point is the assignment of vertex weights in $\mathscr{H}^C$ as they signify the volume incurred in communication operations. The volume incurred in communicating a row j of X is already described in the previous section and here the vertex weight is set accordingly to this quantity

$$w(v_j^C) = s \cdot (|\Lambda(n_j)| - 1).$$

The nets are assigned unit costs

$$c(n_j^C) = 1.$$

Now consider a partition $\Pi^C = \{\mathscr{V}_1^C, \ldots, \mathscr{V}_K^C\}$ of $\mathscr{H}^C$. This vertex partition induces a distribution of communication operations, where the responsibility of the communication operations represented by the vertices in $\mathscr{V}_k^C$ are assigned to processor P_k without loss of generality. A net n_j^C in the cut necessitates messages between the respective processors. In other words, if the fixed vertex v_f^C connected by this net is in part $\mathscr{V}_k^C$, this implies P_k will receive a message from each of the processors corresponding to the vertex parts in $\Lambda(n_j^C) - \{\mathscr{V}_k^C\}$. Hence, the number of messages incurred by this net is equal to $|\Lambda(n_j^C) - \{\mathscr{V}_k^C\}|$, which can be at most $K - 1$ as there are K vertex parts. In the partitioning, the objective of minimizing cutsize according to the connectivity metric [9] hence encodes the total message count. As a part weight indicates the amount of volume that the respective processor is responsible for, the constraint of maintaining balance corresponds to maintaining balance on the volume loads of the processors, and thus bounding the maximum volume.

With the partitioning of $\mathscr{H}$, we address total volume and with the partitioning of $\mathscr{H}^C$, we address total message count and maximum volume. By making use of respective models, we are able to address communication cost metrics that are important for SpMM.

17.4 Experiments

17.4.1 SpGEMM

17.4.1.1 Dataset

We include sparse matrices from Stanford Network Analysis Project (SNAP) [18] and the Laboratory for Web Algorithmics (LAW) [4, 5]. These matrices are downloaded from the University of Florida Sparse Matrix Collection [12] and their properties are given in Table 17.1. These matrices commonly represent the adjacency matrices of road networks or web-related graphs such as relations between products, etc. On such graphs, APSP algorithms can be used to find distances among the entities according to a predetermined metric. In this work, we only consider squaring the original adjacency matrix once, as a representative of the APSP algorithm [11].

Table 17.1 Properties of input and output matrices

Matrix	Input matrices							Output matrix
	Number of			nnz in row		nnz in column		
	rows	columns	nonzeros	avg	max	avg	max	nnz
Road networks								
belgium_osm	1,441,295	1,441,295	3,099,940	2	10	2	10	5,323,073
luxembourg_osm	114,599	114,599	239,332	2	6	2	6	393,261
netherlands_osm	2,216,688	2,216,688	4,882,476	2	7	2	7	8,755,758
roadNet-CA	1,971,281	1,971,281	5,533,214	3	12	3	12	12,908,450
roadNet-PA	1,090,920	1,090,920	3,083,796	3	9	3	9	7,238,920
roadNet-TX	1,393,383	1,393,383	3,843,320	3	12	3	12	8,903,897
Web-related graphs								
144	144,649	144,649	2,148,786	15	26	15	26	10,416,087
amazon-2008	735,323	735,323	5,158,388	7	10	7	1,076	25,366,745
amazon0505	410,236	410,236	3,356,824	8	10	8	2,760	16,148,723
amazon0601	403,394	403,394	3,387,388	8	10	8	2,751	16,258,436

nnz: number of nonzeros

17.4.1.2 Experimental Setup

In order to verify the validity of the proposed parallelization method, an SpGEMM code is implemented and run on a BlueGene/Q system, named Juqueen. For partitioning the hypergraphs, PaToH [9] is used with default parameters except the allowed imbalance ratio, which is set to be equal to 10 %.

As a baseline algorithm, we implemented a binpacking-based method which only considers computational load balancing. This method adapts the best-fit-decreasing heuristic used in K-feasible binpacking problem [16]. The outer-product tasks are assigned to one of K bins in decreasing order of the number of scalar multiplications incurred by the outer products. The best-fit criterion is assigning the task to the minimally loaded bin, whereas the capacity constraint is not used in this method.

17.4.1.3 Results

The performances of the proposed HP-based method and the baseline binpacking-based method are compared in terms of speedups in Figs. 17.1 and 17.2. As seen in these figures, in all cases, the proposed HP-based method performs substantially better than the baseline method. Thus, this improvement verifies the benefit of reducing total message volume. As seen in the figures, the parallel efficiency of HP-based method remains above 20 % in almost all instances.

17.4.2 SpMM

17.4.2.1 Dataset

We test 10 matrices for SpMM. The properties of these matrices are given in Table 17.2. These sparse matrices are also from Stanford Network Analysis Project (SNAP) [18] and the Laboratory for Web Algorithmics (LAW) [4, 5], and they are all downloaded from the University of Florida Sparse Matrix Collection [12]. A common operation that can be performed on these matrices for any kind of graph analysis is the execution of the breadth-first search from multiple sources. This corresponds to multiple SpMM iterations.

17.4.2.2 Experimental Setup

We tested our model for SpMM on SuperMUC supercomputer, which runs IBM System x iDataPlex servers. A node on this system consists of two Intel Xeon Sandy Bridge-EP processors clocked at 2.7 GHz and has 32 GB of memory. The communication interconnect is based on Infiniband FDR10.

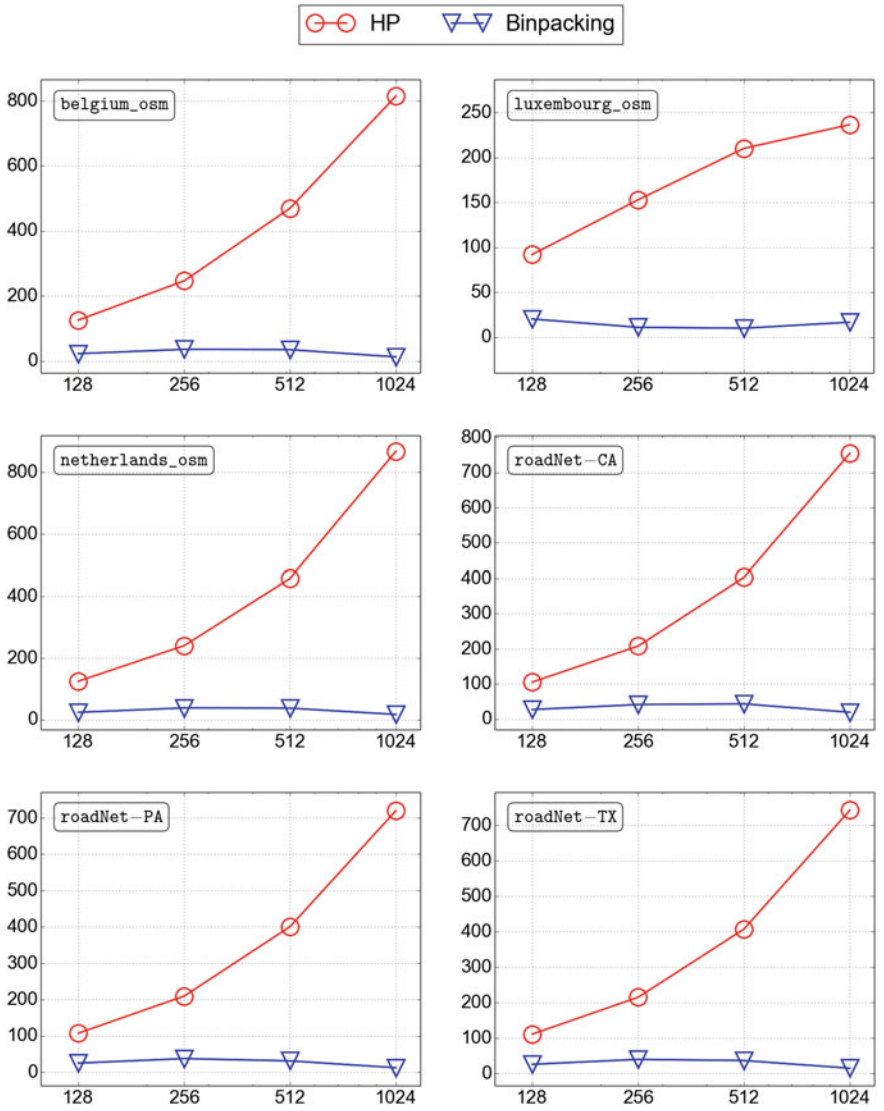


Fig. 17.1 Speedup curves comparing performances of the proposed and baseline SpGEMM algorithms on road networks

We test two models in our experiments. This first is the plain hypergraph partitioning for SpMM described in Sects. 17.3.1 and 17.3.2. This model aims at only reducing communication volume and is abbreviated with HP. The second model we test is the one that extends the HP as described in Sect. 17.3.3. This model aims at reducing communication volume and balancing communication volume in two separate phases and is abbreviated with HPVB. The dimension of the dense matrices is used as

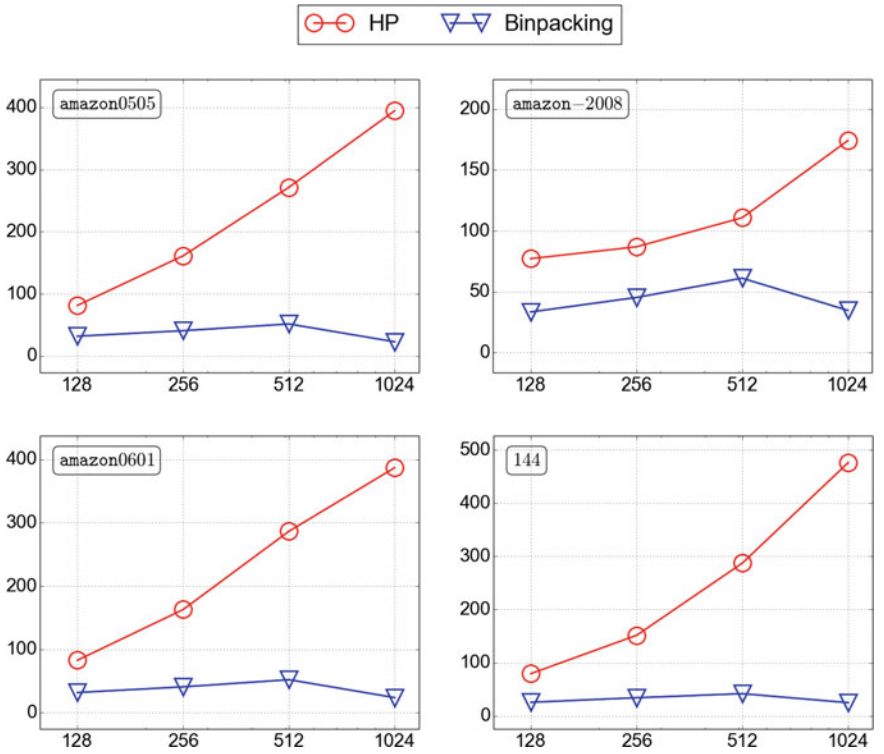


Fig. 17.2 Speedup curves comparing performances of the proposed and baseline SpGEMM algorithms on web link matrices

Table 17.2 Properties of matrices tested for SpMM

Name	Kind	#rows/cols	#nonzeros
amazon_2008	Amazon book similarity graph	735,323	5,158,388
amazon0312	Amazon product co-purchasing network	400,727	3,200,440
eu-2005	crawl of the .eu domain	862,664	19,235,140
roadNet-CA	road network	1,971,281	5,533,214
roadNet-PA	road network	1,090,920	3,083,796
roadNet-TX	road network	1,393,383	3,843,320
333SP	car mesh	3,712,815	22,217,266
asia_osm	road network	11,950,757	25,423,206
coPapersCiteseer	citation network	434,102	32,073,440
coPapersDBLP	citation network	540,486	30,491,458

$s = 5$. SpMM kernel is repeated in a loop for certain number of times and a warming up phase is included for healthier timing. We test for $K \in \{64, 128, 256, 512, 1024\}$ processors.

17.4.2.3 Partitioning and Runtime Results

We present the partitioning and runtime results in Table 17.3 for $K = 512$ and $K = 1024$. There are two communication cost metrics we consider and display in the table: total volume, indicated via "TV" column and volume load imbalance as percent, indicated via "VI (%)" column. Communication volume is in terms of communicated words and volume imbalance is on the send volumes of processors. The time of a single SpMM obtained by the two compared methods is given under the "runtime" column and it is in terms of microseconds. The lower runtimes for a specific test case are indicated via bold text.

As seen in the table, HPVB obtains significant improvements over HP in communication volume load imbalance. At $K = 512$ processors, it achieves up to $8\times$ improvement and at $K = 1024$ processors it achieves up to $9\times$ improvement. It

Table 17.3 Partitioning and runtime results for SpMM for $K = 512$ and $K = 1024$

K	Matrix	TV		VI (%)		Runtime (us)	
		HP	HPVB	HP	HPVB	HP	HPVB
512	amazon-2008	563,327	800,380	209	61	1855	**1195**
	amazon0312	324,433	447,455	155	56	1340	**948**
	eu-2005	320,741	428,054	438	57	1205	**751**
	roadNet-CA	33,764	51,524	120	37	275	**266**
	roadNet-PA	29,944	45,996	83	34	124	**120**
	roadNet-TX	28,901	44,124	120	37	181	**177**
	333SP	132,105	207,317	95	31	**713**	729
	asia_osm	10,149	15,517	451	58	**1213**	1236
	coPapersCiteseer	832,267	1,078,184	159	50	2489	**1724**
	coPapersDBLP	1,863,018	2,240,814	128	48	3860	**3082**
1024	amazon-2008	649,869	908,269	197	57	1977	**1049**
	amazon0312	383,068	518,843	168	181	1310	**1132**
	eu-2005	485,614	623,844	479	92	1325	**1174**
	roadNet-CA	52,915	81,568	111	36	**125**	128
	roadNet-PA	44,564	69,284	121	34	71	**65**
	roadNet-TX	44,288	67,460	134	40	**158**	160
	333SP	208,154	324,505	73	31	395	**374**
	asia_osm	17,971	27,222	523	58	530	**504**
	coPapersCiteseer	1,021,506	1,277,537	213	55	1797	**959**
	coPapersDBLP	2,112,593	2,522,300	233	52	3737	**1959**

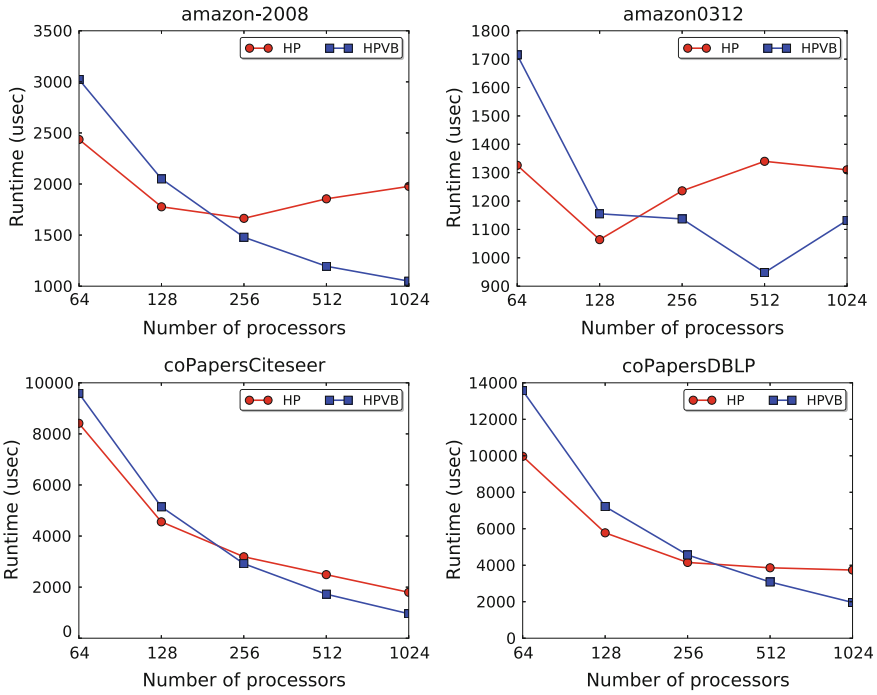


Fig. 17.3 Runtimes for SpMM

increases total volume compared to HP, causing an increase up to 57 and 56 % at 512 and 1024 processors, respectively. However, the reduction in maximum volume proves to be more vital for performance as HPVB almost always performs superior to HP as seen from the runtimes. Out of 20 instances at $K = 512$ and $K = 1024$ processors, HPVB obtains lower runtimes in 16 of them.

We present the obtained speedups in four test matrices in Fig. 17.3. As seen from the figure HPVB scales better than HP as its performance usually gets better compared to HP with increasing number of processors.

17.5 Conclusion

We described efficient parallelization of two important sparse matrix kernels for distributed systems that frequently occur in big data applications: sparse matrix–matrix multiplication and sparse matrix–dense matrix multiplication. For these two kernels, we proposed partitioning models in order to reduce the communication overheads and hence improve scalability. Our experiments show that efficient parallel performance

for big data analysis on distributed systems requires careful data partitioning models and methods that are capable of exploiting certain communication cost metrics.

Acknowledgments This work was supported by The Scientific and Technological Research Council of Turkey (TUBITAK) under Grant EEEAG-115E212. This article is also based upon work from COST Action IC1406 (cHiPSet).

References

1. Intel math kernel library (2015). https://software.intel.com/en-us/intel-mkl
2. Agarwal, V., Petrini, F., Pasetto, D., Bader, D.A.: Scalable graph exploration on multicore processors. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, SC '10, pp. 1–11. IEEE Computer Society, Washington, DC, USA (2010). doi:10.1109/SC.2010.46
3. Akbudak, K., Aykanat, C.: Simultaneous input and output matrix partitioning for outer-product–parallel sparse matrix-matrix multiplication. SIAM J. Sci. Comput. **36**(5), C568–C590 (2014). doi:10.1137/13092589X
4. Boldi, P., Rosa, M., Santini, M., Vigna, S.: Layered label propagation: a multiresolution coordinate-free ordering for compressing social networks. In: Srinivasan S., Ramamritham K., Kumar A., Ravindra M.P., Bertino E., Kumar R. (eds.) Proceedings of the 20th International Conference on World Wide Web, pp. 587–596. ACM Press (2011)
5. Boldi, P., Vigna, S.: The WebGraph framework I: compression techniques. In: Proceedings of the Thirteenth International World Wide Web Conference (WWW 2004), pp. 595–601. ACM Press, Manhattan (2004)
6. Boman, E., Devine, K., Heaphy, R., Hendrickson, B., Heroux, M., Preis, R.: LDRD report: Parallel repartitioning for optimal solver performance. Tech. Rep. SAND2004–0365, Sandia National Laboratories, Albuquerque, NM (2004)
7. Buluç, A., Gilbert, J.R.: Parallel sparse matrix-matrix multiplication and indexing: implementation and experiments. SIAM J. Sci. Comput. (SISC) **34**(4), 170–191 (2012). doi:10.1137/110848244; http://gauss.cs.ucsb.edu/~aydin/spgemm_sisc12.pdf
8. Buluç, A., Madduri, K.: Parallel breadth-first search on distributed memory systems. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC '11, pp. 65:1–65:12. ACM, New York, NY, USA (2011). doi:10.1145/2063384.2063471; http://doi.acm.org/10.1145/2063384.2063471
9. Catalyurek, U.V., Aykanat, C.: Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. IEEE Trans. Parallel Distrib. Syst. **10**(7), 673–693 (1999)
10. CP2K: CP2K home page (Accessed at 2015). http://www.cp2k.org/
11. D'Alberto, P., Nicolau, A.: R-kleene: A high-performance divide-and-conquer algorithm for the all-pair shortest path for densely connected networks. Algorithmica **47**(2), 203–213 (2007). doi:10.1007/s00453-006-1224-z
12. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. ACM Trans. Math. Softw. (TOMS) **38**(1), 1 (2011)
13. Dostál, Z., Horák, D., Kučera, R.: Total FETI-an easier implementable variant of the FETI method for numerical solution of elliptic PDE. Commun. Numer. Meth. Eng. **22**(12), 1155–1162 (2006)
14. Feng, Y., Owen, D., Peri, D.: A block conjugate gradient method applied to linear systems with multiple right-hand sides. Comput. Meth. Appl. Mech. Eng. **127**(14), 203–215 (1995). http://dx.doi.org/10.1016/0045-7825(95)00832-2; http://www.sciencedirect.com/science/article/pii/0045782595008322

15. Heroux, M.A., Bartlett, R.A., Howle, V.E., Hoekstra, R.J., Hu, J.J., Kolda, T.G., Lehoucq, R.B., Long, K.R., Pawlowski, R.P., Phipps, E.T., et al.: An overview of the Trilinos project. ACM Trans. Math. Softw. (TOMS) **31**(3), 397–423 (2005)
16. Horowitz, E., Sahni, S.: Fundamentals of Computer Algorithms. Computer Science Press (1978)
17. Kang, U., Tsourakakis, C.E., Faloutsos, C.: Pegasus: A peta-scale graph mining system implementation and observations. In: Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, ICDM '09, pp. 229–238. IEEE Computer Society, Washington, DC, USA (2009). doi:10.1109/ICDM.2009.14
18. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data (2014)
19. Marion-Poty, V., Lefer, W.: A wavelet decomposition scheme and compression method for streamline-based vector field visualizations. Comput. Graphics **26**(6), 899–906 (2002). doi:10.1016/S0097-8493(02)00178-4; http://www.sciencedirect.com/science/article/pii/S0097849302001784
20. Mattson, T., Bader, D., Berry, J., Buluc, A., Dongarra, J., Faloutsos, C., Feo, J., Gilbert, J., Gonzalez, J., Hendrickson, B., Kepner, J., Leiserson, C., Lumsdaine, A., Padua, D., Poole, S., Reinhardt, S., Stonebraker, M., Wallach, S., Yoo, A.: Standards for Graph Algorithm Primitives. ArXiv e-prints (2014)
21. NVIDIA Corporation: CUSPARSE library (2010)
22. O'Leary, D.P.: The block conjugate gradient algorithm and related methods. Linear Algebra Appl. **29**(0), 293–322 (1980). http://dx.doi.org/10.1016/0024-3795(80)90247-5; http://www.sciencedirect.com/science/article/pii/0024379580902475. Special Volume Dedicated to Alson S. Householder
23. O'Leary, D.P.: Parallel implementation of the block conjugate gradient algorithm. Parallel Comput. **5**(12), 127–139 (1987). http://dx.doi.org/10.1016/0167-8191(87)90013-5; http://www.sciencedirect.com/science/article/pii/0167819187900135. Proceedings of the International Conference on Vector and Parallel Computing-Issues in Applied Research and Development
24. Sarıyuce, A.E., Saule, E., Kaya, K., Çatalyurek, U.V.: Regularizing graph centrality computations. J. Parallel Distrib. Comput. **76**(0), 106–119 (2015). http://dx.doi.org/10.1016/j.jpdc.2014.07.006; http://www.sciencedirect.com/science/article/pii/S0743731514001282. Special Issue on Architecture and Algorithms for Irregular Applications
25. Sawyer, W., Messmer, P.: Parallel grid manipulations for general circulation models. In: Parallel Processing and Applied Mathematics. Lecture Notes in Computer Science, vol. 2328, pp. 605–608. Springer, Berlin (2006)
26. Selvitopi, O., Aykanat, C.: Reducing latency cost in 2D sparse matrix partitioning models. Parallel Comput. **57**, 1–24 (2016). http://dx.doi.org/10.1016/j.parco.2016.04.004; http://www.sciencedirect.com/science/article/pii/S0167819116300138
27. Selvitopi, R.O., Ozdal, M.M., Aykanat, C.: A novel method for scaling iterative solvers: avoiding latency overhead of parallel sparse-matrix vector multiplies. IEEE Trans. Parallel Distrib. Syst. **26**(3), 632–645 (2015). doi:10.1109/TPDS.2014.2311804
28. Shi, Z., Zhang, B.: Fast network centrality analysis using gpus. BMC Bioinf. **12**(1), 149 (2011). doi:10.1186/1471-2105-12-149
29. Uçar, B., Aykanat, C.: Encapsulating multiple communication-cost metrics in partitioning sparse rectangular matrices for parallel matrix-vector multiplies. SIAM J. Sci. Comput. **25**(6), 1837–1859 (2004). doi:10.1137/S1064827502410463
30. Van De Geijn, R.A., Watts, J.: Summa: scalable universal matrix multiplication algorithm. Concurrency-Pract. Experience **9**(4), 255–274 (1997)

Chapter 18
Delivering Social Multimedia Content with Scalability

Irene Kilanioti and George A. Papadopoulos

18.1 Introduction

CDNs aim at overcoming Internet issues and ensuring smooth and transparent content delivery. They principally replicate content in locations as near as possible to the user that is bound to consume it. CDNs handle altogether the major issues of (i) the most efficient placement of surrogate servers in terms of high performance and less infrastructure cost; (ii) the best content diffusion placement, namely the decision of which content will be copied in the surrogate servers and to what extent; and (iii) the temporal diffusion, related to the most efficient timing of the content placement [15]. They are, however, very dissimilar in terms of the services provided and their geographic coverage. The optimization of their overall efficiency, as far as user is concerned, is practically achieved with the automatic detection of the medium (either computer or mobile -smartphone/tablet), optimized management of the browser cache, server load-balancing, the consideration of specific nature of the content of the media provider (video on demand, live videos, geo-blocked content, etc.,) or features of certain operators, such as real-time compression, session management, etc.

Utilization of CDNs is likely to have profound effects on large data download through enhanced performance, scalability, and cost reduction. Extended use of OSNs, and the increasing popularity of streaming media are the factors that drive the HTTP traffic growth [4]. The amount of traffic generated on a daily basis by online multimedia streaming providers is multiplied by the transmission over OSNs (with more than 400 tweets per minute including a YouTube video link [3] being

I. Kilanioti (✉) · G.A. Papadopoulos (✉)
Department of Computer Science, University of Cyprus, 1 University Avenue,
P.O. Box 20537, 2109 Nicosia, Cyprus
e-mail: ekoila01@cs.ucy.ac.cy

G.A. Papadopoulos
e-mail: george@cs.ucy.ac.cy

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_18

published per minute). Subsequently, CDN users can benefit from an incorporated mechanism of social awareness over the CDN infrastructure. In [15, 16] Kilanioti and Papadopoulos introduce a dynamic mechanism of preactive copying of content to an existing validated CDN simulation tool and propose an efficient copying policy based on prediction of demand on OSNs along with its variations.

18.1.1 A Toy Example of Our Approach

Let us consider Bob, located in London and assigned to the London CDN servers of an OSN service. Most of Bob's social friends are geographically close to him, but he also has a few friends in Europe and Australia assigned to their nearest servers. Bob logs into the OSN and posts a video that he wants to share. Pushing the video content to all other geographically distributed servers immediately before any requests occur would be the naive way to ensure that this content is as close as possible to all users. Aggregated over all users, pushing can lead to traffic congestion, and users would experience latency in accessing the content, which, moreover, could not be consumed at all. The problem of caching would be intensified when Alice, the only friend of Bob in Athens, would be interested in that content, and with many such Alices in various places.

Rather than pushing data to all surrogates, we can proactively distribute it only to friends of Bob likely to consume it and only at the time window that signifies a non-peak time for the upload in London area and a non-peak-time for the download in Athens area, thus taking advantage of the timezone differences of our geo-diverse system. The content will be copied only under certain conditions (content with high viewership within the media service, copied to geographically close timezones where the user has mutual friends with high influence impact). This would contribute to smaller response times for the content to be consumed (for the users) and lower bandwidth costs (for the OSN provider).

18.1.2 Contributions

In this work we modify the Social Prefetcher algorithm [15, 16] to incorporate best performing caching mechanisms. We conduct experiments over a large corpus of YouTube videos and use Twitter, where information propagates via retweeting across multiple hops in the network [19]. Social cascades are directly analyzed, as the real dataset of User Generated Content (UGC) used includes multimedia links over the OSN. The Twitter dataset contains geographic locations, follower lists and tweets for 37 million users, spreading of more than one million YouTube videos over this network, a corpus of more than 2 billions messages and approximately 1.3 million single messages with an extracted video URL. The wide popularity and massive user base of YouTube and Twitter allow us to obtain safe insights regarding user navigation

behavior on other similar media and microblogging platforms, respectively. The implemented variations are incorporated in a validated simulator for CDNs, and restrictions of the CDN infrastructure (storage issues, network topology) are taken into account.

The present work goes beyond the Social Prefetcher algorithm [16] in terms of performance. The latter surpasses performance improvement of similar works in pull-based methods, that are employed by most CDNs, whereas moreover uses more refined topology of data centers and does not neglect storage issues. Storage costs are still a significant challenge despite the proliferation of cloud computing. In this work we examine which caching schemes in the surrogate server affect the CDN metrics the most. We further enhance the performance of Social Prefetcher algorithm, an optimization analysis of which is presented by the authors in [15]. The findings of present work can be exploited for future policies complementary to existing CDN solutions or incorporated to OSN providers mechanisms, to handle larger scale data.

The rest of this paper is structured as follows: Sect. 18.2 reviews previous related work. Section 18.3 formally describes the addressed problem. The proposed algorithm is described in Sect. 18.4. Section 18.5 gives an outline of the methodology, along with the preparation of the employed datasets. Our main findings are presented in Sect. 18.6. Section 18.7 concludes the paper and discusses directions for future work.

18.2 Related Work

Systems that leverage information from OSNs with various research goals, such as the decision for copying content or improvement of policy for temporary caching, include [20, 21, 23]. Traverso et al. [23] improve QoS by exploiting time differences among sites and the access patterns that users follow. Sastry et al. [20] analyze social cascades and access to social profiles via a third-party page.

As long as the behavior of users in different media services is concerned, the traffic on YouTube is described in several studies [4, 10–12, 18], with emphasis on the characteristics of media content, such as file size, bitrate, usage patterns, and popularity. In [11], the authors study the YouTube workload to discover that there are many similarities between traditional Web and media streaming workloads. The authors in [7] find a strong correlation among YouTube videos, because the links to related videos generated by uploaders depict small-world characteristics. In [9] the authors analyze how the popularity of individual YouTube videos evolves.

Authors in [16] extend the Social Prefetcher algorithm proposed in [15] to include information about peak-time of various timezones of a geo-diverse system, as well as contextual information about the viewership of video content within the media service. The basic algorithm gives a near-optimal solution to the problem of content delivery and addresses memory usage issues related to the very large graph dataset accommodated. The suggested mechanism added to a CDN simulator overcomes the testing limitations of other existing CDN platforms, such as the blackbox treatment of CDN policies or the need for the participation of third users.

18.3 Problem Description

We aim at mitigating the inherent Internet performance issues by improving the CDN infrastructure mechanisms. We aim at the reduction of the response time for the user, increase of the hit ratio of our request, as well as restriction of the cost of copying from the origin server to surrogate servers. We consider the network topology, the server location, and restrictions in the cache capacity of the server. Taking as input data from OSNs and actions of users over them, we want to recognize objects that will eventually be popular in the realm of the OSN platform.

We search a policy such that given a graph $G(V, E)$, a set of R regions, where the nodes of the social network are distributed, and the posts P of the nodes, it will recognize the set of objects O that will be popular only in a subset of the regions (Table 18.1), where the content is likely to be copied. The policy is represented by the function $Put(n_i, Predict(G, P, R, O))$, which takes as input a surrogate server $n_i \in N$ and the results of function *Predict* (set of g objects that will be globally popular and λ objects that will be locally popular), such that

$$\frac{Q_{hit}}{Q_{total}} \tag{18.1}$$

is maximum, whereas constraint

$$\sum_{\forall i \in O} S_i f_{ik} \leq C_k \tag{18.2}$$

is fulfilled, where:

$$f_{ik} = \begin{cases} 1 & \text{if object } i \text{ exists in the cache of surrogate server } k \\ 0 & \text{if object does not exist} \end{cases} \tag{18.3}$$

Function $Put(n_i, Predict(G, P, R, O))$ returns the set of objects $o \in O$ that have to be placed in surrogate server $n_i \in N$.

18.4 Proposed Dynamic Policy

The proposed algorithm encompasses an algorithm for each new request arriving in the CDN and an algorithm for each new object in the surrogate server. Internally, the module communicates with the module processing the requests and each addressed server separately (Fig. 18.1).

Table 18.1 Notation overview

$G(V, E)$	Graph representing the social network
$V = \{V_1, \ldots, V_n\}$	Nodes representing the social network users
$E = \{E_{11}, \ldots, E_{1n}, \ldots, E_{nn}\}$	Edges representing the social network connections, where E_{ij} stands for friendship between i and j
$R = \{r_1, r_2, \ldots, r_\tau\}$	Regions set
$N = \{n_1, n_2, \ldots, n_u\}$	The surrogate servers set. Every surrogate server belongs to a region r_i
$C_i, i \in N$	Capacity of surrogate server i in bytes
$O = \{o_1, o_2, \ldots, o_w\}$	Objects set (videos), denoting the objects users can ask for and share
$S_i, o_i \in O$	Size of object i in bytes
i_κ	Object accessed at the κ-th iteration, κ: the counter maintained and incremented each time there is a request for an object
ΔT_{i_κ}	Number of accesses since the last time object i was accessed
Π_i	Popularity of object i, $i \in O$
$q_i = \{t, V_\psi, o_x\}, 1 < x < w, 1 < \psi < n$	Request i consists of a timestamp, the id of the user that asked for the object, and the object id
$P = \{p_{12}, p_{13}, \ldots, p_{nw}\}$	User posts in the social network, where p_{ij} denotes that node i has shared object j in the social network
$pts_i, pte_i, 1 < i < \tau$	peak time start and peak time end for each region in secs
$Q = \{q_1, q_2, \ldots, q_\zeta\}$	Object requests from page containing the media objects, where q_i denotes a request for an object of set O
Q_{hit}, Q_{total}	Number of requests served from surrogate servers of the region of the user/total number of requests
$X, Y \in R$	Closest timezones with mutual followers/with highest centrality metric (HITS) values

18.4.1 For Every New Request in the CDN

The main idea is to check whether specific time has passed after the start of the cascade, and then define to what extent the object will be copied. Initially, we check whether it is the first appearance of the object. The variable *o.timestamp* depicts the timestamp of the last appearance of the object in a request and helps in calculating the timer related to the duration of the cascade. If it is the first appearance of the object, the timer for the object cascade is initialized and *o.timestamp* takes the value of the timestamp of the request. If the cascade is not yet complete (its timer has not surpassed

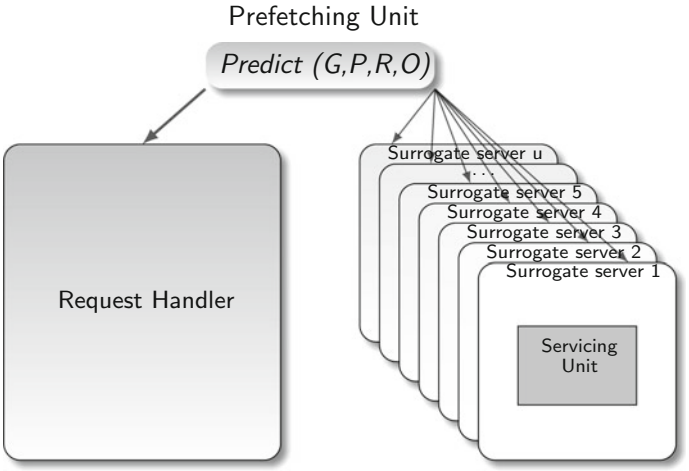


Fig. 18.1 The prefetching unit

a threshold), we check the importance of the user applying the Hubs Authorities (HITS) algorithm and checking its authority score, as well as the viewership of the object in the media service platform (Fig. 18.2).

For users with a high authority score, we copy the object to all surrogate servers of the user's timezone and to the surrogate servers serving the timezones of all

```
1:  if o.timestamp == 0 then
2:      o.timer = 0;
3:      o.timestamp = request_timestamp;
4:  else if o.timestamp != 0 then
5:      o.timer = o.timer + (request_timestamp - o.timestamp);
6:      o.timestamp = request_timestamp;
7:  end if
8:  if o.timer > time_threshold then
9:      o.timer = 0;
10:     o.timestamp = 0;
11: else if o.timer < time_threshold and user.authority_score > authority_threshold then
12:     copy object o to surrogate that serves user's V_i timezone;
13:     for all user V_y that follows user V_i do
14:         find surrogate server n_j that serves V_y's timezone;
15:         copy object o to n_j;
16:     end for
17: else if o.timer < time_threshold and o.Π_i > Π_i_threshold then
18:     copy object o to surrogates n_j that Subpolicy I decides;
19: end if
```

Fig. 18.2 Algorithm for every new request (*timestamp*, V_i, o) in the CDN

1: find X timezones where (user V_i has mutual followers **and** they are closer to user's V_i timezone);
2: find the $Y \subseteq X$ that (belong to X **and** have the highest HITS score);
3: **for all** timezones that belong to Y **do**
4: find surrogate server n_j that serves timezone;
5: copy object o to n_j;
6: **end for**

Fig. 18.3 Subpolicy I

followers of the user (global prefetching). Otherwise, selective copying includes only the surrogates that the subpolicy decides (local prefetching).

Centrality is measured with the HITS algorithm [17], a link analysis algorithm that rates web pages. Twitter uses a HITS style algorithm to suggest to users which accounts to follow [13], as well. A so-called good hub represents a page that points to many other pages, whereas a good authority represents a page that is linked by many different hubs. Memory usage issues for the very large graph dataset accommodated led to calculation of HITS with the MapReduce technique. Subpolicy (Fig. 18.3) checks the X closest timezones where a user has mutual friends and out of them, the Y with the highest value of the centrality metric as an average. Highest value of the metric means that the object is likely to be asked for more times. Copying is performed to the surrogate servers that serve the above timezones.

18.4.2 For Every New Object in the Surrogate Server

Surrogate servers keep replicas of the web objects on behalf of content providers. In the case that the new object does not fit in the surrogate server's cache, we define the *time_threshold* as the parameter for the duration that an object remains cached.

We check for items that have remained cached for a period longer than the *time_threshold* and we delete those with the largest timestamp in the cascade. In case there exist no such objects or all objects have the same timestamp, we apply various policies for the removal of objects

- *Least Recently Used (LRU)* In the most straightforward extension of LRU for handling nonhomogeneous-sized objects we prune the least recently used items first. The algorithm keeps track of what was used when, to make sure that it discards the least recently used item.
- *Least Frequently Used (LFU)* The algorithm keeps track of the number of times an object is referenced in memory. When the cache is full and more room is required, it purges the item with the lowest reference frequency. We simply employ an LFU algorithm by assigning a counter to every object that is loaded into the cache. Each time a reference is made to that object the counter is increased by one. When the cache reaches capacity and a new object arrives, the system will search for the object with the lowest counter and remove it from the cache.

Table 18.2 Applied caching schemes

Name	Primary key	Secondary key
LRU	Time since last access	
LFU	Frequency of access	
SIZE	Size	Time since last access

- *Size-adjusted LRU (SIZE)* The optimization model devised in [1] to generalize LRU is approximately solved by a simple heuristic and the policy is called Size-adjusted LRU or SIZE. In this policy the objects are removed in order of size with the largest object removed first. In case two objects have the same size, objects longer cached since their last access are removed first. Objects in the cache are reindexed in order of increasing values of $S_i \cdot \Delta T_{i_\kappa}$ and highest index objects are greedily selected and purged from the cache until the new object fits in.

Varying algorithms depending on the caching scheme used (Table 18.2) are depicted in Figs. 18.4, 18.5 and 18.6. The heuristics applied in our approach are based on the following observations [15]: Users are more influenced by geographically close friends, and moreover by mutual followers, with the most popular users acting as authorities. Social cascades have a short duration, and in our prefetching algorithm we take into account the observation that the majority of cascades end within 24 h. However, we introduce a varying time threshold for the cascade effect and the time that an object remains in cache. Values given in the time threshold variable also include 48 h, as well as threshold covering the entire percentage of requests.

```
1:  if o.size + current_cache_size ≤ total_cache_size then
2:      copy object o to cache of surrogate n_k;
3:  else if o.size + current_cache_size > total_cache_size then
4:      while o.size + current_cache_size > total_cache_size do
5:          for all object o' in current_cache do
6:              if (current_timestamp - o'.timestamp) + o'.timer > time_threshold then
7:                  copy o' in CandidateList;
8:              end if
9:              if CandidateList.size>0 and CandidateList.size != total_cache_size then
10:                 find o' that o'.timestamp is maximum and delete it;
11:             else if CandidateList.size==0 or CandidateList.size==total_cache_size then
12:                 use LRU to delete any object o ∈ O;
13:             end if
14:         end for
15:     end while
16:     put object o to cache of surrogate n_k;
17: end if
```

Fig. 18.4 VariationA—Algorithm for every new object o in the surrogate server n_k

```
1: if o.size + current_cache_size ≤ total_cache_size then
2:     copy object o to cache of surrogate n_k;
3: else if o.size + current_cache_size > total_cache_size then
4:     while o.size + current_cache_size > total_cache_size do
5:         for all object o' in current_cache do
6:             if (current_timestamp - o'.timestamp) + o'.timer > time_threshold then
7:                 copy o' in CandidateList;
8:             end if
9:             if CandidateList.size>0 and CandidateList.size != total_cache_size then
10:                find o' that o'.timestamp is maximum and delete it;
11:            else if CandidateList.size==0 or CandidateList.size==total_cache_size then
12:                use LFU to delete any object o ∈ O;
13:            end if
14:        end for
15:    end while
16:    put object o to cache of surrogate n_k;
17: end if
```

Fig. 18.5 VariationB—Algorithm for every new object o in the surrogate server n_k

```
1: if o.size + current_cache_size ≤ total_cache_size then
2:     copy object o to cache of surrogate n_k;
3: else if o.size + current_cache_size > total_cache_size then
4:     while o.size + current_cache_size > total_cache_size do
5:         for all object o' in current_cache do
6:             if (current_timestamp - o'.timestamp) + o'.timer > time_threshold then
7:                 copy o' in CandidateList;
8:             end if
9:             if CandidateList.size>0 and CandidateList.size != total_cache_size then
10:                find o' that o'.timestamp is maximum and delete it;
11:            else if CandidateList.size==0 or CandidateList.size==total_cache_size then
12:                use SIZE to delete any object o ∈ O;
13:            end if
14:        end for
15:    end while
16:    put object o to cache of surrogate n_k;
17: end if
```

Fig. 18.6 VariationC—Algorithm for every new object o in the surrogate server n_k

Prinicipally we check whether specific time has passed after the start of cascade and, only in the case that the cascade has not ended, define to what extent the object will be copied (*algorithm for every new request*). This check is also performed in *algorithm for every new object*, where we define the *time_threshold*. The latter roughly expresses the average cascade duration, as it defines the duration that an object remains cached.

18.5 Experimental Evaluation

For the experimental evaluation, we used a stand-alone CDN simulator for CDNs. The configuration of the simulation values is depicted in Table 18.3. For the extraction of a reliable output, we had to conclude to a specific network topology, as well as make assumptions regarding the input dataset. The simulator takes as input files describing the underlying CDN and the traffic in the network, and provides an output of statistical results, discussed in the next Section.

- *Network Topology* There follows a short description of the process defining the nodes in the topology. These nodes represent the surrogate servers, the origin server, and the users requesting the objects (Fig. 18.7). For an in-depth analysis you can refer to [15]. To simulate our policy and place the servers in a real geographical position, we used the geographical distribution of the Limelight network [14]. For the smooth operation of the simulator the number of surrogate servers was reduced by a ratio of 10 %, to ultimately include 423 servers (Table 18.4). Depending on the closer distance between the surrogate region defined by Limelight and each of the timezones defined by Twitter (20 Limelight regions, 142 Twitter timezones), we decided where the requests from each timezone will be redirected. The population of each timezone was also taken into consideration. The INET generator [4] allowed us to create an AS-level representation of the network topology. Topology coordinates were converted to geographical coordinates with the NetGeo tool from CAIDA [5], a tool that maps IP addresses and Autonomous System (AS) coordinates to geographical coordinates [22], and surrogate servers were assigned to topology nodes.
  After grouping users per timezone (due to the limitations the large dataset imposes), each group of users was placed in a topology node. We placed the user groups in the nodes closer to those comprising the servers that serve the respective timezone requests, contributing this way to a realistic network depiction.
- *Number of Requests* 1 million requests were considered sufficient, with the number of objects being the dominant factor increasing the memory use of the simulation tool. Also similar concept approaches use similar number of requests ([23] on a daily basis and [21]), and same number of distinct videos for generation of requests.

Table 18.3 Simulation characteristics

Number of nodes in the topology	3500
Redirection policy	Cooperative environment (closest surrogate)
Number of origin servers	1
Number of surrogate servers	423
Number of user groups	162
Bandwidth	100 Mbit/s

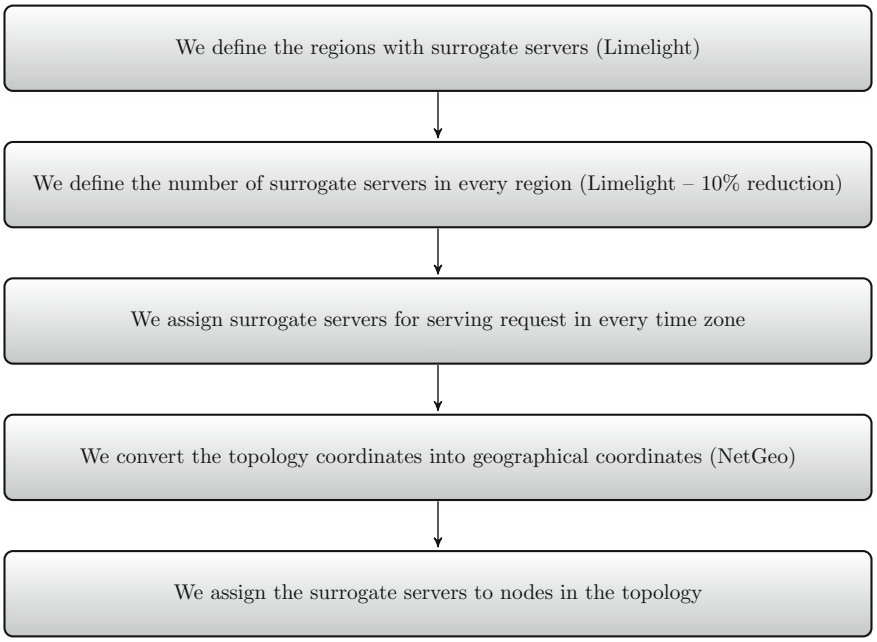


Fig. 18.7 Methodology followed

Table 18.4 Distribution of servers over the world for the experimental evaluation

City	Servers	City	Servers
Washington DC	55	Toronto	12
New York	43	Amsterdam	20
Atlanta	11	London	30
Miami	11	Frankfurt	31
Chicago	37	Paris	12
Dallas	19	Moscow	10
Los Angeles	52	Hong Kong	8
San Jose	37	Tokyo	12
Seattle	15	Changi	5
Phoenix	3	Sydney	1

- *Cache Size* The requests generated from the generator follow a long-tail distribution, thus 15 % of the whole catalog size was considered to be sufficient.
- *Threshold Values* Experimenting was conducted for time thresholds of 24 h and 48 h, as well as for the time threshold that covered all the requests. The threshold for media service viewership was set at 402.408 (average media viewership in the dataset). The authority threshold score was tested for various values (0.006/0.02/0.04).

18.6 Main Findings

The statistic reports produced by the simulator are used to evaluate the proposed policy. There follows a short explanation of the metrics used in our experiments for the extraction of statistical results.

18.6.1 Client Side Metrics

They refer to activities of clients, i.e., the requests for objects.

- *Mean Response Time* indicates how fast a client is satisfied. It is defined as

$$\frac{\sum_{i=0}^{M-1} t_i}{M}$$

where M is the number of satisfied requests and t_i is the response time of the ith request. It starts at the timestamp when the request begins and ends at the timestamp when the connection closes.

18.6.2 Surrogate Side Metrics

They are focused on the operations of the surrogate servers.

- *Hit Ratio*: is the percentage of the client-to-CDN requests resulting in a cache hit. High values indicate high quality content placement in the surrogate servers.
- *Byte Hit Ratio*: is the hit ratio expressed in bytes, counting the corresponding bytes of the requests. High values indicate optimized space usage and lower network traffic.

18.6.3 Network Statistics Metrics

They run on top of TCP/IP and concern the entire network topology.

- *Active Servers* refers to the surrogate servers being active serving clients.
- *Mean Utility of the Surrogate Servers* is a value that expresses the relation between the number of bytes of the served content against the number of bytes of the pulled content (from the origin server or other surrogate servers). It is bounded to the range [0, 1] and provides an indication about the CDN performance. High net utility values indicate good content outsourcing policy and improved mean response times for the clients.

We conducted a multitude of experiments (55 for each caching scheme and time threshold combination). Table 18.5 presents the average values of four parameters for six cases of testing. The lowest mean response times appear for the cases of the time threshold covering all requests for all caching schemes. We observe that LFU scheme outperforms LRU and SIZE in terms of mean response times and hit ratios achieved.

- *Hit Ratio* To begin with, Fig. 18.8 illustrates how the hit ratio of the requests is affected by modifying the number of timezones with highest centrality metric examined. The caching scheme of LFU appears to perform better than the LRU scheme. LRU and LFU offer comparable results, whereas they both outperform SIZE. We come to the conclusion that there is a realistic room for performance improvement by implementing various web caching characteristics in a CDN infrastructure, even though the social cascading mechanisms have already been activated to improve its performance.
- *Mean Utility of the Surrogate Servers* For a fixed number of 10 closest timezones with mutual followers LRU scheme appears to depict the highest mean utility of the surrogate servers, followed by LFU and SIZE (Fig. 18.9).

Table 18.5 Average metric values for $X = 10$ timezones of close mutual friends

	Mean response time (Avg, 10^-2 s)	Hit ratio (Avg, %)	Active servers	Mean utility (Avg, %)
LFU—24-h	1.1383	32.81	326	96.01
LFU—48-h	1.1352	33.08	325	96.01
LFU—all-h	1.1112	34.69	324	96.01
SIZE—24-h	1.1541	32.10	327	95.94
SIZE—48-h	1.146076	32.03	326	95.98
SIZE—all-h	1.1274	33.17	326	96.00
LRU—24-h	1.1412	32.12	326	95.99
LRU—48-h	1.1377	32.42	325	96.02
LRU—all-h	1.1181	34.16	325	96.04

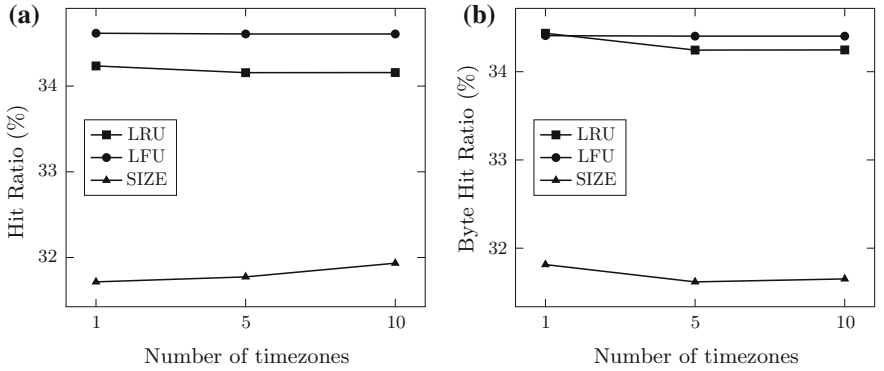


Fig. 18.8 Effect of timezones used as Y on **a** Hit Ratio and **b** Byte Hit Ratio for $X = 10$ closest timezones with mutual followers for LRU, LFU and SIZE

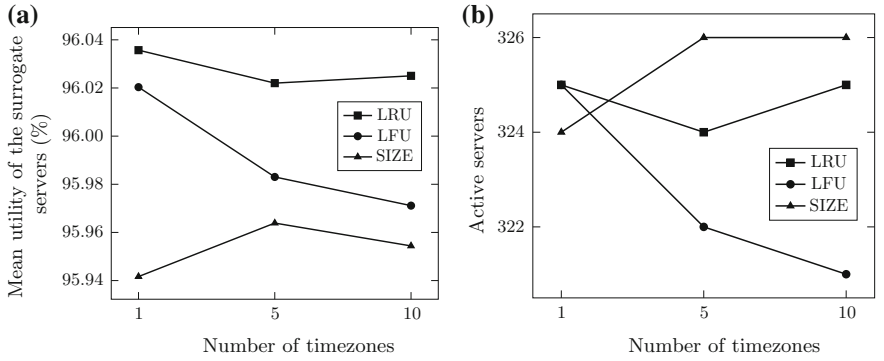


Fig. 18.9 Effect of timezones used as Y on **a** Mean Utility of the Surrogate Servers and **b** Active Servers for $X = 10$ closest timezones with mutual followers for LRU, LFU, and SIZE

- *Active Servers* For a fixed number of 10 closest timezones with mutual followers LFU appears to use less active servers after the first timezone of highest centrality in the scenario of time threshold covering all the requests. SIZE depicts higher values of active servers for the cases of 5 and 10 timezones examined (Fig. 18.9).
- *Mean Response Time* For the most representative case of all requests for LRU and LFU schemes, the trade-off between the reduction of the response time and the cost of copying in servers is expressed with a decrease of the mean response time as the timezones increase, and a point after which the mean response time starts to increase again (Figs. 18.10 and 18.11). This decrease in the mean response time occurs with approximately five timezones out of the 10 used for LRU scheme (for a fixed number of closest timezones with mutual followers), and with seven timezones for LFU. After this point the slight increase in the mean response time is attributed to the delay for copying content to surrogate servers. The cost for every copy is related to the number of hops among the client asking for it and

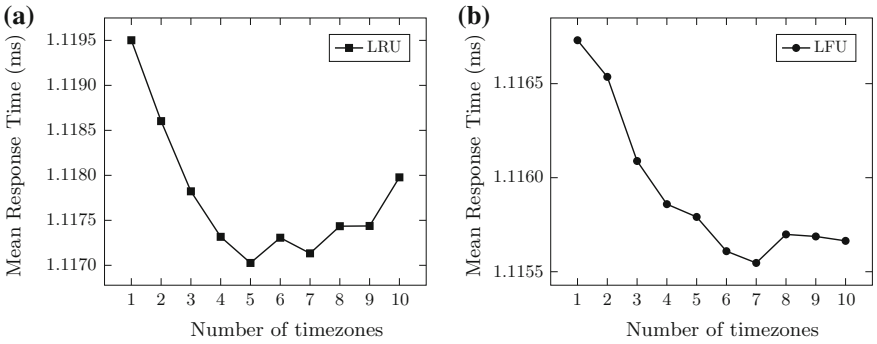


Fig. 18.10 Effect of timezones used as Y on Mean Response Time for $X = 10$ closest timezones with mutual followers for **a** LRU and **b** LFU

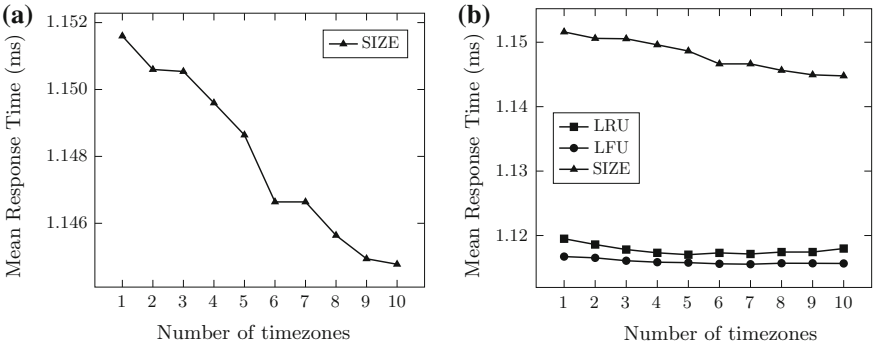


Fig. 18.11 Effect of timezones used as Y on Mean Response Time for $X = 10$ closest timezones with mutual followers for **a** SIZE and **b** all caching schemes

the server where copying is likely to be made, according to the *Put* function. We observe that SIZE depicts a poor performance, since it does not take advantage of the frequency skew.

18.7 Conclusions

CDN infrastructures rapidly deliver multimedia content cached on dispersed geographical servers to Web browsers worldwide. The growing demands for quick and scalable delivery, also due to HTTP traffic increase, can be satisfied with efficient management of the content replicated in CDNs. Specifically, we need Web data caching techniques and mechanisms on CDNs, as well as policies recognizing the patterns of social diffusion of content, to ensure satisfying performance in a constantly changing environment of continuing data volume growth.

In the present work, we further extended a dynamic policy of OSN content prefetching implemented with temporal and other contextual parameters, to depict how various caching schemes can affect the content delivery infrastructure. Bandwidth-intensive multimedia delivery over a CDN infrastructure is experimentally evaluated with realistic workloads, that many works in the related literature lack. While recognizing that we used one media service and one OSN platform for our experimentation, we believe that our results are generally applicable, with a potentially high impact for large-scale systems where traffic is generated by online social services and microblogging platforms. We aim to generalize our proposed policies in the future, to deal with multiple OSN platforms, as well as mobile CDN providers.

Acknowledgments For the development of algorithms and conducting of the accompanying experiments, the cloud infrastructure of the Department of Computer Science of the University of Cyprus, as well as Amazon Web Services, were used.

References

1. Aggarwal, C., Wolf, J.L., Yu, P.S.: Caching on the World Wide Web. IEEE Trans. Knowl. Data Eng. **11**(1), 94–107 (1999). doi:10.1109/69.755618
2. Bakshy, E., Rosenn, I., Marlow, C., Adamic, L.A.: The role of social networks in information diffusion. In: Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, 16–20 April 2012, pp. 519–528 (2012). doi:10.1145/2187836.2187907
3. Brodersen, A., Scellato, S., Wattenhofer, M.: YouTube around the world: geographic popularity of videos. In: Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, 16–20 April 2012, pp. 241–250 (2012). doi:10.1145/2187836.2187870
4. Cha, M., Kwak, H., Rodriguez, P., Ahn, Y., Moon, S.B.: I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. IN: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, IMC 2007, San Diego, California, USA, 24–26 Oct 2007, pp. 1–14 (2007). doi:10.1145/1298306.1298309
5. Center for Applied Internet Data Analysis. https://www.caida.org. Accessed 30 Jun 2016
6. Chard, K., Caton, S., Rana, O., Bubendorfer, K.: Social Cloud: cloud computing in social networks. In: Proceedings of the 3rd IEEE International Conference on Cloud Computing, CLOUD 2010, Miami, FL, USA, 5–10 July 2010, pp. 99–106 (2010). doi:10.1109/CLOUD.2010.28
7. Cheng, X., Dale, C., Liu, J.: Statistics and social network of YouTube videos. In: Proceedings of the 16th International Workshop on Quality of Service, IWQoS 2008, University of Twente, Enskede, The Netherlands, 2–4 June 2008, pp. 229–238 (2008). doi:10.1109/IWQOS.2008.32
8. Easley, D.A., Kleinberg, J.M.: Networks, Crowds, and Markets—Reasoning About a Highly Connected World. Cambridge University Press (2010)
9. Figueiredo, F., Benevenuto, F., Almeida, J.M.: The tube over time: characterizing popularity growth of YouTube videos. iN: Proceedings of the 4th International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, 9–12 Feb 2011, pp. 745–754 (2011). doi:10.1145/1935826.1935925
10. Finamore, A., Mellia, M., Munafò, M.M., Torres, R., Rao, S.G.: YouTube everywhere: impact of device and infrastructure synergies on user experience. In: Proceedings of the 11th ACM SIGCOMM Conference on Internet Measurement, IMC 2011, Berlin, Germany, 2–4 Nov 2011, pp. 345–360 (2011). doi:10.1145/2068816.2068849
11. Gill, P., Arlitt, M., Li, Z., Mahanti, A.: YouTube traffic characterization: a view from the edge. In: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, IMC

2007, San Diego, California, USA, 24–26 Oct 2007, pp. 15–28 (2007). doi:10.1145/1298306.1298310
12. Gill, P., Arlitt, M., Li, Z., Mahanti, A.: Characterizing user sessions on YouTube. In: Proceedings of the SPIE Multimedia Computing and Networking Conference, MCN 2008, San Jose, California, USA, 30–31 Jan 2008, pp. 6818060–6818068 (2008). doi:10.1117/12.775130
13. Gupta, P., Goel, A., Lin, J., Sharma, A., Wang, D., Zadeh, R.: WTF: the who to follow service at Twitter. In: Proceedings of the 22nd International World Wide Web Conference, WWW 2013, Rio de Janeiro, Brazil, 13–17 May 2013, pp. 505–514 (2013). doi:10.1145/2488388.2488433
14. Huang, C., Wang, A., Li, J., Ross, K.W.: Measuring and evaluating large-scale CDNs. In: Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement, IMC 2008, Vouliagmeni, Greece, 20–22 Oct 2008, pp. 15–29 (2008)
15. Kilanioti, I.: Improving multimedia content delivery via augmentation with social information. The Social Prefetcher approach. IEEE Trans. Multimedia **17**(9), 1460–1470 (2015). doi:10.1109/TMM.2015.2459658
16. Kilanioti, I., Papadopoulos, G.A.: Socially-aware multimedia content delivery for the cloud. In: Proceedings of the 8th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2015, Limassol, Cyprus, 7–10 Dec 2015, pp. 300–309 (2015). doi:10.1109/UCC.2015.48
17. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM (JACM) **46**(5), 604–632 (1999). doi:10.1145/324133.324140
18. Mitra, S., Agrawal, M., Yadav, A., Carlsson, N., Eager, D.L., Mahanti, A.: Characterizing web-based video sharing workloads. TWEB **5**(2), 8 (2011). doi:10.1145/1961659.1961662
19. Rodrigues, T., Benevenuto, F., Cha, M., Gummadi, P.K., Almeida, V.A.F.: On word-of-mouth based discovery of the web. In: Proceedings of the 11th ACM SIGCOMM Conference on Internet Measurement, IMC 2011, Berlin, Germany, 2–4 Nov 2011, pp. 381–396 (2011). doi:10.1145/2068816.2068852
20. Sastry, N., Yoneki, E., Crowcroft, J.: Buzztraq: predicting geographical access patterns of social cascades using social networks. In: Proceedings of the 2nd ACM EuroSys Workshop on Social Network Systems, SNS 2009, Nuremberg, Germany, 31 March 2009, pp. 39–45 (2009). doi:10.1145/1578002.1578009
21. Scellato, S., Mascolo, C., Musolesi, M., Crowcroft, J.: Track globally, deliver locally: improving content delivery networks by tracking geographic social cascades. In: Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28–April 1, 2011, pp. 457–466 (2011). doi:10.1145/1963405.1963471
22. Torres, R., Finamore, A., Kim, J.R., Mellia, M., Munafò, M.M., Rao, S.G.: Dissecting video server selection strategies in the YouTube CDN. In: Proceedings of the 31st International Conference on Distributed Computing Systems, ICDCS 2011, Minneapolis, Minnesota, USA, 20–24 June 2011, pp. 248–257 (2011). doi:10.1109/ICDCS.2011.43
23. Traverso, S., Huguenin, K., Trestian, I., Erramilli, V., Laoutaris, N., Papagiannaki, K.: Tailgate: handling long-tail content with a little help from friends. In: Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, 16–20 April 2012, pp. 151–160 (2012). doi:10.1145/2187836.2187858

Chapter 19
A Java-Based Distributed Approach for Generating Large-Scale Social Network Graphs

Vlad Şerbănescu, Keyvan Azadbakht and Frank de Boer

19.1 Introduction

Distributed systems and applications require large amounts of resources in terms of memory and computing power and are becoming a standard for large businesses and enterprises [12] within and outside the domain of Computer Science. A very important topic for distributed applications is Big Data management and more specifically the generation of large-scale social networks graphs where the number of nodes reaches very large numbers. Analysis of such networks is of importance in many areas, e.g., data mining, network sciences, physics, and social sciences [3]. The need for efficient and scalable methods of network generation is frequently mentioned in the literature [8], particularly for the preferential attachment process [1, 13, 14]. Barabasi–Albert model, which is based on preferential attachment (PA) [4], is one of the most commonly used models to produce artificial networks, because of its explanatory power, conceptual simplicity, and interesting mathematical properties [13]. Nevertheless the large number of nodes in such graphs may not fit in the memory on one machine. The need for efficient solutions which provide scalability also requires more computational resources as well as implementation considerations. As such, distribution and synchronization are two main challenges. In this chapter, we investigate as a case study a distributed solution for PA-based graph generation which avoids low level synchronization management, thanks to the notion of cooperative scheduling and futures.

V. Şerbănescu (✉) · K. Azadbakht · F. de Boer
Centrum Wiskunde and Informatica, Science Park 123, 1098 Amsterdam, XG, Netherlands
e-mail: vlad.serbanescu@cwi.nl

K. Azadbakht
e-mail: kazadbakht@cwi.nl

F. de Boer
e-mail: frb@cwi.nl

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_19

In several examples of distributed applications such as high-energy physics applications, research digital libraries or secure banking systems with millions of users, communication between remote machines is a significant challenge. Modeling distributed communication such that it can be analyzed at development phase for synchronization issues or deadlocks allows the design of reliable and efficient software that will not require extensive testing and debugging. We provide in this paper a library that allows a more intuitive mapping from a modeling language, the abstract behavioral specification language(ABS) [6], to a programming language. Since the modeling language is tailored toward asynchronous communication, the distributed implementation of the library will not require any remote object field access or synchronous remote method invocations.

The expressive power of the above-mentioned library is shown in the PA-based graph generation case study where the array representing the graph is partitioned and located on remote machines, each is owned by an actor. According to PA, the slots of each array are resolved by the values from (the same or) other arrays by which the new connections between the nodes are formed. The *future* construct provides a means for each actor on a machine to synchronize on the return value from a process (i.e., a runtime method execution of an actor) on remote machines. The process itself can also suspend on a boolean condition (i.e., the continuation of the process can be activated when the condition is evaluated to True), and then returning the value. As shown in the case study, using such powerful constructs eliminates the need for low level synchronizatoin mechanisms.

While the Java language is one of the most popular, intuitive and easy to use languages in terms of software development and has significant support for parallel and distributed programming, its most basic entity for representing a block of running code is a Thread which is very expensive memory-wise. There are several interfaces which abstract lightweight tasks and subtasks, but in order to schedule and preempt these tasks they still need to be encapsulated in the form of a Thread. The issue appears when an application requires multiple context switches between several tasks which have significant call stack sizes. As this number becomes very large the thread explosion affects the main memory thus the application's performance. The major contribution of this chapter is to provide a library with distributed support of the actor-based model in the Java language, with enhanced future synchronization capabilities that become available in a distributed setting. Furthermore we introduce data structures to manage propagation of futures and allow them to be garbage collected on each machine. We also provide a notification mechanism for future resolution in order to avoid inefficient busy-waiting loops. The rest of this paper discusses related work in Sect. 19.2, followed by a detailed explanation of the new library in Sect. 19.3. We present a case study that is suitable for evaluating our solution in Sect. 19.4 and its high-level implementation in Sect. 19.5. We draw the conclusions and present the next steps for this work in Sect. 19.6.

19.2 Background and Related Work

In the ABS, the core language semantics imposes that all objects created in the program are actors with an independent behavior, with a possibility to communicate with other actors and use futures to synchronize at certain execution points. The language groups actors into Concurrent Object Groups (COG) which allow objects to communicate with each other synchronously, but apart from the actors in the same COG, all other actors are considered remote and may only invoke each other asynchronously. The physical location of an actor in ABS is completely transparent as there are no virtual machines or IP addresses inserted in this high-level language. The language features one construct for the asynchronous communication, another construct for blocking an actor's execution on a future and a third and very powerful construct for suspending and scheduling methods within one single actor, a construct that introduces the notion of cooperative scheduling.

These two latter constructs can be preceded by annotations that allow custom schedulers to be defined in order to satisfy an application's specific requirements. Further annotations can be associated to method calls to specify costs and deadlines in order to create a very powerful scheduler. All these constructs are written in a very simple and concise way in ABS, in order to allow system designers a simple view of their application which can be even large enough to be deployed in a cloud environment [7]. However, from this modeling language we need to generate code for programs to execute on multiple resources, tasks to be submitted to those resources and also incorporate the powerful schedulers. To this end ABS has several execution backends in a simulation language (Maude), functional language (Haskell), and object-oriented languages (Erlang and Java). Our focus for this contribution is the Java language backend for which the translation process of distributed applications may create multiple redundant objects, threads, and data structures that significantly impact performance.

Listing 1 Scheduling in ABS example

```
interface Ainterface {
    Int recursive_m(Int i);
}

interface Binterface{
    Int  compute( );
}

class A implements Ainterface{

    Int recursive_m(Int i){
        if (i>0){
            this.recursive_m(i – 1);
        }
        else{
            B computation = new B ( );
            Future f = computation ! compute( );
            await f ?
}   }   }

class B implements Binterface {

    Int compute( ){
        Int result;
```

```
        /*do some work */
        return result;
}   }

{ // Main block:
    Int i = 0;
    A master_i = new A ( );
    List futures = EmptyList;

    while (i < 100){
        Future f = A ! recursive_m (10);
        futures = cons( f, futures );
    }

    while ( futures != EmptyList ){
        Future f = head (futures);
        futures = tail (futures);
        f.get ;
}   }
```

To illustrate how cooperative scheduling works, we look at a simple program in Listing 1. The program creates an object of Class A which contains method "recursive_m." The construct "o.recursive_m" is a regular synchronous call that must be executed without any preemption. Inside the method, we create an object of class B which has a method "compute." The construct "computation!compute( )" is an asynchronous call that allows the object of class B to execute in parallel the method "compute." At this point there are two constructs for synchronizing with a call that executes in parallel with the current object. The first construct, "f.get" is at the level of the object and forces the current object to block and wait for the completion of method "compute" that was captured in the future f. The second construct "await f ?" is more fine grained in the sense that only the current method that is executing this statement, namely "computation!compute( )" blocks, while subsequent calls of "computation!compute( )" resulting from the main for loop, can be scheduled and run by the object o. An important observation here is to understand that all calls like "computation!compute( )" are inserted into a queue that each object has and are scheduled to be run by the same object and not in parallel. The degree of parallelism is determined by the number of objects created.

The ABS-API library [9–11] was introduced as a solution to translate ABS code into production code initially for parallel applications. Java 8 new features allow wrapping of method calls into lightweight lambda expressions such that they can be put into a scheduling queue of an Executor Service to which the running objects are mapped, significantly reducing the number of idle Threads at runtime. Furthermore we minimized the number of threads created by saving the call stack of suspended methods within an actor caused by the "await" statement. Our first solution to achieve this was to add a central context for all actors in the system and follow this execution sequence.

- Each asynchronous call/invocation is a message delivered in the corresponding object's queue.
- All objects in the same Concurrent Object Group (COG) compete for one Thread.
- A *Sweeper Thread* decides which task should be created and be available for execution from the available queues.
- A thread pool executes available tasks based on a work stealing mechanism.

- On every await statement, we try to optimally suspend the message thread until the continuation of the call is released.

This *Sweeper Thread* however becomes a bottleneck when the number of actors is very large while also making actors dependent on each other. The new library that is the main contribution of this chapter, however, is tailored to support distributed actor-based programming and therefore requires a different organization of thread management and future propagation. Furthermore in a distributed setting there can be no centralized thread for all actors, therefore we propose a new solution that replaces the purpose of this thread.

19.3 Description of the Distributed ABS-API Library

In this section we describe the newest version of the ABS-API library that is written in Java to support an actor programming model in a distributed setting, with enhanced cooperative scheduling, distributed future control and garbage collection. We present a whole new and simple format for classifying actors based on their intended behavior. In this version we optimize even further the memory management of actor by reducing the number of Threads created at runtime. We use certain triggers to determine the start and ending of a live context and eliminate the use of redundant Threads that correspond to the running process of an actor. We introduce a class hierarchy of actors running on remote hosts, on local host and actors whose functionality is reachable from a remote host. This hierarchy simplifies garbage collection and reduces the number of peer-to-peer communications between remote hosts, as well as offering a clear separation between an application running on a single machine or in a distributed environment.

19.3.1 Actor Class Hierarchy, Naming Scheme and Asynchronous Communication

In the previous implementation of the ABS-API we had one single interface to allow an actor programming paradigm. This single interface encapsulated the entire behavior of the actor that comprised of the continuous running cycle, the task message queue, the single thread restriction and the cooperative scheduling of its suspended tasks. However, when deploying actors in a distributed environment they have particular locality and visibility characteristics. It is also important to take into account how actors will communicate with each other depending on their location and what elements need to be specific to the machine. Therefore an actor needs to have a global identity (represented by a Java URI) making it discoverable on all the machines of the application while its internal structure exists on only one machine. The URI takes the format of "IP:actorName", where IP is the host machine's address and *actorName* is

a unique identifier that distinguishes between actors on this machine. This allows for a scheme where local generation of actors avoids inconsistencies at a global level. The discovery mechanism is very simple:

- An actor is instantiated on only one machine and is given a URI comprised of the machine's IP and a unique name.
- To communicate with a remote actor, a machine requires a reference with the unique global identifier.

For inter-machine communication each two machines in the system are connected by one socket and actors send messages through the machine's socket streams. Furthermore, each machine maintains a *actorMap* of all of its actors in order to have a mapping between the Java URI and the Java reference of the actor such that it can forward remote requests to the correct actor. A certain special type of actor is introduced that is classified as local and has no global identity such that it is not reachable by other machines and can only receive messages from other actors on the machine it runs on. These particular characteristics allow for a simple classification of the actors according to the class diagrams in Fig. 19.1 and provide a clear separation between a parallel and a distributed setting.

The diagram in Fig. 19.1, presents how we classify actors based strictly on the machine on which they reside and therefore their physical existence on the machine. The API has a parent abstract class called *DeploymentComponent* which maintains data specific to the machine. Firstly, it contains a customized ThreadPoolExecutor to which the actors residing on the machine will submit their tasks. This ThreadPool

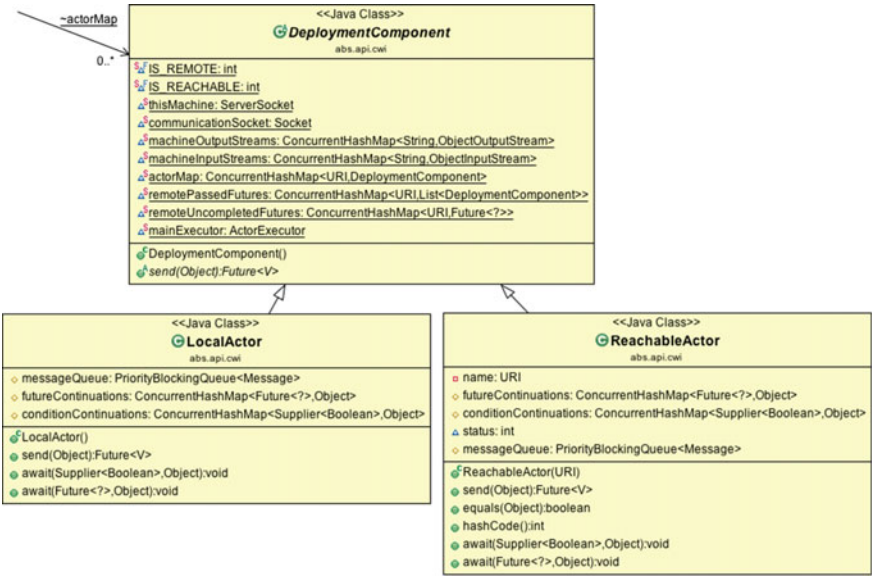


Fig. 19.1 Class diagram based on ABS transparency

Executor is available to all actors on one machine and it has an overridden *afterExecute* method to control several behaviors specific to actors which we will discuss further in this section. Secondly, the class also contains the *actorMap* of all the actors that are initialized on one machine such that remote messages can be routed to the correct actor. Finally, the class contains a table of the socket streams with all other machines in the system that grows and shrinks dynamically, as more machines are added to the system. An important observation is to notice that socket streams are initialized only when a remote actor belonging to a node that was previously unknown is instantiated in the system and a *listener* thread is assigned to the stream processing either incoming messages to the machine and as such, only if the setting is distributed. The machines communicate through serialized messages and objects that can be of four types that will be explained in the next two subsections:

- asynchronous method invocations.
- futures that are passed as parameters.
- a resolved future result.
- actor URI or futureID used to identify actors and futures on remote machines.

The DeploymentComponent class is subclassed into a *LocalActor* class which represents the simplest abstraction of an actor that is not connected to the outside of the machine. This actor has a message queue that can receive asynchronous messages that are executed in a FIFO order with submissions to the machine's thread pool. An important optimization is introduced in the execution instance of a standard actor. Instead of spawning a task that continuously loops through an actor's queue from the point it is instantiated, the task is now spawned only when the first message is introduced in the queue and finishes once the queue is empty. Therefore actors no longer have a live thread corresponding to their run if they are idle. The second subclass of the *DeploymentComponent* is the *ReachableActor* which has two very distinct behaviors, but ensures the transparency of the ABS language presented in Sect. 19.2. This class can identify either an actor that is extended with distributed support to receive remote calls or a remote actor which forwards messages to the correct machine. In both cases the actor is instantiated with the global identifier that we discussed earlier in this section, and this identifier distinguishes its general behavior on the machine. If its identifier's host IP is the same with the machine IP, then it behaves like a standard actor, only it is included in the machine's *actorMap* such that incoming messages can be forwarded to it for execution. On the other hand, if the IP differs from the machine IP, the actor is a remote actor and it is only a reference used for transparency on the machine. In this case, asynchronous messages are forwarded to the machine's output stream such that they can be sent via the machine socket to the machine where the remote actor actually resides.

19.3.2 Distributed Futures Control

The most important feature of our library is that it now has support for programming with distributed actors. A more detailed illustration in Fig. 19.2 explains the role of the *ReachableActor* on both a local and remote machine. In this setting we have an actor *a1* with a unique global identifier which is a Java URI that is "IP2:a1", where IP2 is the IP address of Node 2 on which the actor was instantiated and a1 is a unique identifier of the actor. Node 1 has a reference to this actor and its interface which contains method *m()* is also available. An important objective of our solution is to avoid actors entering a busy-waiting loop in order to check the resolution of futures. To achieve this, we insert a *remoteUncompletedFutures* data structure which retains all the futures that were generated by calls to remote actors. The machine then sends *a serialized lambda expression* of the asynchronous method call to the socket. Each machine is aware of the senders of incoming messages, therefore when an actor completes a remote call, the *serialized result* of the actor can be sent back as a reply. This behavior is part of the *afterExecute* method of the machine's main executor and

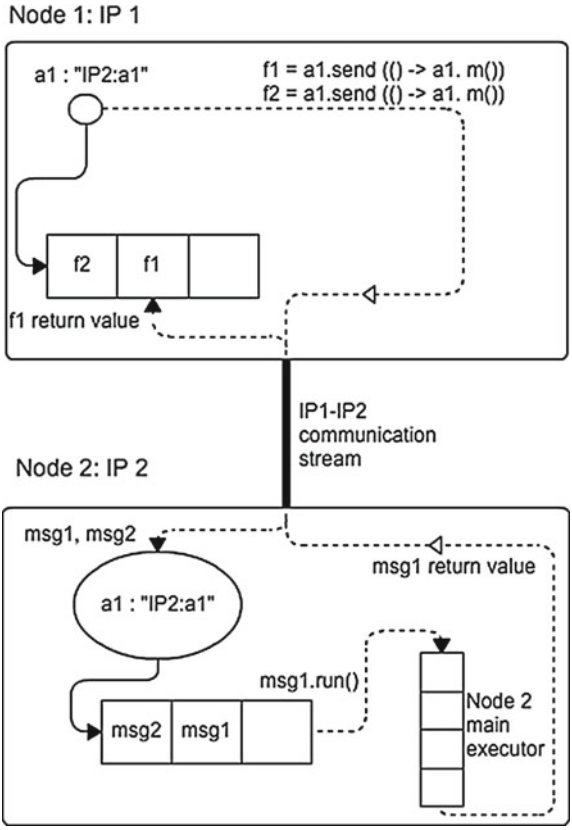


Fig. 19.2 Future Flow

is illustrated by the state *afterExecture* in the state diagram of the actor Fig. 19.4. In Sect. 19.2 we discussed how messages in an actor are executed in the order that they arrive in its queue and how the *await* statement allows for the rescheduling of these messages. To allow remote actors to identify which reply belongs to which future in the queue we introduce a naming scheme in the form of "IP:f" where IP is the address of the actor that will complete the future and f the unique global identifier (futureID) of the future.

The general mechanism is best described in terms of an example scenario with two asynchronous calls to the same actor:

1. Node 1 sends the following sequence of messages to actor *a1* on Node 2.
   - A futureID "IP2:f1" identifying the future that will be generated by the following asynchronous method call.
   - A pair $< IP2 : a1, m() >$ representing the first asynchronous method call to actor *a1*.
   - A futureID "IP2:f2" identifying the future that will be generated by the next asynchronous method call.
   - A pair $< IP2 : a1, m() >$ representing the second asynchronous method call to actor *a1*.
2. The two uncompleted futures *f1* and *f2* and their corresponding identifiers are stored as mappings as *remoteUncompletedFutures*.
3. Actor *a1* receives from the socket stream the two identifiers and two messages *msg1* and *msg2* in the same order and inserts them in the message queue.
4. Actor *a1* schedules *msg1* and *msg2* in a FIFO order on Node 2 main executor unless rescheduled by an *await* statement.
5. When either message has finished executing, the *afterExecute* method of the main executor sends back the corresponding futureID within either pair $< IP2 : f1, result >$ or $< IP2 : f2, result >$back to the socket stream where the message came from.
6. The socket stream forwards the result to Node 1.
7. Future *f1* or *f2* is completed with the received result depending on the futureID.

The semantics of ABS allows actor references and futures to be passed remotely through asynchronous method calls. However the semantics restrict actors from accessing fields of remote references or making synchronous calls on these references. The transparency feature of ABS means that remote futures are accessible by any actor and can be used together with the *await* and *get* statements to synchronize. A more difficult challenge is how futures are propagated throughout the system as parameters of method calls and when they become available for garbage collection on each machine. While remote objects that may be passed as parameters are handled by the class hierarchy, remote futures need a heuristic to be propagated and notified of completion once they are passed as parameters. A *serialized future* object, together with the futureID, needs then to be sent before the actual method call that contains it, such that it can be identified on the remote machine. This is a different

type of message from the one that just sends a futureID like in the previous scenario, as the remote actor actually needs the object to call *get* and *await* statements on. This future is then inserted into a table of *remotePassedFutures* and the corresponding list of machines to which they have been passed as parameters, or the table is simply updated with another machine if the future already exists. Whenever an actor passes a future as a parameter of a remote asynchronous class it takes the following steps:

1. It checks if the future is completed and if so, sends it via the socket stream before sending the asynchronous call.
2. If it is uncompleted, the future is still sent before the call, but also saved in a map with the format $< futureID : List < DeploymentComponent >>$ where the list contains all the remote actors that have received this future as part of an asynchronous method call.
3. The received future is stored by the actor in the *remoteUncompletedFutures* map.
4. When the future is completed either by:
   - a local actor.
   - a remote actor as explained in the protocol before.
   - a remote actor explained in the next step.

   the list of machines that require the future is retrieved and the entry in the map is removed.
5. The actor sends a pair $< futureID, result >$ to all the machines in the list that require it.
6. When a machine receives such a pair it completes the future identified by the *futureID* with the result and possibly runs steps 3 and 4 itself if it propagated this future as well.

19.3.3 Actor Execution Context

Actors run in a parallel and distributed environment through simple messages that are presented in Fig. 19.3. In addition to the usual object-oriented implementation, an actor exposes a method *send* which allows it to receive asynchronous method calls form other actors and this provides parallel execution between the actors. The class simply creates a lambda expression that takes the form of a Java Runnable or Callable and subsequently a wrapper future which may be used for synchronization purposes. In our previous version of the API, each actor had an execution lock that limited it to one method running at a time.

For a single machine, there was a single thread, called a *Sweeper*, available across all actors, that continuously checked all "unlocked" actors and submitted the head of their queue to the executor service. Actor execution is now demand-driven as shown in Fig. 19.4, with a single thread that is spawned into the state *ready* once the first message is received in the actors queue, moves to state *execute* and runs all messages in the queue and goes into state *stop* once the queue becomes empty, restarting once

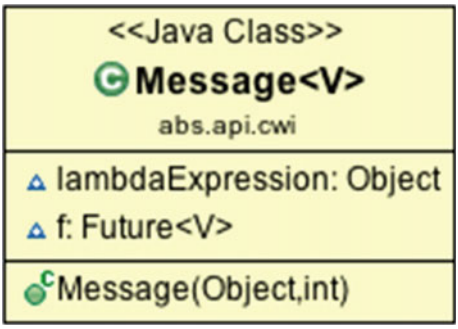


Fig. 19.3 Message encapsulation

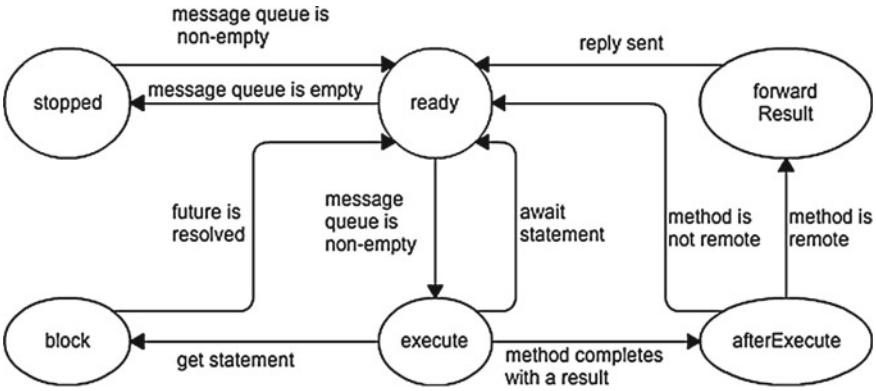


Fig. 19.4 Actor state diagram

another message is inserted in the queue. This makes actors completely independent from each other unless they explicitly call the synchronization mechanisms *get* and *await*.

19.3.4 Synchronization and Cooperative Scheduling

A key feature of the *Sweeper* thread was that it allowed efficient scheduling of tasks within an actor. It prevented redundant thread creation by having suspended tasks of an actor given priority once they were released to compete for the actor's lock. With the *Sweeper* thread deleted from the model of the API, we introduce a new mechanism to support cooperative scheduling. First of all when a *get* statement is called on a future, the actor moves to the state *block* until the future is resolved. If an *await* statement is encountered, the actor invokes another exposed method *await* which receives a boolean condition or a future to suspend on and a continuation in

the form of a lambda expression. The actor will then store a mapping of the continuation and the condition or future in a separate map as either *futureContinuations* or *conditionContinuations* specific to each actor and moves to state *ready*. The main executor introduced in the library is now responsible when, a thread completes, to run the *afterExecute* method which verifies if the method is remote in which case it has to *forward the result* to the socket from which it came to avoid a busy-waiting thread that may do this work. If the method invocation is from a local actor, the after execute method has to search each actor's continuation maps to identify the continuations that may have been resolved by this method (either an existing boolean condition or the actual future that has been resolved).

19.4 Description of the Preferential Attachment Algorithm

In this paper, we represent social networks through the notion of a graph where nodes are the members of the network and edges are the connections between them. The notion of Preferential Attachment (PA) is a specific model of adding a new member preferentially to a social network. We consider the above-mentioned preference to be the degree of the nodes, that is, roughly speaking, the more the degree of a node in the existing graph, the higher probability that it makes a connection with the new node. In this model, an existing graph of n nodes has a discrete probability distribution for the nodes with the probabilities $P_1, P_2, \ldots, P_n$ where $\sum_{i=1}^{n} P_i = 1$ and

$$P_i = \frac{deg(i)}{\sum_{j=1}^{n} deg(j)}$$

where $deg(i)$ returns the degree of the node i. One of the existing nodes is then randomly selected based on the above distribution to make connection with the new node. The Barabasi–Albert model [4], which is designed to generate scale-free networks using the preferential attachment mechanism, is one of the most commonly used models to produce artificial networks, because of its explanatory power, conceptual simplicity, and interesting mathematical properties [13]. The procedure to generate a PA-based graph with n nodes starts with a given initial clique with $m0$ nodes (a small complete graph). The remaining nodes are then added to the graph so that each new node makes m distinct connections with the existing graph ($1 \leq m \leq m0$) based on the distribution. The nodes are added sequentially (i.e., addition of the next node starts after terminating the addition of the current node) since, as shown in the above definitions, adding each new node influences the whole distribution.

Adopted from *Copy Model*, [8], we employ the array data structure to represent the graph. As depicted in Fig. 19.5a, from left to right each pair of array slots represents an edge of the array. In order to set up an array which represents the graph with the above-mentioned parameters, the array size is

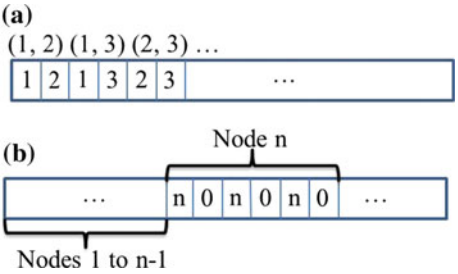


Fig. 19.5 The array representing the graph

$$S = init + 2m * (n - m0)$$

where *init* is the size of initial graph which can be a complete graph, $init = m0 * (m0 - 1)$. Figure 19.5b shows an abstraction of the array where the node n will be attached to the existing graph and $m = 3$. The array can be optimized in terms of memory since one slot of each pair for all the edges is calculable (e.g., n in Fig. 19.5b). However we ignore this optimization in this section. The next step is to resolve the unresolved slots for the node n (depicted by 0) according to the probability distribution of the existing nodes (i.e., $P_1, \ldots, P_{n-1}$). We simply use a uniform distribution over all the slots placed previous to the slots regarding node n since the number of occurrences of each node equals to its degree. Note that the values for the three unresolved slots must be distinct, which is simply checked via a function.

The sequential solution is fairly straightforward. Given the array with the initial graph at the beginning slots, according to above properties, the sequential solution is achieved via adding the nodes sequentially to the array.

However, the solution is more challenging in a parallel or distributed setting. To this aim, first of all the nodes (and the corresponding array) should be partitioned so that each partition is resolved by an actor. In the array, each node is represented by a sequence of slots where the first slot of each pair is the node's id (e.g., the slots regarding node n in Fig. 19.5b). If we consider all the partitioned arrays to be one virtual global array (like what we expect in the sequential approach) then the direction of dependencies and computations is depicted in Fig. 19.6. The arrow x in this figure shows a special kind of dependency, unresolved dependency, which

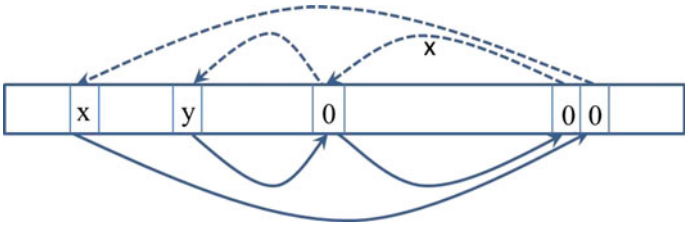


Fig. 19.6 The direction of dependencies (*right to left*) and computations (*left to right*)

shows the slot whose resolution is dependent on yet another unresolved slot, *target* slot. It is not difficult to see that unresolved dependencies only appear in the parallel solution. In order to remain consistent with the original PA model, the distributed approach keeps unresolved dependencies and uses the value of the target slot when it is resolved. How to keep the dependencies and use their target results after resolution is a low-level challenge. Figure 19.7 shows two different strategies to deal with this challenge. The first approach (Fig. 19.7a) is already examined in [1]. The second one (Fig. 19.7b) is adopted from [2], which is for multicore architecture, and tailored to fit the distributed setting. In the former case, the actor **b** places the request explicitly in a data structure and replies to it when the corresponding slot is resolved by the Actor. On the other side, actor **a** needs to keep track of the number of required responses corresponding to the requests. The latter however does not require such low level explicit synchronization management since it utilizes the notion of cooperative scheduling [5] via *await* on boolean conditions [2] to introduce a higher level of abstraction. Our implemented model follows the latter case.

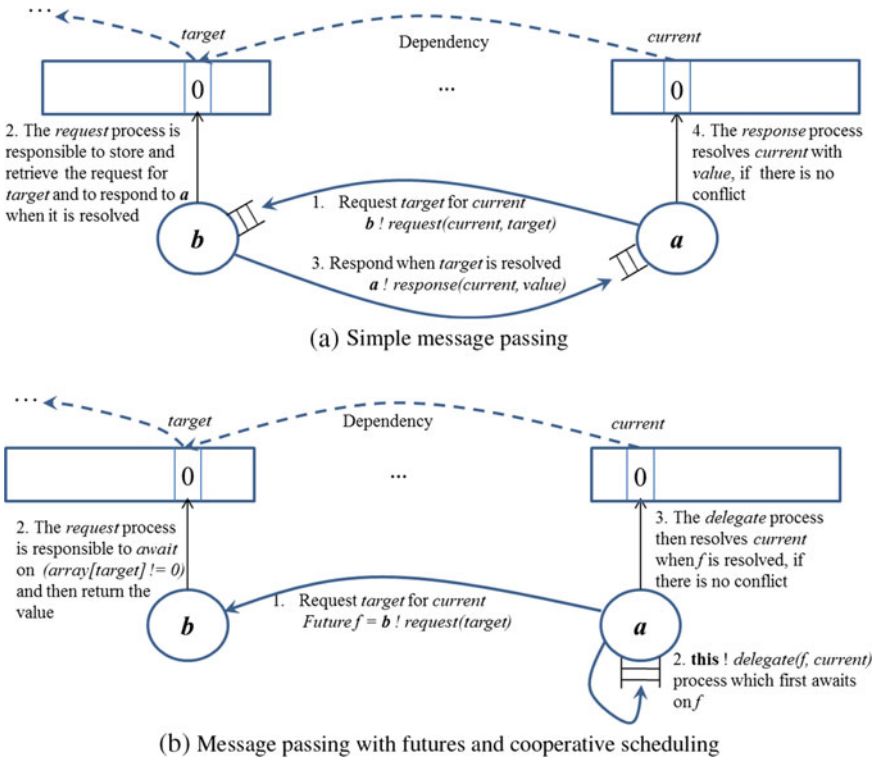


Fig. 19.7 The process of dealing with unresolved dependencies in an actor-based distributed setting

19.5 Implementation of the Algorithm Using the ABS-API Library

The implementation of the preferential attachment algorithm assumes a setting where the number of machines and actors is established a priori such that the application can assign predetermined global names to all the actors. In this manner all machines already have *RemoteActor* references to the actors they need to communicate with and corresponding communication streams setup as soon as all actors are instantiated and initialized. Figure 19.8 specifies our solution in a high-level pseudocode, which

```
 1: Each actor O executes the following in parallel
 2: run(...) : void
 3: for each Node i in the partition do
 4:     for  j = 2 to 2m do j = j + 2 step
 5:         target ← random[1..(i − 1) ∗ 2m]
 6:         current = (i − 1) ∗ 2m + j
 7:         x = whichActor(target)
 8:         actor[x]! request(target)
 9:         this! delegate( f , current)
10:
11:
12: request(target : Int) : Int void
13: localTarget = whichSlot(target)
14: await (arr[localTarget] ≠ 0)
15:                                          ▷ At this point the target is resolved
16: return arr[localTarget]
17:
18:
19: delegate( f : Future, current : Int) : void
20: await f?
21: value = f.get
22: localCurrent = whichSlot(current)
23: if duplicate(value, localCurrent) then
24:     target = random[1..current/(2m) ∗ 2m]
25:                                  ▷ Calculate the target for the current again
26:     x = whichActor(target)
27:     f = actor[x]! request(target)
28:     this. delegate( f , current)
29: else
30:     arr[localCurrent] = value                                    ▷ Resolved
31:
32:
33: duplicate(value : Int, localCurrent : Int) : Boolean
34: for each i in (indices of the node to which localCurrent belongs) do
35:     if arr[i] == value then
36:         return True
37: return False
```

Fig. 19.8 The sketch of the proposed approach

represents the scheme depicted in Fig. 19.7b. Each actor is responsible to resolve one partition of the virtual global array.

As shown in Fig. 19.5b, each node (as a new node) is associated with $2m$ slots of the array. Each actor starts processing its corresponding partition. The way array is partitioned has a considerable influence on the performance since it has a direct impact on the number of the unresolved dependencies (e.g., Consecutive and Round Robin Node Partitioning). In this section we abstract from the partitioning details. To this aim, we introduce two functions in the code. The function *whichSlot(i)* returns the local index corresponding to the virtual global index i, and the function *whichActor(i)* returns the actor index whose associated partition contains the local index corresponding to the virtual global index i.

The process *request* suspends on the boolean condition until it evaluates to *True*. The continuation is then queued and activated according to the actor's scheduling policy. The process *delegate* is also suspended until the future f is resolved. $f?$ returns a boolean value which represents whether the future is resolved or otherwise. Therefore the await on the future suspends the process until the future is resolved. The exclamation and dot are for asynchronous and synchronous method calls respectively. Finally the method *duplicate* checks whether the obtained value will cause a conflict, that is, a node makes two connections to the same target.

19.6 Conclusion

In this paper we presented a library to generate executable code in Java for an actor-based modeling language with very fine-grained scheduling heuristics formal analysis and verification tools. In this implementation we added support for distributed actors, future propagation and significantly reduced the number of threads created and alive throughout the application's lifetime ensuring efficient memory consumption and performance. We offered an enhanced API for distributed communication and explicit control of synchronization. Our focus was on the abstract behavioral specification language which represents an excellent solution for modeling cloud applications and this implementation allows the language to be extended with cooperative scheduling capabilities and powerful scheduling algorithms. We presented the details of how our new solution uses the latest Java 8 concurrent library to map the ABS constructs that invoke the scheduler and also ensure transparency with respect to actor's locations. We motivated our contribution by outlining the implementation of a specific scenario for generating large-social network graphs which can be deployed in a distributed environment using this library. The next step to this scientific work is to integrate this implementation into the ABS compiler that is currently in use to translate ABS code into executable Java code and investigate how to provide syntactic sugaring for ABS asynchronous method invocations. This will allow the direct modeling of our case study using ABS and testing it against the state-of-the art implementation in MPI.

Acknowledgments Partly funded by the EU project FP7-610582 ENVISAGE: Engineering Virtualized Services (http://www.envisage-project.eu). Partly funded by the EU project FP7-612985 UPSCALE: From Inherent Concurrency to Massive Parallelism through Type-based Optimizations (http://www.upscale-project.eu).

References

1. Alam, M., Khan, M., Marathe, M.V.: Distributed-memory parallel algorithms for generating massive scale-free networks using preferential attachment model. Proceedings of the International Conference on High Performance Computing, p. 91. Storage and Analysis, ACM, Networking (2013)
2. Azadbakht, K., Bezirgiannis, N., De Boer, F.S., Aliakbary, S.: A high-level and scalable approach for generating scale-free graphs using active objects. In: Proceeding of the ACM/SIGAPP Symposium on Applied Computing, To appear (2016)
3. Bader, D.A., Madduri, K.: Parallel algorithms for evaluating centrality indices in real-world networks. In: International Conference on Parallel Processing, 2006. ICPP 2006, 539–550. IEEE (2006)
4. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
5. De Boer, F.S., Clarke, D., Johnsen, E.B.: A complete guide to the future. In: Programming Languages and Systems, pp. 316–330. Springer (2007)
6. Johnsen, E.B., Hähnle, R., Schäfer, J., Schlatte, R., Steffen, M.: Abs: A core language for abstract behavioral specification. In: Formal Methods for Components and Objects, pp. 142–164. Springer (2010)
7. Johnsen, E.B., Schlatte, R., Tarifa, S.L.T.: Modeling resource-aware virtualized applications for the cloud in real-time abs. In: Formal Methods and Software Engineering, pp. 71–86 Springer (2012)
8. Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tomkins, A., Upfal, E.: Stochastic models for the web graph. In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, 2000, pp. 57–65. IEEE (2000)
9. Nobakht, B., de Boer, F.S.: Programming with actors in java 8. In: Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications, pp. 37–53. Springer (2014)
10. Serbanescu, V., Azadbakht, K., Boer, F., Nagarajagowda, C., Nobakht, B.: A Design Pattern for Optimizations in Data Intensive Applications Using Abs and Java 8. Practice and Experience, Concurrency and Computation (2015)
11. Serbanescu, V., Nagarajagowda, C., Azadbakht, K., de Boer, F., Nobakht, B.: Towards type-based optimizations in distributed applications using abs and java 8. In: Adaptive Resource Management and Scheduling for Cloud Computing, pp. 103–112. Springer (2014)
12. Serbanescu, V.N., Pop, F., Cristea, V., Achim, O.M.: Web services allocation guided by reputation in distributed soa-based environments. In: 2012 11th International Symposium on Parallel and Distributed Computing (ISPDC), pp. 127–134. IEEE (2012)
13. Tonelli, R., Concas, G., Locci, M.: Three efficient algorithms for implementing the preferential attachment mechanism in yule-simon stochastic process. WSEAS Trans. Inf. Sci. Appl. **7**(2), 176–185 (2010)
14. Yoo, A., Henderson, K.: Parallel generation of massive scale-free graphs (2010). arXiv preprint arXiv:1003.3684

Chapter 20
Predicting Video Virality on Twitter

Irene Kilanioti and George A. Papadopoulos

20.1 Introduction

The diffusion of video content is fostered by the ease of producing online content via media services. It mainly happens via ubiquitous Online Social Networks (OSNs), where social cascades can be observed when users increasingly repost links they have received from others. Twitter is one of the most popular OSNs with its core functionality centered around the idea of spreading information by word-of-mouth [16]. It provides mechanisms such as retweet (forwarding other people's tweets), which enable users to propagate information across multiple hops in the network.

If we knew beforehand when a social cascade will happen or to what range it will evolve, we could exploit this knowledge in various ways. For example, in the area of content delivery infrastructure, we could prefetch content by replicating popular items and subsequently spare bandwidth. The knowledge of the evolution of social cascades could lead to reduction schemes for the storage of whole sequences of large social graphs and the reduction of their processing time.

Towards this direction, in this work we present a model for efficiently calculating the number of retweets of a video. The number of retweets is associated with a score depicting the influence of its uploader in the Twitter dataset, the increasing or decreasing trend the score depicts as well as the distance of content interests among users of the YouTube and Twitter community.

I. Kilanioti (✉) · G.A. Papadopoulos (✉)
Department of Computer Science, University of Cyprus, 1 University Avenue,
P.O. Box 20537, 2109 Nicosia, Cyprus
e-mail: ekoila01@cs.ucy.ac.cy

G.A. Papadopoulos
e-mail: george@cs.ucy.ac.cy

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_20

20.1.1 Contributions

Our work focuses on video virality over an OSN. Study of social cascades is active aiming at the prediction of the aggregate popularity of a resource or the individual behaviour of a user. Few works, however, combine detailed information both of the OSN and the media service with a small and easily extracted feature set. Our study proposes a prediction model that performs better than methods like support vector machines (SVM), stochastic gradient descent (SGD) and K-Nearest Neighbours (KNN), among others, and we, furthermore, proceed to incorporate our prediction model into a mechanism for content delivery with substantial improvement for the user experience.

The remainder of this paper is organized as follows. Section 20.2 reviews previous related work. Section 20.3 formally describes the addressed problem. Section 20.4 provides an outline of the methodology, followed by the preparation of the employed datasets. Our main findings are presented in Sect. 20.5, where also a validation is conducted. Section 20.6 investigates the incorporation of the proposed model into a content delivery mechanism. Section 20.7 concludes the work and discusses directions for future work.

20.2 Related Work

The field of predicting social virality is active [2, 5, 6, 13, 15, 17, 22], etc. Many studies focus on the prediction of the amount of aggregate activities (e.g. aggregate daily hashtag use [14]), whereas others focus either on the prediction of user-level behaviour, like retransmission of a specific tweet/URL [7, 15] or on the prediction of growth of the cascade size [5].

Although our work focuses solely on video sharing, we identify the following methods for virality prediction in general. Feature-based methods and time series analysis methods. They are both based on the empirical observation of social cascades. Our approach falls into the first category.

Feature-based methods are based on content, temporal and other features, and the learning algorithms schemes they use are based on simple regression analysis [5, 18], regression trees [2], content-based methods [19], binary classification [8, 9, 12] etc. They do not focus, though, on the underlying network infrastructure, and often encounter difficulty in extracting all the necessary features due to the large volume of accommodated graphs.

Time-series analysis works [20, 21], on the other hand, argue that patterns of a resource's growth of popularity are indicative for its future retransmissions.

Finally, we should mention that one branch of virality research is based on study of the evolution of cascades during a specific time-window [12, 14, 19], whereas there exist works that examine the cascades continuously over their entire duration [5].

20.3 Problem Description

We consider a directed graph $G(t) = (V(t), E(t))$ representing a social network that evolves through time, consisting at time t of V vertices and E edges. Edges between the nodes of the graph denote friendship in case of a social network (e.g. for Twitter B is a follower of A if there is an edge between B and A pointing at A).

Our problem is stated as follows (Table 20.1). We want to predict the number of retransmits of a video link by a user $v \in V$ after $u \in V$ has transmitted the link. User v is a follower of u.

We express this number, intuitively, as a combination of the following features: the $Score(u, t)$ of node u, $dScore(u, t)/dt$ of node u, and content distance between the content interests of the involved users both in the OSN and the media service. The validity of the predictors is analyzed in this paper. The intuition for their selection is based on the notion, that, the higher influence score a node depicts, the more influence it is expected to exert on other nodes of the social graph. Moreover, the $dScore/dt(u, t)$ expresses the popularity rise/fall of the node, and, lastly, the content distance associates the resource with the user context.

Denoting the output, the predicted output and the total number of predicted values by A_{u2v}, $\widehat{A_{u2v}}$ and M, we aim to find the values α, β, γ, so that:

$$A_{u2v} = \alpha \times Score(u, t) + \beta \times \frac{d\,Score(u, t)}{dt} + \gamma \times content\_dist \tag{20.1}$$

and

$$\sqrt{\frac{1}{M}\sum_{i=1}^{M}(\widehat{A_{u2v}} - A_{u2v})^2} \tag{20.2}$$

is minimum.

Table 20.1 Notation overview

$G(t) = (V(t), E(t))$	OSN graph G at time t of V vertices and E edges
A_{u2v}	Number of actions where u influenced v
$\widehat{A_{u2v}}$	Predicted output
M	Total number of predicted values
α, β, γ	Coefficients of feature set variables
U	Vector of YouTube interests of user u
V	Vector of Twitter interests of user v
Features set	
$Score(u, t)$	Score of node u at time t
$dScore = dScore(u, t)/dt$	Derivative of Score of node u at time t
$content\_dist$	Content distance

20.4 Proposed Methodology

20.4.1 Dataset

Interests of users were analyzed in [1] against directory information from http://wefollow.com, a website listing Twitter users for different topics, including Sports, Movies, News & Politics, Finance, Comedy, Science, Non-profits, Film, Sci-Fi/Fantasy, Gaming, People, Travel, Autos, Music, Entertainment, Education, Howto, Pets, and Shows.

The activity of Twitter users was quantified, and a variety of features were extracted, such as the number of their tweets, the fraction of tweets that were retweets, the fraction of tweets containing URLs, etc. Aggregated features of YouTube videos shared by a user in the dataset include the average view count, the median inter-event time between video upload and sharing, etc.

A sharing event in the dataset is defined as a tweet containing a valid YouTube video ID (with a category, Freebase topic and timestamp). We augmented the provided dataset with Tweet content information about the 15 million video sharing events included in the dataset, as well as information about the followers of the 87 K Twitter users.

20.4.2 User Score Calculation

A user score is calculated combining the number n of its followers, reduced by a factor of 1000 to compensate the wide range of followers in the dataset from zero to more than a million, a quantity b catering for users with reciprocal followership, calculated by taking an average of number of a user's followers to the number of users he follows, as well as the effect e of a user's tweet, measured by multiplying average number of retweets with number of user's tweets and normalizing it to correspond to the total number of tweets. The distribution of these combined metrics depicts large variance and we have applied a logarithmic transformation in order to avoid the uneven leverage of extreme values.

$$Score = \log\left(n + \left(\left(\frac{b}{100}\right) \times n\right) + e\right) \tag{20.3}$$

20.4.3 Content Distance

The content distance $content\_dist$ expresses a measure of similarity of user's u YouTube and his follower's v Twitter interests. Content distance is calculated using

cosine similarity between vectors of user's u YouTube and user's v Twitter video interests, as follows:

$$content\_dist = 1 - \frac{U \cdot V}{\|U\|\|V\|} \tag{20.4}$$

20.5 Experimental Evaluation

By combining user ids, followership information, user features and tweet context we build a measure of A_{u2v}, expressing the number of times a user's u tweet is retweeted by his followers v. We aim to associate the independent variables of the features set (X dataframe) with the series depicting A_{u2v} (y) (Table 20.2).

20.5.1 Selection of Predictors

The regression summary of Table 20.3 shows that coefficients of all predictors are significant ($P > |t|$ is significantly less than 0.05). Therefore, $Score$, $dScore$ and $content\_dist$ can be considered as good predictors. We note that t here refers to $t - statistic$, denoting the quotient of the coefficient of dependent variable divided by coefficient's standard error. P refers to the $P - value$, a standard statistical method for testing an hypothesis. $P - value < 0.05$ means we can reject the hypothesis that the coefficient of a predictor is zero, in other words the examined coefficient is significant (Table 20.4).

Table 20.2 Regression results (i)

Dep. variable	A_{u2v}	R-squared	0.396
Model	OLS	Adj. R-squared	0.396
Method	Least squares	F-statistic	1.570e+04
Prob (F-statistic)	0.00	Log-likelihood	−8576.9
No. observations	71952	AIC	1.716e+04
Df residuals	71949	BIC	1.719e+04
Df model	3	Covariance type	nonrobust

Table 20.3 Regression results (ii)

	Coef	Std err	t	$P > \|t\|$	95 %	Conf. int.
$Score$	5.79e-05	3.78e-06	15.300	0.000	5.05e-05	6.53e-05
$dScore$	4.36e-05	4.51e-06	9.667	0.000	3.48e-05	5.25e-05
con_dist	0.389	0.002	213.060	0.000	0.386	0.393

Table 20.4 Regression results (iii)

Omnibus	2091.840	Durbin-Watson	1.723
Prob(Omnibus)	0.000	Jarque-Bera (JB)	2323.421
Skew	0.408	Prob(JB)	0.00
Kurtosis	3.333	Cond. No.	746

The selection of the above predictors comes as a result of comparing the $P - values$ of various metrics in the dataset and the combination of those with the lowest $P - value$. The metrics included the number of distinct users retweeted, fraction of the users tweets that were retweeted, average number of friends of friends, average number of followers of friends, number of YouTube videos shared, the time the account was created, the number of views of a video, etc., among many others.

20.5.2 Effect of Outliers

The regression plots for each predictor in Fig. 20.1 show the effect of outliers on the estimated regression coefficient. Regression line is pulled out of its optimal tracjectory due to the existent outliers. The detailed regression plots for individual predictors ($Score$, $dScore$ and $content\_dist$) appear in Figs. 20.2, 20.3, and 20.4

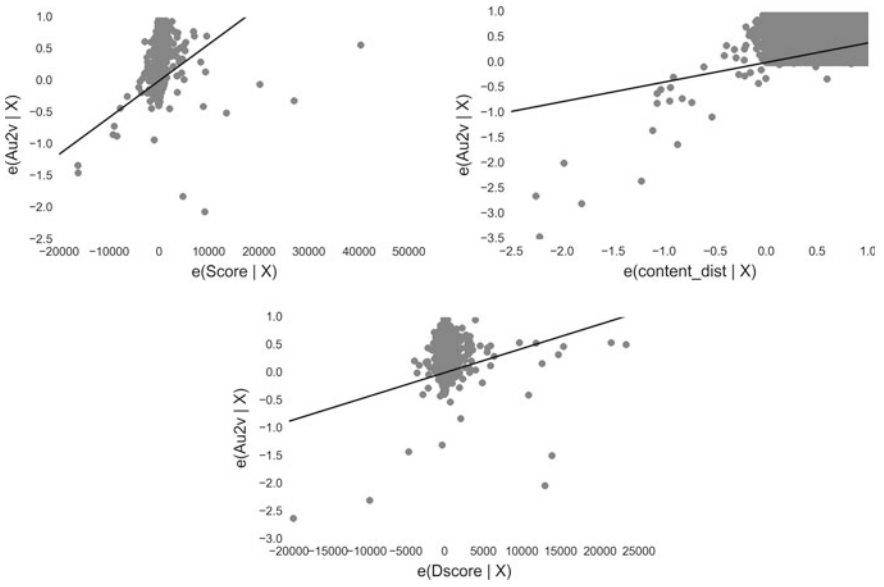


Fig. 20.1 Regression plots for each independent variable

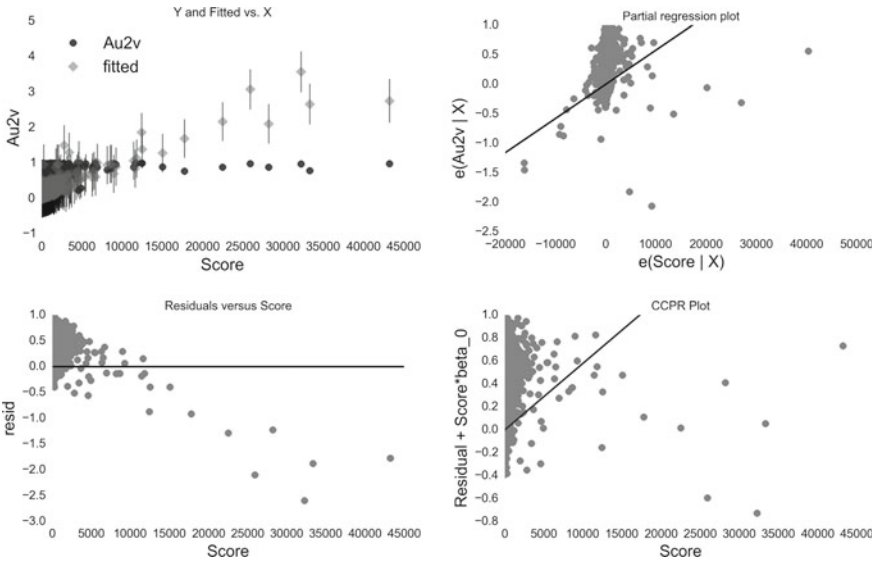


Fig. 20.2 Regression plots for $Score$

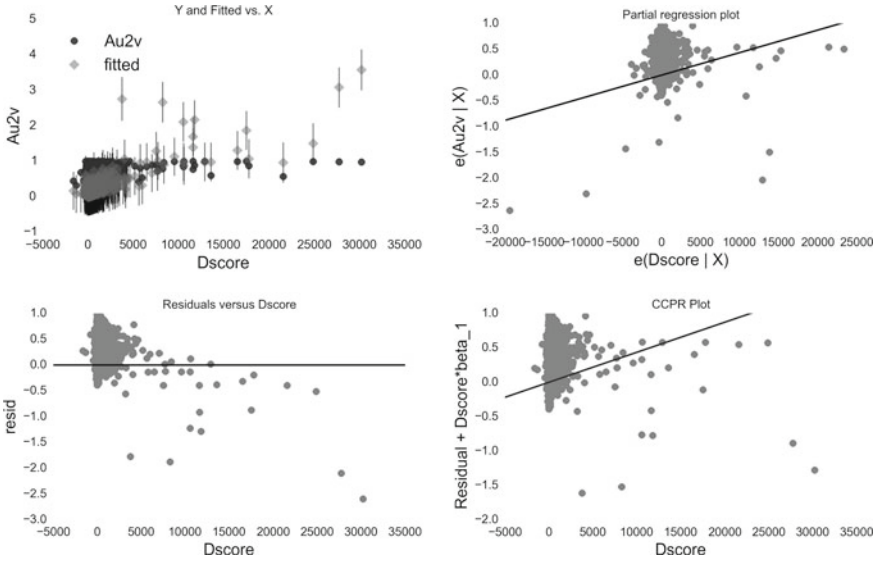


Fig. 20.3 Regression plots for $dScore$

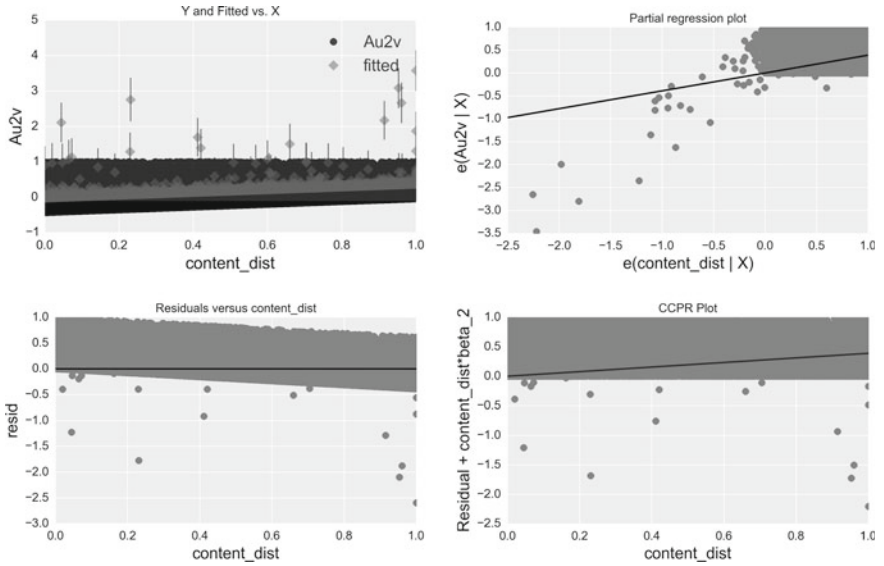


Fig. 20.4 Regression plots for *content_dist*

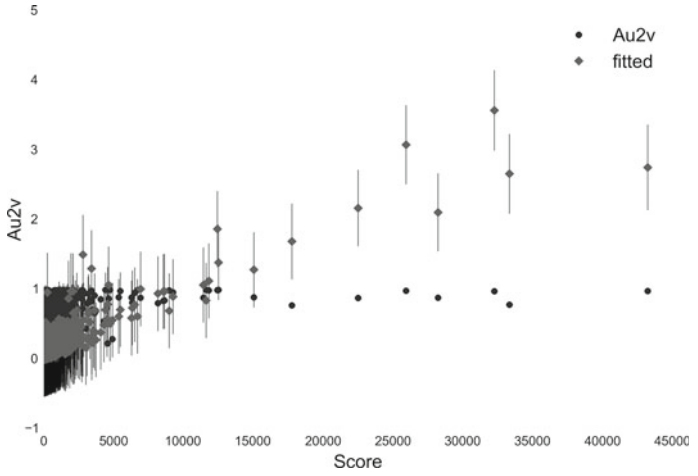


Fig. 20.5 Fitted values of A_{u2v} versus *Score*

respectively. The fitted (predicted) values of A_{u2v} and the prediction confidence for each independent variable appear in Figs. 20.5, 20.6, and 20.7. We observe that fitted values are quite close to the real values of A_{u2v} with the exception of the outliers. This suggests that removal of outliers would yield a better estimate, since it is obvious that the plot is skewed due to their presence.

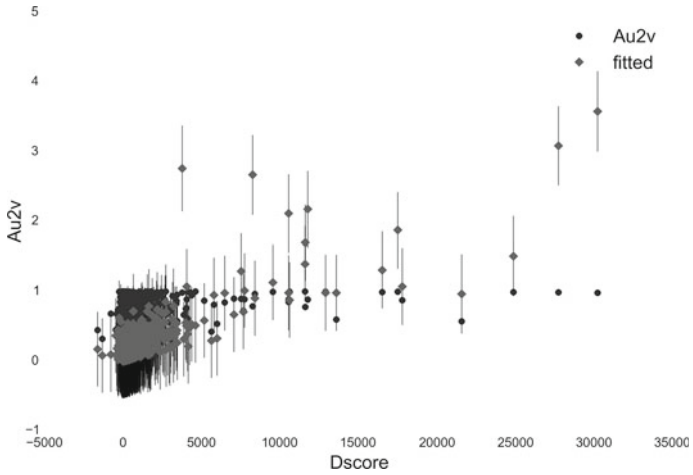


Fig. 20.6 Fitted values of A_{u2v} versus $dScore$

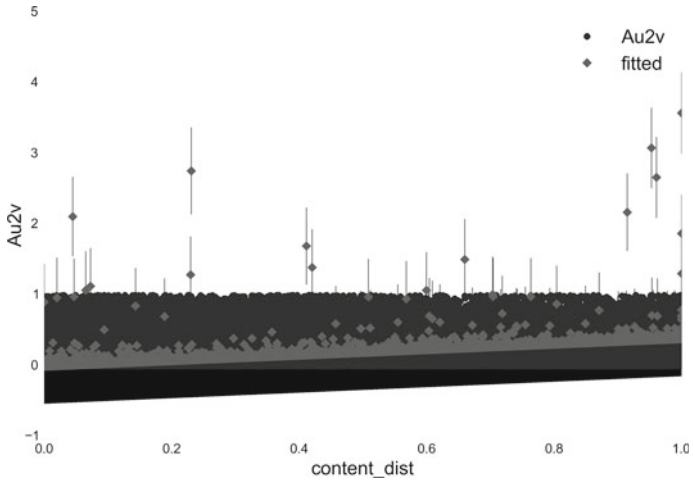


Fig. 20.7 Fitted values of A_{u2v} versus $content\_dist$

A rough estimate of detecting outliers can be based on the quantile distributions of each independent variable in Table 20.5. Observing Table 20.5 with an overview of data distribution we surmise that we could take values of $Score$ and $dScore$ only upto 10 and 5, respectively. The quantiles appearing on the table are calculated when data is rearranged in ascending order and divided into four equal sized parts. Thus, interpreting the second quantile we notice that 50 % of Score values are less than 2.562232. In the table, we notice that we have huge maximum values for $Score$ and $dScore$, but 75 % of the data are below 6.902750 and 2.308805, respectively. Thus,

Table 20.5 Outliers thresholds

	Score	*dScore*	*Content_dist*
Count	71952.000000	71952.000000	71952.000000
Mean	21.227703	15.880803	0.459111
Std	349.102908	292.727717	0.315837
Min	0.000000	–1610.253490	0.000000
25 %	0.787060	0.000140	0.172534
50 %	2.562232	0.526050	0.415394
75 %	6.902750	2.308805	0.724986
Max	43262.678131	30235.027960	1.000000

Table 20.6 Regression results without outliers (i)

Dep. variable	A_{u2v}	R-squared	0.629
Model	OLS	Adj. R-squared	0.629
Method	Least Squares	F-statistic	3.072e+04
Prob (F-statistic)	0.00	Log-Likelihood	13947
No. Observations	54473	AIC	−2.789e+04
Df Residuals	54470	BIC	−2.786e+04
Df Model	3	Covariance Type	nonrobust

Table 20.7 Regression results without outliers (ii)

	Coef	Std err	t	$P > \|t\|$	95 %	Conf.Int.
Score	0.1460	0.001	145.244	0.000	0.144	0.148
dScore	0.0200	0.001	25.819	0.000	0.018	0.022
con_dist	0.1656	0.003	65.690	0.000	0.161	0.171

Table 20.8 Regression results without outliers (iii)

Omnibus	10848.216	Durbin-Watson	1.966
Prob(Omnibus)	0.000	Jarque-Bera (JB)	22428.486
Skew	1.183	Prob (JB)	0.00
Kurtosis	5.070	Cond. No.	5.19

we select 10 and 5 as values to take most of the data and exclude data points with extremely large out of general range values (outliers).

Results of regression model on data obtained after removing outlier data points appear in Tables 20.6, 20.7, and 20.8. The results show considerable improvement with respect to regression with presence of outliers (Tables 20.2, 20.3 and 20.4). Also, Durbin–Watson statistic close to 2 confirms normality assumption of residu-

als, verifying the normality of error distribution, one of the assumptions of linear regression.

Figure 20.10 plots reinforce the argument that after removing outliers we get a better fit of regression line on each independent variable. Namely, the removal of outliers leads to better alignment of the path of regression line to the optimal path.

20.5.3 Tenfold Cross-Validation

We performed a tenfold cross validation on the dataset, fitting the regressor to 90 % of the data and validating it on the rest 10 % for the prediction of A_{u2v} dependent variable from $Score$, $dScore$ and $content - dist$ independent variables. Predictive modeling was conducted after removing outliers from the data. The results of the predictive modeling using linear regression show that we achieve a root mean squared error of 0.1873 (across all folds), which means that our prediction varies by 0.1873 from the real values of A_{u2v}. This shows a considerable improvement in prediction error compared to modeling with original data, where a root mean squared error of 0.2728 across all folds was achieved. Plots in both cases appearing in Figs. 20.8 and 20.9 depict how close our predictions are to the real values of the dependent variable (Fig. 20.10).

20.5.4 Classification and Comparison with Other Models

We predict a user popularity as follows. If A_{u2v} crosses a threshold, e.g. 30 %, i.e., if more than 30 % tweets of user u are retweeted by others users, then user u can be considered as a popular user.

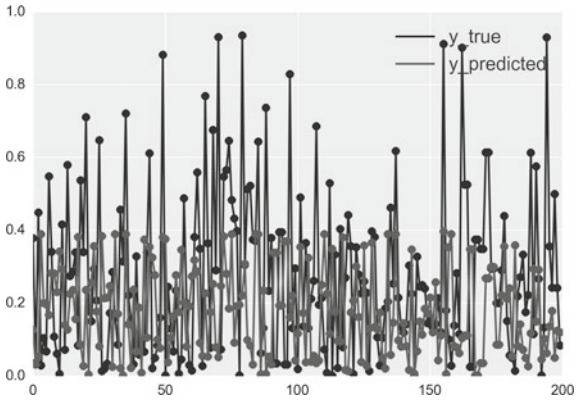


Fig. 20.8 tenfold cross-validation of A_{u2v}

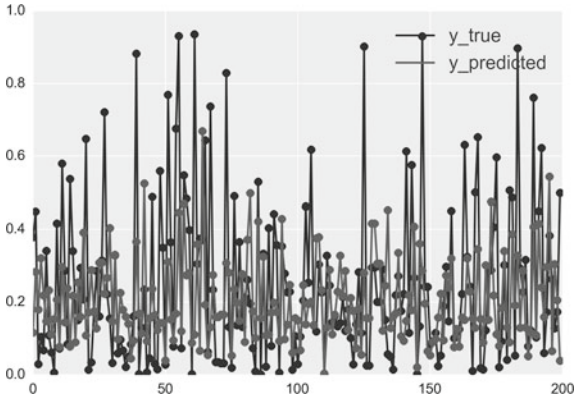


Fig. 20.9 tenfold cross-validation of A_{u2v} without outliers

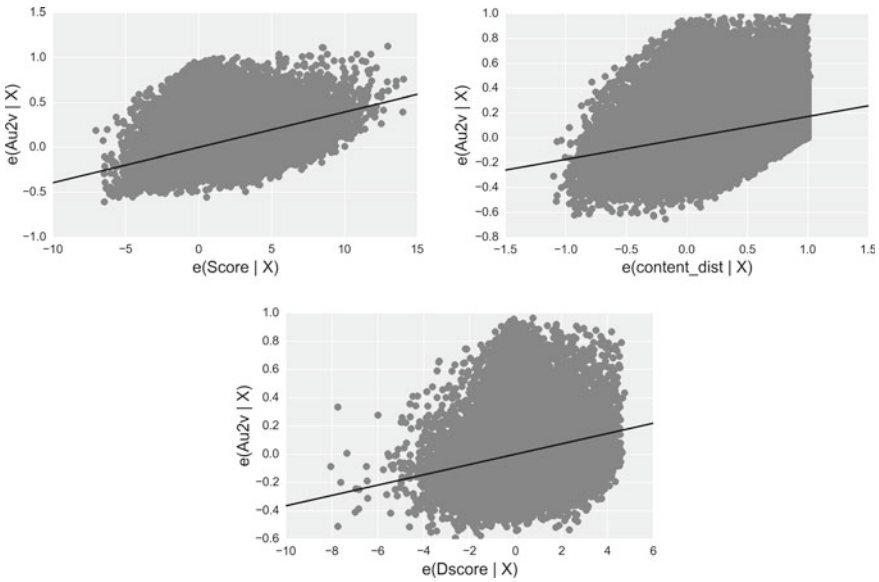


Fig. 20.10 Regression plots for each independent variable

Classification was conducted initially with three different methods: Linear Regression, i.e., the Predictive Model we present in this study, Random Forest and Naive Bayes methods. Area Under the Curve (AUC) is a score that computes average precision (AP) from prediction scores. This average precision score corresponds to the area under the precision-recall curve and the higher AUC represents better performance. Plots in Fig. 20.11 correspond to computed precision-recall pairs for different probability thresholds and the AUC score computes the area under these curves. Best performance is achieved by Linear Regression (0.699), followed by

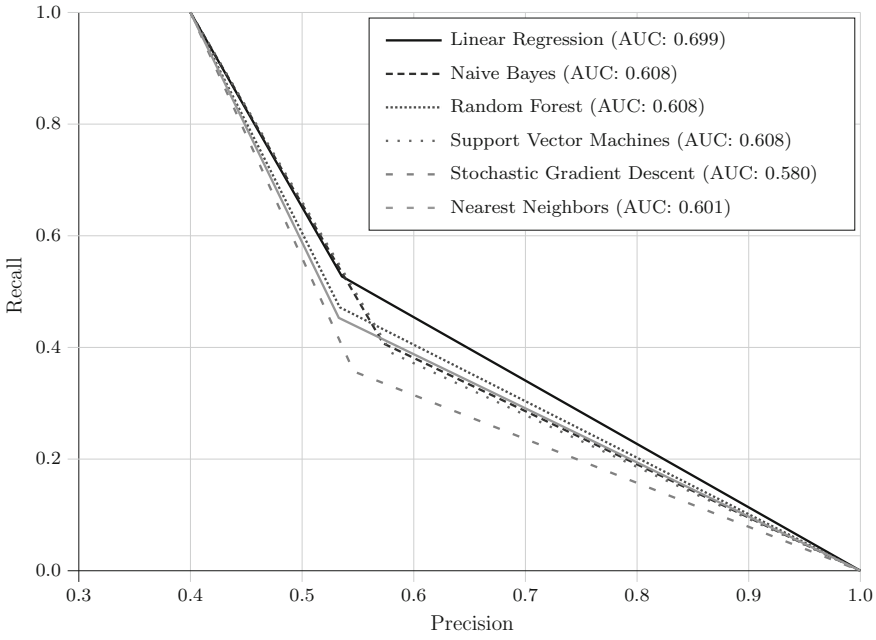


Fig. 20.11 Comparison with other models

Naive Bayes (AUC:0.608) and Random Forest (AUC:0.608). Complementary methods tested were support vector machines (SVM), stochastic gradient descent (SGD) and K-Nearest Neighbours (KNN).

SVM is a supervised learning model with associated learning algorithm that analyzes data used for classification and regression analysis. Given a set of training examples, each marked to belong to one of the two categories (popular/non-popular user), the SVM training algorithm builds a model that assigns new examples into each of the categories, acting as a non-probabilistic binary linear classifier.

Next classification model was stochastic gradient descent (SGD), a gradient descent optimization method for minimizing an objective function written as a sum of differentiable functions. It encompasses a popular algorithm for training a wide range of models in machine learning, including linear support vector machines, logistic regression and graphical models. Its use for training artificial networks is motivated by the high cost of running backpropagation algorithm over the full training set, as SGD overcomes this cost and still leads to fast convergence.

The last classifier implemented here was K-Nearest Neighbours (KNN), a method classifying objects based on closest training examples in the feature space. The input consists of positive, typically small, integer -15 in our case—of closest training examples in the feature space. In KNN classification, the output is a class membership (popular/non-popular user), whereas an object is classified by a majority vote of its

neighbours, with the object being assigned to the class most common among its K-Nearest Neighbours.

After plotting the results of computed precision-recall pairs for various probability thresholds we observe that best performance is noticed in the case of our Predictive Model, followed by Naive Bayes (AUC:0.608), Random Forest (AUC:0.608), SVM (AUC:0.608), KNN (AUC:0.601), and, lastly, SGD (AUC:0.580).

20.6 Incorporation into Content Delivery Schemes

Content Distribution Networks (CDNs) aim at improving download of large data volumes with high availability and performance. Content generated by online media services circulates and is consumed over OSNs (with more than 400 tweets per minute including a YouTube video link [3] being published per minute). This content largely contributes to internet traffic growth [4]. Consequently, CDN users can benefit from an incorporated mechanism of social-awareness over the CDN infrastructure. In [10, 11] Kilanioti and Papadopoulos introduce a dynamic mechanism of preactive copying of content to an existing validated CDN simulation tool and propose various efficient copying policies based on prediction of demand on OSNs.

Rather than pushing data to all surrogates, they proactively distribute it only to social connections of the user likely to consume it. The content is copied only under certain conditions (content with high viewership within the media service, copied to geographically close timezones of the geo-diversed system used where the user has mutual social connections of high influence impact). This contributes to smaller response times for the content to be consumed (for the users) and lower bandwidth costs (for the OSN provider). Herein, we incorporate the proposed Predictive Model in the suggested policy [11] and prove that it further improves its performance.

The proposed algorithm encompasses an algorithm for each new request arriving in the CDN and an algorithm for each new object in the surrogate server (Table 20.9). Internally, the module communicates with the module processing the requests and each addressed server separately (Fig. 20.12).

- *For Every New Request in the CDN*
  Prinicipally we check whether specific time has passed after the start of cascade and, only in the case that the cascade has not ended, define to what extent the object will be copied. We introduce the *time_threshold* that roughly expresses the average cascade duration. The main idea is to check whether specific time has passed after the start of the cascade, and then define to what extent the object will be copied. Initially, we check whether it is the first appearance of the object (Fig. 20.13). The variable *o.timestamp* depicts the timestamp of the last appearance of the object in a request and helps in calculating the timer related to the duration of the cascade. If it is the first appearance of the object, the timer for the object cascade is initialized and *o.timestamp* takes the value of the timestamp of the request. If the cascade is

Table 20.9 Content delivery verification—notation overview

$G(t) = (V(t), E(t))$	Graph representing the social network
$V(t) = \{V_1(t), \ldots, V_n(t)\}$	Nodes representing the social network users
$E(t) = \{E_{11}(t), \ldots, E_{1n}(t), \ldots, E_{nn}(t)\}$	Edges representing the social network connections, where E_{ij} stands for friendship between i and j
$R = \{r_1, r_2, \ldots, r_\tau\}$	Regions set
$N = \{n_1, n_2, \ldots, n_\upsilon\}$	The surrogate servers set. Every surrogate server belongs to a region r_i
$C_i, i \in N$	Capacity of surrogate server i in bytes
$O = \{o_1, o_2, \ldots, o_w\}$	Objects set (videos), denoting the objects users can ask for and share
$S_i, o_i \in O$	Size of object i in bytes
Π_i	Popularity of object $i, i \in O$
$q_i = \{t, V_\psi, o_x\}, < x < w, 1 < \psi < n$	Request i consists of a timestamp, the id of the user that asked for the object, and the object id
$P = \{p_{12}, p_{13}, \ldots, p_{nw}\}$	User posts in the social network, where p_{ij} denotes that node i has shared object j in the social network
$pts_i, pte_i, 1 < i < \tau$	Peak time start and peak time end for each region in secs
$Q = \{q_1, q_2, \ldots, q_\zeta\}$	Object requests from page containing the media objects, where q_i denotes a request for an object of set O
Q_{hit}, Q_{total}	Number of requests served from surrogate servers of the region of the user/total number of requests
$X, Y \in R$	Closest timezones with mutual followers/with highest centrality metric values

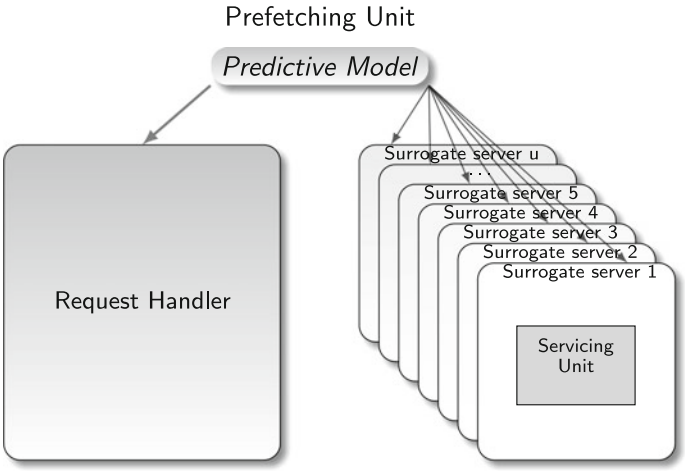


Fig. 20.12 The social-aware CDN mechanism

```
1: if o.timestamp == 0 then
2:     o.timer = 0;
3:     o.timestamp = request_timestamp;
4: else if o.timestamp != 0 then
5:     o.timer = o.timer + (request_timestamp - o.timestamp);
6:     o.timestamp = request_timestamp;
7: end if
8: if o.timer > time_threshold then
9:     o.timer = 0;
10:    o.timestamp = 0;
11: else if o.timer < time_threshold and user.Score > Score_threshold then
12:    copy object o to surrogate that serves user's V_i(t) timezone;
13:    for all user V_y(t) that follows user V_i(t) do
14:        find surrogate server n_j that serves V_y(t)'s timezone;
15:        copy object o to n_j;
16:    end for
17: else if o.timer < time_threshold then
18:    copy object o to surrogates n_j that Subpolicy decides;
19: end if
```

Fig. 20.13 Algorithm for every new request ($timestamp$, $V_i(t)$, o) in the CDN

not yet complete (its timer has not surpassed a threshold), we check the importance of the user applying its Score.

For users with Score surpassing a threshold (average value: 1.2943 in the dataset), we copy the object to all surrogate servers of the user's timezone and to the surrogate servers serving the timezones of all user's followers. Otherwise, selective copying includes only the surrogates that the subpolicy decides. Subpolicy (Fig. 20.14) checks the X closest timezones where a user has mutual friends and out of them, the Y with the highest value of the combined feature set (Predictive Model($Score$, $dScore$, $content\_dist$)) as an average. Copying is performed to the surrogate servers that serve the Y timezones of highest combined feature set value, according to the coefficients derived from our analysis. We note here that variations of the Subpolicy include the replacement of the timezones depicting the highest average values of Predictive Model($Score$, $dScore$, $content\_dist$), with those being derived from the application of Naive Bayes, Random Forest, SVM, SGD, and KNN schemes.

```
1: find X timezones where (user V_i(t) has mutual followers and they are closer to user's V_i(t)
   timezone);
2: find the Y ⊆ X that (belong to X and depict the highest average values of Predictive
   Model(Score, dScore, content_dist));
3: for all timezones that belong to Y do
4:     find surrogate server n_j that serves timezone;
5:     copy object o to n_j;
6: end for
```

Fig. 20.14 Subpolicy

```
1: if o.size + current_cache_size ≤ total_cache_size then
2:     copy object o to cache of surrogate n_k;
3: else if o.size + current_cache_size > total_cache_size then
4:     while o.size + current_cache_size > total_cache_size do
5:         for all object o′ in current_cache do
6:             if (current_timestamp - o′.timestamp) + o′.timer > time_threshold then
7:                 copy o′ in CandidateList;
8:             end if
9:             if CandidateList.size>0 and CandidateList.size != total_cache_size then
10:                find o′ that o′.timestamp is maximum and delete it;
11:            else if CandidateList.size==0 or CandidateList.size==total_cache_size then
12:                use LRU to delete any object o ∈ O;
13:            end if
14:        end for
15:    end while
16:    put object o to cache of surrogate n_k;
17: end if
```

Fig. 20.15 Algorithm for every new object o in the surrogate server n_k

- *For Every New Object in the Surrogate Server*
  Surrogate servers keep replicas of the web objects on behalf of content providers. In the case that the new object does not fit in the surrogate server's cache, we define the *time_threshold* as the parameter for the duration that an object remains cached. We check for items that have remained cached for a period longer than the *time_threshold* and we delete those with the largest timestamp in the cascade. In case there exist no such objects or all objects have the same timestamp, we prune the least recently used items first. To ensure that least recently used items are discarded, the algorithm keeps track of their usage (Fig. 20.15).

The nodes representing the surrogate servers, the origin server, and the users requesting the object (Fig. 20.16) in the simulated network topology are analyzed in detail in [10]. To simulate our policy and place the servers in a real geographical position, we used the geographical distribution of the Limelight network.

For the smooth operation of the simulator the number of surrogate servers was reduced by a ratio of 10 %, to ultimately include 423 servers. Depending on the closer distance between the surrogate region defined by Limelight and each of the timezones defined by Twitter (20 Limelight regions, 142 Twitter timezones), we decided where the requests from each timezone will be redirected. The population of each timezone was also taken into consideration. The INET generator [4] allowed us to create an AS-level representation of the network topology. Topology coordinates were converted to geographical coordinates with the NetGeo tool from CAIDA, a tool that maps IP addresses and Autonomous System (AS) coordinates to geographical coordinates, and surrogate servers were assigned to topology nodes. After grouping users per timezone (due to the limitations the large dataset imposes), each group of users was placed in a topology node. We placed the user groups in the nodes closer to those comprising the servers that serve the respective timezone requests, contributing this way to a realistic network depiction.

The heuristics applied in [11] are based on the observation that users are more influenced by geographically close friends, and moreover by mutual followers, as

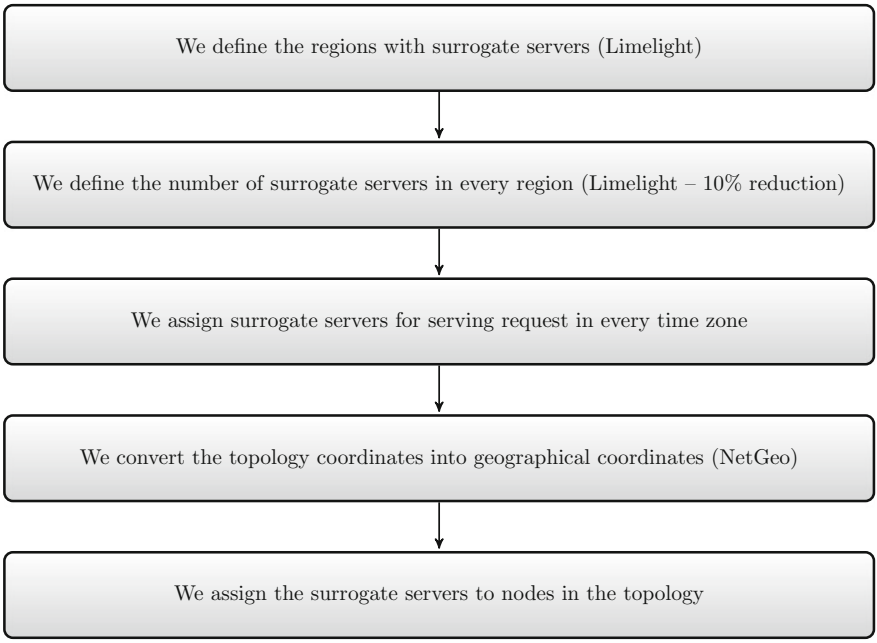


Fig. 20.16 Methodology followed

well as on the short duration of social cascades (about 80 % of the cascades end within 24 h, with 40 % of them ending in less than 3 h). In our prefetching algorithm, we introduce varying time thresholds for the cascade effect and the time an object remains in cache. Values given in the time threshold variable include thresholds covering the entire percentage of requests.

We examine Mean Response Time (MRT), a client-side metric that indicates how fast a CDN client is satisfied, for the most representative case of time threshold covering all the examined requests of our dataset. The trade-off between the reduction of the response time and the cost of copying in servers is expressed for all schemes used (Linear Regression, Naive Bayes, Random Forest, SVM, SGD, KNN) with an MRT decrease as the timezones increase and a point after which the MRT starts to increase again (Fig. 20.17). For the scheme augmented with our Predictive Model, namely the Linear Regression, this shift occurs with approximately 6 timezones out of the 10 used (for a fixed number of closest timezones with mutual followers). After this point the slight increase in the MRT is attributed to the delay for copying content to surrogate servers. The cost for every copy is related to the number of hops among the client asking for it and the server where copying is likely to take place. We observe that Linear Regression outperforms all the other schemes, depicting MRTs smaller than their respective. We note here that timezones with highest average values for each scheme, that Subpolicy defines, are precalculated, in order to reduce computational burden in the simulations.

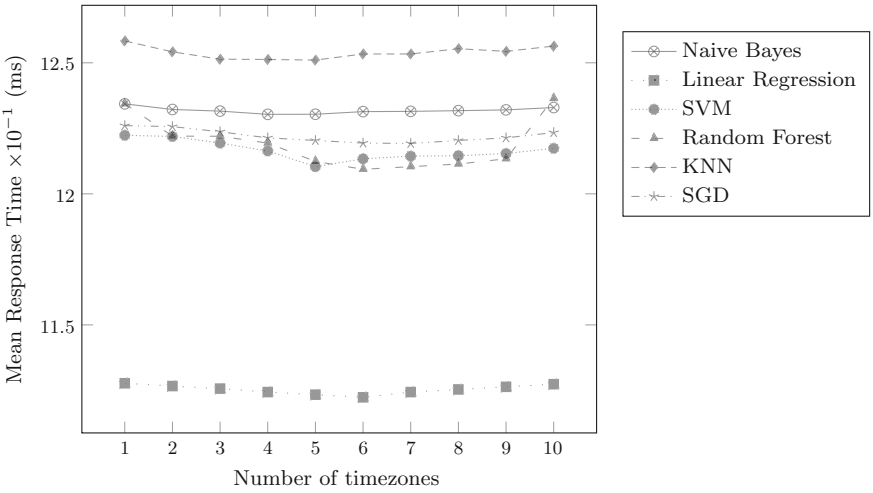


Fig. 20.17 Effect of timezones used as Y on Mean Response Time for various schemes ($X = 10$ closest timezones with mutual followers)

20.7 Conclusions

We come to the conclusion that video sharings over an OSN platform can be predicted with a small set of features extracted from both the platform and the media service. Despite the focused scope of this work and the limitations of its conduction solely with Twitter and YouTube data, the scale of the medium allows us to make assumptions for generalization across different OSNs and microblog platforms. We plan to extensively analyze this generalization in the future. Future extensions also include experimentation with variations of content distance interpretation among users, with various score assignment formulas, as well as subsequent verification in the realm of content delivery. We hope that our findings will broaden the view on the spread of information in web today.

References

1. Abisheva, A., Garimella, V.R.K., Garcia, D., Weber, I.: Who watches (and shares) what on YouTube? And when?: Using Twitter to understand Youtube viewership. IN: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, 24–28 Feb 2014, pp. 593–602 (2014). doi:10.1145/2556195.2566588
2. Bakshy, E., Hofman, J.M., Mason, W.A., Watts, D.J.: Everyone's an influencer: quantifying influence on Twitter. In: Proceedings of the 4th International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, 9–12 Feb 2011, pp. 65–74 (2011). doi:10.1145/1935826.1935845

3. Brodersen, A., Scellato, S., Wattenhofer, M.: YouTube around the world: geographic popularity of videos. In: Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, 16–20 April 2012, pp. 241–250 (2012). doi:10.1145/2187836.2187870
4. Cha, M., Kwak, H., Rodriguez, P., Ahn, Y., Moon, S.B.: I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, IMC 2007, San Diego, California, USA, 24–26 Oct 2007, pp. 1–14 (2007). doi:10.1145/1298306.1298309
5. Cheng, J., Adamic, L.A., Dow, P.A., Kleinberg, J.M., Leskovec, J.: Can cascades be predicted? In: Proceedings of the 23rd International World Wide Web Conference, WWW 2014, Seoul, Republic of Korea, 7–11 April 2014, pp. 925–936 (2014). doi:10.1145/2566486.2567997
6. Dow, P.A., Adamic, L.A., Friggeri, A.: The anatomy of large Facebook cascades. In: Proceedings of the 7th International Conference on Weblogs and Social Media, ICWSM 2013, Cambridge, Massachusetts, USA, 8–11 July 2013
7. Galuba, W., Aberer, K., Chakraborty, D., Despotovic, Z., Kellerer, W.: Outtweeting the Twitterers—Predicting Information Cascades in Microblogs. In: Proceedings of the 3rd Workshop on Online Social Networks, WOSN 2010, Boston, MA, USA, 22 June 2010
8. Hong, L., Dan, O., Davison, B.D.: Predicting popular messages in Twitter. In: Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28–April 1, 2011 (Companion Volume), pp. 57–58 (2011). doi:10.1145/1963192.1963222
9. Jenders, M., Kasneci, G., Naumann, F.: Analyzing and predicting viral tweets. In: Proceedings of the 22nd International World Wide Web Conference, WWW 2013, Rio de Janeiro, Brazil, 13–17 May 2013, Companion Volume, pp. 657–664 (2013)
10. Kilanioti, I.: Improving multimedia content delivery via augmentation with social information. The Social Prefetcher approach. IEEE Trans. Multimedia **17**(9), 1460–1470 (2015). doi:10.1109/TMM.2015.2459658
11. Kilanioti, I., Papadopoulos, G.A.: Socially-aware multimedia content delivery for the cloud. In: Proceedings of the 8th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2015, Limassol, Cyprus, 7–10 Dec 2015, pp. 300–309 (2015). doi:10.1109/UCC.2015.48
12. Kupavskii, A., Ostroumova, L., Umnov, A., Usachev, S., Serdyukov, P., Gusev, G., Kustarev, A.: Prediction of retweet cascade size over time. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM 2012, Maui, HI, USA, October 29–November 02, 2012, pp. 2335–2338 (2012). doi:10.1145/2396761.2398634
13. Kwak, H., Lee, C., Park, H., Moon, S.B.: What is Twitter, a social network or a news media? In: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, 26–30 April 2010, pp. 591–600 (2010). doi:10.1145/1772690.1772751
14. Ma, Z., Sun, A., Cong, G.: On predicting the popularity of newly emerging hashtags in Twitter. JASIST **64**(7), 1399–1410 (2013). doi:10.1002/asi.22844
15. Petrovic, S., Osborne, M., Lavrenko, V.: RT to win! Predicting message propagation in Twitter. In: Proceedings of the 5th International Conference on Weblogs and Social Media, ICWSM 2011, Barcelona, Catalonia, Spain, 17–21 July 2011
16. Rodrigues, T., Benevenuto, F., Cha, M., Gummadi, P.K., Almeida, V.A.F.: On word-of-mouth based discovery of the web. In: Proceedings of the 11th ACM SIGCOMM Conference on Internet Measurement, IMC 2011, Berlin, Germany, 2–4 Nov 2011, pp. 381–396 (2011). doi:10.1145/2068816.2068852
17. Suh, B., Hong, L., Pirolli, P., Chi, E.H.: Want to be retweeted? Large scale analytics on factors impacting retweet in Twitter network. In: Proceedings of the 2nd IEEE International Conference on Social Computing, SocialCom / IEEE International Conference on Privacy, Security, Risk and Trust, PASSAT 2010, Minneapolis, Minnesota, USA, 20–22 Aug 2010, pp. 177–184 (2010). doi:10.1109/SocialCom.2010.33
18. Szabó, G., Huberman, B.A.: Predicting the popularity of online content. Commun. ACM **53**(8), 80–88 (2010). doi:10.1145/1787234.1787254
19. Tsur, O., Rappoport, A.: What's in a hashtag? Content based prediction of the spread of ideas in microblogging communities. In: Proceedings of the 5th International Conference on Web

Search and Web Data Mining, WSDM 2012, Seattle, WA, USA, 8–12 Feb 2012, pp. 643–652 (2012). doi:10.1145/2124295.2124320
20. Yang, J., Leskovec, J.: Patterns of temporal variation in online media. In: Proceedings of the 4th International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, 9–12 Feb 2011, pp. 177–186 (2011). doi:10.1145/1935826.1935863
21. Yang, S., Zha, H.: Mixture of mutually exciting processes for viral diffusion. In: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16–21 June 2013, pp. 1–9 (2013)
22. Zaman, T.R., Herbrich, R., Van Gael, J., Stern, D.: Predicting information spreading in Twitter. In: Proceedings of the Workshop on Computational Social Science and the Wisdom of Crowds, Nips, vol. 104, pp. 17, 599–601 (2010)

Chapter 21
Big Data Uses in Crowd Based Systems

Cristian Chilipirea, Andreea-Cristina Petre and Ciprian Dobre

21.1 Introduction

Crowd Sensing represents a new paradigm whose potential and limitations still require a lot of research. Classically if someone wanted to gather information about an area she would deploy a large enough number of sensors that provided the required data. This solution has some severe limitations. First of all, the number of sensors can be extremely high depending on the application. Take for instance, measuring the noise level inside a city. This would require microphone-like sensors deployed on every street. These sensors would need to communicate in order to send the data to a central location. This in turn means deploying a wire or wireless infrastructure. Furthermore, the sensors require energy. They can either use high power expensive batteries, that require maintenance or a connection to an electrical infrastructure, which is not easily available. All these problems: the sensors, the communication, the power requirements impose an extremely high price for the simple application of noise monitoring. And the problem does not stop with the price. Sensors need to be replaced when they are broken, they need to be protected so that the quality of the measurements is not disturbed and because they are static they are limited in their view of the noise in parts of the city where they are not deployed.

Here comes the idea of using a crowd. Instead of deploying an expensive, difficult to maintain infrastructure, one could use the power of the crowd in order to obtain the same, or even better measurements. The power of the crowd is well known. The first example of this was made by Wikipedia [64], where a collection of articles,

C. Chilipirea · A.-C. Petre · C. Dobre (✉)
University Politehnica of Bucharest, Spl. Independentei 313, Bucharest, Romania
e-mail: ciprian.dobre@cs.pub.ro

C. Chilipirea
e-mail: cristian.chilipirea@cs.pub.ro

A.-C. Petre
e-mail: andreea.petre@cti.pub.ro

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_21

created, and edited by individuals across the Internet built the largest encyclopedia. This work was not managed or coordinated in any way. All participants contributed with as much or as little as they felt.

The idea of using a crowd is further explored by applications such as Amazons Mechanical Turk [3]. This is a marketplace application that proposes the use of crowds in order to solve small tasks in exchange for small amount of money. Many similar specialized marketplaces have appeared over time.

All these systems are based on crowd sourcing. Having a large number of individuals solve one small problem. In most cases crowd sourcing requires the direct involvement of the individual in order for the problem to be solved.

Crowd sensing is a subset of crowd sourcing. If crowd sourcing proposes the use of crowds in order to solve a problem, crowd sensing requires that the problem be of a "sensing" one by nature. The previously presented noise problem is a sensing one.

To take advantage of crowd sensing in order to solve the noise monitoring problem one would require only a simple application, installed on the phones of many users. Smartphones are an ideal platform: they are now ubiquitous; they all have one of three popular operating systems, Android OS, iOS, or Windows Mobile; they are very powerful complex pieces of hardware with a processor, internet access and many sensors; and finally they are very mobile. Particular to the noise application, they also have microphones already included in them.

With an application that takes microphone data, process it to determine a noise level, and sends it to a central location one can use the power of crowds to monitor large cities. This is all done with minimal costs or resources. Because of human mobility, many different areas are constantly being monitored. The only thing required is the good will of the crowds, and this has been shown time after time again through platforms such as Wikipedia. Furthermore, the goodwill can always be replaced with incentives, monetary, or of different nature, such as gamification.

Multiple crowd sensing tasks can be run at the same time. For instance, one task can measure the noise level, while another measures the pollution level or even pedestrian density. With more tasks, more data is being generated and more difficult it is to process. But with more data more information can be extracted and specialists can make more sense of what is going on.

There are multiple factors that introduce an extremely high scale to the crowd sensing problem: Large number of users that generate the data; Multiple concurrent monitoring tasks; Constant monitoring, all devices are periodically gathering data. This creates an extremely large volume of data.

Volume is the first of the "three V's of Big Data" as defined by Gartner [41]. The V's are: Volume, Velocity and Variety. Crowd sensing data has all these features. The data is constantly being generated at a very high speed by many sensors, this represents Velocity. The data can be generated by different, complex sensors, representing Variety. The volume is given by the scale of this data set.

IBM added a fourth V as a Feature of Big Data [35]. This V represents Veracity, the fact that the data is uncertain, or unclear. This is especially true in crowd sensing, where there can be only little trust in the data being generated.

Crowd sensing raises interesting problems that never existed in the case of any other sensing system. Crowd sensing is extremely dependent on crowd dynamics. Sensors are where people are. This does mean they can cover a large area, but this also means they are not always available. For instance, take a task of sensing pollution levels in a city. During the night people are mostly inside their homes and very few are outside offering readings. In the day, most people are mobile and so are the sensors they are carrying, covering large areas, and delivering large amounts of relevant data. This makes it very important to first understand crowd dynamics so that crowd sensing data can be correctly evaluated and used.

Fortunately, there is currently a large amount of research that tries to make sense of crowd dynamics. Crowd tracking applications make use from anything from GPS, WiFi signals to inertial sensors. The data from these applications have very similar properties to the crowd sensing data. There is a high volume of data, given by the large number of individuals, with a high velocity given by the constant location updates, as well as, high variety and veracity given by the different methods in which this data can be acquired.

In the next section we offer related work and a motivation for this research. This is followed by a detailed analysis on Crowd Sensing systems. Next we present different methods of measuring Crowd dynamics, information relevant to the crowd sensing systems. We continue by presenting different types of Context that can be used in conjunction with crowd tracking and crowd sensed data. Finally, we discuss how Big Data can be used to extract information from this data sources and finish with our Conclusions.

21.2 Related Work

Crowd based systems are raising more interest. More and more research projects aim the construction of such a system. In this chapter we are interested in two categories of crowd-based systems, in systems that use the crowd for the purpose of sensing and in systems that try to monitor a crowd.

Sensing data using a crowd, or simply called crowd sensing represents an interesting promise, of cheap, scalable, powerful data gathering system. These systems make use of the mobility of crowds in order to extract sensed from multiple locations. On overview on crowd sensing and a listing of many possible applications is available in [31].

With the increase of interest in crowd sensing systems, in recent years, many platforms have appeared that try to implement and solve this problem. Medusa [54] represents the most popular crowd sensing platform. It consists of a programming framework where people can define sensing tasks. These tasks are then spread to the users and they can gather the sensor data and forward it to the requester. In order to reward the users Medusa uses Amazon Mechanical Turk [3].

In comparison Metador [15, 16] represents a platform for crowd sensing that concentrates on context information. Tasks are created and delivered only to the

individuals that are in the right context. For instance, information of a crowd sensing campaign requiring a picture of a building is sent only to individuals in proximity of the building.

Mosden [36] represents a platform for crowd sensed data whose main idea is to separate logic behind communication, storage and data acquisition. The platform is demonstrated using an application that uses crowd sensed data to determine noise pollution inside a city.

Another platform for crowd sensing is implemented on top of Cupus [6]. It uses a publish/subscribe interface based on the cloud. It is meant for Internet of Thing systems. The platform is used with an application for air quality monitoring. The air quality sensors are small dongles that can connect with Bluetooth to a smartphone. Using the Cupus publish/subscribe systems the data from multiple such devices is collected and processed in the cloud.

Probably the first platform that can be used for crowd sensing purposes is the mCrowd platform [67]. It integrates ChaCha [17] and Amazon Mechanical Turk [3] and offers tasks set on these systems on mobile platforms. The tasks take mostly the form of image or text responses and require human interaction to complete.

Platforms for mobile crowd sensing have appeared in order to make it simple to deploy crowd sensing campaigns or applications. There are a lot of problems that make it difficult to deploy crowd sensing applications. Manny of them are detailed in [65]. Among the problems identified we note: heterogeneity of mobile devices software and difficulty of installing applications, especially if each crowd sensing task requires a different application. There are also problems of scalability and resource utilization. Platforms for crowd sensing have the potential of solving most problems. With the large number of crowd sensed platforms there are also multiple applications for crowd sensing. For instance, in [2] the authors present MAP++ a system where semantic data is automatically added to maps using a crowd sensing system. Their system uses only sensors from inside smartphones, such as the accelerometer, and require no user interaction. Multiple road features can be detected such as bumps or roundabouts. However, their system is calibrated to work only for smartphones carried inside vehicles.

A similar system that uses only sensors that already exist is smartphones is presented in [63]. Here the authors show how Bluetooth modules can be used in order to determine the density of people in different areas. Their application makes use of multiple smartphones that continuously record Bluetooth signals, making it a crowd sensing application. The authors prove that their system has an accuracy of 75 %.

The Mahali project [51] represents probably the most daring of crowd sensing applications. It proposes the use of a dual frequency GPS receiver, in order to measure ionosphere features. This information can be used to get high resolution weather maps. Smart phones are likely to have dual frequency GPS receivers in the near future due to the need of precise localization. In the meantime, they are still useful to fill the gap in connecting remotely placed static sensors, without network access by being an intermediary node in connecting it to the internet. This method is called delay tolerant networking.

Crowd sensing can be used to further many science experiments that would otherwise require large costs and man power to complete. In [32] the authors propose a crowd sensing application intended for use by students. The application asks the users to go to specific locations and take pictures of different plants. These pictures are then used by environmental scientists in order to better understand the flora of the campus. The application has as basis the idea of gamification, the incentive of taking the pictures being a simple set of in-game points. The method of gamification is called floracahing and is similar to geocaching [28].

Not all applications that need sensing data require the deployment of a new crowd sensing campaign. For instance, in [22] the authors use Foursquare data in order to determine the growth of cities. Because of the scale of this data set popular locations in cities can be observed and the increase in density of these locations can be measured. Cities themselves have require many types of data if they are to become what is known as smart cities. The authors of [33] conduct an in depth look of multiple types of sensing that can be done in smart cities. Many of these can be applied to crowd sensing.

With all these platforms and application for mobile crowd sensing appearing in recent years it is clear that the potential given by these system is quite large. But, crowd sensing platforms and solutions all suffer from privacy issues. A few papers try to solve this problem. The work presented in [40] takes privacy as the most important consideration. In order to preserve the privacy of people generating the data they propose a cloud-based approach with a system that gathers the data and obfuscates it before it is being sent to the data requester. Similarly, the authors of [38] create an application that provides anonymous authentication to their service. This application of crowd sensing permits park operators to determine important characteristics of the crowds that move around their park. For instance, the operators are able to determine queue lengths and even recommend better routes for incoming visitors. A survey on privacy of crowd sensing applications is available in [53].

Other works try to improve the reliability of crowd sensing systems. The authors of [27] tackle the problem of how to insure that crowd sensed data is accurate. For instance, when an application requires pictures from an event, how to confirm the pictures are actually valid. The solution of the authors is given by a Bluetooth communication between devices. Multiple devices use their GPS data and increase the trust that the image is indeed taken in that location. This is done without any human intervention. A similar solution is provided in [49] where trust in the data is treated. The requester needs to have trust in the data being sent by random users. In order to improve trust this paper proposes a way of discovering the "truth" in data by correlating multiple reports.

Finally, one has to determine the optimal number of users for a crowd sensing application. In order to determine this number a simulation environment is proposed [26]. This environment is specifically targeted at urban parking. The information extracted from the crowd sensing application being used in order to determine where a parking lot is available.

In order to completely understand crowd sensing and the data it offers, it is vital that data analysts have access to crowd dynamics information. There is a need to

first understand the position of individuals that participate in the sensing process and then to understand the flows of crowds.

Measuring crowd dynamics is classically done using visual systems [57]. These system consists of a number of video cameras placed around the city and powerful software that can identify a face and track it across multiple cameras. Because facial recognition is still not a completely solved problem and the results are highly probabilistic, the results of these systems are not very accurate. Furthermore, implementing this type of systems requires huge investments in infrastructure, starting from cameras and the communication network for their data to be gathered at a central location, to large processing centers required to run the face recognition algorithms.

Other solutions have appeared in the literature. GPS [8] is the most known and most highly used positioning technology. It is present in most modern cars and in almost all smartphones, which are now ubiquitous. This solution has been implemented in the case of large crowd monitoring [10]. Taking advantage of the large number of smart phones the authors built an app that gathered GPS data from any device that had the app installed to a central location. They proved the feasibility of this method by deploying it during a three-day Swiss festival.

Requiring people to install an app in order to provide the required information is not a solution that scales very well. Individuals need to have a level a trust in the app and its use needs to be widespread, which in turn might require incentives. Methods that do not require the involvement of the individuals being tracked are preferred.

An increasingly popular alternative is given by the use of communication signals. Whenever a phone transmits data it uses one of several standardized protocols. By having scanners that can record these signals it is then possible, within a limited accuracy, to identify where a device is and how it is moving. Because protocols such as WiFi includes the address of the device within most packets it is possible to track one device across multiple scanners.

Using WiFi in order to track crowds of people and understand human mobility has been done in for music festivals [11] or for campus monitoring [37]. More interesting applications try to use this type of data in order to inform on building facility planning [55] or even measure flows of people through airport checkpoints [56]. The technology has also been proved useful in measuring human queues [61].

Other works have searched ways of improving the use of WiFi scanning techniques. In [19] we try to show a number of filters that can be used in order to obtain a smaller data set with less noise. The solution for the RSS variance problems in this type of monitoring is solved presented in [59]. Improvements have also taken place in the form of building better tools for visualizing movement data [5].

In the search for accurate device positioning and tracking some works have tried using multiple technologies at the same time. This is apparent in [50] where multiple communications methods and magneto metric sensors are used at the same time, or in [25] where WiFi is used at the same time as inertial sensors.

21.3 Crowd Sensing

Crowd sourcing is becoming a very powerful tool for solving large tasks (even at a scale that makes them seem impossible) that can be broken down in a large number of simple tasks that require just a bit of human involvement. This idea is the basis of Wikipedia [64], a website where anyone can contribute to build one massive, open encyclopedia. This same website for instance offers an extensive list of cases where crowd sourcing is used in order to solve otherwise impossible tasks.

Crowd sensing is a part of crowd sourcing where the people inside the crowd are tasked with gathering sensor data. The authors of [30] define crowd sensing as a natural extension of participatory sensing. Where participatory sensing represents a task where the crowd is used to gather sensor data, and crowd sensing uses this data in conjunction with offline or previously gathered data. An even more important difference is given by the lack of need of participation that was implied by the participatory sensing systems. For instance, a campaign that tries to measure noise pollution requires that the mobile devices send data with regards to noise, but they dont require the mobile device owner to actively do anything. In contrast a campaign that requires images of a building or of an event, requires that the owner of the mobile device actively take these pictures.

To have a crowd sensing campaign, one requires that individuals inside an area carry a device that has the sensing capabilities required by the crowd sensing campaign. The data gathered by the devices then needs to be sent to the campaign manager, the one that requested the data, or to an individual to whom the data is useful.

Because of the popularity of smartphones, it is now possible to have large scale deployments of crowd sensing applications, where as many users that own smartphones can participate. Smartphones are ubiquitous and with this the popularity of crowd sensing applications has grown. However, all crowd sensing applications that make use of smartphones are limited by the features that are found inside them. A solution is to extend the capabilities of smartphones with external hardware, like a Bluetooth dongle that contains only the needed sensor. An example of this is available in [66]. The smartphone is still an important part of the system because with its powerful CPU and communication capabilities it dramatically lowers the price of the scanner. With powerful sensors, such as pollution sensors, or sensors that can detect humidity or pressure, crowds can gather all types of data.

Campaigns that require external sensor are more difficult to deploy. They are more expensive because of the extra hardware and they require people to carry another device with them, which is inconvenient. Smartphones are now powerful computers with an already large variety of sensors. These sensors are the most popular for crowd sensing campaigns. The most popular sensors inside smartphones are

- *Accelerometer*—The accelerometer in smart phones is commonly used as an inertial sensor, capable of inferring the speed with which the person is moving, as well as, offer support for many other applications as impressive as gesture recognition [34] or even authentication [48]. When a large number of these sensors

are used applications such as earthquake detection and measuring [27] can be implemented.

- *GPS*—This sensor is used primarily for localization purposes. In conjunction with maps and interactive applications it can be used to assist with choosing the shortest path through a city. With many of these sensors crowds and congestions can be detected [47].
- *Photo/Video Camera*—Smartphones are now capable of taking high resolution photos and videos. People use them extensively and post photos on social networks and these can be used in crowd sensing campaigns [23]. The photos can also be used to enable citizen journalism [68].
- *Microphone*—Communication remains the main feature of a smartphone and microphones are the central sensor that enables it. With enough microphones crowd sensing campaigns can measure noise pollution [45].
- *Magnetometer*—Are sensors that detect magnetic fields, they are used in conjunction with accelerometers in order to improve positioning.
- *Thermometer*—Are mainly used to measure the temperature of the CPU to insure that it is working appropriately. However, these sensors can also be used to measure the temperature of the environment [7].
- *WiFi*—WiFi represents the module used to communicate data, mainly for internet usage. Because of the details of the 802.11 protocol this module can be used in order to detect other WiFi enabled devices through WiFi scanning [4]. Making a list of static WiFi devices is done through a technique called war-driving [21] and it can be used to improve localization.
- *Bluetooth*—Similar to WiFi, Bluetooth represents a data communication technology. It is mainly used for personal area networks and is meant to connect smartphones to wearables or headphones. They can be used to facilitate communication with other devices and this means they can also be used to detect them and with them the size of crowds [63].

The data gathered by the sensors can be sent to a central authority. Alternatives are to store the data until an Internet connection is available or to send the data in a distributed manner. For instance, if the sensors measure the traffic on a street, the information can be useful to people located on adjacent streets.

In order for any crowd sensing campaign to be successful it requires people to participate in the crowd sensing process. This means people have to be incentivized in order for them to participate in the campaign. There are many articles that propose incentives for crowd sensing systems [14, 42–44] as this is the primary assumption that needs to be addressed in order to implement any such campaign.

With appropriate incentives crowd sensing systems can reach the required number of users. Because the sensors are usually small range, take for instance microphones which work for only few meters, this means lots of people need to take part in the crowd sensing system in order to be able to monitor large areas.

Crowd sensing itself is usually opportunistic. Sensed data is only gathered in locations where people are. If they do not go to different areas, then there will simply be no data from those areas. In order to improve this, active systems could be

implemented in order to guide people to where sensed data needs to be gathered. This is the case for floracache [32] where people are told where to go to gather the data.

Outside of smartphones there are a few other classes of devices that can prove useful for crowd sensing initiatives. Cars now have many powerful computers and sensors as well as communication capabilities with central locations or even other cars [24]. The computers and sensors in these cars can be repurposed in order to participate in crowd sensing networks. Because of their large volume and carrying capabilities, cars can be equipped with larger more complex sensors. We ask the reader to consider if it is not fit for the cars that contributed to pollution to be the ones that are tasked to measure it.

Wearables [58] represent a large class of devices that people can wear. They are now popular in the form of smart watches and fitness bands, but they can take many forms. They have the advantage of being in extremely close proximity to the user. With increases in processing power and multiple sensors they will soon become another interesting alternative for crowd sensing.

To our knowledge there is no real open data set available that shows the results of a crowd sensing campaign.

With many alternatives for applications and many available platforms crowd sensing is likely to be an important part of our daily lives. But in order to understand the accuracy and the benefits of crowd sensed data it is vital that there is access to crowd dynamics information. Understanding crowd dynamics enables us to measure the quality of the sensing data, by understanding what areas are covered and which are not, as well as, be able to predict which areas will be covered in the future. This even enables campaign runners to deploy static sensors in areas that do not get enough traffic.

21.4 Measuring Crowd Dynamics

Crowd dynamics represent all movements and stationary actions that crowds make. They can apply to a small area such as an indoor facility or be as large as an entire city. Crowds display flows and they can merge or split throughout their movement. They are affected by the density of people that make out a crowd.

Ideally a crowd tracking system would be able to offer a precise positioning for every person at any time. No known system is capable of such a task, but instead they all offer approximations.

The most popular method measuring crowds and tracking them is by use of video feeds. This is mostly made apparent by CCTV systems [29]. These systems however require expensive infrastructure and large processing centers aimed at analyzing the video feeds. Because face recognition and individual tracking is still difficult to implement with computers, these systems still have a large room for improvement.

Newer systems make use of smartphones which are considered ubiquitous. This can be done by installing an application of the devices of many citizens or by scanning for the communication signals they send. The application can use GPS positioning to

determine the location of the device. Unfortunately, GPS only works outdoors and has large errors, in the order of meters. Scanning for WiFi hotspots and matching them with locations obtained from war-driving represents a possible alternative which has the advantage of being energy efficient. However, the accuracy is even lower than GPS and the exact location of hotspots is not always known.

Installing software at a large scale is difficult. People need incentives and they are worried about the security of their device as well as their own personal privacy. Because of this, methods that simply scan for communication signals are simpler to implement and deploy.

It is important to understand that all communication signals suffer from a high variety of noise sources. First of all, not all devices are built the same. Then, the weather or surroundings can affect the way in which the signal propagates. This phenomenon is made even worse when we consider mobile features of the surroundings such as cars or other pedestrians.

Not all people own smartphones and from those that do own them not everyone uses the WiFi module. Other people stop the WiFi module when they are not using it due to energy consumption constraints and the fact that they wish to preserve battery. This means that only a portion of the population is being tracked using these systems.

Another important factor that needs to be considered when deploying crowd monitoring applications is the privacy of the people being monitored. The data needs to be obfuscated in a way in which it remains useful but hides the identities of all the individuals being tracked.

If the crowd dynamics information is obtained by deploying applications on the user devices this makes it a crowd sensing application that is able to reveal for instance density. There are many applications that can result from this type of data, for instance we were able to infer the map of Roma [18] given only GPS localization data from several taxis.

21.5 Crowd Data for Crowd Dynamics

As we mentioned previously crowd sensing data is not openly available. We did manage to identify an important source for open crowd tracking, and crowd related data. This source is called CRAWDAD [39] and it hosts a large number of scientific data sets related to wireless data.

In order to have a better picture of what crowd tracking data is and what it can be used for we make a small analysis on five data sets from the CRAWDAD website. All these data sets contain localization data.

The five data sets we chose are

- *Reality* [62]—Represents a data set gathered from 100 subjects from MIT during an academic year. The people, consisting mainly of students carrying mobile phones capable of recording the cells of GSM towers.

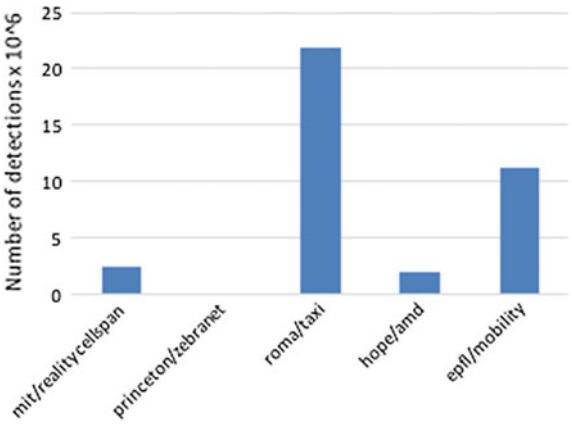


Fig. 21.1 No. of detections

- *Zebranet* [60]—Applied on the Sweetwaters Game Reserve near Nanyuki, Kenya. Zebras were fitted with collars that contained a GPS receiver, a CPU as well as storage and wireless transceiver. Their system uses opportunistic communication in order to gather the data set.
- *Roma Taxis* [12]—In the city of Rome taxis were fit with special devices and software that recorded their movements.
- *Amd* [1]—At the hackers on planet earth conference guests were given RFID tags. These tags were detected by static sensors placed in different rooms of the conference.
- *Epfl* [52]—Similar to the Roma data set, the data is gathered from taxis. This data set is gathered in the city of San Francisco.

The statistics on the five data sets are available in Fig. 21.1 where the number of recorded data elements is displayed. In Fig. 21.2, we compare the number of devices in each of the data sets.

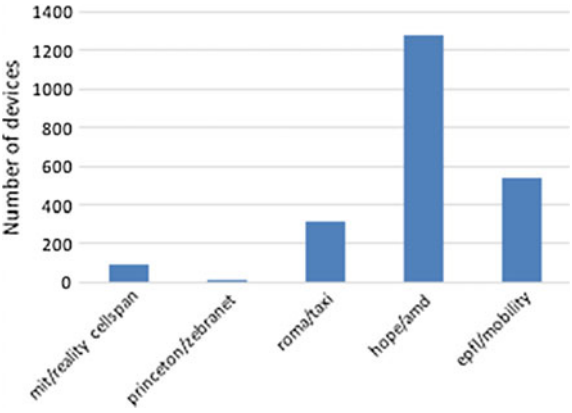


Fig. 21.2 No. of devices

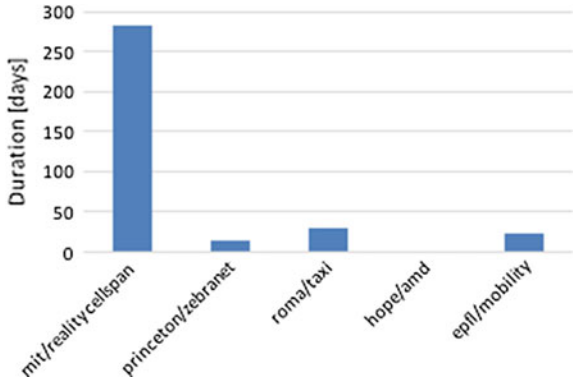


Fig. 21.3 Duration

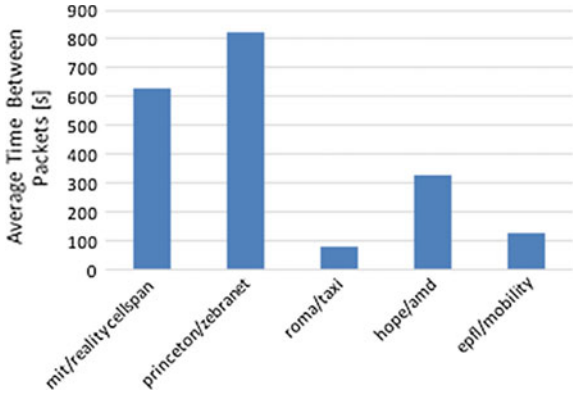


Fig. 21.4 Average time between consecutive packets

Figure 21.3 shows the duration of each of the data sets. These first three figures show how diverse the data sets can be. They can contain data recorded over any number of days with a large variation in the number of elements in the data set or the number of devices. The large variety in the data sets permits anyone to do analysis on very different scenarios and use cases.

It is not only the scenario that differs but the technology used to gather the data set. These technologies span from GPS recordings to WiFi detection to even RFID tags. Because of this difference there is also a difference in the average time between two consecutive detections of the same device. This is best made apparent in Fig. 21.4. This difference is made not only by the technologies but by the settings and the behaviors of the participants.

Finally, we tried to identify the similarities between the data sets. Given our experience with various crowd tracking data sets we learned to expect a day night pattern in any data set that involves people. We managed to identified this feature in all of the data sets. In Fig. 21.5, we display the number of detections or data set elements as they change during the day. The data is normalized by having a ratio to the maximum number of detections for each data set. The day was chosen randomly

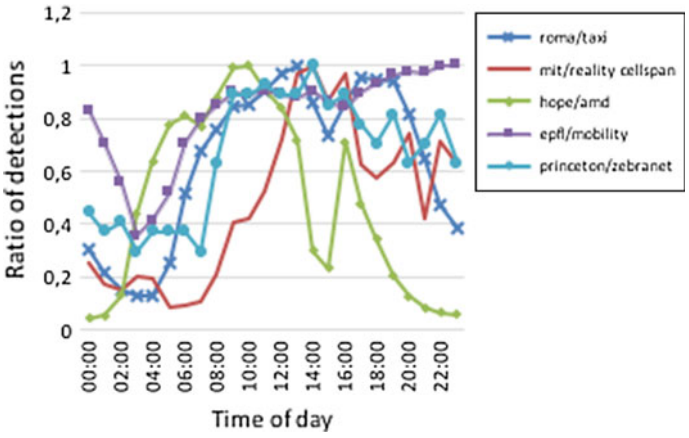


Fig. 21.5 No. of detections over time

from the data sets. It is clear that in all of these the number of packets raises during the day and drops during the night.

We offered only a small overview on some characteristics of crowd tracking data. Further analysis can reveal more interesting features such as patterns that people take each day in their movements. Obtaining these features and this information requires extensive data analysis usually done with the help of Big Data.

21.6 Context

Context represents all types of data set that can help in making more sense of the crowd data. Unlike the data sets we presented in the previous sections, context data represents unstructured data. It usually comes in the form of text and requires natural language processing systems [20]. Unlike the crowd sensed data and crowd tracking data, the origin of context data is not clear. There are various possible open internet sources as well as closed ones. The trust put in the data is dependent on these sources.

We have identified a few potential context data types

- *Weather*—It directly affects the behavior of people. In case of bad weather people are more likely to remain indoors. This can explain a drop in the number of individuals performing crowd sensing outside.
- *Schedules*—There are many types of schedules, from the mostly static once of shops and school, to schedules of complex events. Using schedules one can better understand the reasoning behind crowd movements. This is made specifically apparent in day/night patterns observed in crowd data.

- *Social network data*—People post a high variety of information on social networks from picture to restaurant rating. Based on this data we reasonable to expect certain behaviors in crowd movements. For instance, restaurants with bad reviews are avoided while the ones with good ones have a lot of traffic.
- *News*—Various events are not predictable. This is especially true for high impact disasters. News sources can be used in order to understand what is going on, beyond the normal schedule, or social network information.

There are many sources of context data and many types of it. Making sense with it and matching it with the crowd sense data requires a lot of resources. This is where Big Data jumps in. It is specialized in processing large amounts of structured or unstructured data. Because of its variety context can be continuously extended as new sources of context data are identified and integrated with the existing ones.

21.7 Crowd Data as Part of Big Data

Big Data [46] represents a new paradigm in data processing. It goes beyond traditional database queries where extracting information was done in a straight forward manner. In contrast, instead of giving simple information such as the maximum or average of a data set, Big Data queries use machine learning techniques [13] to offer answers without the need for a question. They offer new information about the data without the analyzer even knowing what to look for. A good example of this is usually given in machine learning courses as the "beer and diapers" example. At a large supermarket chain Big Data analysis revealed that people who buy diapers also buy beer. The example sticks because it is clear how unlikely it is that someone would search through data sets to see if people buy diapers and beer at the same time, and the result has a simple application, place the two products next to each other. This type of information can relevant for crowds. Take for instance groups of people that visit the same two shops in the same order. This would enable the prediction of crowd flows.

We showed in the previous sections that crowd data, be it crowd sensing or crowd tracking, exhibits the features of Big Data. Crowd data has all four versus: Volume, Velocity, Variety, and Veracity. In order to better fit with Big Data we propose the use of crowd sensing as well as crowd tracking. Crowd tracking provides important information as to where the crowd sensed data is gathered from. Context with its many forms bring only improvements to these two data sets, and all of them taken together pose an interesting challenge to Big Data and opens the door to many applications.

By applying Big Data techniques on crowd data we open the door to what is known as smart cities [9]. Places where real time-data and information offers better living conditions, less traffic and overall increase in the quality of life. The information extracted permits automated systems that respond to citizen needs and better planning for the authorities.

But Big Data does not show only human behavior in order for decisions to be made based on past data, it can also help in making predictions on what will happen. For instance, when a large event is organized, having knowledge of possible crowd behaviors can help with many of the decisions.

Another important aspect is given by disaster scenarios. By having real life data as well as appropriate models of human movement, the rescue teams can respond faster, with more knowledge and more information. Even more, facilities can be built in such a way that evacuation is simple and it fits with appropriate models of human behavior.

Let us finish with a possible example of Big Data application with crowds. This is just one of many possible uses. Given a crowd tracking data set, which enables us to know where people are gathered and what their flows are, as well as a crowd sensing data set which enables us to map the noise level throughout the city. In the case of an unexpected event, we have more information on the behavior of crowds and how better to guide them. Adding context data can offer even more insight, it can determine if the event was a music event starting or if it was a dangerous event and emergency response units need to be deployed. The decision can even be taken before anyone even succeeds in reporting what happened.

Big Data opens the doors to a variety of applications and use cases that may be unimaginable, the more we explore it the more we can push its limits and be able to build truly smart cities and smart environments.

21.8 Conclusions

In this chapter, we presented crowd sensing and crowd tracking applications. We discussed the high potential they have and their current limitations. We continued with a discussion on different types of context and how this context information can have a high impact in extracting information from crowd sensed and crowd tracking data.

Finally, we showed how Big Data can be used to put everything together. We showed that crowd data is a type of Big Data and by using multiple crowd data sources in conjunction with context we can obtain various and even surprising new information.

Because the field is still young there is a lot that remains for future work. Multiple crowd sensing applications need to be deployed at a large scale and the data needs to be exchanged in an open manner. Privacy implications need to be considered in order to make this possible. Once the data is open multiple groups can identify different ways to best analyze and extract information from the given data sets. The multitude of crowd data sets and the high potential they have opens a lot of doors for future applications.

Acknowledgments The research presented in this paper is supported by projects: MobiWay, Mobility beyond Individualism: An Integrated Platform for Intelligent Transportation Systems of Tomorrow—PN-II-PTPCCA-2013-4-0321; DataWay, Real-time Data Processing Platform for Smart Cities: Making sense of Big Data—PN-II-RUTE-2014-4-2731. We would like to thank the reviewers for their time and expertise, constructive comments and valuable insight.

References

1. Aestetix, P.C.: CRAWDAD dataset hope/amd (v. 2008-08-07). http://crawdad.org/hope/amd/20080807, doi:10.15783/C7101B
2. Aly, H., Basalamah, A., Youssef, M.: Map++: A crowd-sensing system for automatic map semantics identification. In: 2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), pp. 546–554. IEEE (2014)
3. Amazon: Amazon Mechanical Turk (2016). https://www.mturk.com/mturk/welcome. Accessed 1 July 2016
4. Anand, A., Manikopoulos, C., Jones, Q., Borcea, C.: A quantitative analysis of power consumption for location-aware applications on smart phones. In: 2007 IEEE International Symposium on Industrial Electronics, pp. 1986–1991. IEEE (2007)
5. Andrienko, G., Andrienko, N., Wrobel, S.: Visual analytics tools for analysis of movement data. ACM SIGKDD Explor. Newsl. **9**(2), 38–46 (2007)
6. Antonić, A., Marjanović, M., Pripužić, K., Žarko, I.P.: A mobile crowd sensing ecosystem enabled by cupus: Cloud-based publish/subscribe middleware for the internet of things. Future Gener. Comput. Syst. **56**, 607–622 (2016)
7. Aram, S., Troiano, A., Pasero, E.: Environment sensing using smartphone. In: Sensors Applications Symposium (SAS), 2012 IEEE, pp. 1–4. IEEE (2012)
8. Bajaj, R., Ranaweera, S.L., Agrawal, D.P.: Gps: location-tracking technology. Computer **35**(4), 92–94 (2002)
9. Batty, M., Axhausen, K.W., Giannotti, F., Pozdnoukhov, A., Bazzani, A., Wachowicz, M., Ouzounis, G., Portugali, Y.: Smart cities of the future. The Eur. Phys. J. Spec. Top. **214**(1), 481–518 (2012)
10. Blanke, U., Tröster, G., Franke, T., Lukowicz, P.: Capturing crowd dynamics at large scale events using participatory gps-localization. In: 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pp. 1–7. IEEE (2014)
11. Bonné, B., Barzan, A., Quax, P., Lamotte, W.: Wifipi: Involuntary tracking of visitors at mass events. In: 2013 IEEE 14th International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp 1–6. IEEE (2013)
12. Bracciale, L., Bonola, M., Loreti, P., Bianchi, G., Amici, R., Rabuffi, A.: CRAWDAD dataset roma/taxi (v. 2014-07-17). http://crawdad.org/roma/taxi/20140717, doi:10.15783/C7QC7M
13. Carbonell, J.G., Michalski, R.S., Mitchell, T.M.: An overview of machine learning. In: Machine Learning, Springer, pp. 3–23 (1983)
14. Cardone, G., Foschini, L., Bellavista, P., Corradi, A., Borcea, C., Talasila, M., Curtmola, R.: Fostering participaction in smart cities: a geo-social crowdsensing platform. IEEE Commun. Mag. **51**(6), 112–119 (2013)
15. Carreras, I., Miorandi, D., Tamilin, A., Ssebaggala, E.R., Conci, N.: Crowd-sensing: Why context matters. In: 2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 368–371. IEEE (2013)
16. Carreras, I., Miorandi, D., Tamilin, A., Ssebaggala, E.R., Conci, N.: Matador: Mobile task detector for context-aware crowd-sensing campaigns. In: 2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), IEEE, pp. 212–217. IEEE (2013)

17. ChaCha: Cha Cha (2016). http://www.chacha.com/. Accessed 29 June 2016
18. Chilipirea, C., Petre, A., Dobre, C., Pop, F., Xhafa, F.: Enabling vehicular data with distributed machine learning. In: Transactions on Computational Collective Intelligence XIX, pp. 89–102. Springer (2015)
19. Chilipirea, C., Petre, A.C., Dobre, C., van Steen, M.: Filters for wi-fi generated crowd movement data. In: 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pp. 285–290. IEEE (2015)
20. Chowdhury, G.G.: Natural language processing. Annu. Rev. inform. Sci. Technol. **37**(1), 51–89 (2003)
21. Constandache, I., Choudhury, R.R., Rhee, I.: Towards mobile phone localization without war-driving. In: Infocom, 2010 Proceedings IEEE, pp. 1–9. IEEE (2010)
22. Daggitt, M.L., Noulas, A., Shaw, B., Mascolo, C.: Tracking urban activity growth globally with big location data. R. Soc. Open Sci. **3**(4),150, 688 (2016)
23. Demirbas, M., Bayir, M.A., Akcora, C.G., Yilmaz, Y.S., Ferhatosmanoglu, H.: Crowd-sourced sensing and collaboration using twitter. In: 2010 IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM), pp. 1–9. IEEE (2010)
24. Eichler, S., Schroth, C., Eberspächer, J.: Car-to-car communication. In: VDE-Kongress 2006, VDE VERLAG GmbH (2006)
25. Evennou, F., Marx, F.: Advanced integration of wifi and inertial navigation systems for indoor mobile positioning. Eurasip J. Appl. Sig. Process. **2006**, 164–164 (2006)
26. Farkas, K., Lendák, I.: Simulation environment for investigating crowd-sensing based urban parking. In: 2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), pp. 320–327. IEEE (2015)
27. Faulkner, M., Olson, M., Chandy, R., Krause, J., Chandy, K.M., Krause, A.: The next big one: Detecting earthquakes and other rare events from community-based sensors. In: 2011 10th International Conference on Information Processing in Sensor Networks (IPSN), pp. 13–24. IEEE (2011)
28. Geocaching: Website (2016). https://www.geocaching.com. Accessed 29 June 2016
29. Gill, M., Spriggs, A.: Assessing the Impact of CCTV. Home Office Research, Development and Statistics Directorate London (2005)
30. Guo, B., Yu, Z., Zhou, X., Zhang, D.: From participatory sensing to mobile crowd sensing. In: 2014 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 593–598. IEEE (2014)
31. Guo, B., Wang, Z., Yu, Z., Wang, Y., Yen, N.Y., Huang, R., Zhou, X.: Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm. ACM Comput. Surv. (CSUR) **48**(1), 7 (2015)
32. Han, K., Graham, E.A., Vassallo, D., Estrin, D.: Enhancing motivation in a mobile participatory sensing project through gaming. In: 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), pp. 1443–1448. IEEE (2011)
33. Hancke, G.P., Hancke Jr., G.P., et al.: The role of advanced sensing in smart cities. Sensors **13**(1), 393–425 (2012)
34. Hartmann, B., Link, N.: Gesture recognition with inertial sensors and optimized dtw prototypes. In: 2010 IEEE International Conference on Systems Man and Cybernetics (SMC), pp. 2102–2109. IEEE (2010)
35. IBM: IBM Big Data & Analytics Hub (2016). http://www.ibmbigdatahub.com/infographic/four-vs-big-data. Accessed 29 June 2016
36. Jayaraman, P.P., Perera, C., Georgakopoulos, D., Zaslavsky, A.: Efficient opportunistic sensing using mobile collaborative platform mosden. In: 2013 9th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), pp. 77–86. IEEE (2013)
37. Kalogianni, E., Sileryte, R., Lam, M., Zhou, K., Van der Ham, M., Van der Spek, S., Verbree, E.: Passive wifi monitoring of the rhythm of the campus. In: Proceedings of The 18th AGILE

International Conference on Geographic Information Science; Geographics Information Science as an Enabler of Smarter Cities and Communities, Lisboa (Portugal), 9–14 June 2015; Authors version, Agile
38. Konidala, D.M., Deng, R.H., Li, Y., Lau, H.C., Fienberg, S.E.: Anonymous authentication of visitors for mobile crowd sensing at amusement parks. In: International Conference on Information Security Practice and Experience, pp. 174–188. Springer (2013)
39. Kotz, D., Henderson, T.: Crawdad: A community resource for archiving wireless data at dartmouth. IEEE Pervasive Comput. **4**(4), 12–14 (2005)
40. Krontiris, I., Dimitriou, T.: Privacy-respecting discovery of data providers in crowd-sensing applications. In: 2013 IEEE International Conference on Distributed Computing in Sensor Systems, pp. 249–257. IEEE (2013)
41. Laney, D.: 3D data management: Controlling data volume, velocity and variety. META Group Res. Note **6**, 70 (2001)
42. Lee, J.S., Hoh, B.: Dynamic pricing incentive for participatory sensing. Pervasive Mobile Comput. **6**(6), 693–708 (2010a)
43. Lee, J.S., Hoh, B.: Sell your experiences: a market mechanism based incentive for participatory sensing. In: 2010 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 60–68. IEEE (2010)
44. Luo, T., Tan, H.P., Xia, L.: Profit-maximizing incentive for participatory sensing. In: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, pp. 127–135. IEEE (2014)
45. Maisonneuve, N., Stevens, M., Ochab, B. (2010) Participatory noise pollution monitoring using mobile phones. Inf. Polity **15**(1, 2), 51–71
46. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Byers, A.H.: Big Data: The Next Frontier for Innovation, Competition, and Productivity (2011)
47. Marfia, G., Roccetti, M.: Vehicular congestion detection and short-term forecasting: a new model with results. IEEE Trans. Veh. Technol. **60**(7), 2936–2948 (2011)
48. Mayrhofer, R., Gellersen, H.: Shake well before use: Authentication based on accelerometer data. In: International Conference on Pervasive Computing, pp. 144–161. Springer (2007)
49. Meng, C., Jiang, W., Li, Y., Gao, J., Su, L., Ding, H., Cheng, Y.: Truth discovery on crowd sensing of correlated entities. In: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, pp. 169–182. ACM (2015)
50. Mirowski, P., Ho, T.K., Yi, S., MacDonald, M.: Signalslam: Simultaneous localization and mapping with mixed wifi, bluetooth, lte and magnetic signals. In: 2013 International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1–10. IEEE (2013)
51. Pankratius, V., Lind, F., Coster, A., Erickson, P., Semeter, J.: Mobile crowd sensing in space weather monitoring: the mahali project. IEEE Commun. Mag. **52**(8), 22–28 (2014)
52. Piorkowski, M., Sarafijanovic-Djukic, N., Grossglauser, M.: CRAWDAD dataset epfl/mobility (v. 2009-02-24). http://crawdad.org/epfl/mobility/20090224, doi:10.15783/C7J010
53. Pournajaf, L., Xiong, L., Garcia-Ulloa, D.A., Sunderam, V.: A survey on privacy in mobile crowd sensing task management. Tech. rep., Technical Report TR-2014-002, Department of Mathe-matics and Computer Science, Emory University (2014)
54. Ra, M.R., Liu, B., La Porta, T.F., Govindan, R.: Medusa: A programming framework for crowd-sensing applications. In: Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, pp. 337–350. ACM (2012)
55. Ruiz-Ruiz, A.J., Blunck, H., Prentow, T.S., Stisen, A., Kjærgaard, M.B.: Analysis methods for extracting knowledge from large-scale wifi monitoring to inform building facility planning. In: 2014 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 130–138. IEEE (2014)
56. Schauer, L., Werner, M., Marcus, P.: Estimating crowd densities and pedestrian flows using wi-fi and bluetooth. Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, pp. 171–177. Networking and Services, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (2014)
57. Siebel, N.T., Maybank, S.: The advisor visual surveillance system. In: ECCV 2004 workshop applications of computer vision (ACV), Citeseer, vol. 1 (2004)

58. Starner, T.: Human-powered wearable computing. IBM Syst. J. **35**(3.4), 618–629 (1996)
59. Tsui, A.W., Chuang, Y.H., Chu, H.H.: Unsupervised learning for solving rss hardware variance problem in wifi localization. Mob. Networks Appl. **14**(5), 677–691 (2009)
60. Wang, Y., Zhang, P., Liu, T., Sadler, C., Martonosi, M.: CRAWDAD dataset princeton/zebranet (v. 2007-02-14). http://crawdad.org/princeton/zebranet/20070214, doi:10.15783/C77C78
61. Wang, Y., Yang, J., Liu, H., Chen, Y., Gruteser, M., Martin, R.P.: Measuring human queues using wifi signals. In: Proceedings of the 19th Annual International Conference on Mobile Computing & Networking, pp. 235–238. ACM (2013)
62. Wang, Y., Chen, Y., Ye, F., Yang, J., Liu, H.: Towards understanding the advertiser's perspective of smartphone user privacy. In: 2015 IEEE 35th International Conference on Distributed Computing Systems (ICDCS), pp. 288–297. IEEE (2015)
63. Weppner, J., Lukowicz, P.: Bluetooth based collaborative crowd density estimation with mobile phones. In: 2013 IEEE international conference on Pervasive computing and communications (PerCom), pp. 193–200. IEEE (2013)
64. Wikimedia Foundation, Inc.: Wikipedia (2016). https://en.wikipedia.org/wiki/Main_Page. Accessed 28 June 2016
65. Xiao, Y., Simoens, P., Pillai, P., Ha, K., Satyanarayanan, M.: Lowering the barriers to large-scale mobile crowdsensing. In: Proceedings of the 14th Workshop on Mobile Computing Systems and Applications, p. 9. ACM (2013)
66. Xu, C., Li, S., Zhang, Y., Miluzzo, E., Chen, Y.F.: Crowdsensing the speaker count in the wild: Implications and applications. IEEE Commun. Mag. **52**(10), 92–99 (2014)
67. Yan, T., Marzilli, M., Holmes, R., Ganesan, D., Corner, M.: Mcrowd: a platform for mobile crowdsourcing. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, pp. 347–348. ACM (2009)
68. Zaslavsky, A., Jayaraman, P.P., Krishnaswamy, S.: Sharelikescrowd: Mobile analytics for participatory sensing and crowd-sourcing applications. In: 2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW), pp. 128–135. IEEE (2013)

Chapter 22
Evaluation of a Web Crowd-Sensing IoT Ecosystem Providing Big Data Analysis

Ioannis Vakintis, Spyros Panagiotakis, George Mastorakis and Constandinos X. Mavromoustakis

22.1 Introduction

The uprising of the Internet of Things (IoT) [1] has brought new opportunities in data analysis and management from a multitude of new databases. This is due to the fact that the development of IoT applications is booming so much that in a few years it is expected to change the current state of Internet in a fully integrated Internet with billions of devices [2]. With the Internet of Things devices like smartphones and tablets generate tremendous quantity of data in a daily base. All this new information is generated and stored in different formats such as structured or unstructured data that varies from text to pictures or audio. Figure 22.1 illustrates an Internet of Things ecosystem with a variety of devices and a variety of choices for storing the retrieved information. The diversity of data has led to the creation of new databases, which can face the new distributed trend, and cut off from the traditional bases. In this context, NoSQL databases, also known as "Not only SQL" databases, have made their appearance in the web development market to offer an alternative solution. The main differences between them and SQL databases is, first that they do not use table format and second that they do not use SQL as query language [3].

I. Vakintis · S. Panagiotakis (✉) · G. Mastorakis
Department of Informatics Engineering, Technological Educational Institute of Crete, 71004 Heraklion, Crete, Greece
e-mail: spanag@ie.teicrete.gr

I. Vakintis
e-mail: vakintis@gmail.com

G. Mastorakis
e-mail: gmastorakis@staff.teicrete.gr

C.X. Mavromoustakis
Department of Computer Science, University of Nicosia, Nicosia, Cyprus
e-mail: mavromoustakis.c@unic.ac.cy

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_22

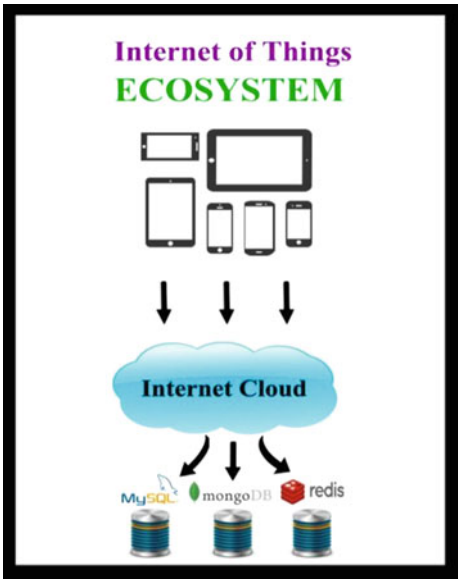


Fig. 22.1 Internet of things ecosystem

The typical Internet provides us with some capabilities such as email, e-shop, and social networking. The IoT vision would expand all these capabilities by letting the user to interact with IoT devices. As IoT devices referred the devices that can be controlled or monitored remotely by Web technologies. Web technologies along with IoT are also known as the Physical Web [4]. IoT applications have been used for a wide range of activities in our society. Transportation and Civil Infrastructure Monitoring [6, 7], Environmental Monitoring [8–14], Health Care and Fitness [15], urban sensing [16] and traffic monitoring, social networks [17] are some areas which benefit from physical web. Smart devices have played a major role to this trend. Smartphones, tablets, music players, sensor embedded gaming systems, and in-vehicle sensing devices (e.g., GPS navigators) are flooding the market and feed with sensor data the Internet. They are equipped with various sensors (e.g., accelerometer, ambient light, camera, microphone, gyroscope, proximity, and meteo sensors), so they transform a near-ubiquitous smart device into a global mobile sensing device [18, 19].

As it is obvious, the sensing capabilities of smartphones can recognize individual or community phenomena. The category of individual phenomena includes several actions of a specific device's owner, which usually are divided into three categories (a) movement patterns such as walking and running, (b) modes of transportation such as biking, driving, or taking a bus, and (c) activities such as listening to music and making coffee. Most of the time, the user can have access to his personal data which are presented graphically as analytics. On the other hand, community phenomena are related to the actions of a set of people and are not limited to a specific user. Community phenomena include real-time traffic patterns, air [20], water or noise pollution, and pothole patrol. The way in which users are involved in the process of collect-

ing sensor data distinguishes sensing of community phenomena to participatory and opportunistic.

One of the biggest challenges for IoT is that many people have the belief that physical infrastructure is separated from Internet infrastructure. But with IoT, all physical components (etc., chips and networks) act as a unified infrastructure. Except from smart devices there are also ordinary objects that can be smart and connected and it is possible to control the physical world from a distance. As it has been understood, all those physical devices produce a significant amount of data, known as Big Data. IoT applications need flexibility, agility, and scalability to handle Big Data and this is the next challenge for IoT. Normally, IoT data are processed at cloud side. There are two options for handling Big Data, relational databases, and NoSql databases. Relational databases are more suitable for structured data and horizontal scalability. Databases need to adopt and meet these new IoT requirements with greater data processing agility, multiple analytical tools including real-time analytics, and consistent views of the data.

In this chapter, we evaluate a web-based crowd-sensing cross-platform in the fields of access networks and databases. The architecture of the platform [5] is based upon various HTML5 APIs in order to sense and collect useful information data for the ambient environment of its users. After the collection of the data, we group and graphically present the retrieved data following statistical processing, so they are available for public use. To the best of our knowledge, this is the first crowd-sensing platform that exclusively uses HTML5 APIS for the collection of sensor data. The application is multisensor as it can collect data from almost all sensors of mobile devices and is totally based on HTML5 features. Besides the use of the platform as a participatory and opportunistic sensing application [21], our endmost aim is to be used with other Internet of Things equipment for the introduction to the third generation of Web characterized as ubiquitous Web [22]. More details about the architecture, the technologies and the capabilities of the platform will be discussed in the second section. Next, we will present the main scope of this chapter that is to evaluate the performance of such a platform in terms of latency when it is accessed via various wireless networks (e.g., Wi-Fi, 2G, 3G) or various databases (MySQL, MongoDB, and Redis) are used for data storage.

22.2 Web Platform Overview and Related Technologies

22.2.1 Architecture

In this section, we briefly describe the architecture of our crowd-sensing platform. To the best of our knowledge it is the first platform that uses HTML5 APIs to deliver real-time sensor data to end users. Our platform is a modern, real-time web application system for gathering sensor data from mobile devices and illustrating them in real time. The basic functionality of the platform include: (1) Gathering

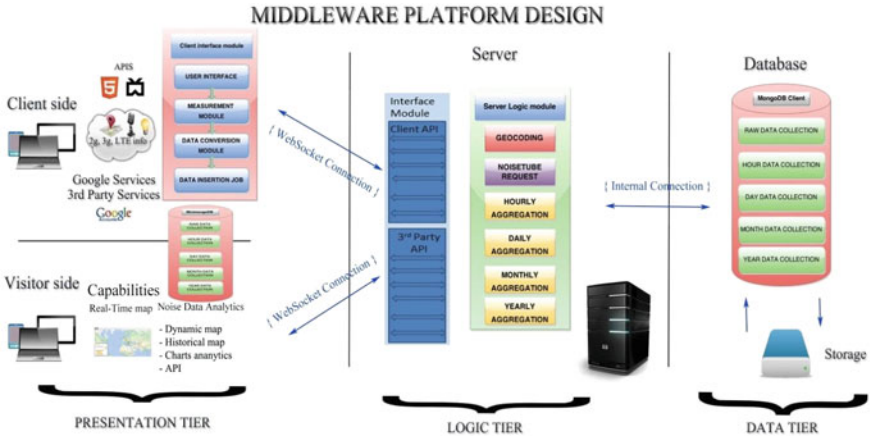


Fig. 22.2 Middleware platform architecture

mobile sensor data (e.g., noise intensity, luminous intensity, and connection-type information) (2) Displaying sensor data in real time (3) Analyzing them through the cloud and presenting the results to the community in order to provide interesting statistical reports. Figure 22.2 shows the design of the platform, which is based on the multitier paradigm [23]. In software engineering, multitier or n-tier architecture is a client–server architecture in which, presentation, application processing and data management are logically separated processes.

The presentation tier includes the client-side, which is the collector of the sensor data and the visitor side, which is responsible for the presentation of the sensor data following statistical processing at server-side. The collected sensor data are illustrated in a fully interactive world map and in nice informative, responsive charts. Also, it offers the data to the community via web services API. The sensor data could then be used for further purposes such as for making surveys, scientific research, or experiments.

22.2.1.1 Client Component

The client component is a web application responsible for the implementation of HTML5 APIs for the communication with the device sensors, the storing of sensor data to the local database and their transfer to the server. The implementation of the client component is based on the Meteor framework. Meteor is a full stack real-time framework that uses MongoDB as its main database and Distributed Data Protocol (DDP), based on websockets, as the communication channel between its components. We use Meteor because its nature is to be real-time by default. Also, MongoDB is a next-generation document-oriented database, which is storing data in a JSON-like format, making the integration of data in certain types of applications

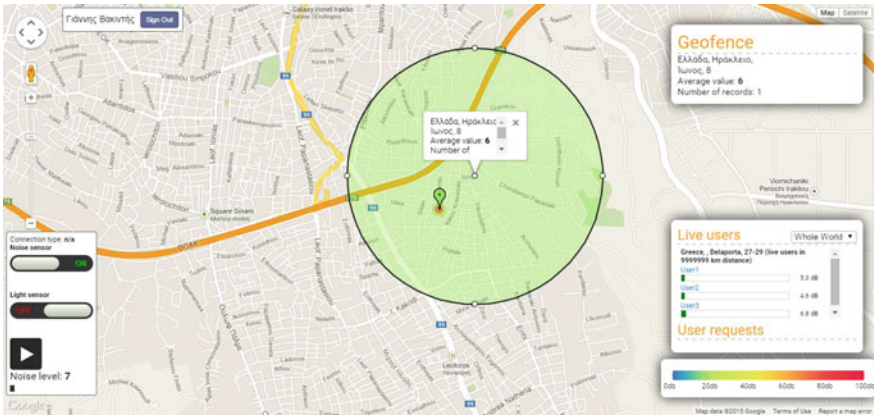


Fig. 22.3 Client application

easier and faster. MongoDB provides scalability and flexibility to the developer. It is perfect for IoT applications, which need very large databases, and also for real-time analytics that need lightweight data. Meteor acts as an application server between the structural components of our platform. In particular, it undertakes to transfer the information quickly and safely to the server via DDP. Probably, the most critical job for Meteor is the synchronization of the data in both client- and server-side. For implementation purposes we have chosen to collect location data and ambient noise and light data provided by the location sensor, the microphone device, and the light sensor of clients, respectively. To this end, we use the Geolocation, the Web Audio, and the Ambient Light APIs, while exploit the getUserMedia() function with a gain node analyzer. The client component is the main source of sensor data in our architecture. The user interface of the client application is essentially a real-time application which includes an interactive Map. It depicts all the online users that use the client application. Figure 22.3 shows a screenshot from the client application. This service displays points with noise and geolocation information over a Google map. After the page is rendered, we initialize the Google map. Also we initialize the overlays for the info panels and the controls.

22.2.1.2 Server Component

The server component is responsible for storing the sensor information from the clients, processing and distributing it to various collections for visualization purposes. The main server jobs are: Reverse geocoding, time aggregation, and NoiseTube data request. With reverse geocoding the server updates user data by translating location coordinates to country, locality, and place information using Google services. Time aggregation job is to manipulate and insert sensor data into collections in order to achieve a spatiotemporal visualization in the visitors' page. Finally, there is the

NoiseTube data request job with which the server fetches data from the NoiseTube API to create more crowd-full visualization charts for demonstration purposes.

22.2.1.3 Visitor Component

The visitor web page component, similar to the client component, is built upon the Meteor framework. The visitor component provides a friendly way for visualizing the data collected by the client devices of our framework in real time and is available to anyone visits our web platform, client or not. By real-time we mean that the data are manipulated by the server at regular intervals so the generated charts visualize the last samples with statistical ways. The visitors' web application provides two ways for visualization:

(1) ***Mapping services.*** The data for the real-time map are derived from two sources of data: Client components and the NoiseTube API. This service displays the last gathered values over a Google map as points with noise and geolocation information. Figure 22.4 shows a screenshot from the real-time map of the visitors' page.
(2) ***Analytics' charts.*** This service displays spatial–temporal analytics. The logic behind this service is to convert sensor data from the collection of records in the database to chart data format. The charts can display daily, monthly, and yearly data for all the countries that participate in the project. In particular the daily graph illustrates hourly information for an exact date, the monthly chart daily information for the selected month, and the yearly chart monthly information for the selected year. Figure 22.5 shows a screenshot of such a chart page. Analytics charts are provided in two formats: (a) time charts, aggregated by country and locality information and (b) averages' graphs, aggregated by country data.

Fig. 22.4 Reactive real-time map

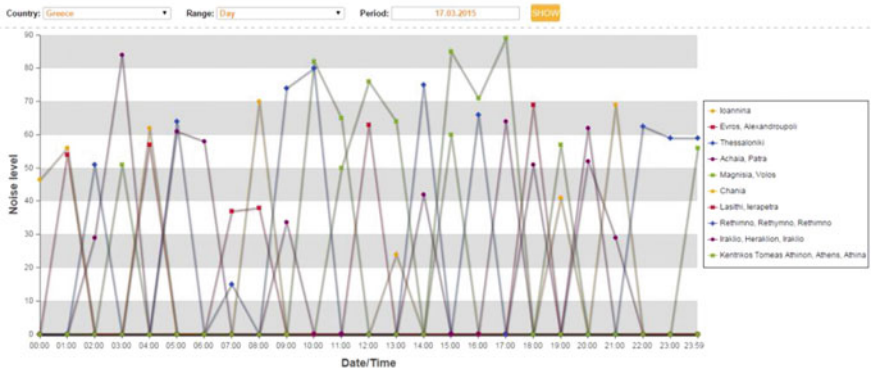


Fig. 22.5 Spatial-temporal analytics daily chart

Averages' graphs are provided as both 2D graphics by the Ext JS framework and 3D graphics by the X3Dom framework.

Another service of the visitors' application is the Heatmap. We use Heatmap in two separate services. The first service provides a world map with historical noise data based on a specific date. There is a filter area which gives users the capability to select data based on location, date, or noise volume range. Figure 22.6 is a screenshot from the historical map.

The second service upon Heatmaps offers the capability to any visitor user to upload custom sensor data to our server, which are then displayed as a Heatmap. Figure 22.7 shows a sample of such noise data that are appeared as heat points in a Google map. Also, the dynamic map service can store such data in the sensor collection of our database, so it can be also used for illustration and statistical purposes.

Fig. 22.6 Historical world heatmap

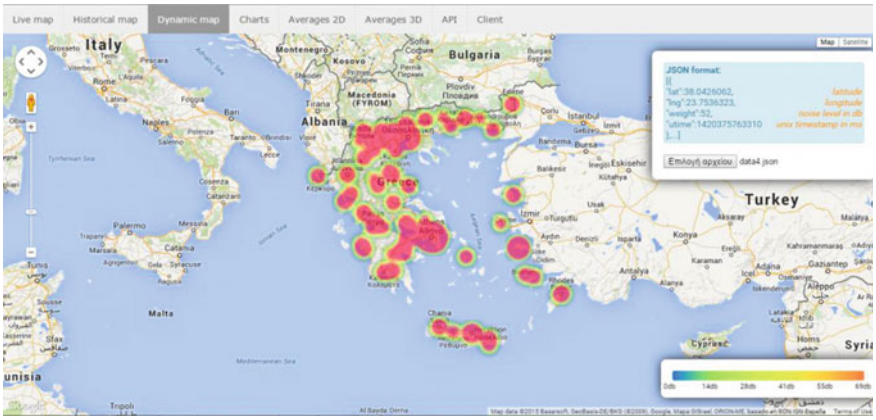


Fig. 22.7 Heatmap from uploaded data

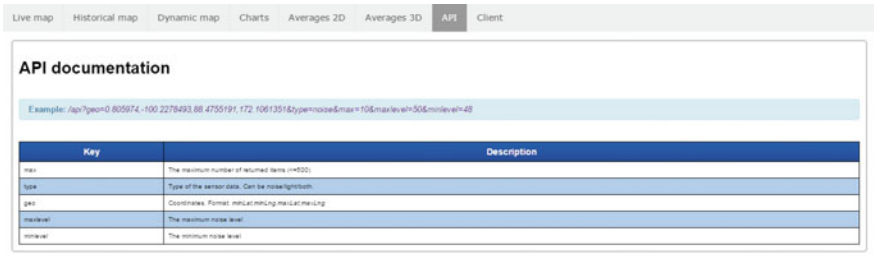


Fig. 22.8 Access data API documentation

The last service is the Access data API that allows users to have access to raw sensor data from our server by specifying parameters such as the maximum number of returned data, the type of the data (noise, light or both), coordinates, maximum and minimum level of sensor data. Figure 22.8 shows the Access Data API documentation from the User Interface of our visitor page.

A more detailed presentation of our platform can be found in [5], which starts with the structural components of the architecture, analyzes the incorporated technologies and services to conclude with several implementation details.

22.2.2 Related Technologies

Our web platform has been constructed with an aggregation of web technologies. Starting with the client component we use various HTML5 APIs to gather the required sensor information such as the geolocation API for user location, the ambient light

sensor API for luminosity, and the network information API for network information. The transmission of sensor data from the client component to the server is made with Websockets in Json format. Also, the platform contains a variety of ways to present statistical information such as Google maps, Ext JS framework, X3D and X3DOM technologies. Finally, the development of our platform made on top of the Meteor web platform. Meteor is considered to be one of the dominant web technologies in the near future since it combines a full stack isomorphic system using the same language (Javascript) in both frontend and backend [24]. Table 22.1 summarizes the related technologies.

22.3 Platform Performance Evaluation

22.3.1 Introduction

The evaluation and performance tests we have performed are twofold. In the first set of tests we evaluate the performance of our platform under various wireless access technologies (e.g., Wi-Fi, 2G, 3G) in terms of latency. In particular, we measure the latency involved in the performance of several tasks such as: (1) the appearance of the marker from a client login to the visitors' page, (2) the delay between the uploading of data to the server and their visualization on a dynamic map, (3) the elapsed time to visualize data in the historical map, charts and averages, (4) the time to retrieve data from the Access data API.

In the second part of evaluation tests, we create a test bed to compare the current database of Meteor (Mongo DB) with two others databases, MySQL and Redis. In more specific, we are benchmarking the three databases to evaluate their differences into fundamental operations such as read and write. In this context, we keep the database size stable and are conducting six performance tests: Data insertion test, Data reading test, Data reading with sorting test, Data searching test, Data removing test, and Data aggregation test.

22.3.2 Latency Tests for Different Access Networks

22.3.2.1 Methodology

The purpose of making the performance tests is to identify how fast our platform services are delivered to the visitors' page under various conditions (i.e., different wireless network technologies, or transferred data sizes). In order to have reliable results we repeat each test 15 times and write down the average latency value. The

Table 22.1 Web platform technologies

Web platform technologies	
HTML5	HTML5 [25–28] is a programming language used for describing the layout and presenting the contents of Web pages
Geolocation API	Geolocation API [29] allows the client-side device to provide geographic positioning information to javascript web applications
Ambient Light Sensor API	The Ambient Light Sensor API [30] senses the environment of the device to provide web applications with the measured luminosity in lux units
Media Capture and Streams API	The Media Capture and Streams API (or GetUserMedia API) [31] offers to web applications access to multimedia streams, such as video and audio, from local devices (webcam or microphone) through a browser
Network Information API	The Network Information API [32] measures the available bandwidth and offers to the developers the ability to adapt web media elements, as images, videos, audios and fonts, accordingly for a better user experience with multimedia content
WebSockets	The WebSocket protocol [33, 34], provides a bidirectional communication channel using a single TCP connection
Web Audio API	The Web Audio API is a high-level versatile JavaScript API for controlling, processing, and synthesizing audio
Google Maps and Google Maps API v3	Google Maps [35] is a web mapping service and technology for desktop and mobile devices that provided by Google. The capabilities of the specific platform are satellite imagery, street maps, and street view perspectives
Google Geocoding APIv3	Geocoding or forward geocoding is the procedure of translating addresses (e.g., Delaporta, Heraklion 71409, Greece) into geographic coordinates (latitude 35.3191579 and longitude 25.1483078) [36, 37]
Geo-fence	A geo-fence [38] is a virtual boundary around a real-world geographical area which defines a point of interest
Meteor	Meteor is a real-time Javascript web application framework which is written on top of Node.js and concludes various packages like MongoDB and jQuery
JSON	JSON or else JavaScript Object Notation [39] is a way to store information in an organized, easy-to-access manner
BSON	BSON [40] is based on JSON objects and the “B” is referred to Binary data. It is a data interchange format that is used mainly as data storage
GeoJSON	GeoJSON [41] is an open standard format for encoding a variety of geographic data structures and is based to JavaScript Object Notation
Ext JS framework	Ext JS [42, 43] is a Javascript application framework suitable for interactive web applications. Ajax, DHTML and DOM scripting are some of the techniques that Ext JS use to present graphics in web pages
X3D and X3DOM	HTML5 [44, 45] through its canvas element enabled the presentation of X3D graphics from web pages without requirement for any plugin.

majority of our tests concern the tier between visitors' page and server. Only our marker display test concerns the wholeness of our architecture, i.e., the tree tiers: client, server, and visitor.

The tool that helps us to complete the elapsed time tests is the "*new Date()*" function of Javascript. The "*new Date()*" returns a data object in the following form: Tue Mar 10 2015 00:03:44 GMT+0200 (Χειμερινή ώρα GTB). When it is specified as current time a new date() minus zero, it responds with the Unix time stamp format of the time. The format of Unix Timestamp has the following structure 1421091697521. It is a big number in seconds which counts the time from January 1, 1970. Hence, specifying a second new date() function at the end of each test and abstracting the first time stamp from the second, we can calculate the duration of each experiment.

In order to emulate different wireless network technologies at the tier between client and server or between visitors and server, we used a proxy server between the client and the server. We emulate 2G, 3G, and Wi-Fi network via the proxy server. The proxy server we used is the WinGate, version 8.2.5 [46]. The proxy server acts as an intermediary between the endpoint device, in our case the computer, and another server from which the client is requesting the service. It provides us with options to adjust the link bandwidth in our network to the one we wish. In the bandwidth control panel we can set up the restrictions for our network.

22.3.2.2 Results

Marker Test

In the marker test we measure the time that our system needs to display a marker in the visitors' live map following the login of a client in our system. To be more precise we measure the time from the moment that the client grants the client application with the right to read his sensor data, pushing the OK button, until a marker associated with his presence is displayed in the live map of our system (both at client and visitors' page). Below is the process schema that we follow:

The client user presses the button ➔ the Client application sends the noise level data to the database (every 1000 ms) ➔ the Geocoding job updates the data collection ➔ the Data aggregation job inserts the data into the live user collection ➔ the Client application updates the live map (every 1000 ms).

Code used:

```
On Client application

insert "new Date()-0" as attribute “user” into collection “sensor”
```

```
On Visitor application

if(parseInt(res[k].user)>142108320){

        var datenow=new Date()-0;

        console.log("created: "+res[k].user+";
                displayed:"+datenow+"; difference:"+(datenow-
                parseInt(res[k].user))+"ms");

                        }

where “res[k]” is the data for each point
```

The final result is the difference between the two time stamps. The first time stamp is created when the data is sent to the database and the second when the marker is displayed in the google map.

```
"created: 1421091697521; displayed:1421091698956; difference:1435ms"
```

Figure 22.9 illustrates the results of the marker test. The test conducted assuming three different wireless technologies between client and server: Wi-Fi (4 Mbps), 2G (250 kbps), and 3G (750 kbps). As it is depicted in the figure the difference between the three access technologies is small. Wi-Fi is the fastest one (1700 ms), 3G is second (2200 ms), and 2G follows with 2400 ms. The small difference between these measurements denotes that most of the elapsed time is consumed in data processing than in transmission or propagation.

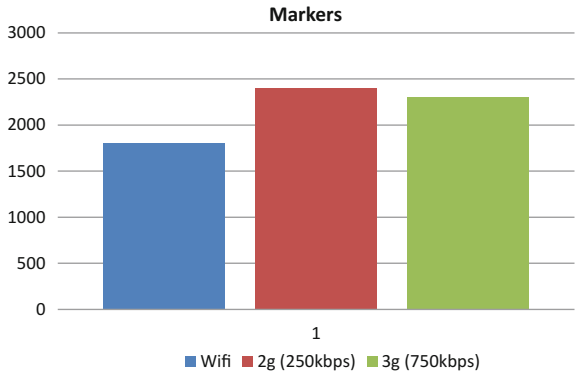


Fig. 22.9 Marker test results

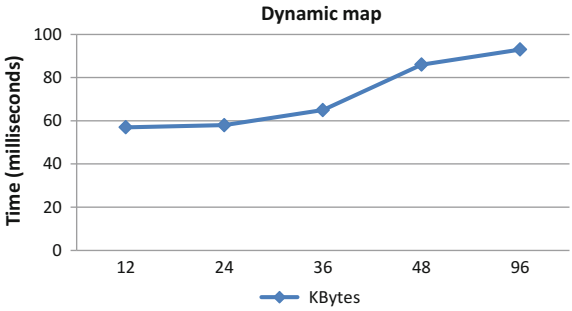


Fig. 22.10 Dynamic map results

Dynamic Map Test

In the dynamic map test we measure the time that elapses between the moment the user presses the button to upload a bundle of sensor data to the server until the time they are displayed in our dynamic map (on the visitors' page). To this end, we make as samples 4 different JSON files of different sizes. The structure of the JSON file has similar structure as our sensor data. It contains 4 fields, two with coordinates, one with time (in Unix time stamp format) and one with noise value. The tested samples are of 12, 24, 36, 48, and 96 KB. Figure 22.10 depicts the results of the dynamic map test. As we see in the figure the system takes more time to display the data when the size of the document is bigger.

Access Data API Test

In the Access Data API test we measure the time that our system needs to return the requesting JSON objects from the database. We perform 3 queries via the API with different parameters. The first query returns 10 objects, the second 100 and the third 500 objects. Figure 22.11 shows the results of the API test. As we see in the chart, the response time of our system for 10,100 and 500 objects is 15, 30 and 80 ms, respectively.

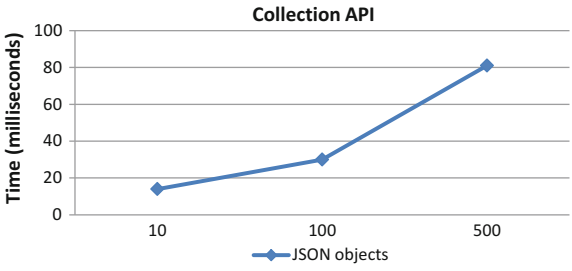


Fig. 22.11 Collection API results

Country, Locality, Averages Charts Tests and Historical Map Test

In the country charts test we measure the time it takes our graphs to be displayed in the visitors' page. The specific test was conducted on January 31, 2015 and the total number of countries that were displayed that day in the visitors' page was 12. Figure 22.12 depicts the results of the country charts test. We repeated the test for three different wireless access technologies: Wi-Fi (4 Mbps), 2G-GPRS (250 kbps), and 3G (750 kbps). As it is depicted in Fig. 22.12, a Wi-Fi connection takes less time to display the charts than the other two technologies. The Wi-Fi connection takes 2745 ms to conclude, with 2G and 3G to take almost the same times, 2842 and 2840 ms, respectively.

Similar tests with country charts were made with the country averages charts. The specific test was also conducted on the January 31, 2015 and the total number of countries that were displayed in the visitors' page was 64. Figure 22.13 shows the results of the country averages charts test. As it is depicted, the Wi-Fi connection takes less time to display the charts with small difference from the 3G connection. The slowest connection is 2G.

Locality charts test was conducted on January 31st and the total number of localities that were displayed in the visitors' page was ten. We take as an example localities of Greece. Figure 22.14 depicts the results of the locality charts test. As it is depicted,

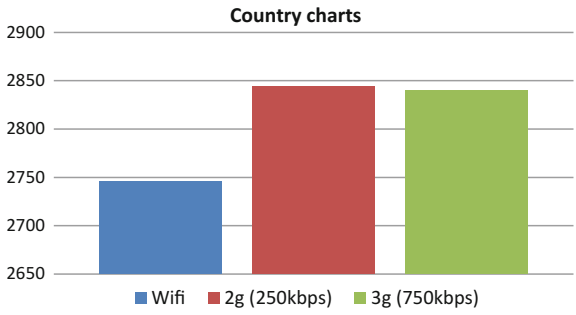


Fig. 22.12 Country charts results

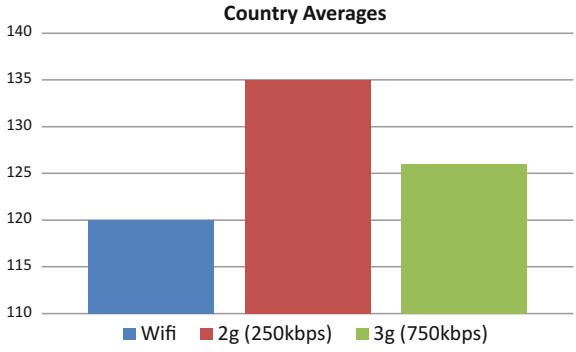


Fig. 22.13 Country averages results

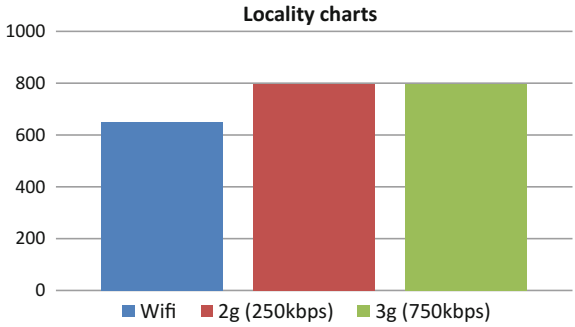


Fig. 22.14 Locality charts

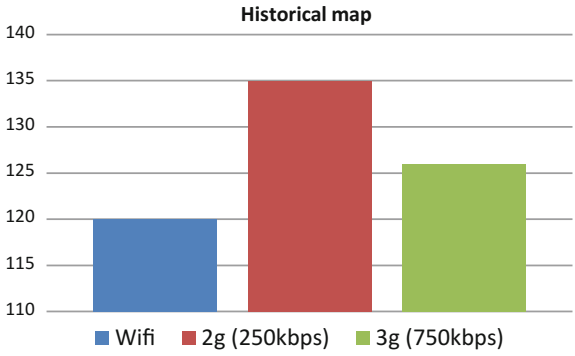


Fig. 22.15 Historical map results

the Wi-Fi connection takes less time to display the charts. The other two connections 2G and 3G had similar times for the specific charts.

The last test was conducted with the historical map. Figure 22.15 shows the results of the historical map test. As it is depicted, the Wi-Fi connection takes less time to display the historical map than the other two wireless connections 3G and 2G. The difference between Wi-Fi and 3G is less than the one between Wi-Fi and 2G.

22.3.3 Benchmarking Database Ecosystems

22.3.3.1 Introduction

Regularly, crowd-sensing applications can be developed with one of the following three models: Remote procedure calls (RPC), publish/subscribe, and database-centric approach. Our platform is based on the third approach, the database-centric. Database-centric is a software architecture where a database plays critical role to all the procedures that take place inside the application. It is the most preferred solution

when the application has to do with Big Data. Generally, a database can provide fault tolerant and reliable transactions. In this benchmark we compare two kind of databases: SQL (MySQL) and NoSQL (MongoDB, Redis).

22.3.3.2 NoSQL Versus SQL

SQL (Structured Query Language) was developed in the 1970s by IBM and since then has become the standard query language for Relational DataBase Management Systems (RDBMS). Databases that belong to SQL category are MySQL, Oracle, and SQLServer. They have slightly different syntaxes but there is not required any significant change when switching from one such system to another. A RDBMS is organized into relations between entities, each of which is represented by a table consisting of rows and columns. The header of the table consists of the list of the columns and the body of the table consists of the rows. RDMS are based in the key concept, which is used to order data or map data to relations. The most important key of a table is the primary key which uniquely identifies the rows of the table. A SQL uses the CRUID tasks to access database entries. The initials CRUD is referred to: Creating, Reading, Updating, and Deleting data. SQL databases [47] are used for low-volume and low-velocity data such as customer data and billing.

- Pros: Follow ACID (Atomicity, Consistency, Isolation, Durability) rules, data integrity, data reliability
- Cons: Scaling problem with growing data volume and workload demands.

NoSQL (not only SQL) is another type of DBMS that can be used on the cloud. NoSQL, refers to a well-known group of non-relational database management systems, where databases are not built primarily on tables, and generally do not use SQL for data manipulation [47]. The NoSQL database is the generation of DBMS which have as scope to eliminate the weak points of relational databases. There are many types of NoSQL databases, such as documents, graph, key-value pairs, and column family. All types have as common that are non-relational. NoSQL deals with data that are more flexible or need a simpler structure. They do not have limitations on data structure, allowing nested documents or multidimensional arrays. Also, they are meant to be schema-free and suitable to store data that is simple, schema-less, or object-oriented. Primary uses of NoSQL Database are [48]:

(1) Large-scale data processing
(2) Basic machine-to-machine information look-up and retrieval
(3) Exploratory analytics on semi-structured data
(4) Large volume data storage.

Table 22.2 summarizes the main differences between general SQL databases and NoSQL databases.

Table 22.2 SQL versus NoSQL

SQL	NoSQL
Relational model	Non-relational data (schema-less, unstructured, simpler)
Tables	Key-value, document, graph, column family stores
ACID	BASE
Consistency	Availability, Performance
Single server	Cluster of servers (Horizontal scalability)
SQL query	Simpler and different API

22.3.3.3 NoSQL Categories

NoSQL databases can be classified into four major categories: Key-Value stores, document stores, column Family stores, and graph databases. In this section we will analyze Key-Value stores and document stores because we use databases that belong to these categories.

Key-Value Stores

Key-value stores are the simplest type of NoSQL databases. They store data in pairs of key and value. The value is a block of data that have any type and any structure. They do not need any schema to be defined and let the user to define the semantics for the values and how to parse the data. They are easy to build and scale and they have good performance. The basic API to have access to the data are: (1) put (key, value), (2) get (key), and remove (key). Redis belongs to the type of key-value store databases.

Document Stores

A document store database uses a database as a collection of documents. It is one step higher than key/value stores. Every document consists of various named fields and one of them is the unique documentID. Document databases are schema free. The data can be of any structure and different among documents. The data types allowed for use vary from strings, numbers, and dates to more complex ones such as trees, dictionaries, or nested documents. The output format can be JSON, BSON, or XML. This is a characteristic that makes document stores databases very popular to developers because the server can support not only simple key-value lookup but also queries on the document contents. MongoDB belongs to the type of document store databases.

22.3.3.4 Overview of Tested Databases

In this section we describe the characteristics of the 3 databases that we use in our benchmark: MySQL, MongoDB and Redis. We analyze the theoretical approach of those databases and we also underline the main functionalities that each supports.

MySQL

MySQL belongs to the category of SQL databases and it is the most popular in business industry. It is owned by Oracle. Many famous applications use SQL such as Facebook, Lindedln, Google, and Twitter. MySQL is a relational database and organizes its data into tables, rows, and columns. For accessing data it uses SQL statements. The basic statements are: INSERT, SELECT, UPDATE, and DELETE. There are also others functionalities such as join, group by, and views.

MySQL supports a variety of storage engines with different characteristics to manipulate data. The default storage engine is the InnoDB after the version 5.5. It is ACID compliant and supports various kinds of transactions such as commit, roll back, and crash-recovery. For caching data in memory it uses a pool buffer. Pool buffer is a linked list of pages, keeping heavily accessed data at the head of the list by using a variation of the least recently used (LRU) algorithm.

MongoDB

MongoDB belongs to the category of document store—NoSQL databases. It has been developed by 10gen and written in C++. It is the most famous NoSQL database and well-known applications such as foursquare use it. Its main characteristic is scalability and speed, so it is suitable to work with large amount of data. The structure of its data has flexible schema. The database itself contains multiple collections and every collection contains multiple documents. Data is stored in BSON format, which is a binary-encoded format of JSON. BSON objects are lightweight, traversable and efficient, so they are very fast in encode and decode operations. Every object contains a unique ID that the system automatically adds it to the object if the user does not assign it. In Fig. 22.16 we can see the structure of the Object_id.

In the field of querying, MongoDB has a very large set of available queries and user does not need to write MapReduce functions. Mongo shell is the MongoDB client that communicates with the database from a command line. The commands to query the database are insert, find, update, and remove. MapReduce operations and a simple aggregation framework are responsive to the aggregations tasks.

Redis

Redis belongs to the category of in-memory key-value store—NoSQL databases. It is considered as a very fast database due to the in-memory storage. It offers high performance and more flexibility than a usual key-value database. A database in

Fig. 22.16 MongoDB object_id

Redis is characterized by dictionaries that are pairs of keys and values. Redis offers a lot of choices for data structure. Data can be stored in: String, list of strings, set of strings, sorted set of strings and a hash. Every data structure has its own set of commands.

Redis store data in RAM in order to achieve high performance. Sometimes, this is a drawback because RAM can be used by other applications or services. Redis can also store data in disks. It uses three methods for data persistence: append-only file, snapshots, and a combination of both. Snapshots save a dataset periodically when a number of keys is changed. On the other hand, append-only file logs all the write operations.

22.3.3.5 Test Bed of Databases

Experimental Methodology and Setup

In this section, we describe the methodology used to evaluate the performance of the three databases: MySQL, MongoDB and Redis, when deployed in our platform. For each experiment that was conducted, we will describe the benchmark functions and commands, the procedure and the final results.

Experimental Overview

The main goal of these tests was to compare the performance of MongoDB, the default database of Meteor, with two other databases MySQL and Redis. The databases were implemented in the cloud server of our platform and the benchmark is considered to be a database benchmark over sensor data. The benchmark architecture also includes 3 different database clients, one for each database implementation. The benchmark performs basic read, write, search, and remove operations and some more advanced, such as search with sorting and aggregation. Each database is evaluated and tested separately, meaning that only one test is running upon time for the current database. Every test comprises a series of requests, such as read and write, from server to database. We set a database client in the server-side of the application. In the following sections we will analyze each test separately. The performance of the databases is evaluated by measuring the elapsed time for each request to conclude. Measurements for each test are taken multiple times in order to maintain reliability. Specifically, the values illustrated in the charts that follow have been derived from the average value of a 10-time run of each test. This is a safe way to ensure that the underlying network will not alter the results.

Test Bed Environment

The whole benchmark was implemented in the server-side of our Meteor platform. The server was deployed on a virtual machine instance running 64-bit Ubuntu 14.14 on a Digitalocean instance (droplet) (1 GB memory, 30 GB SSD Disk). The Database editions that we tested are MySQL 5.6.10, MongoDB 2.6.7, and Redis 2.8.9. In order

to connect and interact with the database servers, the following libraries and drivers were used for the implementation of the database clients:

- MySQL: numtel: mysql [49]
- MongoDB: native Meteor driver
- Redis: slava: redis-livedata [50]

In the case of Redis we needed to run a Redis server and a url to connect to it. This is because Redis is not yet shipped with Meteor. Hence, the following command is needed to be passed in the server in order to start the Redis server:

REDIS_CONFIGURE_KEYSPACE_NOTIFICATIONS=1.

Benchmark Implementation

The scope of the benchmark was to run various common operations for databases and record their performance upon Big Data from sensors. All tests were made on the server-side of our middleware platform (located in the cloud). The benchmark comprises 6 separate tests: data insertion test, data reading test, data reading with sorting, data searching, data removal, and data aggregation. The visitor of the client page of our platform can easily run the benchmark test by the administrator page we have created. The benchmark test's architecture is shown in Fig. 22.17.

All tests are the same for every database. In every test we feed the database with a certain number of documents, namely 10, 100, 500, 1000, and 2000. The scope is to measure the time that elapses between the submission of the query and the time we get the result from the database. We are getting start time upon the submission of the query from the server to the database and we obtain the difference with the

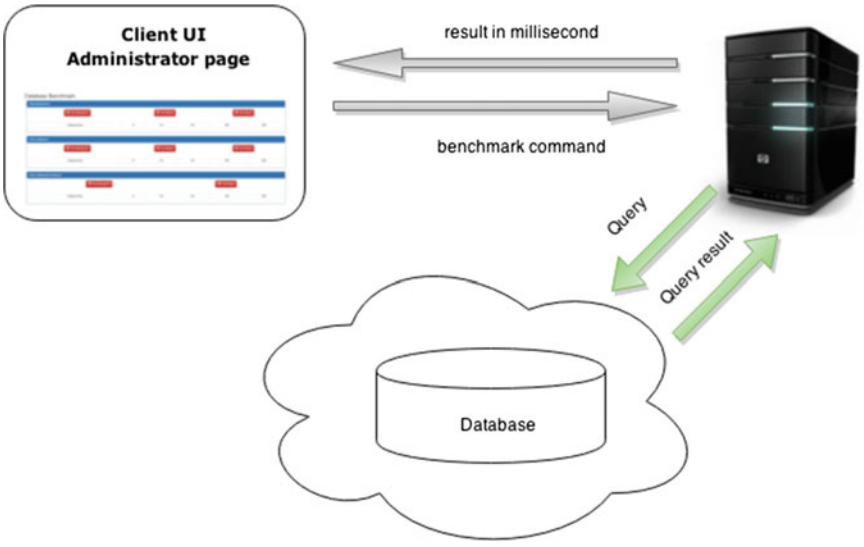


Fig. 22.17 Benchmark architecture

timestamp upon the receipt of the results by the server. After that, the server method returns the difference in milliseconds to the client-side function, which displays it. All the tests include a remove and insert method for data. In every test we start from scratch and this is the reason that it takes some time to proceed. So, for every test, we have two constants:

- The data have the same structure for all records.
- The database collection has no data when we start the tests.

We run the administrator page from the client-side of the Meteor platform. The benchmark command enables the query at the server-side and then the benchmark process starts. To start the administrator page and run the benchmark tests we need to configure the url of MongoDB driver and enable keyspace notifications for Redis:

(1) MONGO_URL=mongodb://localhost:27017/noiseserver
(2) REDIS_CONFIGURE_KEYSPACE_NOTIFICATIONS=1
(3) ROOT_URL=http://html5platform.tk:3400meteor-port3400

Data Structure

The common data structure for all records is shown in Table 22.3. Each document has exactly the same structure as the one we obtain from the client application. It has four numbers and one string. The first two numbers represent the location's latitude and longitude and their values derive from a number array with 10 pairs of coordinates. The third number contains the date in unix format and the fourth contains the sensor data. Finally, the string represents the locality and it is fed from an array of 10 localities.

Experimental Results

In this section, we report the results of all the tests conducted. Figures 22.19, 22.20, 22.21, 22.22, 22.23 and 22.24 display the charts with the measurements in each category. Figure 22.18 shows the administrator page of the benchmark test. Although NoSQL databases claim that they deliver faster performance with Big Data than classic RDBMS databases, we reach to the conclusion that MySQL has better performance in our tests. Also, MongoDB, the default database of Meteor, has very good response time in comparison to Redis. However, Redis is not fully supported

Table 22.3 Sensor data structure

Name	Type	Example
Lat	Double	32.8133112
Lng	Double	4.1426565
locality	String	New York City
Hour	Double	1422012856380
Noise	Double	30.5

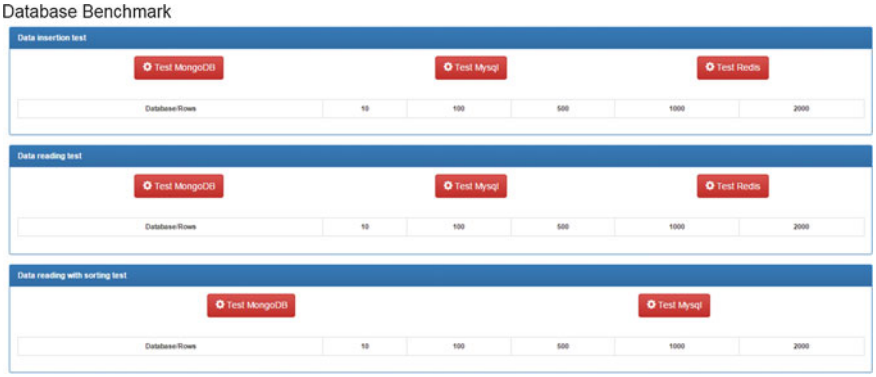


Fig. 22.18 Administrator page

yet in Meteor. Redis in Meteor does not support documents so it makes for each part of the document one record. In case Redis was fully supported, it would definitely perform much better.

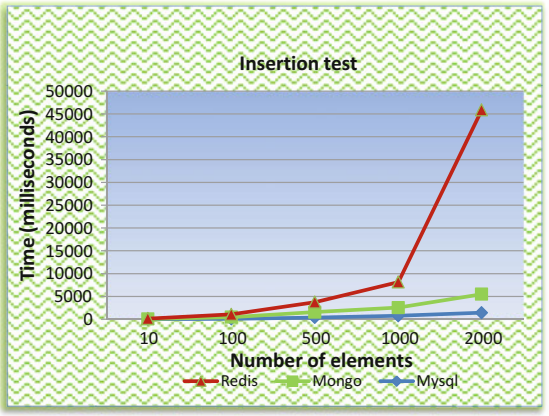
A general outcome from the results is that all databases (MySQL, Mongo, and Redis) have almost equal response time in case of small number of entries but when this number increases, MySQL and Mongo perform significantly better than Redis. It is worth noting, here, that the initial thought of Meteor developers was to use MySQL as the default database when they started to create Meteor. The reason for choosing Mongo, finally, was that it is consistent with Javascript and the Meteor developers was familiarized with it.

All the tests of the benchmark work with the same way: We are getting the start time at the beginning of each procedure and obtaining the difference with current time stamp at the end. When the procedure is over, the server returns the difference in millisecond. The result denotes server time, since the server sends the query to the database and gets the results back.

Insertion Test

Every time we run the benchmark test for a database we consider 5 cases, each with 10, 100, 500, 1000 and 2000 documents, respectively. All tests have the same structure when they start. As we described above, we take some precautions in order to have the same characteristics for every database. When the user presses the button to make a query, we at first remove all the documents from the database and then we insert the necessary documents. Just before the insertion procedure we keep the starting time. When the insertion procedure ends we keep the ending time and we abstract it from the starting time. Figure 22.19 shows the results from the insertion test. MySQL and Mongo has very close response times in almost all cases except for 2000 documents. On the other hand, Redis has satisfactory times until the insertion of 1000 documents but in case of 2000 its performance falls in very low values.

Fig. 22.19 Insertion test



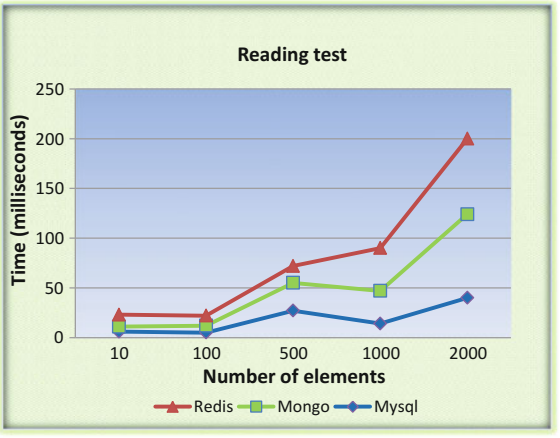
Reading Test

As we referred above, we also have 5 cases for the insertion of documents. Also, we take the same precautions with insertion test. In fact, hereafter we will have a removing and insertion operation exactly before the starting of the main test. When the insertion job finishes we keep the finishing time. The abstraction of insertion finishing time with the reading finishing time gives us the desired reading time. Figure 22.20 shows the results from the reading test. MySQL has a clear superiority against Mongo and Redis.

Reading with Sorting Test

The "read with sorting" test is the same with the reading one with just ordering of the results in the returned recordset. Recordset is descendingly sorted by the "locality"

Fig. 22.20 Reading test



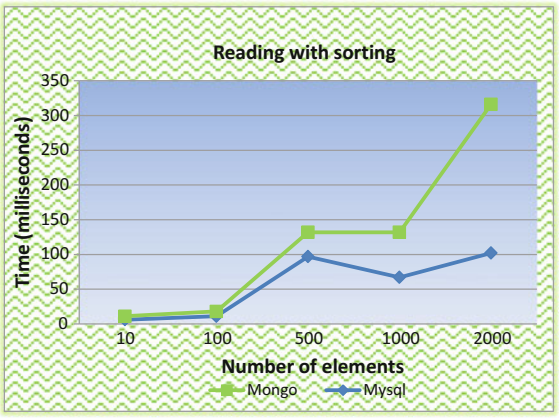


Fig. 22.21 Reading with sorting

field. For MySQLl we use "order by" command and for Mongo we use "sort". There is no ordering support in Redis driver for the moment. Figure 22.21 shows the results from reading test. MySQL has a clear superiority against Mongo in large number of documents. When the procedure starts both databases have similar response times.

Searching Test

In searching test we search the database by locality "Manila". Figure 22.22 shows the results of searching test. MySQL along with Mongo has almost equal times. An important notice is that both MySQL and Mongo has the same performance for almost all the procedure. On the other hand, Redis has the same rate with the other two databases until the last set of documents. The searching time rises from 5 to 324 ms when redis searches between 2000 documents. We also tested Redis in searching on more than 2000 documents and the additional time was disappointing.

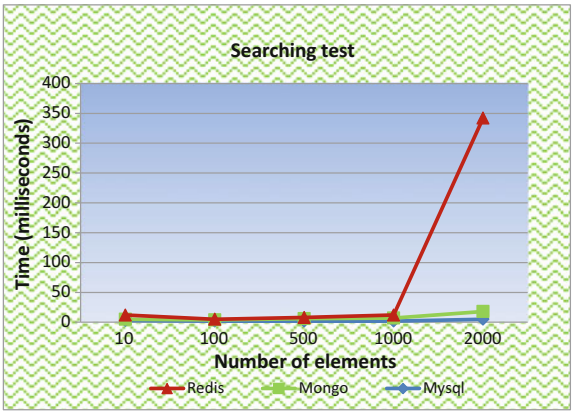


Fig. 22.22 Searching test

Removing Test

In removing test, MySQL is also the dominant database and Mongo the second. For one more time Redis comes third. All databases have close times which sometimes are equal. For 1000 documents both MySQL and Redis need 6 ms to finish the test. Mongo is 1 ms faster. Figure 22.23 illustrates our results.

Aggregation Test

The last test is the aggregation test. MySQL and Mongo were tested in aggregation procedure. By the time we made the tests there was not support for aggregation in Redis. For aggregation we used the "*group by*" command in Mysql and the *meteorhacks:aggregate* library in Mongo. At first we grouped by locality and then we summarized the results of each group. Finally we computed the average value for each group. The aggregation pipeline provides an alternative to map-reduce and may be the preferred solution for aggregation tasks where the complexity of map-reduce may be unwarranted. Figure 22.24 depicts the results of the aggregation test. For one more time MySQL has better times than Mongo. In both databases it takes more time to aggregate small datasets than large ones.

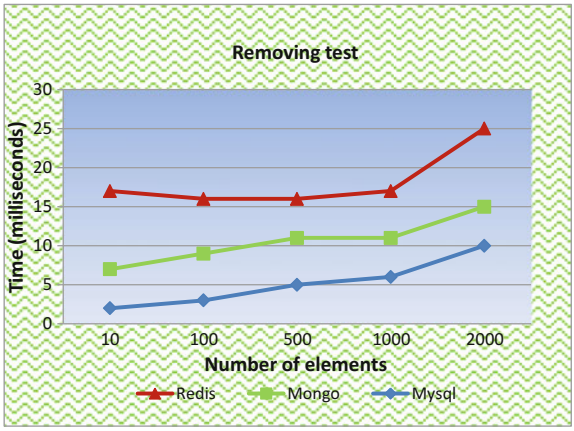


Fig. 22.23 Removing test

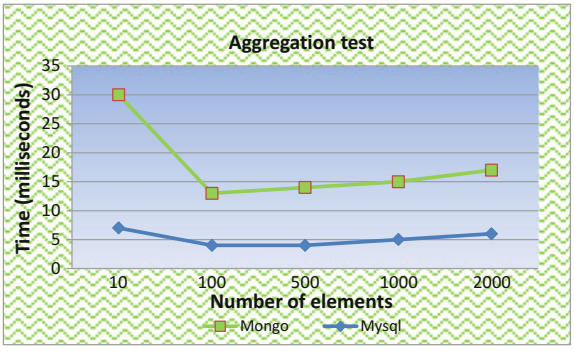


Fig. 22.24 Aggregation test

22.4 Conclusion

In this chapter we presented the implementation of a web middleware platform which is interfaced with the real world through various mobile sensors. HTML5 offers the capability to the developers to interact with mobile and standalone sensors with a web manner. HTML5 sensor APIs offer access to the device hardware with only some lines of Javascript code. In that context, the fundamental functionality of our platform is to gather, process, and visualize the information that acquires from the device sensors. Furthermore, the platform groups and graphically presents the retrieved data following its statistical processing. The uniqueness of the platform is that it solely uses HTML5 APIs to deliver real-time sensor data to the end users. Google maps and rich-interactive charts are some of the visualization ways that are applied. The platform, also, provides more advanced capabilities including an API which integrates with the database for collecting bulk sensor measurements or accessing the collected data. All these web services are offered to the end users via a visitors' component. Sensor data, in general, can be a very important source of information. With appropriate analysis they can offer better understanding of the environment that surrounds us. In that context, the almost real-time analysis offered by our platform based on the events from sensors can be a serious help for various urban communities. Of course, raw sensor data is just numbers. This is why our platform processes, analyzes, and stores the information in forms with human value including maps, graphs, or aggregations.

As we refer above, the purpose of the platform is to transfer sensor data from clients to visitors in an almost real-time manner. For that reason we evaluated the response time of the platform by doing performance tests in various tasks. Performance tests measure the time it takes for delivering sensor data from the initial component (client) to the destination component (visitor). Also, we measured the time it takes for the visitors' page to retrieve data from the database through the server component. The performance tests made under various access communication networks such as Wi-Fi, 2G, and 3G. All the networks had similar execution times with very small differences between them, with Wi-Fi to perform faster followed by 3G and 2G. We also evaluated the latency performance of three well-known databases: MySQL, MongoDB and Redis. We tested the databases in basic operations such as read and write, but also in most complicated ones such as aggregation and read with sorting. These tests can be characterized as competitive tests between two different families of databases: SQL (MySQL) and NoSQL (MongoDB and Redis) ones, over big sensor data volumes. A general outcome from the results is that all databases (MySQL, Mongo and Redis) scored almost an equal execution time with a small number of entries. When the number of entries was increased, MySQL and Mongo had significant superiority over Redis. It is worth mentioning here, that the initial thought of the Meteor's developers was to use MySQL as the default database when they started creating Meteor.

References

1. Wikipedia.: Internet of Things. http://en.wikipedia.org/wiki/Internet_of_Things Retrieved 2015-03-20
2. Jun-Wei, H., Shouyi, Y., Leibo, L., Zhen, Z., Shaojun, W.: A crop monitoring system based on wireless sensor network. Procedia Environ. Sci. **11**, 558–565 (2011)
3. Hammes, D., Hiram, M., Harrison, M.: Comparison of NoSQL and SQL Databases in the Cloud. In: Southern Association for Information Systems (SAIS) Proceedings. Paper 12 (2014)
4. Guinard, D., Vlad, T.: Towards the web of things: Web mashups for embedded devices. In: Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain (2009)
5. Vakintis, I., Spyros, P.: "Middleware platform for mobile crowd-sensing applications using HTML5 Apis and web technologies", Accepted for publication as chapter contribution in the HandBook "Internet of Things (IoT) in 5G Mobile Technologies". Springer (2016)
6. Thiagarajan, A. et al.: VTrack: accurate, energy-aware road traffic delay estimation using mobile phones. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems. ACM (2009)
7. UC Berkeley/Nokia/NAVTEQ.: Mobile Millennium. http://traffic.berkeley.edu/. Accessed 10 March 2015
8. Maisonneuve, N., Matthias, S., Bartek, O.: Participatory noise pollution monitoring using mobile phones. Inform. Polity **15**(1), 51–71 (2010)
9. D'Hondt, E., Matthias, S.: Participatory noise mapping. In: Demo Proceedings of the 9th International Conference on Pervasive (2011)
10. D'Hondt, E., Matthias, S., An J.: An: Participatory noise mapping works! An evaluation of participatory sensing as an alternative to standard techniques for environmental monitoring. Pervasive Mob. Comput. **9**(5), 681–694 (2013)
11. Maisonneuve, N. et al.: NoiseTube: Measuring and mapping noise pollution with mobile phones. Information Technologies in Environmental Engineering, pp. 215–228. Springer, Berlin, Heidelberg (2009)
12. Maisonneuve, N. et al.: Citizen noise pollution monitoring. In: Proceedings of the 10th Annual International Conference on Digital Government Research: Social Networks: Making Connections between Citizens, Data and Government. Digital Government Society of North America (2009)
13. Drosatos, G. et al.: A privacy-preserving cloud computing system for creating participatory noise maps. In: Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual. IEEE (2012)
14. Mun, M. et al. PEIR, the personal environmental impact report, as a platform for participatory sensing systems research. In: Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services. ACM (2009)
15. Consolvo, S. et al.: Activity sensing in the wild: a field trial of ubifit garden. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM (2008)
16. Andreas, K., Eric, H., Aman, K., Feng, Z.: Toward community sensing. In: Proceedings of the 7th International Conference on Information Processing in Sensor Networks, p. 481–492, 22–24 April 2008
17. Miluzzo, E. et al.: Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems. ACM (2008)
18. Campbell, A.T. et al.: The rise of people-centric sensing. Int. Computi. IEEE **12.4**, 12–21 (2008)
19. Pintus, A. et al.: Connecting smart things through web services orchestrations. Springer, Berlin, Heidelberg (2010)

20. Dutta, P. et al.: Common sense: participatory urban sensing using a network of handheld air quality monitors. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems. ACM (2009)
21. Panagiotakis, S. et al.: Towards Ubiquitous and Adaptive Web-Based Multimedia Communications via the Cloud. In: Resour. Manage. Mob. Cloud Comput. Networks Environ. 307 (2015)
22. http://www.labnol.org/internet/web-3-concepts-explained/8908/. Accessed 10 Dec 2015
23. WikiPedia, Multitier architecture. http://en.wikipedia.org/wiki/Multitier_architecture. Accessed 10 March 2015
24. Meteor official website. https://www.meteor.com/. Accessed 10 Dec 2015
25. Sabari website. http://www.sabarimarketing.com/blog/html5-the-fifth-revision-of-the-hypertext-markuplanguage-html. Accessed 10 Dec 2015
26. W3C, HTML5. http://www.w3.org/2014/10/html5-rec.html.en. Accessed 10 Dec 2015
27. W3C, HTML5 recommendation. http://www.w3.org/html/wg/drafts/html/master/. Accessed 10 Dec 2015
28. WikiPedia, HTML5. http://en.wikipedia.org/wiki/HTML5. Accessed 10 Dec 2015
29. W3C, Geolocation API. http://www.w3.org/TR/geolocation-API/. Accessed 10 Dec 2015
30. W3C, Ambient light. http://www.w3.org/TR/ambient-light/. Accessed Dec 2015
31. W3C, Media capture and streams. http://www.w3.org/TR/mediacapture-streams/. Accessed Dec 2015
32. W3C, Network information API. http://www.w3.org/TR/netinfo-api/. Accessed 10 Dec 2015
33. Websocket official webpage. https://www.websocket.org/. Accessed 10 Dec 2015
34. HTML5rocks, websockets. http://www.html5rocks.com/en/tutorials/websockets/basics/. Accessed 10 Dec 2015
35. Google maps official webpage. https://www.google.gr/maps/. Accessed 10 Dec 2015
36. WikiPedia, Geocoding process. http://en.wikipedia.org/wiki/Geocoding/. Accessed 10 Dec 2015
37. The Google Geocoding API. https://developers.google.com/maps/documentation/geocoding/. Accessed 10 Dec 2015
38. WikiPedia, Geo-fence. http://en.wikipedia.org/wiki/Geo-fence. Accessed 10 Dec 2015
39. WikiPedia, JSON. http://en.wikipedia.org/wiki/JSON. Accessed 10 Dec 2015
40. WikiPedia, BSON. http://en.wikipedia.org/wiki/BSON. Accessed 10 Dec 2015
41. WikiPedia, GeoJSON. http://en.wikipedia.org/wiki/GeoJSON. Accessed 10 Dec 2015
42. WikiPedia, Ext_JS framework. http://en.wikipedia.org/wiki/Ext_JS. Accessed 10 Dec 2015
43. Ext JS, documentation. http://docs.sencha.com/extjs/4.2.1/#/guide/charting. Accessed 10 Dec 2015
44. W3C, HTML5. http://www.w3.org/html/wg/drafts/html/master/. Accessed 10 Dec 2015
45. WEB3D. http://www.web3d.org/wiki/index.php/X3D_and_HTML5. Accessed 16 Dec 2015
46. Wingate, proxyserver. http://www.wingate.com/download/wingate/download.php
47. Heroku, NoSQL databases. https://blog.heroku.com/archives/2010/7/20/nosql. Accessed 10 March 2015
48. Moniruzzaman, A.B.M., and Syed Akhter, H.: Nosql database: New era of databases for Big Data analytics-classification, characteristics and comparison. arXiv:1307.0191 (2013)
49. Reactive MySQL for Meteor. https://github.com/numtel/meteor-mysql
50. Redis Livedata. https://github.com/meteor/redis-livedata

Chapter 23
A Smart City Fighting Pollution, by Efficiently Managing and Processing Big Data from Sensor Networks

Voichita Iancu, Silvia Cristina Stegaru and Dan Stefan Tudose

23.1 Introduction

World population is on the rise and becoming more and more urbanized which, in the short term, translates to urban areas becoming increasingly populated. This trend pushes to the limits the existing fabric of urban facilities and gives rise to a need for a better and more efficient city infrastructure, that can withstand the effects of an increasing population density.

This is why, one of the major environmental concerns of our time is *air pollution*. Apart from severely degrading the natural environment, air pollution directly affects our health. Short-term and long-term effects range from light allergic reactions, such as: irritation of the nose, throat and eyes; to serious conditions like: bronchitis, pneumonia, aggravated asthma, lung and heart diseases.

In this chapter, we propose a system, representing a *smart environment* that should increase air quality in crowded urban areas, such as crossroads or highways. The goal of the resulting *Smart City* is to monitor the air quality by pinpointing polluted areas of the city in *real time*.

Both the *sensor networks*, in the context of *Internet of Things*, and *Big Data* services are offered to citizens to provide various types of functionalities, that they will be able to use in everyday life. They are of strong interest for large communities, which can benefit from

V. Iancu · S.C. Stegaru · D.S. Tudose (✉)
Computer Science and Engineering Department, University Politehnica of Bucharest,
Splaiul Independenţei 313, Sector 6, Bucharest, Romania
e-mail: dan.tudose@cs.pub.ro

V. Iancu
e-mail: voichita.iancu@cs.pub.ro

S.C. Stegaru
e-mail: silvia.stegaru@cs.pub.ro
URL: https://cs.pub.ro/

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_23

1. a detailed perspective upon the surrounding world, via *sensors*;
2. the robust and reliable infrastructure behind the *Big Data services*.

The two aspects are forecasted to continue to grow in the near future, which is why we think that they are likely to represent a solution for today's cities *de-noising and cleansing problems*, also known as *pollution*.

Nowadays cities rely more and more on the *fast growing amounts of data*, coming from sensor networks. Consequently, these networks heavily increase their node number as they expand geographically. The involved sensors are measuring a wide range of aspects, such as: temperature, power consumption, gas consumption, radioactivity, earthquakes, pollution, etc. They may be interconnected via *wired* or *wireless*, even *mobile*, networks (e.g., PANs, such as Bluetooth, Zigbee, 6LowPan, etc.), which usually take the information gathered from the sensors to a data interpreting and processing center. This interpreting and processing center is needed because of the individual sensor's reduced storage and processing power. That is why the sensors themselves are unable to process and interpret the information extracted from the environment, in order to obtain relevant information about it. This relevant information can only be obtained by processing *large volumes of data* and possibly interpolating them over time. The processing of the *real-time Big Data* coming from the sensors needs to be *performant*, *scalable robust* and *reliable*, in order to be able to issue relevant actions, in due time, for the envisaged Smart City. Thus, the *Big Data* processing systems are able to extract important relevant information. They do this by processing large volumes of apparently insignificant things that gather over time and are organized in a *historical* manner. This is possible by using AI (*A*rtificial *I*ntelligence) data mining or by means of HPC (*h*igh *p*erformance *c*omputing) techniques, depending on the type of information to be processed and on the expected types of results.

In the remainder of this book chapter, we will detail our view of a Smart City which benefits from the combined help of the sensors and of Big Data techniques. We will focus mostly on the description of its *architecture*, namely on: (1) the management of a *reliable* and *trustworthy* mobile and static sensor network, which will gather the data; and (2) the *spatial* and *temporal* Big Data storage organization. We will also point out hints about the way in which we envisage that the Smart City actions will be derived from the Big Data and the way in which they will actually be materialized by the Smart City's actuators. In Sect. 23.2 there is a description of the related scientific literature. In Sect. 23.3 there is an overview of the *highly available architecture* of our Smart City. For the architecture, we detail each individual component and the interactions among them, with a special accent on *load balancing*, *fault tolerance*, *scalability*, and *trust*. The Big Data will prevail into discussion in Sect. 23.4, which is a special section arguing about the way to organize the data, both physically and logically, in a scalable and highly available manner. It is also about how Big Data systems are of help for a Smart City fighting pollution. A short hint toward techniques that could be of use to extract meaningful information from the data gathered using the sensors, will be given in Sect. 23.5. Based on those data processing techniques,

we have introduced in Sect. 23.6 a detailed description of effective measures that the Smart City could take, based on its internal mechanisms. Last but not least, the conclusions and directions for future developments, regarding our Smart City's architecture, can be found in Sect. 23.7.

23.2 State of the Art

Even if real Smart Cities do not currently exist, we base our proposal for a *Smart City fighting against pollution* mainly on the scientific research within the fields of Big Data and of both wired and wireless sensor networks.

We rely on many efficient existing high availability mechanisms [1], since extra care has been taken to:

1. *balance the load* between the sensor data gathering nodes, in a DHT-like (*D*istributed *H*ash *T*able) manner;
2. use *fault tolerant* techniques, such as data replication, journaling, or even interpolation based on neighboring data that has not been lost, in order not to lose the recent data gathered by the sensors;
3. organize the data gathering nodes in a *scalable architecture*, in order to obtain *high availability* for the whole Smart City system.

In our work, we have only addressed security in terms of *trust* between the sensor nodes.

For Big Data, the scientific background we base our work on is related to: (1) on one hand, the yet classical NoSQL system based on MAP-REDUCE, which is Google Big Table [2], optimized by Blobseer [3] for more reliable Big Data organization, by Berkeley Spark cluster for very efficient data retrieval [4, 5] and very recently by the brand new and radically different Google Spanner [6]; and (2) on the other hand, data center and High Performance cluster techniques and tools, such as LUSTRE FS[7] and tape libraries, which are techniques that are in use or we could use within the University Politehnica of Bucharest's cluster.

Sensor Networks are a technology that can offer a significant contribution in completing the ubiquitous computing paradigm and they now represent a new revolution in computing. Growing importance of context-awareness, as an enabler for more intelligent, invisible, and autonomous applications and services has highlighted the need for a greater integration of the physical with the digital world [8], which is also the case of a Smart City. We propose to study how sensor networks scale to the standards of a Smart City infrastructure, with emphasis on pollution tracking [9]. Among the topics of interest, we consider remote monitoring and control of the sensor network [10] and also integration of the data gathered from sensors into a Big Data management and processing infrastructure.

23.2.1 Big Data Logical Organization Techniques

Big Data is a domain that has emerged as a result of a long chain of data storage techniques, including: (1) SQL databases, such as MySQL [11]; (2) database clusters, such as MySQL cluster [12]; (3) distributed file systems, such as NFS (Network File System) [13]; (4) and even the World Wide Web [14], viewed as a huge collection of unstructured, but still useful, data. Data gathering from a large collection of sensors has not been exploited so much until recently, but it contains a huge challenge for the Big Data storage, retrieval and processing techniques.

The forerunner of Big Data storage platforms would be the peer-to-peer data sharing systems, such as the DHTs (Distributed Hash Table): Chord [15], CAN [16], Pastry [17], and Tapestry [18]. They have appeared in the early 2000 s and have grown very popular in sharing especially multimedia content. The stored data can have a variable size and is not usually something to be stored in an SQL-like database, but in a distributed filesystem.

The most representative DHT is Chord, being an elegant and efficient solution for a fast, fault tolerant, highly scalable, and fairly load-balanced DHT solution. Based on the SHA-1 [19] hash, Chord is fit not only for data sharing, but also for uniformly hashing distributed data among Internet users. Chord FS (CFS) [20] is a distributed filesystem, which is a layer above Chord that has better performance than NFS when sharing large amounts of data coming from volatile nodes.

We consider that DHT systems are still fit for storing and partitioning larger collections of data into smaller and more manageable ones.

DHTs and the other peer-to-peer systems have been data sharing finding and retrieving techniques that were developed by small companies or by research organizations, having both practical and scientific characteristics, to illustrate the feasibility and the usefulness of such tools. A more systematic and visible approach, from the point of view of the ordinary Internet users, has the proprietary solutions developed by companies such as Google. In 2006, the company has published a well-known paper about Google Big Table [2], concerning the organization of data that would be fit for web page search queries. In [2], Google first introduces the MAP-REDUCE concept, a flavor of fork-join for large sets of parallel and distributed tasks.

Based on Google Big Table [2], the open source community, composed of both scientific and industrial players from the Internet field, has developed the very popular system called Hadoop [21], which stores its data in the HDFS (Hadoop File System) [22] filesystem. Hadoop is a flexible and extensible tool, and, being open-source, it has been subject to many types of improvements, one of which is Blobseer [3], developed to store large amounts of binary data, thus its name: Binary Large OBjectS.

The data retrieval system is very important, and should be extremely efficient when it comes to collections of Big Data that need to be structured and processed. This has been the subject of the PACMan in-memory cache solution [23] and of the

Orchestra data flow reordering and planning solution [4], both being developed by the University of Berkeley, as extensions to HDFS-like storage systems.

Still, none of the above Big Data solutions is really fit for gathering huge amounts of data from sensor nodes planted throughout a large city. The nodes can transmit frequently data toward the datacenter, in order to store the data for later, periodical processing. We plan to complement the features of each of the above described tools, in order to achieve the goal of a Big Data storage system for the data gathered via the sensor nodes of a Smart City.

A Big Data storage system is extremely useful within the Smart City when it comes to detecting failures in a large network of nodes.

23.2.2 Failure and Abnormal Behavior Detection (FABD) Based on the Big Data

The Johnson Space Center describes the goal of failure management as being: "to effectively detect faults and accurately isolate them to a failed component in the shortest time possible" [24]. As such, maintenance operations become necessary in a Smart City, in order to establish a trust system for the data coming from the nodes.

From an architectural point of view, failure detection modules for sensor networks can be classified into server-centric and self-organizing systems.

In the first case, a server is used to gather data from the sensor nodes and process it to obtain an accurate, global view of the activity in the network. Usually, only part of the data in the network is analyzed in order to avoid congestion and occupying the bandwidth. Consequently, the analysis that takes place at the server side is not fully informed. However, depending on the types of operations applied on the data, this approach can lead to a strong or a weak detection.

Sympathy [25] is a server-based tool, useful for detecting and debugging failures in wireless sensor networks. The analysis involves comparing data from the sensors against certain metrics and classifying the result into a small set of possible scenarios.

MANNA [26, 27] is a different type of server-based tool which only analyzes the battery power of the sensor nodes in a WSN. It creates an energy map for each node in the network and only exchanges messages when a node's internal state has changed. This event-driven method can only detect crashes and can generate false positives, if there are connectivity problems in the network.

A different approach for detecting failures is ANFD (Adaptive Neighborhood Failure Detection) [28]. The authors use piggybacking in order to minimize the communication overhead. Moreover, this server-based method also uses self-organizing techniques, by letting the nodes decide when a sensor node should be investigated. These local perceptions of neighboring nodes are contained in lists, which the nodes exchange among each other and later piggyback to the central server. This hybrid approach can identify crashed nodes, and also communication problems.

As opposed to the server-based approach, self-organizing systems (i.e., TinyD2 [29], LD2 [30]) do all the processing directly on the nodes. This method avoids the extra overhead implied by sending messages to a central instance. Unfortunately, this method is even poorer informed than the previous approach because it lacks feedback from non-neighboring nodes and its input can be biased if nodes have a small number of neighbors.

It follows that the failure and abnormal behavior detection model strictly depends on the sensor network design.

23.2.3 City-Wide Wireless Sensor Network Infrastructure

Wireless sensor networks have been the focus of intense research in the past decade. Most of these deployments are designed to be vertically integrated as an infrastructure [31–33]. In such a system, the sensor network is designed with only one specific application in mind, in which is the system that has full knowledge of all the parameters and functionalities the sensor network can provide.

However, a recent trend is for an application to spread over different sensor networks in order to have a more comprehensive view of the system. The large number of sensor networks an application might manage can lead to unwanted complexity and can become increasingly cumbersome, to a point at which it cannot further maintain coherency.

Another aspect is the reuse of an existing infrastructure for multiple applications, which could avoid the deployment of similar sensor networks at the same physical location and, thus, allow for a minimal investment cost and higher returns. Recent research has been focused on changing the tightly coupled vertical system to a more flexible sensor integration framework [34–36].

The aim is system horizontalization; the breakup of initial systems into reusable components which can be addressed by any application. These frameworks provide functionalities that significantly reduce the complexity of interaction between the application and the underlying system.

Some sources of information on air pollution already exist and are publicly available [37–39]. However, they do not cover the entire monitored area as they are based on measurements performed at fixed locations. Thus, it is not easy to discover localized pollution sources. Our system has the potential of collecting much more data than a traditional one due to its multi-agent architecture. The more users in the system, the larger the area covered will be and better granularity. In addition, our devices could be installed on public transportation vehicles in order to provide an accurate overview of the pollution generated by traffic.

IrisNet [34] is one of the first systems to try and aggregate data and provide access to different geographically distributed sensor networks over the Internet. It works by reusing the existing infrastructure of sensor networks and sharing existing sensor feeds among different sensing services through the use of a distributed database in which sensor data is collected.

Hourglass [40] creates a framework that connects homogeneous sensor networks which are geographically distributed to applications that use and manage the sensor information. Hourglass delivers a data collection network which manages service naming and discovery, routing from sensor networks to applications and a support for integrating services to perform data aggregation and buffering.

Sensor Web Enablement (SWE) [41] is a set of standards developed by the Open Geospatial Consortium which enable the discovery, exchange, and processing of sensor information and also the access to sensor networks via the Internet. SWE desires to achieve a true plug-and-play integration of sensor networks and to enable their access and control via web-based applications.

SenseWeb [42] shares information from globally distributed sensor networks to applications. At the center of the framework lies a coordinator which manages access of applications and sensor network publishers. This central broker coordinates information access of all applications to their relevant sensor networks.

The Urban Sensing Project [36] has three types of applications: urban, social, and personal. The personal application uses only end-user information and is targeting only the end users. The social application is similar to a social networking site where users share data for free. The urban scenario has users sharing data between them but also with the general public, which imposes some new constraints on privacy and security. The authors argue that a new network architecture is required to share the data in order to ensure sharing and quality checking. Through these scenarios, the authors want to reveal the evolution of networks from single domain, to collective sharing of WSN data and finally to a large-scale fully featured infrastructure, that is likely to be used in a Smart City.

23.3 A Highly Available Architecture for the Smart City

We have seen the existing elements and premises on which we could build a *Smart City*, in Sect. 23.2. In this section we focus, from the Big Data organization point of view, on how we think of partitioning the very large set of data, gathered from all over the city, into smaller sets, using DHT-like techniques. Furthermore, we plan to manage the obtained partitions by means of enhanced Hadoop-like systems, being able in the end to efficiently retrieve the necessary data at any time in the future. This is a generic model, which can be used for many Smart City scenarios, and which we are going to fully describe in the case of pollution sensors.

We present an overview of the highly available architecture of our proposed Smart City, by detailing its individual components and the interactions among them, with a special accent on load balancing, fault tolerance, scalability, and trust derived from failure detection.

23.3.1 General Overview of the Architecture

The architecture of the Smart City's mechanisms, which help it fight against pollution, can be observed in Fig. 23.1. Starting from lowest to highest level, its components are

1. The Smart City's actuators, which is a set of active elements that can be usually found all over the city, which can also react to some commands, and slightly change their behavior. They can be *smart traffic lights*, *reversible traffic lanes*, etc. They are controlled by the *data processing unit*, which will be described into more detail in Sect. 23.3.5.
2. The pollution sensor network is the wired and wireless sensor network gathering information about the types and quantities of particles within the air surrounding the sensor. This network sends its data to the *data acquisition module* and is controlled by the *node control module*, both of which are described in Sect. 23.3.3. This component will be detailed in Sect. 23.3.2.
3. The data acquisition module is a distributed software component that serves groups of sensors located in the same vicinity, with each of these data acquisition modules furtherly sending the data one level upwards, toward the *Big Data*

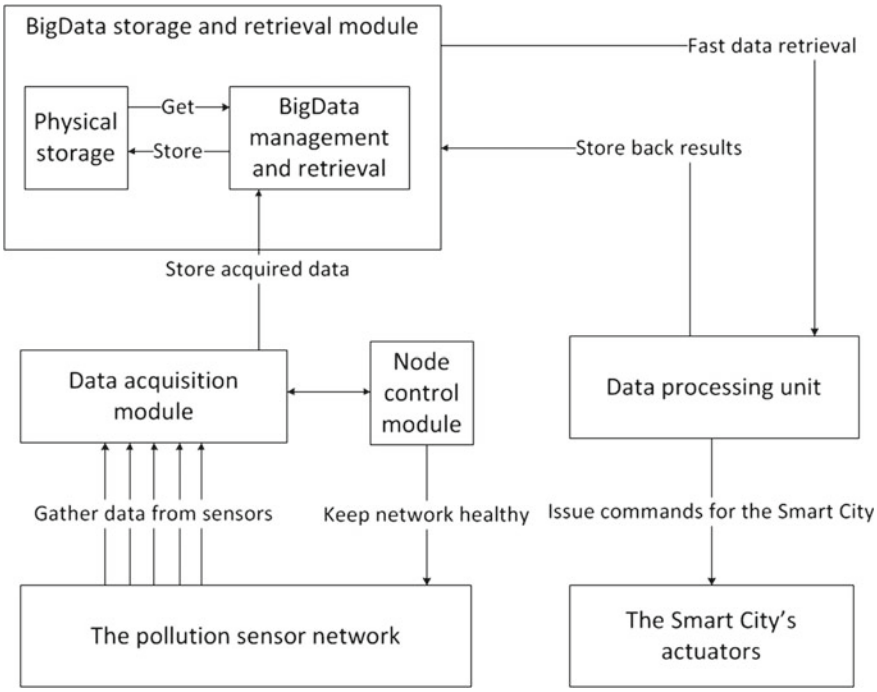


Fig. 23.1 The overall architecture of the elements involved in the Smart City fighting pollution

storage and retrieval module. This component will be described in more detail in Sect. 23.3.3.

4. The sensor control module is a distributed software component that communicates both with the *data acquisition module* and with the *pollution sensor network*, in order to maintain a healthy and functional *sensor network*. This component will be described into more detail in Sect. 23.3.3.
5. The Big Data component represents a combination of the hardware and software solutions that are used for storing the data coming from the sensors, both on a geographical and on a historical dimension. Its subcomponents are, thus, the *Big Data physical storage solution* and the logical *Big Data management and retrieval solution*. As a whole, this component interacts with the *data acquisition module*, to store the data acquired by it, and with the *data processing unit*, for fast data retrieval and for storing back into the Big Data physical storage the processed results. The present component will be described into more detail in Sect. 23.3.4.
6. The data processing unit is a software component that periodically extracts meaningful information from the data acquired from the sensors, data that has been previously stored in the *Big Data storage and retrieval module*. The data processing techniques could include data mining techniques and even high performance computing (HPC) techniques. It interacts with the storage platform, for retrieval and write back, and also directly with the *Smart City actuators*, whose adaptive behavior it determines. The present component will be described in more detail in Sect. 23.3.5.

All the above-mentioned components are parts of the Smart City, and they should function as a highly available distributed system. They will be presented, in order, in the remainder of this section.

23.3.2 The Wireless Sensor Network

There are major issues that need to be tackled in urban areas in order for the new infrastructure to work properly. Most of them are related to the environment, privacy and security and network reliability. To meet these requirements, sensors and actuators will probably need to be ubiquitous and, therefore, integrated into every aspect of urban life. This will lead to the mapping of physical space into a virtual network, or creating the paradigm of Internet of Things (IoT), which will allow users to interact both locally and remotely with the physical environment.

When applied to a large-scale urban scenario, the IoT paradigm gives rise to a so-called City Information Model (CIM) in which the design of every aspect of urban fabric, such as buildings, sewage, water, gas, and power distribution, street management, public transportation, etc., are shaped and governed by this new paradigm.

Buildings, whether residential or offices will be automated by wireless sensor and actuator networks in order to enhance and improve their users' daily activities. The envisioned applications of such networks are multiple, from sensing environmen-

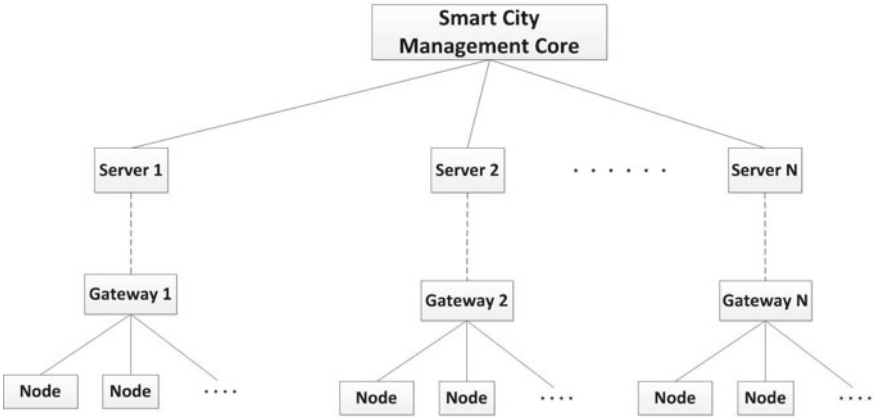


Fig. 23.2 Smart City infrastructure

tal conditions like pollution, temperature, humidity, luminosity, user presence in a certain room, or measuring energy consumption rate. The system can then automatically adjust itself to a certain energy management policy by switching on lighting or heating only when certain parameters are met.

In our Smart City, buildings can have status and performance indicators which are reported in real time and are calculated by constant measurements of physical data from sensors. The sensor nodes placed on buildings are wired to the Smart City network, thus being static. However, each involved sensor node has its own unique identifier, for example the IPv6 address, which enables even ordinary people to provide data to the network, such as information about the degree of pollution. The sensor nodes in this case are mobile, i.e., sensors on mobile phones or any devices that have an active Internet connection.

Each sensor node is connected to a gateway, as shown in Fig. 23.2. Each root gateway will communicate with its dedicated server, forwarding to it the information collected from the nodes, as described in Sect. 23.3.3.

The solution offered by our system relies on devices that can detect an abnormally high concentration of an air pollutant. The devices record the measured data along with location coordinates and can periodically transfer them to a central processing system. Gathered data can be used not only to monitor pollution, but can also have very interesting secondary usages in measuring overall efficiency of buildings and products, population changes, energy production and consumption, and even individual movement patterns.

The same concept of traffic management which is in common use today for monitoring and altering car traffic on crowded roads and intersections can be applied on another scale in public transportation systems. For example, usage of subway or buses can be monitored by measuring how many passengers pass through ticketing barriers or validate their tickets.

By capturing this huge amount of data into a storage system, such as the one described in Sect. 23.3.4, management systems can optimize passenger flow, reducing wait times and crowding.

23.3.3 The Data Acquisition and Node Control Model

Given that the Smart City generates a large amount of information in short periods of time, the data acquisition model is a vital component within the Smart City architecture. Designing this component is not straightforward, since it has to handle real- time data. Moreover, the model has to take into account that nodes can fail and information can be misleading.

Before beginning to describe the architecture of a trustworthy data acquisition system for a Smart City, we need to analyze the types of faults that should be addressed. The authors of [43] have classified faults into two major categories

- Functional faults: usually manifest themselves as node crashes or transmission timeouts.
- Data faults: the node sends erroneous data, either because it malfunctioned or because it has been compromised in terms of security.

We propose to mainly take into account data faults, but also handle faults in which nodes have stopped responding.

The network is composed of nodes and gateways, the latter providing routes to the application server. In order to minimize the overhead of the messages sent, they should mainly be piggybacked together with relevant data. The server does not ask for data from the nodes, unless it has to investigate a fault and requires more information. Instead, it receives periodical samples from the nodes in the network. The samples that have to be processed or that are very recent can be stored in a local database. However, as soon as the samples have been processed, they should be moved to the *fresh storage system* described in Fig. 23.5 to make room for new data from the sensor nodes. The purpose of the fresh storage system in this case is to maintain a list of structured entries with the previous values from the nodes and to eventually process data at a higher level, namely by the Big Data processing unit described in Sect. 23.3.5.

The data acquisition and node control functionalities are combined to form the *F*ailure and *A*bnormal *B*ehaviour *D*etection (FABD) module. This module itself consists of several components, which can be visualized in Fig. 23.3. The local database, also called the *fresh storage system*, is used to store the data received from the nodes, through the communications handler, and provides data for the data analysis component. This component transforms the data into useful output, that the management component can be used to decide upon a node's *degree of trust* (e.g., the data analysis component only provides the rough analysis results, but the management component applies the thresholds imposed by an administrator in order to reach a conclusion concerning trust).

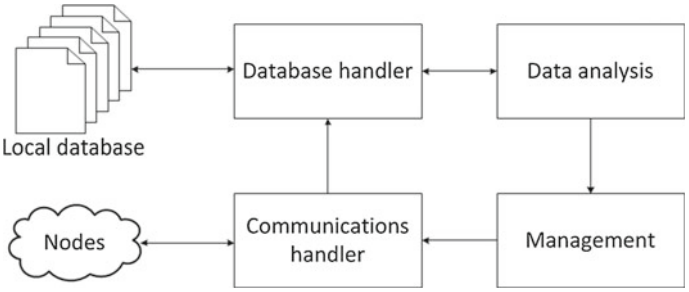


Fig. 23.3 Component interaction in the FABD module

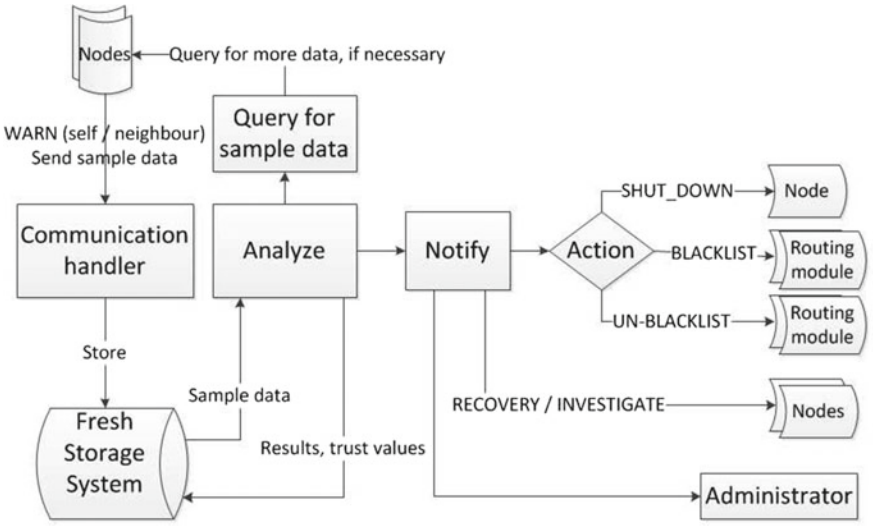


Fig. 23.4 The failure detection flowchart of actions

A basic flowchart of actions for the FABD module can be observed in Fig. 23.4. Following the data analysis step, trust values can be generated for each node, in order to appreciate their activity. The nodes start with a maximum value, which decreases when faults occur. When a node's trust value falls below a threshold, an administrator will be notified to take action.

If the node is detected to be failing, it will be notified, hoping that its self-healing mechanism can identify the problem and fix it. During its recovery period, the node can be temporarily blacklisted so it will not affect other node's data. If the problem persists, actions such as shutting down the node and notifying an administrator should be taken.

Consequently, the Smart City needs the FABD module to prepare the data for the thorough analysis performed by the data processing unit described in Sect. 23.3.5.

However, before reaching this stage, the data will be stored by a specialized component, described in Sect. 23.3.4.

23.3.4 The Big Data Component

We will begin to describe the innovative Big Data component of a Smart City architecture by means of a bottom-up approach. We will only detail the logical component, the physical component being a SAN managed by the Lustre FS, for the newer data, and by libraries of tapes to archive the very old data.

Each FABD module owns a local small database, as seen in Sect. 23.3.3. This is where they store their fresh data, which is periodically flushed into the Big Data permanent storage system, to be later processed by the Big Data processing unit. This then stores back the results and also issues commands to the Smart City's actuators. This is how the whole system is able to help control the pollution in the city.

In order for the FABD servers to scale and be highly available, they should be replicated and connected via a Chord-like DHT network, in the fresh storage system. This way, if a FABD server fails, another one will take its place. This *high availability scheme* can be achieved by installing more independent FABD servers. For this, we also configure a Chord network, which includes all the FABD servers and sensor nodes within the Smart City. The keys in this system represent the ID of the data provided by each individual sensor at a given moment in time, as seen in Fig. 23.5.

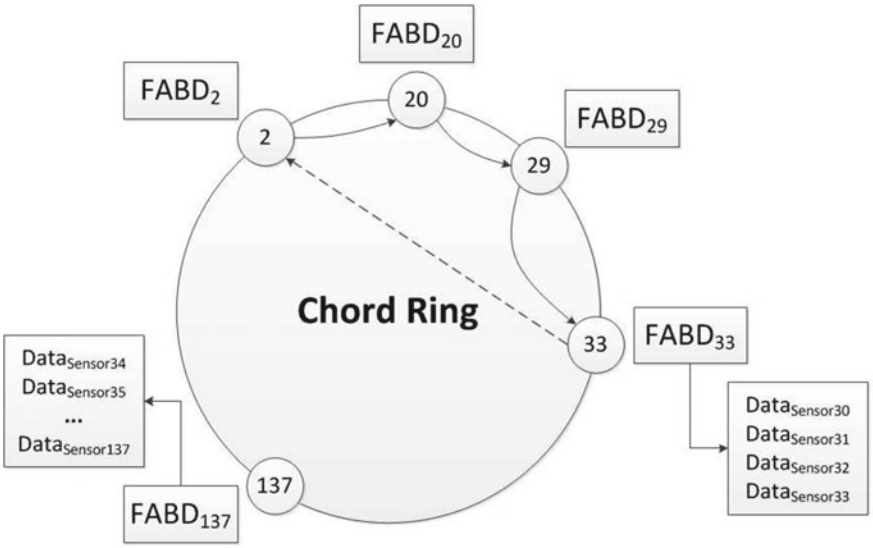


Fig. 23.5 The fresh storage system

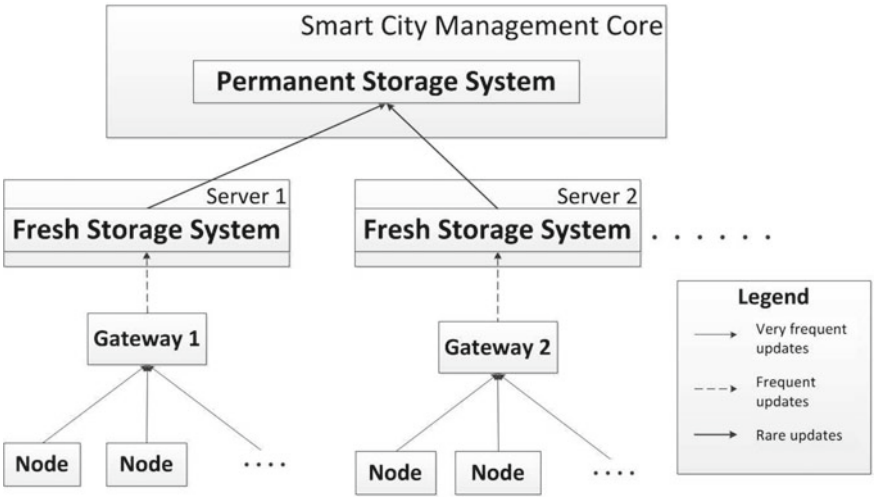


Fig. 23.6 The entire storage system

With an ensemble view, we see the storage system as being composed of two physically independent storage elements, as can be seen in Fig. 23.6:

1. The *fresh storage system*, which is closer to the nodes, resides on the FABD server Sect. 3.3. For the fresh storage system, the storage elements will be ordinary MySQL databases [11], but, in order to achieve a highly scalable storage system that is also resilient to individual failures, each of these databases will be connected in a Chord ring [15], as seen in Fig. 23.5. Thus, the temporary data will be organized by means of a DHT, in the same manner described in [1].
2. The *permanent storage system* is the storage system that holds historical data about the nodes, by using a NoSQL (Not Only SQL) database, for which we have chosen Blobseer [3], as seen in Fig. 23.7. This is an ever extending database, which periodically gets enriched with fresh snapshots about the situation of the pollution in the city, gathered from the sensor nodes via the *fresh storage system*.

More details about the organization of the Smart City's Big Data can be found in Sect. 23.4. This model organizes the data in such a manner that it can be further processed to reach the Smart City goals, in our case to complete the *pollution map*.

23.3.5 The Data Processing Unit

The data access model depends on the type of the processing to be performed on the sensed data, that is stored in the permanent storage system. The data processing unit extracts the *raw data* from the permanent storage system, it processes it via a more or

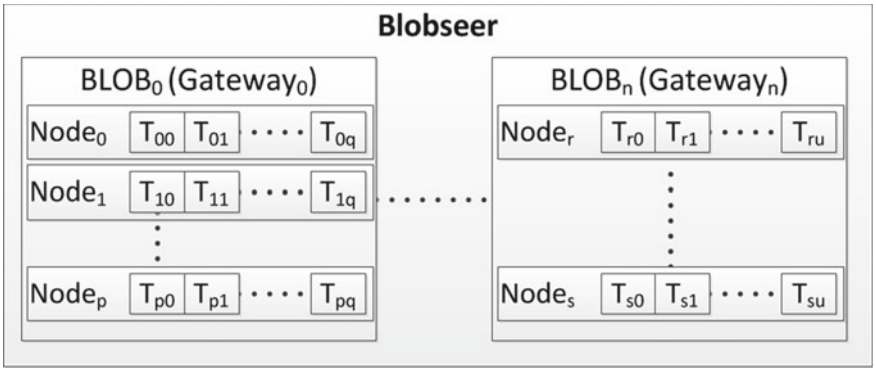


Fig. 23.7 The permanent storage system

less elaborate algorithm. It is not the scope of this chapter to show exactly by which algorithm the pollution information is inferred, but rather we aim at pointing out the interactions between a data processing unit, which is a replaceable component in the system, and the rest of the Smart City architecture.

In brief, for our Smart City fighting against pollution, we are interested in the *time and space evolution* of a function that represents the degree of pollution. In fact, we must have an algorithm to determine the degree of pollution within the given city.

The purpose of the data processing unit is to determine the pollution map of the city. In order to achieve this, it will measure relatively often individual degrees of pollution at different points in the city. After this, an interpolation method will be used to measure how the function of pollution `f(position, time)` varies, and we should also correlate this variation with the physical map.

The outcome of the performed computations should be translated into actions to be performed by the Smart City's actuators. For example, if on a given street we have a degree of pollution higher than a given threshold, `T1`, we should make that a *permanent one-way street*. If on a given street we have a degree of pollution higher than a second given threshold, `T2 > T1`, we should also perceive a *pollution tax* for the street. And, finally, if on a given street we have a degree of pollution higher than a third given threshold, `T3 > T2 > T1`, we should *close the street* for car circulation a number of hours a day.

Less drastic measures, involving *adaptive traffic lights* or *smart reversible lanes* could be envisaged. This can only be possible if the huge amount of data received from the sensors is logically organized, thus prepared for the processing unit.

23.4 Organizing the Smart City's Big Data

This is a special section, arguing about the way to organize the data, both physically and logically, in an *original scalable* and *highly available* manner, that is suitable for the Smart City. It will also describe how the existing Big Data systems presented in Sect. 23.2 are of help for a Smart City fighting against pollution.

The main challenges that we see for a Big Data sharing system would be

- To be fault tolerant and persistent.
- To be scalable, as to easily adapt to increasing volumes of data.
- To be easily and rapidly accessible by the data processing tools.

We will address all the identified requirements for a Big Data system in the remainder of this section.

We have already given a brief description of the Big Data component from the Smart City architecture, in Sect. 23.3.4. We have seen that we have a *fresh storage system*, which stores temporary data associated to individual FABD servers, and also a *permanent storage system*, which periodically receives new data from the sensors, and which keeps the historical data from them.

Based on Fig. 23.6, we can state that the Smart City's Big Data component is a two-dimensional storage system, both from the *physical* and from the *logical* organization point of view. The physical perspective has been described in Sect. 23.3.4. The logical perspective is given by: (1) the Chord Big Data storage solution, that is the *fresh storage system*, which represents the *spatial dimension* of the city; (2) the Blobseer Big Data storage solution, that is the *permanent storage system*, which represents the *temporal dimension* of the city.

In the fresh storage system, we have seen that actually each *FABD server* is responsible for a group of adjacent sensors. The adjacent nodes that a sensor network gateway is responsible for represent the set of all mobile or static sensors in its vicinity. The FABD modules store the raw data from their nodes into their own local databases for raw data, which are periodically flushed into the permanent storage system.
All the sensor network gateways involved into the Smart City are connected together in a Chord ring, which involves *logarithmic* routing time and thus *logarithmic time to data*, as seen in Fig. 23.5. Actually, there is a *caching* that is performed with respect to the data gathered from the sensors, and only after a certain amount of data has been read from the sensors, periodically, there will be a write in the permanent database, i.e., a flush. Of course, replication mechanisms are put in place for the temporary data, i.e., the data that has not already been written into the permanent database. If we do so, we will reduce very much the write bottleneck for the permanent database, by performing some kind of a hierarchical, 2-time, writing, performed at random moments in time, done by each individual gateway. This flushing technique is somewhat similar to collision detection and avoidance in Ethernet, in the following order: (1) not to lose any significant amount of sensor data; and (2) not to create a bottleneck for the operation of writing into the permanent database.

It should be mentioned that, in the fresh storage system, each gateway and its local database can be identified on the Chord ring by having a unique identifier, a hash of its IP address, for example. Furthermore, each individual sensor's identifier can be determined by prefixing its parent server's ID with the sensor's own ID, in order to obtain the routing key in the Chord network

```
sensor_ID = hash(parent_server_IP) + hash(own-IP)
```

As already mentioned earlier, in Sect. 23.3.1, please note that each static or mobile device's and FABD server's IP are considered to be their IPv6 addresses, which are unique identifiers.

In the permanent storage system, a Smart City BLOB defined in our solution in Fig. 23.7, by following the Blobseer spirit and terminology, will be the set of data associated to a specific gateway, which will keep growing over time, and which gets stored periodically by the fresh storage system into the permanent storage system. Actually, the BLOB (Binary Large OBject) will represent a geographical vicinity within the Smart City, managed by a certain, unique gateway, which will concatenate, over a short period of time, the data from all the sensors connected to it, thus creating *pollution snapshots* for that vicinity.

In case a sensor network gateway fails, its functionality will be taken by its successor in the Chord ring, and, as a consequence, the BLOB of its successor will become the reunion between the set which represented its initial BLOB and the set that represented the BLOB of its predecessor. For this to be (almost) instantaneously possible by means of the Chord resilience to failure mechanisms, each node should replicate its data gathered from the sensors onto its successor. This way, the data gathered from the sensors is never lost, thus we have obtained a reliable and persistent fresh data storage system.

Having the *fresh storage system* and the *permanent storage system* as described above in this section, we can safely say that we have covered the first two requirements that we have identified for a Big Data storage system for the Smart City, that is we have found architectural solutions for: data persistence, fault tolerance and for a scalable storage model. The only issue that remains to be accessed is to have a means of *fast access to the Big Data*. We have done this, by using an *in-RAM cache-like data access method*, which significantly speeds up data access, by prefetching and caching more interesting parts of the stored sensor data on the processing nodes. This has been done by customizing a tool called Spark [44], which has been designed to interact with Hadoop-like storage systems [21], such as our permanent storage system: Blobseer [3].

23.5 Smart City's Data Mining

Based on the data organization scheme, presented in Sect. 23.4, this section contains hints toward data mining techniques that could be of use to extract meaningful information from the data gathered from the sensors involved in the Smart City.

As discussed in Sect. 23.3.3, the data gathering process has to be intertwined with a Failure and Abnormal Behavior Detection process, in order to best determine the relevant data coming from trusted nodes. The results provided by the FABD component rely heavily on the metrics used to analyze the data. As such, the metrics in Table 23.1 should be regarded as a minimal set [45].

The first two represent values verifications, which are employed for each value of each sensor node. This is a simple test, that can be easily employed on the sensor nodes as well. However, in the context of Big Data it is important to be able to keep track of the faults that occur, in order to differentiate between one time faults and recurrent faults, that could also form a pattern.

The third table entry completes the values test, revealing more information about how the data has changed in a fixed period of time for a certain sensor node. Together with the first type of tests, patterns can already be formed to classify the data received, as seen in Table 23.2. Moreover, the so-formed patterns can lead to a more fine grained control of the IoT system, and even self-adaptation to context.

Entries number 4 and 5 in Table 23.1 represent average and median variance in the measured data. These types of analysis are also achieved for each node individually, on the set of data received within a time frame. Just as the first two types of analysis, these two have to be utilized together for better results. For instance, there

Table 23.1 The minimal set of metrics used in failure detection

Nr. crt.	Threshold	Description
1	MIN_READING	The minimum allowed value of the sensor readings
2	MAX_READING	The maximum allowed value of the sensor readings
3	MAX_GROWTH_RATE	Growth rate represents how fast the values read from the sensors increase or decrease over a fixed period of time. This threshold represents the maximum allowed growth rate
4	MAX_AVG_VARIANCE	The average variance between nodes in a group. This threshold represents the maximum allowed variance between the average data from the nodes
5	MAX_MEDIAN_VARIANCE	Median variance is similar to the average variance, but instead of computing the average, only the median value of a group of readings is taken into account. This threshold represents the maximum allowed variance between the median data from the nodes
6	MAX_UPTIME	The maximum uptime allowed for a node before maintenance is necessary
7	MAX_MSG_RECV_COUNT	The maximum number of messages received by the server from the node before maintenance is necessary
8	MAX_MSG_SENT_COUNT	The maximum number of messages sent by the server to the node before maintenance is necessary
9	MIN_BATTERY	The minimum estimated voltage of a node before an alert is generated

Table 23.2 Developing patterns from the values and growth rate tests

Values	Growth rate	Pattern
Detected	Detected	Out of range
Detected	Not detected	Bad values in a close range
Not detected	Detected	Acceptable spikes
Not detected	Not detected	No problem

are cases where the average test can result in false positives that the median filter does not acknowledge. For instance, consider the case where most nodes provide small temperature values, but a subset of nodes starts to send bigger values, without exceeding the growth rate or values thresholds. Depending on the number of nodes with this behavior and on the values themselves, there could be a case where the average threshold is triggered both for the failing node and for most of the healthy nodes. However, if there are more than half of the nodes healthy, then the median threshold will only be exceeded by the failing nodes.

Entries 6 through 9 in Table 23.1 present metrics used to monitor statistical data coming from the nodes. This data can be piggybacked on regular messages to the server in order to best utilize the network bandwidth.

The battery power metric is relevant only for wireless sensor networks. In a wired network, the is no need to estimate battery power. However, in the context of Big Data systems, it might be of interest to keep track of the power consumption of the system in order to be able to minimize it. The power consumption can be estimated as a function of the initial battery power, the number of messages exchanged and the cost of transmitting or receiving a message. This approach avoids querying the nodes for their battery state in order to save bandwidth; if piggybacking is used, this estimation can be corrected periodically with data from the nodes themselves.

This data gathering model performs preliminary tests on the data, or samples of data received from the sensor nodes, in order to identify faults in the network infrastructure. The results obtained can be further used to modify trust values for each node. Consequently, these trust values can help validate the data coming from the sensor nodes. The Big Data analysis on the information stored should only use data coming from trusted nodes in order to ensure a certain degree of trustworthiness for the results provided.

23.6 Smart Measures for a Smart City

Based on the Smart City's architecture presented in Sect. 23.3 and on the data organization from Sect. 23.4 and on the associated data mining techniques from Sect. 23.5, this section contains a detailed description of the effective smart measures that the Smart City could take. There are a lot of reasons to have a Smart City, capable of

fighting against pollution by itself. If the city could adapt itself to diminish the degree of pollution, this would in term make life healthier and more pleasant for its citizens. Fighting pollution could be done by means of: (1) individual smart traffic lights, which adapt for the time they stay red or for the time they stay green depending on the amount of traffic they detect; (2) correlated and interacting smart traffic lights, which could cooperate in order not to generate light traffic in some broad areas, but heavy traffic in other parts of the city, if possible; (3) reversible traffic lanes, which can switch directions for traffic depending on the difference between the amounts of traffic in the two different ways; and (4) producing consistent and exhaustive topological and historical data analysis, in order to predict a traffic model for each city street and provide recommendations for the Police Office about how to make one-way streets or even close traffic on certain portions of streets and provide better public transportation coverage in those areas, thus diminishing pollution.

23.6.1 Smart Measures from the FABD Model

The information analyzed by the data gathering component can reveal several faults of the data or hardware issues (i.e., related to connectivity, or crashes). From these results, several objective measures can be taken to alleviate the consequences, or to help maintain the network. Furthermore, such actions, if automated, can form the basis of a self-adaptive Internet of Things system.

Analyzing the data using the tests described in Sect. 23.5 reveals certain patterns of the data. Consequently, transitioning from one pattern to another can denote a significant change in the analyzed system. As such, different data patterns can be created from one or several consecutive analysis of the system. Furthermore, these states can be analyzed from a historical point of view to reveal changes in the patterns that would require attention. For example, in a system consider that we constantly detect an unacceptable growth rate, but the values received are still within their respective thresholds. This, as shown in Table 23.2, means that the system is constantly dealing with acceptable spikes—a fact which represents the pattern exhibited by the data. Now consider that at some point the system also starts receiving bad values. In this case, the change represents the transition to a state of constant out of range spikes. While the first state might just represent a warning that there will soon be a failure, the system still works. The transition means that the system gradually started to fail. However, if the system goes from a no problem state directly to constant bad values, this could mean that the malfunction has physical causes (i.e., natural disasters, fire, vandalism, etc.).

On a larger scale, these faults can also be analyzed groupwise to reveal which regions have a tendency to fail more often and provide possible reasons for these faults. Consequently, special measures can be taken to prevent the failures from happening.

The prevention of hardware faults should be a priority for Big Data systems. There are no studies at this time that we are aware of that investigate the impact of incorrect

data in the context of Big Data analysis. Moreover, such a study would depend heavily on the type of data it is analyzing and the impact of faults could vary from one application to another. Given this case, the data gathering component should also warn an administrator about possible future failures of the Smart City infrastructure. We plan on achieving this by analyzing the different statistical data of the system. Indicators such as the uptime of the nodes, the number of messages transmitted and received, and the estimated battery power can help with maintenance tasks. Moreover, this will lower the costs of maintenance because it enforces maintenance on demand, rather than periodical controls. Of course, the periodical controls will remain an essential duty, but they will become considerably less frequent, a factor that is inherently related to the costs of these actions.

23.6.2 Pollution Measures for the Smart City

One of the most important aspects of the sensor nodes is the fact that the sensors are pluggable. Thus, the users can select from various types of sensors the ones that are most relevant to their situation and use exactly the needed sensors to serve their purposes. A list of sensors that our Smart City should support includes sensors for the following pollutants:

- **Carbon monoxide** is generated by incomplete combustion of carbon. Even relatively small amounts of it can lead to hypoxic injury, neurological damage, and possibly death [46];
- **Ammonia** is one of the most widespread gases. Children with asthma may be particularly sensitive to ammonia fumes; also a significant part of respiratory allergies are related to this gas and prolonged exposure to ammonia may cause nasopharyngeal and tracheal burns, bronchial and alveolar edema, and airway destruction resulting in respiratory distress or failure [47, 48];
- **Hydrogen sulfide** is generated by bacteria as part of organic material decomposing. It can cause eye, throat, and lung irritation, asthma attacks, nausea, headache, nasal blockage, sleeping difficulties, weight loss, and chest pain [49].
- **Gasoline and diesel exhaust** are major pollutants of populated areas. Exposure to this mixture may result in asthma attacks, increase likelihood of cancer, chronic exacerbation of asthma and other health problems [50].
- **Natural gas, propane, methane** and other petroleum derivative gases. These gases are essentially fossil fuels that can cause irritations to the upper respiratory tract, and, in contact to a source of heat, can provoke fires and explosions.
- **Carbon dioxide** and general indoor pollutants are generated by a multitude of human activity. They indirectly increase the likelihood of asthma attacks and may cause a rise in asthma cases among children [51].

Many people will immediately benefit from our system: asthmatics, people who do jogging, etc. On the long term, government agencies that regulate and impose

pollution standards can benefit from the large amounts of data gathered by our system which can result in better statistics and understanding of the way pollutants affect the urban environment. It can also lead to better air quality management and to pinpoint major pollution sources inside of a city.

Due to its modular design, our system can be extended to offer additional functionality. For instance, multiple mobile sensor nodes could be installed on public transportation buses and trams, offering an up-to-date and detailed picture of urban pollution. More complex logic could be incorporated in the server application, allowing the automated identification of problem areas and possibly the prediction of air pollution patterns and expansion, based on meteorological data.

23.7 Conclusion

To our knowledge, our envisaged architecture for a Smart City represents an *original* approach. This is true, since it ingeniously combines the Big Data management techniques with the Internet of Things, i.e., for pollution sensor networks.

We aim at designing some prototype mechanisms that enable the *Smart City* to be *context-aware* and *self-organizing*. To attain this purpose, we make use of: (1) sensor networks, connected by wired or mobile wireless networks; and of (2) Big Data scalable and reliable management techniques.

The main advantages of our design, which also represent *original* components presented within this chapter, are

1. the FABD server, which is at the same time able to monitor and control the sensor node network and to reliably store only the trustworthy pieces of information regarding pollution that are gathered by the sensors;
2. the two-level Big Data storage system, which has both a *provisional spatial dimension* and a *permanent temporal dimension*, both of them building a scalable and robust data management model.

Our chapter includes a special, very important, section about the actual measures that the Smart City should take in order to efficiently reduce pollution, in Sect. 23.6.

Even if it has been beyond the scope of this chapter to detail them, our design has kept in mind that *artificial intelligence data mining techniques* together with *high performance computing techniques* should also be applied in order to obtain an accurate model describing the pollution in the Smart City.

As far as the sensor network is concerned, the sensors we have envisaged for our Smart City were derived from the Pollution Track project [52], thus being similar to them.

Future developments for the Smart City, besides *actually implementing* such a city, are

1. optimized *high performance computing* techniques to determine a more accurate model for the Smart City's intrinsic mechanisms for fighting against pollu-

tion and also designing models of interaction between weather forecast and the Smart City's mechanisms in order to obtain accurate pollution forecast models;
2. an improved *secured communication model* between the Smart City's components, so that the Smart City's functionality could not be altered by any outside or inside attacker.

Last but not least, a very important improvement that can be implemented for the data gathering and failure detection component is making the analysis group aware. Besides, studies have to be done on the impact of faulty data present in Big Data systems. These studies should reveal how small changes in the data set analyzed can influence decision-making within the larger system.

References

1. Almăşan, V.: Using peer-to-peer scalable techniques to increase service availability in SIP networks. PhD thesis, Universitatea Tehnică din Cluj-Napoca, Romania (2011)
2. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: A Distributed Storage System for Structured Data. In OSDI, Seattle, WA, USA (2006)
3. Nicolae, B., Antoniu, G., Bougé, L., Moise, D., Carpen-Amarie, A.: BlobSeer: next generation data management for large scale infrastructures. J. Parallel Distrib. Comput. **71**(2), 168–184 (2011)
4. Chowdhury, M., Zaharia, M., Ma, J., Jordan, M.I., Stoica, I.: Managing data transfers in computer clusters with orchestra. In: SIGCOMM (2011)
5. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M.J., Shenker, S., Stoica, I.: A Fault-tolerant Abstraction for In-memory Cluster Computing. In NSDI, San Jose, CA, USA (2012)
6. Corbett, J.C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J., Ghemawat, S., Gubarev, A., Heiser, C., Hochschild, P., Hsieh, W., Kanthak, S., Kogan, E., Li, H., Lloyd, A., Melnik, S., Mwaura, D., Nagle, D., Quinlan, S., Rao, R., Rolig, L., Saito, Y., Szymaniak, M., Taylor, C., Wang, R., Woodford, D.: Spanner: Google's Globally-Distributed Database. In OSDI, Hollywood, CA, USA (2012)
7. Schwan, P.: Lustre—building a filesystem for 1000-node cluster. In: Proceedings of Linux Symposium (2003)
8. Weiser, M.: Some Computer Science Problems in Ubiquitous Computing. Communications of the ACM (1993)
9. Tudose, D., Patrascu, T.A., Voinescu, A., Tataroiu, R., Tapus, N.: Mobile sensors in air pollution measurement. In: Proceedings of the 18th Workshop on Positioning, Navigation and Communication (WNPC'11), Dresden, Germany, April 2011
10. Tataroiu, R., Tudose, D.: Remote monitoring and control of wireless sensor networks. In: Proceedings of the 17th International Conference of Control Systems and Computer Science (CSCS17), vol. 1, pp. 187–192. Bucharest, Romania (May 2009)
11. The MySQL database. http://dev.mysql.com/
12. Davies, A., Fisk, H.: MySQL Clustering. MySQL Press (2006)
13. Sun Microsystems, I.: NFS: Network File System Protocol Specification. RFC 1094 (Standard) (1989)
14. Wilde, E.: Wilde's WWW. Springer (1998)
15. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In: Proceedings of the 2001 ACM SIGCOMM Conference, pp. 149–160 (2001)

16. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: SIGCOMM '01: Proceedings of the 2001 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, vol. 31, pp. 161–172. ACM Press, October 2001
17. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pp. 329–350, Nov 2001
18. Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.D.: Tapestry: A resilient global-scale overlay for service deployment. IEEE J. Sel. Areas Commun. **22**(1), 41–53 (2004)
19. SHA-1—Secure Hash Standard. http://www.itl.nist.gov/fipspubs/fip180-1.htm
20. Dabek, F., Brunskill, E., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I., Balakrishnan, H.: Building peer-to-peer systems with chord, a distributed lookup service. In: Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), Schloss Elmau, Germany, IEEE Computer Society, May 2001
21. Borthakur, D., Gray, J., Sarma, J.S., Muthukkaruppan, K., Spiegelberg, N., Kuang, H., Ranganathan, K., Molkov, D., Menon, A., Rash, S., Schmidt, R., Aiyer, A.: Apache hadoop goes realtime at facebook. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, SIGMOD '11, pp. 1071–1080. ACM, New York, NY, USA, (2011)
22. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), MSST '10, IEEE Computer Society, pp. 1–10. Washington, DC, USA (2010)
23. Ananthanarayanan, G., Ghodsi, A., Wang, A., Borthakur, D., Kandula, S., Shenker, S., Stoica, I.: PACMan: Coordinated Memory Caching for Parallel Jobs. In NSDI, San Jose, CA, USA (2012)
24. Johnson Space Center: Fault-Detection, Fault-Isolation and Recovery (FDIR) Techniques. NASA Engineering Network, Technique DFE-7 (1994)
25. Nithya, R., Kevin, C., Rahul, K., Lewis, G., Eddie, K., Deborah, E.: Sympathy for the Sensor Network Debugger. In: 3rd Embedded Networked Sensor Systems, pp. 255–267 (2005)
26. Linnyer Beatrys, R., Isabela, G.S, Leonardo, B.O., Hao, C.W., José Marcos S.N., Antonio A.F.L.: Fault Management in Event-Driven Wireless Sensor Networks. In: Proceedings of the 7th ACM international Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (2004). doi:10.1145/1023663.1023691
27. Jinran, C., Shubha, K., Arun, S.: Distributed fault detection of wireless sensor networks. In: Proceedings of the 2006 Workshop on Dependability Issues in Wireless ad Hoc Networks and Sensor Networks (2006). doi:10.1145/1160972.1160985
28. Benhamida, F.Z., Challal, Y., Koudil, M.: Efficient adaptive failure detection for query/response based wireless sensor networks. In: Wireless Days, IFIP (2011). doi:10.1109/WD.2011.6098190
29. Kebin, L., Qiang, M., Xibin, Z., Yunhao, L.: Self-diagnosis for large scale wireless sensor networks. In: IEEE INFOCOM (2011)
30. Qiang, M., Kebin, L., Xin, M., Yunhao, L.: Sherlock is around: detecting network failures with local evidence fusion. In: IEEE INFOCOM (2012)
31. Alan, M., David, C., Joseph, P., Robert, S., John, A.: Wireless sensor networks for habitat monitoring. In: Proceedings of the 1st ACM international Workshop on Wireless Sensor Networks and Applications, WSNA (2002). doi:10.1145/570738.570751
32. Jeongyeup, P., Chintalapudi, K., Govindan, R., Caffrey, J., Masri, D.: A wireless sensor network for structural health monitoring: performance and experience. In: Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors, pp. 1–9. EmNets (2005)
33. Clemens, L., Nagendra, B.B., Daniel, R., Gerhard T.: On-body activity recognition in a dynamic sensor network. In: Proceedings of the ICST 2nd international conference on Body area networks, BodyNets (2007)
34. Phillip, B.G., Brad, K., Yan, K., Suman, N., Srinivasan, S.: IrisNet: An architecture for a worldwide sensor web. IEEE Pervasive Comput. **2**(4), 22–33 (2003). doi:10.1109/MPRV.2003.1251166

35. Adam, D., Richard, G., Sergio, A.M., Arnold, P., Mats, U.: Janus: an architecture for flexible access to sensor networks. In: Proceedings of the 1st ACM workshop on Dynamic interconnection of networks, DIN, pp. 48-52 (2005). doi:10.1145/1080776.1080792
36. Mani, S., Mark, H., Jeff, B., Andrew, P., Sasank, R.: Wireless Urban Sensing Systems (2006)
37. Jung, Y.J., Lee, Y.K., Lee, D.G., Ryu, K.H., Nittel, S.: Air pollution monitoring system based on geosensor network. In: Geoscience and Remote Sensing Symposium, IGARSS (2008); IEEE International, vol. 3 (2009)
38. Kularatna, N., Sudantha, B.: An environmental air pollution monitoring system based on the IEEE 1451 standard for low cost requirements. IEEE Sens. J. **8**(4) (2008)
39. Tsow, F., Forzani, E., Rai, A., Wang, R., Tsui, R., Mastroianni, S., Knobbe, C., Gandolfi, A.J., Tao, N.: A wearable and wireless sensor system for real-time monitoring of toxic environmental volatile organic compounds. IEEE Sens. J. **9**(12) (2009)
40. Jeff, S., Peter, P., Jonathan, L., Mema, R., Margo, S., Matt, W.: Hourglass: An Infrastructure for Connecting Sensor Networks and Applications (2004)
41. Botts, M., Percivall, G., Reed, C., Davidson, J.: OGC Sensor Web Enablement: Overview and High Level Architecture, ed. pp. 175–190. Springer (2006)
42. Aman, K., Suman, N., Jie, L., Zhao, Feng: SenseWeb: an infrastructure for shared sensing. IEEE Multimedia **14**(4), 8–13 (2007). doi:10.1109/MMUL.2007.82
43. Shuo, G., Ziguo, Z., Tian, H.: FIND: faulty node detection for wireless sensor networks. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, pp. 253-266. Berkeley, California (2009). doi:10.1145/1644038.1644064
44. Spark – lightning-fast cluster computing. http://spark-project.org/
45. Silvia Stegaru: Failure and Abnormal Behaviour Detection in Wireless Sensor Networks. Master thesis (2013)
46. United States Environmental Protection Agency: Carbon Monoxide (CO). http://www.epa.gov/iaq/co.html
47. Agency for Toxic Substances and Disease Registry: Medical Management Guidelines for Ammonia. http://www.atsdr.cdc.gov/mmg/mmg.asp?id=7&tid=2
48. Healthy child, healthy world: Keep amonia out of your home. http://healthychild.org/easy-steps/keep-ammonia-out-of-your-home/
49. New York Department of Health: Hydrogen Sulfide Chemical Information Sheet. http://www.health.state.ny.us/nysdoh/environ/btsa/sulfide.htm
50. American Lung Association Energy Policy Development: Transportation Background Document. Prepared by M.J. Bradley & Associates LLC (2011)
51. WebMD Asthma Health Center: High Carbon Dioxide Levels May Up Asthma Rate. http://www.webmd.com/asthma/news/20040429/high-carbon-dioxide-levels-may-up-asthma-rate?lastselectedguid=%7b5FE
52. Pollution Track. http://pollutiontrack.com/

Index

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7