BABES-BOLYAI UNIVERSITY, CLUJ-NAPOCA, ROMANIA

Faculty of Economics and Business Administration

Department of Business Information Systems

Using Distributed Computing Resources for Efficient Execution of Complex Business Applications

Ph.D. Thesis Summary

Ph.D. Student:
Gabriela Andreea Morar

Ph.D. Supervisor:
Prof. Dr. Nicolae Tomai

Research Field: Cybernetics and Statistics

2012

Cuvinte cheie engleză

Key words: *distributed systems, cloud computing, performance, workflow, Hadoop, negotiation domain, resource negotiation*

Contents

Contents

Abstract

In the last decade the number of companies and scientists that rely on distributed computing systems in order to remain competitive has greatly increased. As the costs of operating a self-owned supercomputer are rather high, not all companies or research groups can afford acquiring the needed infrastructure for running their applications or experiments. Hence they have to turn to external computational resources. In the beginning clusters and grids were the ones that got the attention of industry and academia but they still had some limitations. The newly emerged cloud computing paradigm promises them all the needed resources at any time based on a pay-as-you-go model. But having the necessary infrastructure at affordable costs at their disposal is not enough for all of them, as some do not possess the necessary means or knowledge to take advantage of the newly emerged technologies.

In this work we show that distributed systems can be effectively used by industry and academia in order to increase the performance of their applications and reduce the execution costs. We present two main paradigms that can help users that do not own advanced parallel programming skills to parallelize their applications: workflows and MapReduce [8].

We also present two use cases consisting of building two workflows based on real-life applications and testing their performance when running on cloud resources. Since the workflows do not fit the needs of all users that want to take advantage of distributed resources, we also conducted a study regarding the efficiency of the MapReduce paradigm. In this case we selected the Hadoop [13] implementation

of the paradigm and in order to prove its efficiency in processing large amounts of data we tested it by running a clustering algorithm on data retrieved from Twitter. We wanted to test how the performance offered by Hadoop is influenced by the type of storage used (solid state disk or hard disk) and by the values of the parameters that can be set by the user when they configure their clusters.

After assessing the performance of applications that run on distributed resources (in our case cloud resources) we have identified the need for resource selection from a variety of possible existing resources without exceeding certain costs. This was done in a best-fit manner regarding a user's requirements. Thus the idea of using intelligent agents for concurrent resource negotiation arose. For this reason we have implemented a specialized negotiation domain based on IaaS cloud resources characteristics. We have considered the case of an academical federated cloud consisting only of private clouds.

We created a mechanism that – based on the data offered by the cloud controller of these clouds – can semi-automatically generate negotiation profiles. We have developed a negotiation mechanism consisting of an agent representing the resource buyer and several agents representing the cloud providers. The semi-automatic creation of negotiation domains can be considered a first step in our intention of achieving on-time cloud resource provisioning for cloud based applications. Integrating the developed mechanism in real platforms is part of future work.

Key words: *distributed systems, cloud computing, performance, workflow, Hadoop, negotiation domain, resource negotiation*

The current document contains a summary of the PhD thesis with the title "Using Distributed Computing Resources for Efficient Execution of Complex Business Applications". The summary is meant to give an overview of the thesis and presents the main achievements obtained in the course of my PhD studies.

1 Research Motivation

Nowadays staying competitive on the market is one of the main problems that companies have to deal with. Sometimes this might require reducing time and costs of processing large amounts of data. Scientists also deal with similar problems. They need to optimize execution times and reduce costs for their experiments or simulations because they constantly have to deal with deadlines and limited budgets.

During the past decades progress in computing technology has contributed greatly to accelerating scientific progress by reducing execution times of scientific and business applications [15]. Several research domains require great computational power for running experiments and validating research hypotheses. Hence scientists need to be able to easily reuse software and to vary input parameters or starting hypotheses for their experiments.

Whenever scientific experiments are conducted we can identify two main problems that might be the cause of failure:

- not having the right scientific ideas or not finding the right simulations to validate these ideas.
- and not being able to implement the scientific idea such that it will facilitate the execution of the experiments.

Not all scientists possess the necessary skills for developing complex applications that need to bring together tools belonging to different areas: data analysis tools, domain-specific tools, etc. This is even a more difficult task when we have to take limited resources like time and financial costs into consideration.

Several technical solutions that come as a helping hand for scientists that need some aid in maximally exploiting the computational resources they possess have arisen in the past few years. First we mention the workflow paradigm and the multiple Workflow Management Systems that were developed in order to facilitate the mapping of the execution of complex scientific applications on distributed resources. This solution better fits applications that need high computational power in order to complete within reasonable time limits. Second we discuss the MapReduce technique as a means for improving execution times in the case of applications that need to process large amounts of data. Of all available MapReduce implementations we focus on the one offered by the Hadoop project.

Even if distributed systems are an obvious solution when it comes to improving execution times of applications, they are not easy to manage and implementing applications that can run on them is a rather challenging task. From the complexity and heterogeneity of these systems a series of problems might arise. Failures can occur in the systems at any time, especially if they are formed by large numbers of heterogeneous resources bound together. Possible solutions for this problem could be: data replication in the case of applications that process large amounts of data or jobs resubmission.

When it comes to distributed computing systems there were three main models that got scientists' and industry's attention: clusters, grids and cloud computing.

Grid computing is a form of distributed computing which is composed of loosely coupled computers and appears to the user as a super computer. A grid consists of heterogeneous machines that offer their computational resources to the public in the form of virtual organizations (VOs)[1][12].

Recently an increased interest in Service Oriented Architectures (SOAs) and virtualization has been manifested by the computer science community. Based on these technologies a new type of distributed system was born: cloud computing.

Cloud computing has emerged as an alternative to previous types of distributed systems, and tries to solve cost problems by using a pay-as-you-go pricing model and by providing users with elastic resources that can be accessed from anywhere without requiring previous booking. In the past decade a considerable

[1]VOs are defined as "A set of individuals, institutions is intended to share resources by following sharing rules"

number of organizations have chosen to expose their resources, both hardware and software, using the everything-as-a-service paradigm.

The costs generated by creating in-house supercomputers are considerably high and accessing already existing infrastructures, like grids, requires lots of time, specialized knowledge, access permission from third parties, in-advance scheduling of resources, etc. Due to their flexibility clouds receive a growing interest from the industry and academia, whereas grids are mostly used by the academia as large scale computing and storage environments.

As the number of cloud providers on the market continues to increase daily, the need for automatic cloud resource management systems emerges. Their purpose is to enforce regulation of the supply and demand of cloud resources [31]. Cloud computing service delivery requires specific Quality of Service (QoS) to be maintained by the providers in order to meet the users' objectives and sustain their operations. In this context, an SLA-oriented resource management system established through a process of negotiation for the interaction between cloud participants is needed.

The conducted research tries to tackle the problem of resource provisioning based on certain QoS requirements in federated cloud environments. We want to observe how we can solve the resource negotiation problem on a competitive market with the use of intelligent agents. The agents will represent the cloud buyers and the cloud providers. Basically, we are interested in how can we get a maximum utility for the buyer when it deals with the problem of acquiring computational resources from various cloud providers.

2 Research Objectives

In this work we try to study the way in which the use of computational resources, provided by distributed systems, can increase the performance of various application types and reduce their execution costs. Keeping this goal in mind we have studied the current types of existing distributed systems, compared their suitability in running complex applications and the cost derived from their usage. Starting from this initial scenario several questions arose:

1. What is the best type of distributed computing resource for an application to run on?

2. What is the best way for a regular user to harvest the performance of such resources?

3. How do the technical characteristics (number of cores, RAM, hard disk, etc.) of these computing resources influence the performance of the application?

4. How to choose the best option for running an application, in the context of cost efficiency, when several resource providers are available?

These are the main questions that the current work tries to give an answer to. First by investigating the characteristic of parallel programs, then identifying the main means for parallelizing an application without needing strong parallel programming skills. Furthermore we will evaluate the way in which applications' performance can be improved and their execution costs reduced by using distributed computational resources. Finally we will try to suggest a mechanism that could help users acquire the needed resources when they are dealing with several resource providers.

3 Thesis Structure

Structura tezei reiese din figura 1:

Part I - Theoretical Notions Regarding Parallel Programming and Distributed Systems - consists of Chapter 2 and Chapter 3 and has the role of providing the reader with an overview regarding parallel computing and distributed systems and why they are needed for improving the performance of todays applications.

Chapter 2: Parallel Computing Overview discuses how parallel computing appeared, gives motivations of why parallel systems and parallel programming are needed, presents the different levels of parallelism and enumerates the main types of parallel hardware that were created based on the Von Neumann architecture. This chapter also gives an insight on how the performance of a parallel program can be assessed (speedup, efficiency, Amdahls law, etc.).

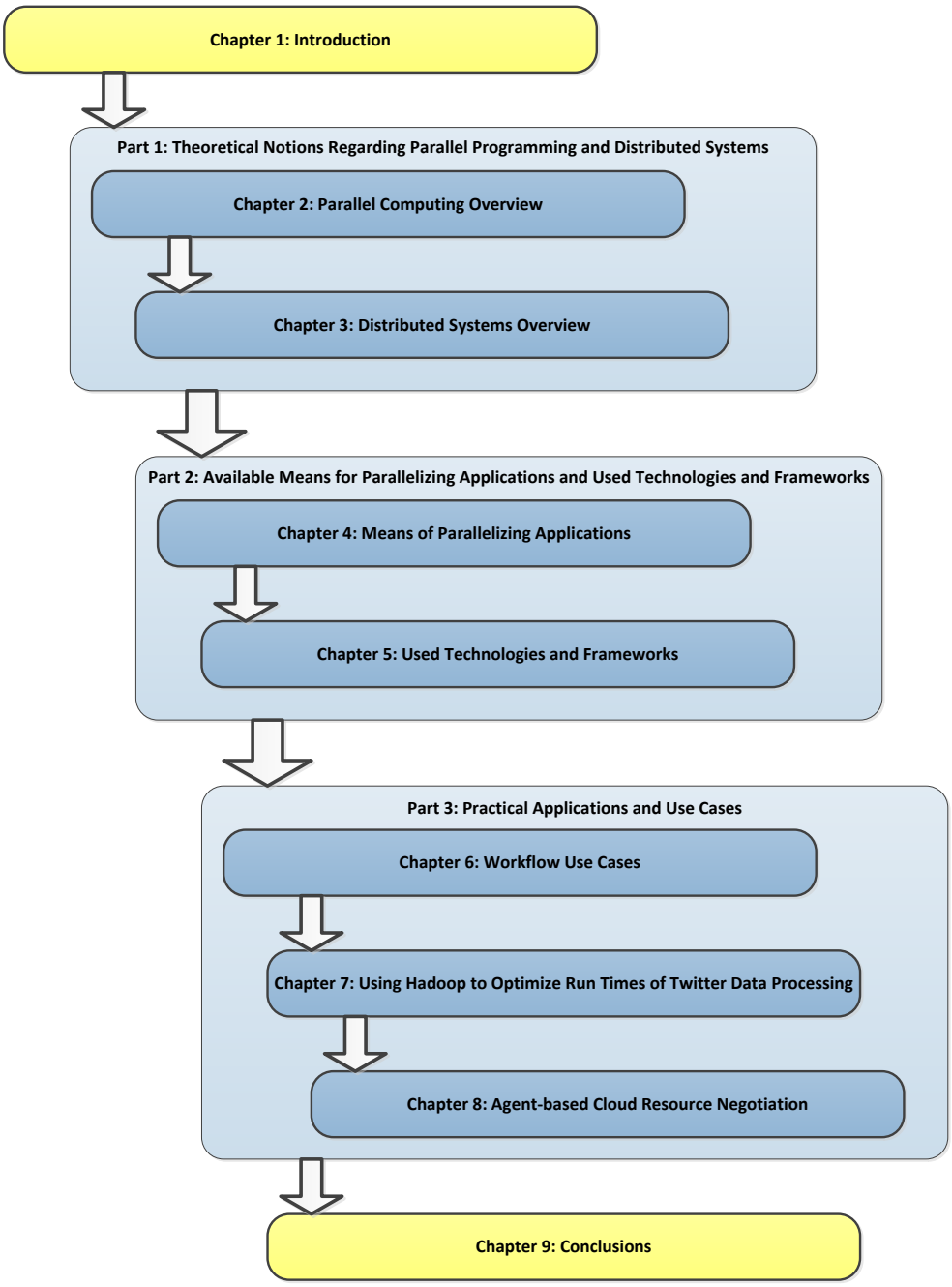

Figure 1: Thesis structure flow.

In [26] a discussion on why parallel systems and parallel programing are needed takes place. The author argues that the need for parallel computing arose with the need for ever-increasing performance. The development of research in areas such as: climate modeling, protein folding, drug discovery, energy research, etc., and the need for fast web searches and more realistic image rendering and computer games lead to the increase of performance appetite. This could only be achieved trough parallel systems and by using parallel programs.

With the time the evolution of single processor performance mainly consisted of increasing the number of transistors on integrated circuits. In order to accommodate more performance in the same component the size of the transistors was gradually decreased as their speed increased. But as the performance of the transistors increased so did their power consumption. This resulted in an increase of heat generated by the component. According to [14] during the first decade of the twenty-first century, air-cooled circuits are close to their limit of their ability to dissipate heat.

Another issue that lead to the need for parallel systems is the fact that there is a minimum voltage required to drive the microprocessor at the desired frequency, which is presented in [4]. This minimum voltage is approximately proportional to the frequency. This leads to the well-known cube-root rule that the speed s is roughly proportional to the cube-root of the power P, or equivalently, $P(s) = s^3$. This means that increasing the speed by a specific amount causes an increase in power consumption by the same amount to the power of three. In order to be able to discuss parallel computer architectures we first have to identify different levels at which parallelism may occur. Based on the data presented in [23], [14] and [26] we have created a short list of the main parallelization levels:

Parallel hardware and software has evolved from conventional serial hardware and software. In 1966 Michael Flynn [**?**] proposed a classification of computer architectures based on the number of concurrent instruction and data streams: SISD (Single Instruction, Single Data), SIMD (Single Instruction, Multiple Data), MISD (Multiple Instruction, Single Data), and MIMD (Multiple Instructions, Multiple Data). When it comes to evaluating the performance achieved by parallelizing a certain program or the execution of an application there are several means that can be of great help. Usually programmers use a combination of them

in order to better assess the results achieved and to help them decide if future changes are still needed. Some of these means of appreciating the effectiveness of the parallelizations are presented in [26]. When it comes to parallelizing applications in order to achieve better execution times the best result that one can hope for is to equally divide the work among the available cores without introducing extra work for them.

If we succeed in doing this, and we run our program with p cores, one thread or process executing on each core at a time, then our parallel program should run, in theory, p times faster than the serial program. If we consider T_{serial} the time needed for the serial execution of the application and $T_{parallel}$ the parallel execution time, then the best we can hope for is $T_{parallel} = T_{serial}/p$. If this happens we can appreciate that our parallel program has a linear speedup.

$$S = \frac{T_{serial}}{T_{parallel}}, \tag{1}$$

Back in the 1960s, Amdahl [3] made an observation thats become known as Amdahl law. It says, roughly, that unless virtually all of a serial program is parallelized, the possible speedup is going to be limited — regardless of the number of cores available.

We can say about a technology that it is *scalable* if it can handle increasing problem sizes. In the case of parallel programs scalability refers to maintaining the same level of efficiency by varying the number of processes/threads and the problem size.

There are several indicators that help us evaluate the performance of the parallelization of a program. The ones presented here represent only a small part of them.

Chapter 3: Distributed Systems Overview presents an overview regarding different types of distributed systems (clusters, grids, clouds), presents their advantages and disadvantages, and focuses more on cloud computing. Its business models, advantages and disadvantages are presented as well as a comparison between the main types of distributed systems presented.

In their book about distributed systems [33] Tanenbaum and Steen define such a system as follows:

A distributed system is a collection of independent computers that appear to the users of the system as a single computer

Such a system contains a number of independent computers that cooperate with one another over a communications network in order to achieve a specific objective. One of the subgroups of distributed computing systems are clusters. [6] and [27] define clusters as follows:

A Cluster is a type of parallel and distributed system, which consists of a collection of inter-connected stand-alone computers working together as a single integrated computing resource.

Another subgroup of distributed computing systems are grids. [6] and [27] define them as follows:

A Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed 'autonomous' resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements.

One vision of 21st century computing is that users will access Internet services from anywhere based on a pay-as-you-go pricing model. [7] envisions the cloud computing power as the 5th utility (after water, electricity, gas, and telephony). This computing utility is supposed to provide the basic level of computing service that is considered essential to meet the everyday needs of the general community. A number of computing paradigms were proposed for delivering such a utility. The newest is cloud computing. This is only a reduced set of the characteristics that were given to cloud computing so far:

- pay-per-use pricing model
- elastic capacity and the illusion of infinite resources
- resources are abstracted or virtualized
- on-demand self-service/elasticity

The majority of work done in this area also contains data about cloud computing business models, among them the works of [20], [35], [1], , [23], [6]:

- **Infrastructure-as-a-Service** – infrastructure providers (IPs) manage a large set of computing resources, such as storing and processing capacity. Through virtualization, they are able to split, assign and dynamically resize these resources to build ad-hoc systems upon customer demand.
- **Platform-as-a-Service** – cloud systems can offer an additional abstraction level: instead of supplying a virtualized infrastructure, they can provide the software platform on which systems run.
- **Software-as-a-Service** – there are services of potential interest to a wide variety of users hosted in cloud systems. This is an alternative to locally run applications.

Part II - Available Means for Parallelizing Applications and Used Technologies and Frameworks - consists of Chapter 4 and Chapter 5 and elaborates on the main means available on the market for application parallelization (workflows and MapReduce paradigms). This part gives an overview of the main frameworks and technologies used in the practical applications section: the Askalon framework for developing scientific workflows that can be run on grids or clouds; Hadoop, a Java implementation of the MapReduce paradigm that allows processing of large data in a parallel manner; and Eucalyptus, a private cloud platform that enables users to create their in-house cloud infrastructures.

Capitolul 4: Available Means for Parallelizing Applications describes the main technologies (workflows, MapReduce) present on the market that allow researchers that have limited parallel programming skills to parallelize their application in order to limit their running costs and at the same time increase their performance.

According to [17] the number of multiprocessor research papers has constantly increased since 2001 and surpassed its peak point in the last years. [32] argues that one of the next challenges for this research area is concurrency. Nowadays software is influenced by the industry's requirements to create larger systems that deal with greater problems and that should exploit the ever-growing capabilities of computing and storage resources.

Another challenge that current and future software will need to deal with is data, the volume of which increases faster than the processing power of compu-

tational resources. [16] estimate that planned and future experiments are going to generate orders of magnitude more data than has been collected in the entire human history. Processing this data will require the usage of more computing and communication power than was possible until a few years ago.

One of the existing ways for parallelizing the execution of applications over distributed heterogeneous resources are workflows. This section will deal with defining the workflow concept and analyzing how this paradigm can influence the execution of scientific/business applications.

> *"A scientific workflow is the process of combining data and processes into a configurable, structured set of steps that implement semi-automated computational solutions of a scientific problem."* [2]

The Workflow Management Coalition define the workflow as:

> *"The automation of a business process, in whole or part, where documents, information or tasks are passed from one participant to another to be processed, according to a set of procedural rules."* [18]

When scientific workflows first appeared in the late 1990s, they were mainly used for visualization purposes. Since then there has been significant development in technology [34]. Some of them that are of higher interest for WFMS are: component-oriented frameworks, data and computational grids, service-oriented architecture and web services, semantic data models and tools to increase domain-specificity, virtual organizations, peer-to-peer networks, virtualization and cloud computing. They all marked the evolution and improvement of WFMS in one way or another. Thus, scientific workflows have evolved to satisfy different scientific requirements, computational technologies and scientific approaches, leading to a complete transformation of the scientific method. With time they evolved from state-of-the-art to commodity, thus having a great impact on scientific studies.

Another mean used for parallelizing applications was the MapReduce paradigm [8], which is designed to simplify the concepts around large scale distributed computing and allows dealing with large datasets.

It is divided into two steps: map and reduce. The map function takes a single instance of the type key/value pair as an input. The output of the function

are key/value pairs that are grouped by key and are used as an input for the reduce function. Based on the key value and the list of values outputted by the map function, the reduce function performs some computations over that list and outputs key/value pairs.

Chapter 5: Used Technologies and Frameworks – gives details on the frameworks used in order to test the efficiency of application parallelization. The Askalon framework is described since it is one of the main workflow development platforms that allow users to harvest cloud resources. For the parallelizing the processing of large data the Hadoop framework was chosen. The main computing infrastructure used for running the experiments was a private cloud based on the Eucalyptus platform.

Askalon [10] is a grid application development and computing environment whose final goal is to provide an invisible grid to the application developers. It was extended in order to be able to run on cloud resources too. In Askalon the user composes grid workflow applications at a high-level of abstraction using an XML-based language (AGWL) that shields the application developer from the grid/cloud. The AGWL representation of a workflow is then given to the middleware services (run-time system) for scheduling and reliable execution.

In 2004 Doug Cutting [25], a well known open-source software developer, decided to create his own implementation of the MapReduce algorithm previously developed by Google. He named the new software after the name of a stuffed elephant his child had.

Hadoop has two main components: HDFS (Hadoop's distributed file system) and MapReduce. HDFS stores files across a collection of servers in a cluster. Files are decomposed into blocks, and each block is written to more than one of the servers (the implicit value is 3, but can be customized by users). This replication provides both fault-tolerance (loss of a single disk or server does not destroy a file) and performance (any given block can be read from one of several servers, improving system throughput).

[24] presents Eucalyptus, an open-source cloud-computing framework that uses computational and storage infrastructure commonly available to academic research groups. Eucalyptus has several components that interact with one another through well-defined interfaces.

The architecture of the Eucalyptus system is simple, flexible and modular with a hierarchical design reflecting common resource environments found in many academic settings. In essence, the system allows users to start, control, access, and terminate entire virtual machines using an emulation of Amazon EC2's SOAP and "Query" interfaces. That is, users of Eucalyptus interact with the system using the exact same tools and interfaces that they use to interact with Amazon EC2.

Each high-level system component is implemented as a stand-alone web service. This has the following benefits: first, each web service exposes a well defined language-agnostic API in the form of a WSDL document containing both operations that the service can perform and input/output data structures. Second, we can leverage existing web service features such as web service security policies for secure communication between components.

Part III - Practical Applications and Use Cases - consists of Chapter 6, Chapter 7, and Chapter 8. This part holds all the study cases conducted in order to assess the efficiency provided by various distributed systems and parallel programming for applications' execution time and costs.

Chapter 6: Workflows Use Cases describes two different workflows: the eBay workflow [1] that is based on an eBay data crawler and the RainCloud [2] workflow that is based on a real life meteorological application. While the first one is I/O intensive the second one is computationally intensive. The two workflows were implemented using the Askalon framework and were run on cloud resources in order to test their performance and estimate their execution costs. The workflow used for this case study was implemented in the Askalon workflow management system. In order to be able to create a workflow for an application several steps need to be followed. These steps are presented as a suggestion in [9]:

[1] part of the work presented in this chapter was published in **Gabriela Andreea Morar**, Cristina Ioana Muntean, Gheorghe Cosmin Silaghi, "Implementing and Running a Workflow Application on Cloud Resources", Economy Informatics, vol. 15, no. 3/2011, pages 15-27

[2] part of the work presented in this chapter was published in **Gabriela Morar**, Felix Schueller, Simon Ostermann, Radu Prodan, and Georg Mayr, "Meteorological Simulations in the Cloud with the ASKALON Environment", CoreGRID/ERCIM Workshop on Grids, Clouds and P2P Computing, Rhodes Island, August, 2012, accepted

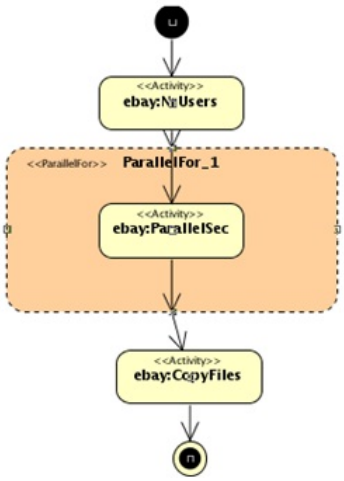

Figure 2: eBay workflow implemented in Askalon.

- Identify sections of the application that have the least connections with each other. These sections will become independent activities in the workflow.
- Establish sections of the code that could be run in parallel. These sections will be included in a parallel section of the workflow.
- If some previously defined activities have too many dependencies between each other, they should be grouped together in a single activity.
- Input and output data needed for each activity should to be identified.
- Correlations between activities must be established.

Figure 2 presents the workflow designed for the eBay data retrieval application.

Nrusers is the activity in which the number of sellers is being computed based on the input file that contains the list with all the sellers. The number of sellers is not fixed because the content of the input file containing the sellers' user names can vary. This is one of the steps that ensure the scalability of the application. In this same activity the large file containing the users' names is divided into several different files that will serve as input for each activity in the parallel section.

ParallelSec is the section of the application that was identified as being parallel. In this activity, data about the sellers is being retrieved based on the

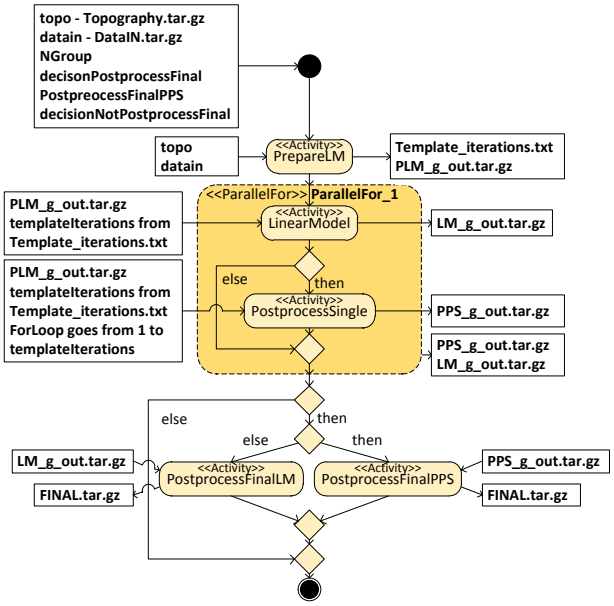

Figure 3: Graphical workflow representation in Askalon.

names contained by the files previously created in the Nrusers activity.

The last activity, **CopyFiles**, has the main purpose of gathering all the data contained in the two collections produced by the parallel for section of the workflow.

We run experiments for the eBay workflow on different types of cloud instances and using input files with 6000 and 12000 users. The results show that the application can successfully run on cloud resources with a speed up of almost 40, speedup computed based on the time archived by running the application in a sequential manner.

The application for which the RainCloud workflow is meant to investigate and simulate precipitation in mountainous regions with a simple meteorological numerical model called linear model of orographic precipitation (LM) [5]. Applications of this model range from climatological studies to hydrological aspects. As LM is a very simple and basic model, it can be run easily in a large number of parameter studies. Figure 3 depicts the graphical representation of the workflow and the input and output files needed by each activity. Based on the desired output and the purpose of running the workflow more scenarios or execution paths were created. We designed the workflow to be run in three "flavors":

- *ideal flavor* belongs to the operational area in which the whole setup uses idealized topography and atmospheric conditions. This workflow is mostly used for model testing or investigation of meteorological phenomena.

- *semi-ideal flavor* belongs to the research area and is designed such that either the topography is idealized or the atmospheric input is simplified, however, at least one part has real world application. For example, the topography is based on real topography data, but the atmospheric conditions are "manually" given. This flavor is used for e.g. interpretation of meteorological measurements.

- *real flavor* belongs to the research area such that both topography and atmospheric conditions are given by a real-world observations or full numerical models. This flavor is used for forecasting/downscaling precipitation.

In general, the execution of one workflow with the experimental input data would cost approximately 2.72$ if executed on Amazon EC2 using 4 `c1.xlarge` instances (0.68$/hour). This result applies to all presented workflow instances, as their execution time is lower than the one-hour payment granularity of EC2. SIMON: the next sentence does not make sense

For a yearly cost of 992.8$ this workflow can be run once every day, which is only a fraction of the amount the purchase of a comparable, dedicated system would cost.

Scientific workflows are now being used on a large scale in order to help scientist run their applications and take advantage of the computational resources that they have at their disposal. At the beginning, workflows appeared as a means for scientists to better identify the steps their applications were made of and to help them scale the applications by identifying the component activities and the correlations existent between them. Nowadays, many workflow management systems exist, some of the most used being mentioned in the introduction section, but the majority of them being employed to run workflows on grid and cluster resources. Only in the near past they were modified in such a manner that they could take full advantages of the newly emerged kind of computational resources, namely cloud resources.

Chapter 7: Using Hadoop to Optimize Run Times of Twitter Data Processing - describes how Hadoop can be used in order to achieve better execution times for applications that need to process large quantities of data. For the application part the Mahout [22] library was used, since it offers a large number of machine learning algorithms and is implemented in such a way that it runs on a Hadoop based infrastructure. We conducted a series of experiments i order to assess how the type of the storage used by the clusters influences Hadoop's performance. We also tried to vary different configurations parameters based on which Hadoop's allows users to tune their clusters according to their needs. Based on these setups we assessed the performance achieved in running the k-means clustering algorithm on users data retrieved from Twitter[1].

Hadoop has over 165+ parameters that allow the user to tune the jobs, maps and reduces and the HDFS. Finding the right configuration that will enable your applications to perform the best is not a trivial task. When configuring a certain Hadoop installation there are several key factors that need to be taken into account.It is the user's duty to tune Hadoop such that it offers the best performance for a certain type of application. Several tips regarding Hadoop performance tuning can be found in [28],[37], and [19].

In order to test the performance of Mahout algorithms we created 2 clusters similar in size but having different storage characteristics. The clusters were created using Oracle VirtualBox 4.1.18 [36] virtual machines having the same characteristics. Both clusters on which Hadoop runs have 1 master node and 3 slave nodes as presented in Figure 4. The main difference between them is the storage type: the first cluster (ClusterHD) runs on a hard disk while the second one (ClusterSSD) runs on an SSD. After setting up the Hadoop clusters it is recommended to test their performance in order to see if they were set up properly. Hadoop comes with a library $hadoop-*-test.jar$ that includes some benchmarks meant to help developers test the performance of their clusters. A short introduction about using these benchmarks is presented in [37]. It is recommended to test the clusters with an input dataset similar to the data that

[1]part of the work presented in this chapter was published in Cristina Ioana Muntean, **Gabriela Andreea Morar**, and Darie Moldovan, "'Exploring the meaning behind Twitter hashtags through clustering"', in Lecture Notes in Business Information Systems, vol. 127, pages 231 - 242. Springer-Verlag Berlin, 2012

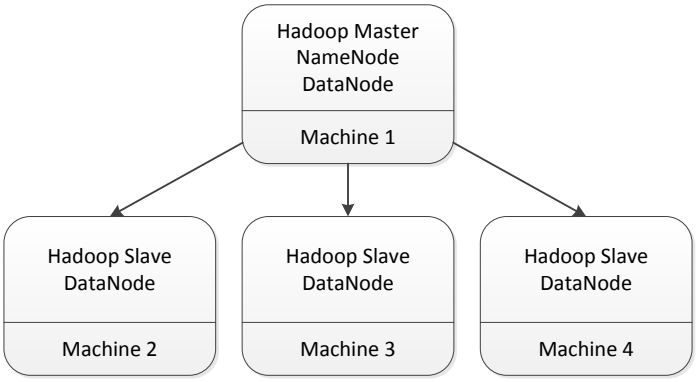

Figure 4: Hadoop cluster architecture.

user will later use.

For our experiments we used a dataset collected through the Twitter Streaming API for a period of one week, starting 10.12.2011 until 16.12.2011. The resulting dataset represents a random sample of 10% of the entire daily activity.

K-means [21] is a rather simple but well known unsupervised learning algorithm for clustering. Given a dataset, the algorithm partitions data into a number of clusters. This number of clusters, k, is fixed a priori.

Our experiments show that Hadoop's performance can vary significantly based on the installation being done on an SSD or a hard disk. Also we can say that execution times can be greatly decreased just by adding extra nodes to the Hadoop cluster. The achieved performance is also influenced by the time of the input size and the network type that connects the clusters nodes.

Due to the fact that Hadoop was created in such way that it can also take advantage of cloud resources we can say that if scientists need to occasionally process large data in a limited amount of time and with low costs, then this could be a good solution to their problems.

Chapter 8: Agent-based Cloud Resource Negotiation presents an example of how intelligent agents can be used in order to negotiate cloud resources that better fit users requirements. In this chapter we described a negotiation domain (CloudDomain)[1] that we implemented using the Genius platform and we

[1]part of the work presented in this chapter was published in **Gabriela Andreea Morar**, and Andreea Ilea, Alexandru Butoi, Gheorghe Cosmin Silaghi, "Agent-based Cloud Resources Negotiation", in Proceeding of 8th International Conference on Intelligent Computer Communi-

test its performance in the context of multi-party negotiations. The negotiation profile is generated in a semi-automatic way based on the data offered by private clouds providers through their Cloud Controllers. We studied the performance of the implemented negotiation domain in the context of cloud resource negotiations where a buyer need to select one cloud provider from a series of 10 available options. Intelligent agents using Bayesian and Q-learning learning mechanisms were used. The negotiations protocols used were the ones described in [29] and [30].

We mention that we have implemented a semi-automatic system for the generation of negotiation profiles. This generates the cloud providers' preferences profiles based on the data given directly by them. The only part of the profiles that can't be generated in automated manner is the importance cloud provider give to each of the negotiated issues. This problem could be solved by implementing a machine learning mechanism that could compute these values based on the preferences expressed by cloud providers in previous negotiation rounds.

4 Contributions

This section will summarize the contributions that were presented in each chapter of the thesis:

Chapter 2: Parallel Computing Overview - presents a wide theoretical study of parallel computing: how it appeared, what were the causes that led to its appearance and gives examples on how the performance of parallel programs can be correctly measured.

Chapter 3: Distributed Systems Overview - contains a large theoretical study regarding the existing types of distributed systems and brings together several comparisons that were made between grid, clusters and cloud computing in the domain literature. It also provides a detailed study regarding cloud computing: the way it appeared, its business models, its advantages and disadvantages, etc.

Chapter 4: Available Means for Paralleling Applications - contains an overview regarding the main means existing on the market that allow scien-

cations and Processing, pag. 297-300 30 August - 1 Septembrie, 2012 in Cluj-Napoca, Romania

tists or businessmen to parallelize their applications in order to achieve better execution times and reduce costs. We focused on the workflow paradigm and the MapReduce paradigm. The first one usually addresses applications that need high computing power, while MapReduce is mainly intended to be used for applications that process large amounts of data. We studies the way in which they work in order to be able to apply them to optimize the execution of certain applications.

Chapter 5: Used Technologies and Frameworks - describes the technologies and platforms used for assessing the above-presented means for parallelizing applications and gives advantages and disadvantages regarding their complexity of use. We thoroughly studies these frameworks and platforms and identified their advantages and disadvantages.

Chapter 6: Workflows Use Cases - presents the two workflows that we created based on two rather different applications: one I/O intensive (eBay user data crawler) and the second one computationally intensive (a real life meteorological application that will be used by the Tyrolean avalanche service for forecasting possible avalanches). Both workflows were implemented using the Askalon framework and their performance was tested on cloud resources. We identify the main sections of both applications, and based on the sections that can only be run in a serial manner and the ones that can run in parallel we implemented the two workflows. We also applied different techniques that lead to lower execution times, resulting in lower costs. Especially for the second application, reduced execution time is of great importance due to the fact that the data processed by it is needed daily at a certain time.

Chapter 7: Using Hadoop to Optimize Run Times of Twitter Data Processing - describes the way in which users, that consider Hadoop as a tool for running their data intensive applications, can create and configure their own clusters in order to achieve better results regarding execution time. This also ensures lower execution costs in case the user chooses cloud resources as their computing infrastructure. We conducted some performance tests on different types of Hadoop cluster configurations and compared the gathered results.

Chapter 8: Agent-based Cloud Resource Negotiation - describes a negotiation domain that we created in order to be used by agents to negotiate cloud

resources in the context of a market were several different cloud providers compete for the same users. CloudDomain contains 5 different issues (CPU, RAM, number of available instances, price and hard disk size). The profile was implemented in order to be used by the Genius [11] framework. The created profiles were used in order to simulate the way in which populations consisting of 10 agents having different structures and Q-learning or Bayesian learning mechanisms behave in the context of cloud resource negotiation. A one-to-many negotiation protocol was used for these experiments. We created a mechanism that can generate in a semi-automated manner the negotiation profiles based on the characteristics revealed by private clouds (in our case an Eucalyptus private cloud).

5 Research Mobility

During my PhD I had the opportunity to go for a research mobility abroad. For eight months I was part of the Distributed and Parallel Systems Group from the Institute of Computer Science, University of Innsbruck, Austria under the supervision of Assistant Prof. Radu Prodan. During my stay there I collaborated with my supervisor on one of his projects: the RainCloud project. The project is a collaboration between the Distributed and Parallel Systems Group, the Institute for Meteorology and Geophysics, University of Innsbruck and Tyrolean avalanche service ("Tiroler Lawinenwarndienst" (LWD)). The final scope of the project was to provide the LWD with a meteorological application which will run each day at the same time and which will provide them with realistic forecasts about the precipitation quantity that might occur in a certain area.

The project had two main parts:

- **First Part** – implementing the meteorological application that based on a certain topology and meteorological input data will generate the forecast. This part was taken care of by the Institute for Meteorology and Geophysics, University of Innsbruck.
- **Second Part** – creating a workflow using the Askalon grid/cloud application development platform. The workflow had optimizing the execution

times of the applications by parallelizing some of its areas as a purpose; finding a way how to run the application with the best the performance and minimum costs. This part was under the responsibility of the Distributed and Parallel Systems Group from the Institute of Computer Science, University of Innsbruck.

As part of the Distributed and Parallel System Group I had the role of implementing the workflow for the meteorological application implemented by the other party. My tasks were:

- collaborating with the person that implemented the meteorological application and identifying the different sections of the application that could be mapped as single activities or parallel activities in the workflow.
- identifying all the dependencies existing between the previously found activities in order to be able to define them in the workflow.
- collaborating with the person that created the application in order to make the necessary changes needed to improve the performance of the application.
- implementing the actual workflow based on the above-presented application.
- testing the workflow's performance on cloud resources; first on the private cloud that the research group had at its disposal and then on real public cloud instances in order to be able to estimate the costs generated by executing the application each day for an entire year.

As a result of my research mobility in Austria I published an article with my coworkers based on the workflow we implemented. The article is: **Gabriela Morar**, Felix Schueller, Simon Ostermann, Radu Prodan, and Georg Mayr, "Meteorological Simulations in the Cloud with the ASKALON Environment", CoreGRID/ERCIM Workshop on Grids, Clouds and P2P Computing, Rhodes Island, August, 2012, accepted.

The research mobility allowed me to gather substantial experience in the field of distributed systems, especially grids and clouds. I also learned how workflows operate, how to implement them and how to modify applications in order

to increase their performance. The entire practical and theoretical experience gathered during this period had a significant impact on my research.

6 Results Dissemination

The results obtained during the time of the PhD were presented at several international conferences or workshops or were published in different journals. A list of the publications is presented bellow:

- **Gabriela Andreea Morar**, Cristina Ioana Muntean, Gheorghe Cosmin Silaghi, "Implementing and Running a Workflow Application on Cloud Resources", Economy Informatics, vol. 15, no. 3/2011, pages 15-27
- Cristina Ioana Muntean, **Gabriela Andreea Morar**, and Darie Moldovan, "'Exploring the meaning behind Twitter hashtags through clustering"', in Lecture Notes in Business Information Systems, vol. 127, pages 231 - 242. Springer-Verlag Berlin, 2012.
- Alexandru Butoi, **Gabriela Andreea Morar**, and Andreea Ilea, "Two-Phased Protocol for Providing Data Confidentiality in Cloud Storage Environments", in Lecture Notes in Business Information Systems, vol. 127, pages 220 - 230. Springer-Verlag Berlin, 2012.
- **Gabriela Andreea Morar**, and Andreea Ilea, Alexandru Butoi, Gheorghe Cosmin Silaghi, "Agent-based Cloud Resources Negotiation", in Proceeding of 8th International Conference on Intelligent Computer Communications and Processing, pages 297-300 August 30 - September 1, 2012 in Cluj-Napoca, Romania
- **Gabriela Morar**, Felix Schueller, Simon Ostermann, Radu Prodan, and Georg Mayr, "Meteorological Simulations in the Cloud with the ASKALON Environment", CoreGRID/ERCIM Workshop on Grids, Clouds and P2P Computing, Rhodes Island, August, 2012, accepted
- Alexandru Butoi, **Gabriela Andreea Morar**, Andreea Ilea, "Agent-Based Framework for Implementing and Deploying of SOA", Journal of Mobile, Vol

4, No 2 (2012), Embedded and Distributed Systems (JMEDS) ISSN: 2067 - 4074, pages 107-113

- Alexandru Butoi, Andreea Ilea and **Gabriela Andreea Morar**, "Conceptual Design for Business SOA using Object-Oriented Paradigm", in Proceedings of "The Eleventh International Conference on Informatics in Economy IE 2012", Bucharest, Romania, pages 41-45

- **Gabriela Andreea Morar**, Cristina Ioana Muntean and Nicolae Tomai , "An Adaptive M-learning Architecture for Building and Delivering Content based on Learning Objects", The Second Romanian Workshop on Mobile Business, Cluj-Napoca, Romania, published Economy Informatics, vol. 10, no. 1/2010, pages 63-73

7 Future Research Direction

The research conducted for the current thesis provided me with a greater knowledge regarding parallel computing, distributed systems (especially cloud computing) and means of negotiating computation resources using intelligent agents. The experience accumulated so far, as well as the need that exists on the market for ensuring on-time computational resources lead to identify possible future research directions:

- implementing a module for Askalon, which will have the role of negotiating cloud resources at runtime based on user's QoS requirements and based on the estimation of resources needed by the next activity in a workflow. In the first phase this negotiation mechanism could be applied to federated private clouds composed of academical clouds.

- implementing a scheduler that will schedule jobs based on the estimated energy consumption of the available resources, as reducing energy consumption is likely to be the main direction that scientist are interested in after cost optimization and time efficiency.

- implementing a module for Genius that will be capable of automatically generating negotiation profiles based on the characteristics exposed by cloud providers, profiles that will be further used by a broker agent that has the role to negotiate with users on behalf of the cloud providers.
- implementing a module that will automatically install Hadoop on nodes newly added to a cluster and configure them according to the master node. Implementing and creating a model between different Hadoop configurations and cluster infrastructures, that will be able to suggest to users the best configuration for their current cluster. This is a multi-objective optimization problem since Hadoop has a large number of configurable parameters and setting them to optimal values is not intuitive for either new or experienced users.

These are only a few of the main future research directions that could be followed based on the research conducted so far. All of them refer to areas that currently are of great interest to the research community.

List of Figures

References

[1] Technical report. 11

[2] I. Altintas, O. Barney, Z. Cheng, T. Critchlow, B. Ludaescher, S. Parker, A. Shoshani, and M. Vouk. Accelerating the scientific exploration process with scientific workflows. In *Journal of Physics: Conference Series*, volume 46, page 468. IOP Publishing, 2006. 13

[3] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM. doi: 10.1145/1465482.1465560. URL `http://doi.acm.org/10.1145/1465482.1465560`. 10

[4] Nikhil Bansal. Dynamic speed scaling to manage energy and temperature. In *In IEEE Syposium on Foundations of Computer Science*, pages 520–529, 2004. 9

[5] Idar Barstad and Felix Schüller. An Extension of Smith's Linear Theory of Orographic Precipitation: Introduction of Vertical Layers. *Journal of the Atmospheric Sciences*, 68(11):2695–2709, November 2011. ISSN 0022-4928. doi: 10.1175/JAS-D-10-05016.1. URL `http://journals.ametsoc.org/doi/abs/10.1175/JAS-D-10-05016.1`. 17

[6] Rajkumar Buyya. *High Performance Cluster Computing: Architectures and Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999. ISBN 0130137847. 11

[7] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Comp. Syst.*, 25(6):599–616, 2009. 11

[8] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. 2, 13

[9] T. Fahringer and Askalon Team. Askalon grid environment, 2007. 15

[10] T. Fahringer, A. Jugravu, S. Pllana, R. Prodan, C. Seragiotto Jr, and H. L. Truong. Askalon: a tool set for cluster and grid computing. *Concurrency and Computation: Practice and Experience*, 17(2-4):143–169, 2005. 14

[11] Cristian Figueroa, Nicolas Figueroa, Alejandro Jofre, Akhil Sahai, Yuan Chen, and Subu Iyer. A Game Theoretic Framework for SLA Negotiation. Technical report, Enterprise Systems Storage Laboratory, HP Laboratories, 2008. http://www.hpl.hp.com/techreports/2008/HPL-2008-5.pdf, consulted on 5 August 2011. 23

[12] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, 2001. ISSN 1094-3420. doi: http://dx.doi.org/10.1177/109434200101500302. 5

[13] Hadoop. Hadoop website. `http://hadoop.apache.org`, 2012. 2

[14] John L. Hennessy and David A. Patterson. *Computer Architecture, Fourth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006. ISBN 0123704901. 9

[15] A. J. G. Hey, S. Tansley, and K. M. Tolle. *The fourth paradigm: data-intensive scientific discovery*. Microsoft Research Redmond, WA, 2009. 4

[16] T. Hey and A. Trefethen. The data deluge: An e-science perspective. In *Grid computing*, pages 809–824. Wiley Online Library, 2003. 13

[17] M.D. Hill and M.R. Marty. Amdahl's law in the multicore era. *Computer*, 41(7):33–38, 2008. 12

[18] D. Hollingsworth. Workflow management coalition: The workflow reference model. Technical report, The Workflow Management Coalition, 1995. 13

[19] S.B. Joshi. Apache hadoop performance-tuning methodologies and best practices. In *Proceedings of the third joint WOSP/SIPEW international conference on Performance Engineering*, pages 241–242. ACM, 2012. 19

[20] Tobias Kurze, Markus Klems, David Bermbach, Alexander Lenk, Stefan Tai, and Marcel Kunze. Cloud Federation. In *Proceedings of the 2nd International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2011)*. IARIA, September 2011. 11

[21] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967. 20

[22] Apache Mahout. Mahout website. `http://mahout.apache.org`, 2012. 19

[23] Dan C. Marinescu. Cloud computing: Theory and practice, 2012. URL `http://www.cs.ucf.edu/~dcm/LectureNotes.pdf`. 9, 11

[24] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The eucalyptus open-source cloud-computing system. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pages 124–131. IEEE, 2009. 14

[25] M. Olson. Hadoop: Scalable, flexible data storage and analysis. *IQT Quarterly*, pages 14–18, 2010. 14

[26] Peter Pacheco. *An Introduction to Parallel Programming*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2011. ISBN 9780123742605. 9, 10

[27] Gregory F. Pfister. *In search of clusters (2nd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998. ISBN 0-13-899709-8. 11

[28] B. T. Rao, N. V. Sridevi, V. K. Reddy, and L. S. S. Reddy. Performance issues of heterogeneous hadoop clusters in cloud computing. *Global Journal of Computer Science and Technology*, 11(8), 2011. 19

[29] L. Sęrban, C. Ştefanache, G. Silaghi, and C. Litan. A qualitative ascending protocol for multi-issue one-to-many negotiations. *Complex Automated Negotiations: Theories, Models, and Software Competitions*, pages 143–159, 2013. 21

[30] Liviu Dan Serban, Cristina Maria Stefanache, Gheorghe Cosmin Silaghi, and Cristian Marius Litan. A qualitative ascending protocol for multi-issue one-to-many negotiations. In *Proc. of the 2011 Workshop on Agent-based Complex Automated Negotiations*, Studies in Computational Intelligence. Springer, 2012. to appear. 21

[31] Kwang Mong Sim. Towards complex negotiation for cloud economy. In *Advances in Grid and Pervasive Computing*, volume 6104 of *LNCS*, pages 395–406. Springer, 2010. 6

[32] H. Sutter. The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobb?s Journal*, 30(3):202–210, 2005. 12

[33] Andrew S. Tanenbaum and Maarten van Steen. *Distributed Systems: Principles and Paradigms (2nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006. ISBN 0132392275. 10

[34] I.J. Taylor. *Workflows for e-science: scientific workflows for grids*. Springer-Verlag New York Inc, 2007. 13

[35] Luis M. Vaquero, Luis Rodero-merino, Juan Caceres, and Maik Lindner. A break in the clouds: Towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, pages 50–55, 2009. URL `http://dx.doi.org/10.1145/1496091.1496100`. 11

[36] Virtualbox. Virtualbox website. https://www.virtualbox.org/, 2012. 19

[37] T. White. *Hadoop: The definitive guide*. Yahoo Press, 2010. 19