Capítulo

5

Resource Allocation in Clouds: Concepts, Tools and Research Challenges

Glauco Estácio Gonçalves, Patrícia Takako Endo, Thiago Damasceno Cordeiro, André Vitor de Almeida Palhares, Djamel Sadok, Judith Kelner, Bob Melander, Jan-Erik Mångs

Abstract

Cloud computing is an attractive computing model since it allows for the provision of resources on-demand. Such a process of allocation and reallocation of resources is the key to accommodating unpredictable demands and improving the return on investment from the infrastructure supporting the Cloud. However, despite the recent growth of the Cloud Computing market, several problems with the process of resource allocation remain unaddressed. This short course introduces essential concepts and technologies regarding Cloud Computing and presents some research questions on the topic, focusing on the challenges and the state-of-the-art solutions in resource allocation.

Resumo

Computação na Nuvem é um modelo de computação atrativo, uma vez que permite que recursos sejam provisionados sob demanda. Tal processo de alocação e realocação de recursos é a chave para acomodar demandas imprevistas e para melhorar o retorno de investimento na infraestrutura que suporta a Nuvem. Entretanto, a despeito da recente expansão do mercado de Computação na Nuvem, diversos problemas relativos à alocação de recursos permanecem abertos. Este minicurso introduz os conceitos e tecnologias essenciais da Computação em Nuvem e apresenta algumas questões de pesquisa na área, focando nos desafios e soluções para alocação de recursos.

5.1. Introduction

Cloud Computing offers an interesting solution for software development and access of content with transparency of the underlying infrastructure locality. The Cloud infrastructure is usually composed of several datacenters and consumers have access to only a slice of the computational power over a scalable network. The provision of these computational resources is controlled by a provider, and resources are allocated in an elastic way, according to consumers' needs.

The use of Clouds as a type of infrastructure for running software is quite different than traditional practices, where software runs over infrastructures often dimensioned according to the worst case use and peak scenarios. To accommodate unforeseen demands on the infrastructure in a scalable and elastic way, the process of allocation and reallocation in Cloud Computing must be dynamic. Furthermore, another essential feature of the resource allocation mechanisms in Cloud Computing is to guarantee that the requirements of all applications are suitably met. According to [Khan, 2009], "a resource allocation is defined to be robust against perturbations in specified system parameters if degradation in the performance feature is limited when the perturbations occur within a certain range".

To achieve this requirement, any allocation mechanism in Cloud Computing should be aware of the status of each element/resource in the infrastructure. Then, the mechanism should apply algorithms to better allocate physical or virtual resources to consumers' applications, according to the requirements pre-established with the cloud provider.

Beyond the benefit of elastic services, Cloud Computing allows consumers to reduce or eliminate costs associated with internal infrastructure for the provision of their services. This opportunity of cost reduction makes Cloud Computing a very attractive alternative for consumers, especially for business initiatives. Enterprises can effectively offload operational risks to cloud providers. From the perspective of cloud providers, the model offers a way for better utilization of their own infrastructure. Authors in [Armbrust et al 2009] suggest that this model benefits from a form of statistical multiplexing, since it allocates resources for several users concurrently. This statistical multiplexing of datacenters is guaranteed through several decades of research in areas such as: distributed computing, grid computing, web technologies, service computing, and virtualization.

However, despite the recent and notable growth of the Cloud Computing market, there are several key issues open with the process of resource allocation. This short course introduces essential concepts and technologies for Cloud Computing. It also presents some research questions on the area, focusing on the challenges and state-of-the-art solutions for resource allocation.

5.1.1. The Emergence of Cloud Computing

Nowadays, there are several definitions for Cloud Computing in literature, covering common terms like IaaS (Infrastructure as a Service), PaaS (Platform as a Service) and SaaS (Software as a Service). However, there is still confusion about the definition due to the lack of standardization. According to [Zhang et al 2010], the main reason for the existence of different perceptions of Cloud Computing is that it is not a new technology, but rather a new model that brings together a set of existing technologies to develop and

run applications in a different way. In fact, technologies such as virtualization and service oriented provisioning are not new, however Cloud Computing uses them to offer a new service to its consumers and, at the same time, to meet new business requirements. Such discussion about the definition of Cloud Computing will be reexamined in more detail on Section 5.2.1.

Since the popularization of the Cloud Computing term in 2007, with IBM Blue Cloud [Vouk 2008], several enterprises have become Cloud Computing providers: Amazon with Elastic Compute Cloud (EC2)[1], Google with Google App Engine[2], Microsoft with Windows Azure Platform[3], and Salesforce with Force.com[4] . Despite offering services with different purposes, all of these solutions are called Clouds. Each solution has their own programmability level and therefore can be put in their specific class. More about the different proposals for the classification of Cloud initiatives will be presented in Section 5.2.3.

As well as contributions from commercial interests, the Open Source Community has been very active in the development of Cloud Computing. They have supplied numerous contributions in related areas such as tools for interaction with existent Clouds, software for automated use of available Clouds, alternatives for standardization, and virtualization. Moreover, the Open Source Community is actively developing solutions for management clouds, especially those employed in academic research around the world. Some of these tools are presented in Section 5.5.

5.1.2. The value of Cloud for Business

Many organizations have invested in Cloud Computing, and to date more than 150 enterprises have entered the industry as Cloud providers[5]. On the side of Cloud consumers, a recent study of more than 600 companies by InformationWeek reported that the number of companies using Cloud Computing increased from 18% in February 2009 to 30% in October 2010, a 67% increase. Moreover, an interesting worldwide survey by Gartner[6] identified Cloud Computing as the top technology priority for CIOs in 2011.

The cited business research on Cloud Computing demonstrates financial interest and reveals the increasing credibility of Cloud Computing. Such growth is motivated by its ongoing consolidation as well as by the revenues that customers and operators are observing with Cloud Computing. It is interesting to notice the different viewpoints of both parties: from the side of enterprises that are using Cloud Computing, it is very attractive since it provides opportunity for cost reductions with internal infrastructure, as well as other advantages. Alternately, from the providers' viewpoint, Cloud Computing makes it possible to increase revenues using their own IT infrastructure. However, such investment should be carefully dimensioned, since consumers have high expectations

---

[1] http://aws.amazon.com/ec2/

[2] http://code.google.com/intl/appengine/appengine/

[3] http://www.microsoft.com/windowsazure/

[4] http://www.salesforce.com/platform/

[5] http://cloudcomputing.sys-con.com/node/770174

[6] http://www.gartner.com/it/page.jsp?id=1526414

about the elasticity of their applications. Metaphorically, one can say that consumers expect that provider's resources be infinite. Questions like "How many physical machines are necessary to accommodate my unpredictable demand?" and "How many consumers are necessary to obtain financial returns in a reasonable time?" should be taken into consideration by providers.

In this way, considering the provider's viewpoint, resource management in this business scenario should be treated very cautiously. Authors in [Buyya et al 2008] believes that market-oriented resource management is necessary to regulate supply and demand of resources to achieve market equilibrium. They suggest that this would provide feedback in terms of economic incentives for both consumers and providers. Furthermore, they promote the use of QoS-based resource allocation mechanisms that differentiate service requests based on their utility.

5.1.3. The value of Cloud for Academy

In parallel to the commercial growth of Cloud Computing, it is prevalent in academic circles and is the theme of many research projects around the world. Many important proposals of open architectures were developed by academic institutes, such as the Cumulus Project [Wang et al 2008], and Eucalyptus [Nurmi et al 2009],.

The Cumulus Project is based on OpenNebula [OpenNebula Project 2011a], and it intends on providing virtual machines, virtual applications and virtual computing platforms for scientific applications. Visualizing the integration of already existing technologies, the Cumulus project uses HP and IBM blade servers running Linux and the open-source Xen hypervisor[7].

Eucalyptus is another open source Cloud Computing framework that provides resources for experimental instrumentation and academic study. Eucalyptus users are able to start, control, access, and terminate entire virtual machines. In its current version, Eucalyptus supports VMs that run atop the Xen supervisor [Barham et al 2003].

Beyond open architectures and tools, resource allocation is another topic that has been attracting attention in the academic community. Some work had been done to address this problem, but with different focuses. At the same time, one can note that the resource allocation issue is not a new problem. There are many solutions focused in the datacenter area ([Urgaonkar et al 2010], [Chen et al 2010], [Kong et al 2010]) and Grid Computing ([Murphy et al 2010], [Xhafa and Abraham 2010]). In this way, some of the new strategies for resource allocation employ these existing solutions by adapting them to be used in Cloud Computing environments ([Lee et al 2010], [Beloglazov and Buyya 2010], [Buyya et al 2010]).

There is also a different way to handle resources in Clouds that is based on the allocation problems in Virtual Network Environments [Chowdhury 2009]. General concepts from the area of networks virtualization can be applied to the field of Cloud Computing. This can be done by considering that virtual networks can be associated with developer's applications, which are composed of several servers and databases as well as the network elements (routers, switches, and links) between them. Section 5.4.3

[7] http://xen.org/

describes the resource allocation problems on network virtualization technology in more detail.

There are many academic papers relating to challenging research in Cloud Computing, such as automated service provisioning, server consolidation, and energy management. Some of these challenges are discussed in Section 5.3. Indeed, this is the major motivation for Cloud Computing in academia: there are many questions to be answered.

5.1.4. Structure of the short course

This paper presents and discusses the state-of-the-art and the challenges of resource allocation for Cloud Computing. To better understand these challenges, the remainder of this paper is organized as follows.

Section 5.2 introduces the basic principles of Cloud Computing by presenting the definitions, agents, and a classification of Cloud operators. This section is important because it demonstrates the authors' view about Cloud Computing and presents definitions used throughout of this paper.

Section 5.3 focuses on the description of research opportunities in Cloud Computing. For didactical reasons, these research questions were classified in three areas: negotiation, resource management, and resource control. Challenges in the negotiation area are related to the creation of innovative interfaces between developers and the Cloud. Challenges in the resource management and resource control areas are related to automated provisioning of resources, including energy-efficient, VM migration and cloning, development of communication protocols, and security and privacy.

Resource management and resource control layers are faced with the problem of selecting the best suitable physical/virtual resources to accommodate applications under previously negotiated requirements. In this short course, we address these issues by describing concepts and solutions of resource allocation problem in Section 5.4.

Section 5.5 is focused on describing some of the existing open source solutions by highlighting their main characteristics and architectures. Finally, a conclusion of this paper and future trends are presented in Section 5.6.

5.2. What is Cloud Computing?

In this section the main concepts of Cloud Computing will be presented. It will begin with a discussion on the definition of Cloud Computing (Section 5.2.1) and the main agents involved in Cloud Computing (Section 5.2.2). After that, classifications of Cloud initiatives are offered in Section 5.2.3. Finally, some of the main technologies acting behind the scenes of Cloud Computing initiatives are presented in Section 5.2.4.

5.2.1. Definitions

Considering the differences between the several Cloud Computing initiatives, one can understand Cloud Computing as a broad area encompassing such different research areas as Grid Computing and Content Delivery Networks (CDNs). Regardless of their coverage, Cloud Computing may be seen as a commonplace for all these areas, since it is able to offer solutions to the problems of each area and it can be applied according to

respective singularities. Thus, far from intruding into those research areas, Cloud Computing establishes relationships with them. Basically there are two relationships: on one hand, Cloud Computing inherits techniques and algorithms from each area; on the other hand, Cloud Computing offers an infrastructure for applications of these areas. This occurs with Grid Computing, for example, since traditional task allocation techniques used in grids can be used on Clouds and, at the same time, uncertainties in availability and performance present in grids can be solved with Cloud's infrastructure [Lee and Zomaya 2010]. With CDNs a similar relationship occurs; Cloud Computing can adapt servers placement algorithms to its own necessities [Presti et al. 2007] and CDNs can use Cloud Computing environments to improve their operations ([Ramaswamy et al. 2005], [Broberg et al. 2009]).

Despite this relationship with other areas and the increasingly widespread use of the Cloud Computing term, the concept lacks a formal definition. Authors in [Vaquero et al. 2008] reviewed the literature for a minimum set of characteristics that Clouds must exhibit. However, they were not able to find a single common feature between current definitions. On the other hand, they noted that the set of characteristics that is most similar to a minimum common definition is: elasticity, pay-per use model, and virtualization. The first two characteristics are essential for Cloud Computing and are complementary, since users want to rent and release resources on-demand and pay for what they use. However, there is no consensus about virtualization. Some other definitions ([Buyya et al. 2009], [Armbrust et al. 2009]) highlight virtualization as an essential technology for Cloud Computing, but leading enterprises in this area, like Google [Sheth 2009], affirm that such technology is dispensable for their solutions. The concept here is that virtualization is an old solution in computer science, frequently identified with some form of abstraction, which can be used with different purposes [Padala 2010]. Several solutions, like Amazon EC2, use server virtualization technologies to "break" a powerful physical computational resource into smaller, virtual entities to be sold. Alternately, other initiatives, like Google App Engine, use distributed processing technologies that join several computational resources into only one virtual computational resource offering an environment like a supercomputer.

From the minimum set of characteristics encountered by [Vaquero et al. 2008] (elasticity, pay-per use model, and virtualization), they also formulated their own definition of Clouds: "Clouds are a large pool of easily usable and accessible virtualized resources. These resources can be dynamically reconfigured to adjust to a variable load, allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLA". Other authors give a similar definition [Buyya et al. 2009]: "A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers". Please note that both definitions cite the use of service-level agreements as a necessary aspect in Cloud Computing.

An interesting definition of Cloud Computing is given by the National Institute of Standards and Technology (NIST) of the United States: "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that

can be rapidly provisioned and released with minimal management effort or service provider interaction" [Mell and Grace 2009]. The definition confirms that on-demand dynamic reconfiguration (previously called elasticity) is a key characteristic. Additionally, the NIST definition highlights another Cloud Computing characteristic: it assumes that there are minimal management efforts required to reconfigure resources. In other words, the Cloud must offer self-service solutions that must attend to requests on-demand. This is important because it excludes from the scope of Cloud Computing those initiatives that operate through the rental of computing resources in a weekly or monthly basis. Hence, it restricts Cloud Computing to systems that provide automatic mechanisms for resource rental in real-time with minimal human intervention.

The NIST definition gives a satisfactory explanation of Cloud Computing as a computing model. But, contrary to the two previously cited definitions, NIST does not cover the main object of Cloud Computing: the Cloud. Thus, in this short course, Cloud Computing is defined as the computing model that operates based on Clouds. In turn, the Cloud is defined as a conceptual layer that operates above an infrastructure to provide services in a timely manner.

This definition encompasses three main characteristics of Clouds. Firstly, it notes that a Cloud is primarily a concept, i.e., a Cloud is an abstraction layer over an infrastructure. Thus, since Clouds are simply a concept, they are independent of the technologies in the infrastructure and therefore one can accept different setups, like Amazon EC2 or Google App Engine, to be named Clouds. Moreover, the infrastructure is defined in a broad sense once it can be composed by software, physical devices, and/or other Clouds, as occurs with Dropbox[8] that is hosted by the Amazon Simple Storage Service (S3)[9]. Secondly, all Clouds have the same purpose: to provide services. This means that a Cloud hides the complexity of the underlying infrastructure while exploring the potential of overlying services, acting similarly to a middleware. Also, providing a service involves, implicitly, the use of some type of agreement that should be guaranteed by the Cloud. Such agreements can vary from pre-defined contracts to malleable agreements defining functional and non-functional requirements. Thirdly, the Cloud must provide services as quickly as possible such that the infrastructure resources are allocated and reallocated to attend the users' needs.

5.2.2. Agents involved in Cloud Computing

Several authors ([Vaquero et al. 2008], [Buyya et al. 2009], [Zhang et al 2010], and [Vouk 2008]) have presented the agents involved in Cloud Computing. However, there is some discordance between the explanations of each one. Therefore, this short course will focus only on three distinct agents in Cloud Computing as shown in Figure 5.1: clients, developers, and provider. The first notable point is that the provider will deal with two types of users that, in this short course, are called developers and clients. Such differentiation is important because, in Cloud Computing, users (developers) can create new services that will be accessed by another type of users (clients). Thus, clients are the customer of a service produced by a developer. They lease services from developers and use these services, but such use generates demand to the provider that hosts the

[8] http://www.dropbox.com/

[9] https://s3.amazonaws.com/

service, and therefore the client can also be considered a user of the Cloud. It is important to highlight that in some scenarios (like scientific computation or batch processing) a developer may behave as a client to the Cloud because it is the end-user of its applications. Please note that the text will use "users" when referring to both classes without distinctions.

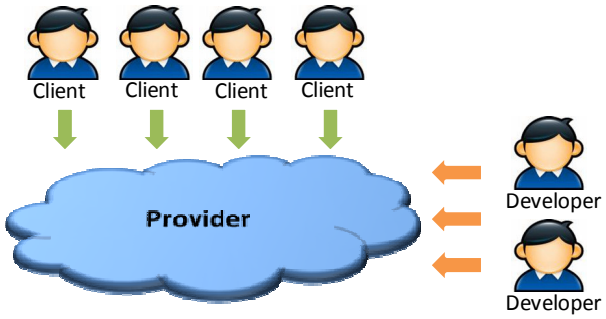

Figure 5.1. Agents in a typical Cloud Computing scenario

Developers can be service providers, independent programmers, scientific institutions, and so on, all who build applications into the Cloud. Please note that, a priori, developers do not need to know about the architectures and technologies that compound the Cloud infrastructure, nor about the specific location of each item in the infrastructure. This concept can be called as transparency of locality. Basically, developers want to create and run their applications while keeping decisions related to maintenance and management of the infrastructure to the provider.

Lastly, it must be clarified that the term application is used to mean all types of software that can be developed on the Cloud. These applications may be requisitioned from clients or from developers, depending on the applications' purpose as determined by its developer. In addition, it is important to notice that the type of applications supported by a Cloud depends exclusively on the purposes of the Cloud determined by the provider. Such variety of purposes generates many different types of Cloud Operators that are discussed in Section 5.2.3.

Please note that the aforementioned agents are a simplification of the entire ecosystem that can emerge from Cloud Computing. The figure of the Cloud Brokers, for example, arises naturally. [Buyya et al. 2009] defines that such brokers have the mission of acting on behalf of developers to select suitable providers and negotiate application requirements with them. Also, brokers operating in the Cloud Computing market will force standardization in the industry, since they will need to compare information from each operator on a solid and clear basis.

5.2.3. Classification of Cloud Operators

Currently, there are several operational initiatives of Cloud Computing; however despite all being called Clouds, they provide different types of services. For that reason, the academic community ([Vaquero et al. 2008], [Buyya et al. 2009], [Mell and Grace 2009], [Zhang et al. 2010], [Youseff et al. 2008]) precisely classified these solutions in order to understand their relationships. The three complementary proposals for classification are as follows.

Classification according to the intended audience

This first simple taxonomy is suggested by NIST [Mell and Grace 2009] and tries to organize providers according to the deployment model of the Cloud, i.e., according to the audience at which the cloud is aimed. There are four classes in this classification: Private Clouds, Community Clouds, Public Clouds, and Hybrid Clouds.

The first three classes accommodate providers in a gradual opening of the intended audience coverage. The Private Cloud class encompasses such types of Clouds destined to be used solely by an organization operating over their own datacenter or one leased from a third party for exclusive use. When the Cloud infrastructure is shared by a group of organizations with similar interests it is classified as a Community Cloud. Furthermore, the Public Cloud class encompasses all initiatives intended to be used by the general public. It is important to highlight that this latter class is the main focus of the literature on Cloud Computing since it offers the main challenges. Finally, Hybrid Clouds are simply the composition of two or more Clouds pertaining to different classes (Private, Community, or Public).

Classification according to the service type

In [Youseff et al. 2008], authors offer a classification as represented in Figure 5.2. Such taxonomy divides Clouds in five categories: Cloud Application, Cloud Software Environment, Cloud Software Infrastructure, Software Kernel, and Firmware/Hardware. By employing the concept of service composition from the realm of Service-Oriented Architecture (SOA), authors arrange the different types of Clouds in a stack, showing that Clouds of higher levels are created using services in lower levels. This idea pertains to the definitions of Cloud Computing discussed previously in Sections 5.2.1 and 5.2.2. Essentially, the Cloud operator does not needs to be the owner of the infrastructure. However, this classification allows each layer to use services of all layers below, permitting a Cloud on the Cloud Application layer to operate over their own physical infrastructure.

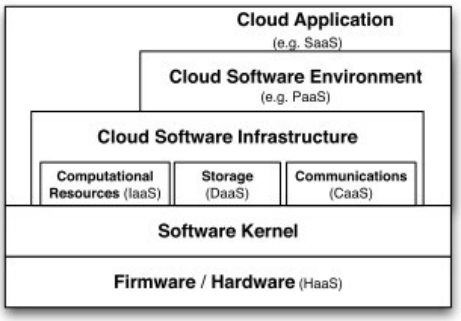

Figure 5.2. Classification of Cloud types (from [Youseff et al. 2008])

The class in the top of the stack, also called Software-as-a-Service (SaaS), involves applications accessed through the Internet, including social networks, webmail, and office tools. Such services provide software to be used by the general public, whose main interest is to avoid tasks related to software management like installation and updating. Moreover, for the same reasons, these services can target businesses as in the case of Customer Relationship Manager (CRM) software provided by Salesforce.com. From the point of view of the cloud operator, SaaS can decrease costs with software implementation when compared with traditional processes. Similarly, the Cloud Software Environment, also called Platform-as-a-Service (PaaS), encloses Clouds that

offer programming environments for developers. Through well-defined APIs, developers can use software modules for access control, authentication, distributed processing, and so on, in order to produce their own applications in the Cloud. Also, developers can contract services for automatic scalability of their software, databases, and storage services.

In the middle of the stack is the Cloud Software Infrastructure class of initiatives. This class encompasses solutions that provide virtual versions of infrastructure devices found on datacenters like machines, databases, links, and so on. Clouds in this class can be divided into three subclasses according to the type of resource that is offered by the Cloud. Computational resources are grouped in the Infrastructure-as-a-service (IaaS) subclass that provides generic virtual machines that can be used in many different ways by the contracting developer. Services for massive data storage are grouped in the Data-as-a-Service (DaaS) class, whose main mission is to save users' data on remote disks, which allows access to these users from anywhere and at anytime. Finally, the third subclass, called Communications-as-a-Service (CaaS), is composed of solutions that offer virtual private links and routers through telecommunication infrastructures.

The last two classes do not offer Cloud services specifically, but they are included in the classification to show that providers offering Clouds in higher layers can have their own software and hardware infrastructure. The Software Kernel class includes all of the software necessary to provide services to the other categories like operating systems, hypervisors, cloud management middleware, programming APIs, libraries, and so on. The class of Firmware/Hardware covers all sale and rental services of physical servers and communication hardware.

Classification according to programmability

The five-class scheme presented above can classify and organize the current spectrum of Cloud Computing solutions, but such a model is limited because the number of classes and their relationships will need to be rearranged as new Cloud services emerge. Therefore, in this short course, a different classification model will be used based on the programmability concept, which was previously introduced in [Endo et al. 2010].

Borrowed from the realm of network virtualization [Chowdhury and Boutaba 2010], programmability is a concept related to the programming features a network element offers to developers, measuring how much freedom the developer has to manipulate resources and/or devices. This concept can be easily applied to the comparison of Cloud Computing solutions. More programmable Clouds offer environments where developers are free to choose programming paradigms, languages, and platforms, thus having more control over their leased resources. Less programmable clouds restrict developers in some way: perhaps by forcing a set of programming languages on the developer or by providing support for only one application paradigm. On the other hand, programmability directly affects the way developers manage their leased resources. From this point-of-view, providers of less programmable Clouds are responsible to manage their infrastructure while being transparent to developers. In turn, a more programmable Cloud leaves more of these tasks to developers, thus introducing management difficulties due to the more heterogeneous programming environment.

Thus, Cloud Programmability can be defined as the level of sovereignty developers have to manipulate services leased from an operator. Programmability is a

relative concept, i.e., it was designed to compare one Cloud with others. Also, programmability is directly proportional to heterogeneity in the infrastructure of the provider and inversely proportional to the amount of effort that developers must spend to manage leased resources.

To illustrate how the concept can be used, one can classify two current Clouds: Amazon EC2 and Google App Engine. One can note that Amazon EC2 is more programmable, since in this Cloud developers can choose between different virtual machine classes, operating systems, and so on. After they lease one of these virtual machines, developers can configure it to work as they see fit: as a web server, as a content server, as a unit for batch processing, and so on. On the other hand, Google App Engine can be classified as a less programmable solution, because it allows developers to create web applications that will be hosted by Google. This restricts developers to the Web paradigm and to some programming languages.

5.2.4. Mediation System

Figure 5.3 introduces an Archetypal Cloud Mediation System that can be used as a reference to the main targets of a Cloud Mediation System. The Archetypal Mediation System was developed while focusing on one principle: resource management as the main service of any Cloud Computing provider. Thus, this reduced mediation system relegates other important services like authentication, accounting, and security, for example, to the group of auxiliary services that is out of the scope of this short course. Clients also do not factor into this view of the system, since resource management is mainly related to the allocation of developers' applications and meeting their requirements.

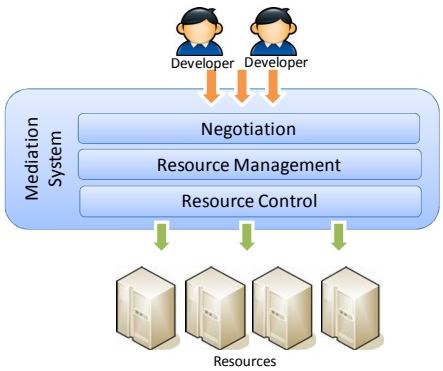

Figure 5.3. Components of a Archetypal Cloud Mediation System

The mediation system is responsible for the entire process of resource allocation in the Cloud. Such a process covers tasks that range from the automatic negotiation of developers requirements to the execution of their applications. It has three main layers: negotiation, resource management, and resource control.

The negotiation layer deals with the interface between the Cloud and developers. In the case of Clouds serving infrastructure services, the interface can be a set of operations based on Web Services for access and control of the leased virtual machines. Alternately, in the case of PaaS services, this interface can assume the form of an API with programming primitives for software development in the Cloud. Moreover, the negotiation layer handles the process of contract establishment between the enterprises and the Cloud. Currently, this process is simple and the contracts tend to be restrictive.

One can expect that in the future, Clouds will offer more sophisticated avenues for user interaction through high level abstractions and service level policies.

The resource management layer is responsible for the optimal allocation of applications for obtaining the maximum usage of resources. This function requires advanced strategies and heuristics to allocate resources that meet the contractual requirements as established with the application developer. These may include service quality restrictions, jurisdiction restrictions, elastic adaptation, and so on.

Metaphorically, one can say that while the resource management layer acts as the "brain" of the Cloud, the resource control layer plays the role of its "limbs". The resource control encompasses all of the functions needed to enforce decisions generated by the upper layer. Beyond the tools used to configure the Cloud resources effectively, all communication protocols used by the Cloud are included in this layer.

5.2.5. Groundwork Technologies

In the following section, some of the technologies that compose the groundwork in current mediation systems on Cloud Computing (namely Service-oriented Computing, Virtualization, MapReduce Framework, and Datacenters) will be discussed.

Service-Oriented Computing

Service-Oriented Computing (SOC) defines a set of principles, architectural models, technologies, and frameworks for the design and development of distributed applications based on the Service-Orientation paradigm. In turn, Service-Orientation is a "design paradigm intended for the creation of solution logic units (services) that are individually shaped so that they can be collectively and repeatedly utilized" [Erl 2009].

The development of software focused on services gave rise to SOA (Service-Oriented Architecture), which can be defined as an architectural model "that supports the discovery, message exchange, and integration between loosely coupled services using industry standards" [Khoshafian 2007]. In other words, SOA defines a set of standards for the design and development of systems, and can be easily reused and combined to adapt to changes in business requirements. The common technology for the implementation of SOA principles is the Web Service that defines a set of standards to implement services over the World Wide Web.

In Cloud Computing, SOA is the main paradigm for the development of functions on the negotiation layer. Operators publish APIs for their services on the web, allowing developers to use the Cloud and to automate several tasks related to the management of their applications. Such APIs can assume the form of WSDL documents[10], REST-based documentation[11], or libraries for popular programming languages[12]. Also, operators can make available SDKs and other toolkits for the manipulation of applications running on the Cloud.

---

[10] http://s3.amazonaws.com/ec2-downloads/ec2.wsdl

[11] http://kenai.com/projects/suncloudapis/pages/Home

[12] http://code.google.com/intl/pt-BR/appengine/docs/

Server Virtualization

Server (or platform) virtualization is a technique that allows a computer system to be partitioned on multiple isolated execution environments similar to a single physical computer, which are called Virtual Machines (VM). Each VM can be configured in an independent way having its own operating system, applications, and network parameters. Commonly, such VMs are hosted on a physical server running a hypervisor, which is a piece of software that effectively virtualizes the server and manages the VMs [Padala 2010].

There are several hypervisor options that can be used for server virtualization. From the open-source community, one can cite Citrix's Xen[13] and the Kernel-based Virtual Machine (KVM)[14]. From the realm of proprietary solutions, some examples are VMWare ESX[15] and Microsoft's HyperV[16].

The main factor that boosted up the adoption of server virtualization within Cloud Computing is that such technology offers good flexibility regarding reallocation of workloads across the physical resources. Such flexibility allows, for example, for operators to execute maintenance without stopping developers' applications (that are running on VMs) or to implement strategies for better resource usage through the migration of VMs. Also, server virtualization is adapted for the fast provisioning of new VMs through the use of templates, which enables providers to offer elasticity services for applications developers [Marshall et al. 2010].

MapReduce Framework

MapReduce [Dean and Ghemawat 2004] is a programming framework developed by Google for distributed processing of large data sets across computing infrastructures. Inspired on the map and reduce primitives present in functional languages, authors of MapReduce developed an entire framework for the automatic distribution of computations. In this framework, developers are responsible for writing map and reduce operations and for using them according to their needs, which is similar to the functional paradigm. These map and reduce operations will be executed by the MapReduce system that transparently distributes computations across the computing infrastructure and treats all issues related to node communication, load balance, and fault tolerance. Also, for the distribution and synchronization of the data required by the application, the MapReduce system also requires the use of a distributed file system called Google File System (GFS) [Ghemawat et al. 2003].

Despite being introduced by Google, there are some open source implementations of the MapReduce system, like Hadoop[17] and TPlatform[18]. The former is popular open-source software for running applications on large clusters built of commodity hardware. This software is used by companies like Amazon, AOL, and

---

[13] http://www.xen.org/products/cloudxen.html

[14] http://www.linux-kvm.org/page/Main_Page

[15] http://www.vmware.com/

[16] http://www.microsoft.com/hyper-v-server/en/us/default.aspx

[17] http://hadoop.apache.org/

[18] http://net.pku.edu.cn/~webg/tplatform/

IBM, as well as in different web applications such as Facebook, Twitter, Last.fm etc[19]. Basically, Hadoop is composed of two modules: a MapReduce environment for distributed computing, and a distributed file system called Hadoop Distributed File System (HDFS). The latter is an academic initiative that provides a development platform for web mining applications. Similarly to Hadoop and Google's MapReduce, the TPlatform has a MapReduce module and a distributed file system called Tianwang File System (TFS) [Peng et al. 2009].

The use of MapReduce solutions is common groundwork technology in PaaS Clouds because it offers a versatile sandbox for developers. Differently from IaaS Clouds, PaaS developers using a general-purpose language with MapReduce support do not need to be concerned about software configuration, software updating, network configurations, and so on. All these tasks are the responsibility of the Cloud provider, which, in turn, benefits from the fact that configurations will be standardized across the overall infrastructure.

Datacenters

Developers who are hosting their applications on a Cloud wish to scale their leased resources, effectively increasing and decreasing their virtual infrastructure according to the demand of their clients. This is also true for developers making use of their own private clouds. Thus, independently of the class of Cloud in question, a robust and safe infrastructure is needed.

Whereas virtualization and MapReduce respond for the software solution to attend this demand, the physical infrastructure of Clouds relies on one or more datacenters. These datacenters are infrastructures composed of TI components providing processing capacity, storage, and network services for one or more organizations [Veras 2009]. Currently, the size of a datacenter (in number of components) can vary from tens of components to tens of thousands of components depending on the datacenter's mission. Also, there are several different TI components for datacenters including switches and routers, load balancers, storage devices, dedicated storage networks, and, the main component of any datacenter, servers [Greenberg et al. 2008].

In terms of servers, the blade variety arises as a good solution for Cloud Computing, since they possess modular design optimized to minimize expenditures of power consumption, cooling capacity, and the use of physical space [Veras 2009]. Once a blade server chassis has been installed, additional servers can be added into bays while the system is up and running, which promotes scalability. However, the initial configuration can be labor-intensive and expensive. This disadvantage comes with the fact that the blade servers are specialized computing equipment and their configuration and administration often requires training provided by the vendor, which may not be inexpensive.

For Cloud Computing, the use of datacenters provides the required power to attend developers' demands on processing, storage, and networking capacities. Also, a large datacenter, running a virtualization solution, allows for better usage of the hardware's power through the statistical multiplexing of developers' applications.

[19] http://wiki.apache.org/hadoop/PoweredBy

5.3. Open Research Problems

Despite the large number of current Cloud solutions, several research challenges remain open. For didactical reasons, the research questions were grouped into three areas following the three layers presented in Section 5.2.4: negotiation, resource management, and resource control. Upcoming sections will discuss the main challenges to address in each one of these areas: Section 5.3.1 will discuss challenges in the negotiation layer and Section 5.3.2 will discuss challenges in the resource management and resource control layers jointly.

5.3.1. Challenges in the negotiation layer

Challenges in the negotiation area are related to the creation of innovative interfaces between developers and the Cloud. Currently, these interfaces assume the form of an API that, according to the programmability level, can range from a web-service toolkit for VM manipulation (e.g. Amazon EC2) to a set of programming primitives used to develop applications (e.g. Google App Engine). In addition, such APIs allow developers to request and control additional functionalities like service quality assessment, load balance, elastic application growth, backup strategies, and so on.

However, the next generation of Clouds – which are associated with the Internet of Services[20] – must offer **sophisticated interaction mechanisms** for human users, which can be based on service-level specifications. This type of interface will demand formal approaches to specifying the operator's offerings and the developer's requirements. Also, it is important to highlight that these requirements and offerings will overcome current requirements while considering issues like geographical restrictions or juridical/political aspects.

Beyond the internal challenges faced by Clouds, the new service-aware markets, as emphasized by [Llorente 2010], will bring both new opportunities and new challenges to the interaction between operators. These operators will provide specific interfaces on which their services/resources can be aggregated by developers (or brokers) to compose new services to be offered. Thus, an upcoming issue in the future that must be overcome is **the need for standardization**.

Currently, the main market providers offer proprietary interfaces to access their services, which can hinder users within their infrastructure as the migration of applications cannot be easily made between cloud providers [Buyya et al 2009]. It is hoped that cloud providers identify this problem and work together to offer a standardized API based on open standards like SOAP and REST.

An important effort in the search for standardization comes from the Open Cloud Manifesto[21], which is an initiative supported by hundreds of companies that aims to discuss a way to produce open standards for Cloud Computing. Their major doctrines are collaboration and coordination of efforts on the standardization, adoption of open standards wherever appropriate, and the development of standards based on customer requirements. Participants of the Open Cloud Manifesto, through the Cloud Computing Use Case group, produced an interesting white paper [OpenCloud 2011] highlighting

[20] http://cordis.europa.eu/fp7/ict/ssai/home_en.html

[21] http://www.opencloudmanifesto.org/

the requirements that need to be standardized in a cloud environment to ensure interoperability in the most typical scenarios of interaction – Use Cases – in Cloud Computing.

Another group involved with Cloud standards is the Open Grid Forum[22], which is intended to develop the specification of the Open Cloud Computing Interface (OCCI)[23]. The goal of OCCI is to provide an easily extendable RESTful interface for the management of Clouds. Originally, the OCCI was designed for IaaS setups, but their current specification [Metsch 2010] was extended to offer a generic scheme for the management of different Cloud services. Despite being currently adopted by Open-source projects only, considerable growth [Edmonds 2010] can be noted in the number of groups adopting OCCI as an alternative interface (OpenNebula, Eucalyptus, and Libvirt, for example) turning it into an interesting candidate for standard interfacing.

According to [Sheth and Ranabahu 2010], Cloud interoperability faces two types of heterogeneities: vertical heterogeneity and horizontal heterogeneity. The first type is concerned with interoperability within a single Cloud and may be addressed by a common middleware throughout the entire infrastructure. Authors highlight the Open Virtualization Format[24] (OVF), standard software used to manage VMs across a Cloud with heterogeneous infrastructure, which is already accepted by some hypervisors like VMWare ESX and VirtualBox[25]. Another solution for the vertical heterogeneity is Libvirt[26], which is a free software API that provides a common generic and stable layer to manage VMs. Libvirt offers an abstraction to programmers since it supports execution on different hypervisors like: Xen, KVM, VirtualBox, and VMWare ESX.

The second challenge, the horizontal heterogeneity, is more difficult to address because it is related to Clouds from different providers, and each provider has its own programmability level. Therefore, the key challenge is dealing with these differences. In this case, a high level of granularity in the modeling may help to address the problem.

In [Sheth and Ranabahu 2010], the same authors describe three interoperability aspects where semantic models would benefit Clouds. The first aspect is about **functional portability**, since semantic models may be used to define and share functional aspects of applications in a platform-agnostic way. The second point is the lack of support for **data migration** between Clouds. Regarding this concept, the authors see an opportunity for using data models such as a Resource Description Framework (RDF). Finally, the last aspect is that of **service description portability**. The growing number of different Cloud APIs makes the composition of cloud services difficult to achieve. Thus, the authors suggest using some scheme of annotation to enrich service descriptions and to permit easy combinations of cloud services.

---

[22] http://www.gridforum.org/

[23] http://occi-wg.org/about/specification/

[24] http://www.dmtf.org/vman

[25] http://www.virtualbox.org/

[26] http://www.libvirt.org/

5.3.2. Challenges in the resource management and resource control layers

Although being two distinct layers, Resource Management and Resource Control face related challenges, which are treated jointly in this section. According to the discussion in [Zhang et al. 2010], the first challenge regarding these layers is to meet their main objective: the **automated provisioning of resources**. Certainly, this problem is not a new one, and several computing areas have faced such type of problem since early operating systems. However, due to the heterogeneous and time-variant environment in a Cloud, the resource provisioning becomes a complex task, forcing the mediation system to respond with minimal turnaround time in order to maintain the developer's quality requirements. More details about this challenge will be treated in Section 5.4 under the discussion about resource allocation problems/solutions.

The project of **energy-efficient Clouds** is another research challenge in Cloud Computing. Energy management is a current worldwide design concept, since there is considerable concern about the reduction of carbon footprints and the need for climate balance. Regarding Clouds, this point is especially relevant as a result of the high demand for electricity to power and to cool the servers hosted on them [Buyya et al. 2010]. According to [Zhang et al. 2010], this challenge involves remodeling technologies employed at datacenters for energy reduction, which can be approached from different perspectives including new datacenter architecture, energy-aware VM management, and the use of energy-efficient network protocols and network devices.

As discussed in Section 5.2.4, operators implement Clouds using huge datacenters that are expensive to construct and require significant energy consumption. Also, their hierarchical architectures suffer from limitations like scalability, stiffness of address spaces, and node congestion in upper levels of the hierarchy. Such waste of resources has motivated research on the **remodeling of datacenter architectures**. Authors in [Verdi et al. 2010] surveyed this theme, highlighted the main problems on network topologies of state-of-the-art datacenters, and discussed literature solutions for these problems. Such solutions involve the development of new strategies for routing and forwarding, the creation of smart/programmable network devices, cross-layer interactions in the network protocols stack, and so forth. [Zhang et al. 2010] also suggest research on the creation of small datacenters, since they can offer a cheaper and more energy efficient consumer alternative to constructing a geographically distributed cloud, which is very adapted for time-critical services and interactive applications.

Regarding energy-aware VM management, savings may be achieved through many different strategies, one of which is the **consolidation** of VMs in servers according to current resource utilization. This makes use of **VM migration and cloning**, a typical technology available on virtualized datacenters. The strategy and the feature both have their own open research points.

VM migration and cloning provides an interesting technology to balance load over servers within a Cloud, provide fault tolerance to unpredictable errors, or reallocate applications before a programmed service interruption. But, although major industry hypervisors (like VMWare or Xen) have implemented VM migration techniques including "live" options, there remains some open problems to be investigated. These include cloning a VM into multiple replicas on different hosts [Lagar-Cavilla et al. 2009] and developing VM migration across wide-area networks [Cully et al. 2008]. Also, the use of migration introduces a network problem, since, after migration, VMs

require adaptation of the link layer forwarding. Some of the strategies for new datacenter architectures explained in [Verdi et al. 2010] offer solutions to this problem.

Server consolidation is a useful strategy for minimizing energy consumption while trying to maintain high usage of the servers' resources. The basic mechanism of this strategy saves the energy consolidating VMs (through migration) onto some servers and puts the idle servers into a standby state. The first open research point on this topic is the development of algorithms and heuristics for this type of problem, which can be mapped to bin-packing problems known to be NP-hard. [Zhang et al. 2010] also cites other related challenges to server consolidation: allocate VMs while maintaining their communication relationships and the prediction of VMs' resource usage for the anticipation of migrations.

Development of communication protocols in Clouds is another open field of research. Such protocols can act as a standard plane for resource management and control in the Cloud, allowing interoperability between devices. It is expected that such type of protocols can control and reserve resources of the different elements including processing servers, switches, routers, load balancers, and storage components present in the datacenter. One possible method of coping with this challenge is to use smart communication nodes with an open programming interface to create new services within the node. One example of this type of open nodes can be seen in the emerging Openflow-enabled switches [McKeown et al 2008].

S**ecurity and privacy** issues are the main hurdles considered by potential users of Clouds. Developers and clients expect that providers will ensure confidentiality in the data access and expect a complete auditing system for confirming that there are no security breaches. Despite being present in prior discussions on distributed systems, security remains as an open research field in Cloud Computing since it requires new solutions (or adaptations of older ones) to cope with specific aspects of the cloud-based computing model. Such type of solution must cover the multiple sub-systems operating on Clouds and must be integrated into all devices and software pieces to allow complete auditing of the Cloud. Also, as indicated by [Chen, Paxson, and Katz 2010], research on this topic must be made from the base, and be initiated by categorizing all the threats to which Clouds are subject.

5.4. Resource Allocation

This short course is focused on the resource management and resource control layers. These layers are related to each other and the main objective is to implement resource allocation mechanisms that provide an automated provisioning of resources. In this way, one of the main purposes of any cloud operator is to schedule developers' applications while aiming for the best utilization of available resources. A developer's application covers, aside from the software, some additional information about the application's needs and services as negotiated previously. Thus, the Cloud Computing operator faces the problem of selecting the most suitable physical and virtual resources to accommodate these applications under the previously negotiated requirements. This section presents the main general concepts and also some already available solutions for this problem.

First, general definitions on resource allocation are given and some terms that will be used throughout this section are clarified. Next, the main challenges related to

resource allocation are presented. More specifically, the fundamental aspects that are present while facing the resource allocation problem will be exhibited. Then some of the existing solutions will be presented, highlighting how its components fit under each one of the fundamental aspects.

5.4.1. Definitions for Resource Allocation

Resource allocation is a subject that has been addressed in many computing areas, such as operating systems, grid computing, and datacenter management. A Resource Allocation System (RAS) in Cloud Computing can be seen as any mechanism that aims to guarantee that the applications' requirements are attended to correctly by the provider's infrastructure. Along with this guarantee to the developer, resource allocation mechanisms should also consider the current status of each resource in the Cloud environment, in order to apply algorithms to better allocate physical and/or virtual resources to developers' applications, thus minimizing the operational cost of the cloud environment.

An important point when allocating resources for incoming requests is how the resources are modeled. There are many levels of abstraction of the services that a cloud can provide for developers, and many parameters that can be optimized during allocation. The modeling and description of the resources should consider at least these requirements in order for the RAS works properly.

Cloud resources can be seen as any resource (physical or virtual) that developers may request from the Cloud. For example, developers can have network requirements, such as bandwidth and delay, and computational requirements, such as CPU, memory and storage. Additionally, as explained in Section 5.3, other requirements are also feasible of Clouds, such as maximum delay between nodes, topology of the network of all nodes, jurisdiction issues, and interaction between different applications.

Generally, resources are located in a datacenter that is shared by multiple clients, and should be dynamically assigned and adjusted according to demand. It is important to note that the clients and developers may see those finite resources as unlimited and the tool that will make this possible is the RAS. The RAS should deal with these unpredictable requests in an elastic and transparent way. This elasticity should allow the dynamic use of physical resources, thus avoiding both the under-provisioning and over-provisioning of resources.

In this way, one may consider that the Cloud resources, resource modeling, and developer requirements are the inputs of a resource allocation system, as shown in Figure 5.4. Each one of these aspects will be described in more detail in the next section, since they are each closely related to the challenges on resource allocation.

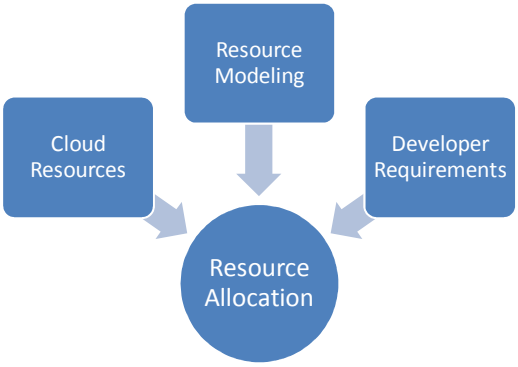

Figure 5.4. Resource Allocation System Inputs

5.4.2. Research Challenges in Resource Allocation

This section presents the main challenges faced while dealing with resource allocation under a Cloud. The challenges that a RAS should face are divided into four categories: resource modeling and description, resource offering and treatment, resource discovery and monitoring, and resource selection.

When developing a resource allocation system, one should think about how to describe the resources present in the Cloud. The development of a suitable **resource model and description** is the first challenge that an RAS must address. An RAS also faces the challenge of representing the applications requirements, called **resource offering and treatment**. Also, an automatic and dynamic RAS must be aware of the current status of the Cloud resources in real time. Thus, mechanisms for resource **discovery** and **monitoring** are an essential part of this system. These two mechanisms are also the inputs for **optimization** algorithms, since it is necessary to know the resources and their status in order to **elect** those that fulfill all the requirements.

Figure 5.5 shows how these challenges are related. First, the provider faces the problems grouped in the **Conception Phase**, where resources must be modeled according to the variety of services the Cloud will provide and the type of resources that it will offer. The next two challenges are faced in the scope of the **Operational Phase**. When requests for resources arrive, the RAS should initiate resource discovery to determine if there are indeed resources available in the Cloud to attend the request. Finally, if resources are available, the RAS may select and allocate them to serve the request. These challenges are described with more details next.

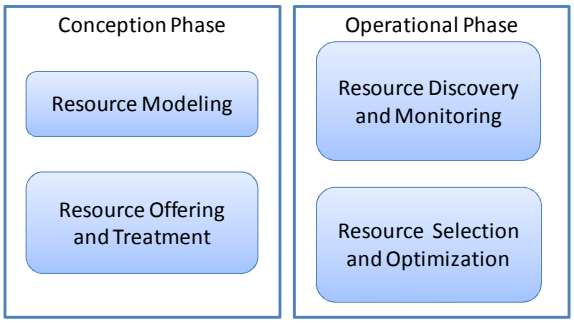

Figure 5.5. Relationship between resource allocation challenges

Resource Modeling and Description

Resource modeling and description defines how the Cloud functions and how it deals with infrastructural resources. This modeling is essential to all operations in the Cloud. Management, control, and optimization algorithms are all dependent on the resource model chosen by the operator. Also, the services that a Cloud will provide to developers depend on this concept.

Network and computing resources may be described by several existing specifications, such as Resource Description Framework (RDF)[27] and Network Description Language (NDL)[28]. However, with Clouds it is very important that resource modeling takes into account schemas for representing virtual resources, virtual networks, and virtual applications. According to [Houidi et al. 2009], virtual resources need to be described in terms of properties and functionalities much like as in the way services and devices/nodes are described in existing service architectures.

It is important to note that if a particular model is very detailed (description at a very low level) it means that it's details are relevant and should not be disregarded, which in turn makes the optimization problem much more difficult to handle and solve. For example, if we consider that a request is composed of all the physical specifications of each machine, we will have to deal with the little details that in another situation could be disregarded or considered irrelevant. In this way, matching the request with available resources is rendered more difficult due to the peculiarities of each request, and achieving a generic solution is considerably more complicated. On the other hand, more details can give more flexibility and allow for a better usage of resources. This concept is called the **granularity of the resources description**.

Solutions for Resource Modeling and Description

There are many solutions that are focused on solving the challenges of resource description. Next, some of these solutions are surveyed and analyzed, with the objective being to exemplify the approaches that are being used to accomplish these challenges.

Note that resource modeling and description is associated with a greater challenge in Cloud Computing: interoperability. The author in [Nelson 2009] describes the "hazy scenario", in which large Cloud providers use proprietary systems, thus hampering the potential for integration between different Clouds. In this way, the main goal of interoperability in Clouds is to enable the seamless flow of data across Clouds from different providers and also between applications inside the same Cloud [Dillon and Chang 2010]. As explained in Section 5.3.1, solutions such as intermediary layers, standardization, and open APIs, are interesting options for interoperability.

In the search for standardization, the Elastra[29] has an interesting initiative. They have published their **Elastic Modeling Languages** that are a set of three languages defined using the Web Ontology Language (OWL): the Elastic Deployment Modeling Language (EDML), the Elastic Computing Modeling Language (ECML), and the Elastic Management Modeling Language (EMML). This set of languages expresses the

[27] http://www.w3.org/RDF/

[28] https://noc.sara.nl/nrg/ndl/

[29] www.elastra.com/technology/languages

application requirements, state of cloud resources, and configurations required to enable automated application deployment and management [Charlton 2009].

The EDML provides a language to enable the description of common servers in datacenters such as web servers, application servers, databases, load balancers, etc. The language allows detailed descriptions of these servers ranging from the CPU architecture and storage technology, to the server configuration and software packages installed. The ECML is used to create a structural description for applications. It uses EDML descriptions to instantiate components and also introduces abstract units called connectors which are used to mediate communication between components. Furthermore, ECML can describe high-level requirements such as scalability, security, backup and others. Finally, the EMML permits information retrieval on ECML and EDML elements, since it maintains information regarding the current state and history of these items in the Cloud.

A different solution is provided by the **Application Descriptor Language** (ADL), which is used to describe web applications in a specific grid operating system, called AppLogic[30]. The ADL modeling considers that resources may be real physical resources, or virtual ones sharing the same hardware. Basically, there are three types of descriptors: component, assembly, and package, as will be addressed below.

The **component descriptor** describes specific network applications (such as an HTTP server or database server) located on a single host. The attributes of the component descriptor are all optional and are used to describe valid properties of the component. "*.migrateable*" is an example of a component attribute that indicates that the component can be migrated to another host. Beyond attributes, entities may also have sub-entities that define component characteristics. For example, "*resource*" defines component's requirements such as CPU, memory, and network bandwidth. The **assembly descriptor** is used to describe a new component composed of several interconnected components, which can in turn be used to describe the structure of an application. Finally, the **package descriptor** is similar to a "table of contents", in which re-usable components may be shared amongst all applications. In the same way as the other descriptors, the package descriptor also has their own attributes and sub-entities.

Service Modeling Language (SML)[31] and **Service Modeling Language Interchange Format (SML-IF)**[32] are a pair of XML-based specifications created by leading information technology companies that have defined a set of XML and XML Schema documents, as well as several rules for the treatment of these documents.

The SML specification defines the main concepts of the model, and is a set of interrelated XML documents that contain information about the different aspects of an IT service as well as the constraints that each aspect must satisfy to function properly. Constraints are captured in two ways. The first constraint is established through XML Schema documents, and the second is through the use of rule documents. SML uses common query languages such as XPath for rule construction. Once a model is defined,

[30] http://doc.3tera.com/AppLogic24/AppLogic.html

[31] http://www.w3.org/TR/2007/WD-sml-20070806/

[32] http://www.w3.org/TR/sml-if/

one can then establish its validity. This involves checking whether all documents satisfy the XML Schema constraints and the rule-based constraints.

The SML-IF specification describes a packaging format for exchanging SML-based models. It defines how all SML documents can be packaged into a single document without any loss of fidelity. The SML-IF specification enables interoperability between different implementations of SML. This is a result of the fact that all conforming implementations must be able to produce and consume SML-IF documents. Note that, in contrast to an SML document, the SML-IF document is not required to be complete. It is possible to have a scenario in which a provider sends an incomplete model to another provider so that they can be integrated together to produce a complete model.

CML (Common Model Library) is a set of rules that uses SML to encode models and SML-IF to communicate between them. CML defines syntax and semantic for expressions for the purpose of managing entities in the IT environment. In summary, CML is a set of XML schemas that describes IT entities. This collection includes infrastructure (e.g. application servers), logical entities (e.g. software license, and IT roles), and relationships [Houidi et al. 2009].

Another XML-based solution is presented in [Houidi et al. 2009], where virtual resources are now considered. In this work, authors consider a network virtualization environment scenario where infrastructure providers should describe their virtual resources and services in order to be able to offer them. In this solution, properties and functionalities of the virtual resources are modeled in a UML class diagram. Authors suggest that their diagram can be represented in a XML-based document, but does not show an example document or a XML-Schema for such type of documents.

Resource Offering and Treatment

Once the resources are modeled, the Cloud provider may offer and handle them through an **interface**. At a lower level, the RAS should **handle** the Cloud resources and, at a higher level, address applications' requirements.

The resource offering **interface** should provide some way for developers to clearly describe the requirements of their application. These requirements may be represented by some form of a Service Level Agreement (SLA) that must be ensured by the provider through continuous monitoring of QoS, due to the dynamic nature of the Cloud [Patel et al. 2009][Yfoulis and Gounaris 2009].

Handling resources requires the implementation of solutions to control all resources available in the Cloud. Such control and management solutions would offer a complete set of signaling protocols to set up hypervisors, routers, and switches. Currently, to delegate these control tasks, each Cloud provider implements their own solution that descends, in general, from datacenter control solutions. They also employ virtual datacenter solutions developed by vendors like VMWare or Citrix[33]. However, in the future, new signaling protocols can be developed for integrated reservation of resources in the Cloud environment.

[33] http://www.citrix.com/

It is important to highlight that resource modeling is not necessarily dependent on the way in which resources are offered to developers. For example, the Cloud provider could model each resource individually, like independent items in a fine-grained scale (such as GHz of CPU or GB of memory), but can offer them like a coupled collection of items or a bundle, such as classes of VM (high memory and high processor types).

As previously noted, in addition to traditional network requirements and computational requirements, new requirements for developers may be present in Clouds. An example of such a requirement is the **topology of the network**, which may be defined by developers. Developers are able to configure nodes' relationships and communication restrictions (down and uplinks, for example). **Jurisdiction** is another example. It is related to where (physically) applications and their data must be stored and handled. Due to some restrictions – such as copyright laws – developers may solicit to store information in chosen places, such as specific countries or continents. The **node proximity** may be another restriction indicating that there should be a maximum (or minimum) physical distance (or delay value) between nodes or locations. Please note that it may directly impact other requirements, such as topology.

Resource Discovery and Monitoring

Resource discovery may be described basically as the task in which the provider should find appropriate resources (suitable candidates) in order to comply with incoming developers requests. Also, more specific questions, like "How should one discover resources with (physical) proximity in a Cloud?", or "How does one cause minimal impact upon network traffic responsibility?" can be seen as a resource discovery responsibility and are not trivial to be answered, given the dynamic nature of the Cloud environment.

Considering that one of the key features of Cloud Computing is the capability of acquiring and releasing resources on-demand [Zhang et al. 2010], **resource monitoring** should be continuous. This is because, in addition to serving as an information provider to handle new requests, it should also be informed of resources' current status to make an informed decision of a possible reallocation, i.e., it must assist resource usage optimization. The timing and the quantity of messages is a relevant issue to be emphasized, because the network bandwidth should be used rationally. In this way, a careful analysis should be done to find a suitable trade-off between the amount of messages and the frequency of information refreshing.

The monitoring may be passive or active. It is considered **passive** when there is an entity (or a set of entities) that collects information from nodes continuously, i.e., the nodes are passive in relation to this entity. The entity may send messages to nodes asking for information or simply may retrieve information when necessary. When the monitoring is **active**, nodes are autonomous and may decide when to send state information to some central entity. Clouds also make use of both alternatives simultaneously to improve the monitoring solution.

A simple implementation of a resource discovery service is the discovery framework used in an advertisement process as described in [Houidi et al. 2009]. Such a framework is proposed for a scenario based on a network virtualization environment, but it can be easily adapted for other scenarios. Such a framework can be used by brokers to discover and match available resources, and typically is composed of

distributed repositories, which are responsible for storing information about and descriptions of physical and virtual resources.

Resource Selection and Optimization

After acquiring information about available resources in the Cloud (during the discovery phase), a set of appropriate candidates is highlighted. The resource selection mechanism elects the candidate solution that fulfills all requirements and optimizes the usage of the infrastructure. In virtual networks, for example, the essence of resource selection mechanisms is to find the best mapping between the virtual graph and the substrate graph. This should achieve the best load balancing in the substrate network resources, with respect to the capacity constrains [Houidi et al. 2008]. By this example, it is clear that selecting solutions from a set of available ones is not a trivial task due to the dynamicity of the scenario and all the different requirements that must be contemplated by the provider.

The resource selection may be done using an optimization algorithm. Many optimization strategies may be used, from simple and well-known techniques such as simple heuristics with thresholds or linear programming, to newer ones, such as Lyapunov Optimization [Urgaonkar et al. 2010]. Moreover, traditional artificial intelligence algorithms – ant colony and game theory [Teng and Magouls 2010], for example – are also viable for Clouds.

One way to classify resource selection strategies is related to the moment when optimization techniques are applied: a priori and a posteriori.

In the case of a priori techniques, the first allocation solution is already an optimal solution. To achieve this goal, the optimization strategy should consider all variables that influence the allocation process. For example, consider that VM instances should be allocated in the Cloud; the optimization strategy should determine the problem, present a solution (or a set of possibilities) that satisfy all constraints, and attempt the goals (such as the minimization of resource reallocations) in an optimal manner.

In the case of a posteriori techniques, once an initial resource allocation is made, which can be a suboptimal solution, the RAS should manage its resources in a continuous way. If necessary, decisions, such as to add more memory or storage capacity to a VM, reallocate an already allocated application, or reallocate resources for other applications, should be taken in order to optimize the system utilization or to comply with a developer's requirements.

Since the resource utilization and provisioning are dynamic (i.e., in addition to the resources being requested and then allocated, they can also be de-allocated), it is more interesting that the a posteriori optimization strategies reach an optimal allocation first, and are able to optimize the old requests and readjust them according to new demand. In this case, the optimization strategy may also fit with the definition of a priori and dynamic classification.

Furthermore, optimization strategies may also be applied to improve specific aspects in the Cloud such as energy saving or to cope with polices for security and fault-tolerance.

5.4.3. Solutions

This section presents existing solutions that address the resource allocation problem and its challenges. These solutions come from several sources including state-of-art papers, existing techniques coming from other technologies (such as grid computing), and others. The reader should notice that the treatment given here is not extensive and can be seen only as a survey of information about these solutions. Also, one must keep in mind that the solutions presented here accomplish many of the described challenges.

The resource allocation problem is not new. There are many solutions focusing on datacenters ([Urgaonkar et al. 2010], [Chen et al. 2010], [Kong et al. 2010]) and grid computing ([Murphy and Goasguen 2010], [Xhafa and Abraham, 2010]). Some of the new strategies for resource allocation employ these legacy solutions adapting them to be used in Cloud Computing environments ([Lee and Zomaya 2010], [Beloglazov and Buyya 2010], [Buyya et al. 2010]). Also, as will be described here, an approach is to treat the resource allocation on Cloud Computing based on a network virtualization viewpoint.

As many of the solutions found try to optimize energy consumption, we will discuss it here as a major section. As an example, an RAS can achieve optimal energy consumption by observing the energy consumption in real time and by making decisions about reallocation, such as VM migration to liberate underused servers.

The second major subsection is related to the time-response management of applications in the Cloud. In this section, the selected papers focus on optimizing the time that tasks take to be executed. This is important in order to fulfill the developers' requirements and the overall performance of Clouds.

Resource allocation based on virtual network environments is also reviewed. General concepts from this area can be applied to the field of Cloud Computing by considering that a virtual network can be associated with a developer application that is composed of several servers and databases as well as the network that interconnects them. More details on virtual networks can be found in [Chowdhury 2009].

Energy management

As mentioned previously in Section 5.3, energy management is an important current worldwide design criterion especially in Cloud Computing. According to [Buyya et al. 2010], lowering the energy usage is a challenging and complex issue because computing applications and data are growing so quickly that increasingly larger servers and disks are needed to process them fast enough within smaller time periods. However, energy saving can be achieved through many strategies, such as consolidating VMs according to current resource utilization, turning off or setting CPU hibernation when servers become idle, moving workloads or VMs, and so on.

Authors in [Urgaonkar et al. 2010] use queuing information to implement a resource allocation and energy management mechanism in virtualized datacenters. It assumes that to attend each application there is a certain number of VMs hosted by the same number of servers. The servers' queuing information is used to make online control decisions, according to changes in the application's workload. The energy saving is done by switching idle servers into inactive mode when their workload is low, and turning them on again when workload increases. To achieve this goal, the authors

present the Datacenter Control Algorithm (DCA), which addresses client's requests according to the following three steps: Admission Control, Routing, and Resource Allocation.

Each application has its own Admission Control module, which receives new requests and is responsible for verifying if new requests can be admitted using a threshold-based solution. After that, admitted requests can be routed between available servers, prioritizing active servers with the smallest requests queue. Finally, in a fully distributed way, servers allocate resources for each VM in order to minimize time to attend to their requests.

Another energy management system for virtualized datacenters is proposed in [Beloglazov and Buyya 2010] and [Buyya et al. 2010]. The papers consider an environment where developers request VMs in the Cloud. In this scenario, providers have to deal with the tradeoff between energy and performance. Specifically, they must obtain minimum energy consumption while meeting developer requirements. Reduction of energy consumption is performed by switching off idle servers.

The proposed system architecture is composed of the Dispatcher, the Global Manager, and the Local Manager. Clients requesting VMs send their requests to the Dispatcher that distributes requests between VMs. The Local Manager **monitors** the resources, specifically the usage of VMs and the state of servers. This information is sent to the Global Manager that is responsible for processing information from Local Managers and for **resource selection**. These decisions are divided into two parts: the admission of new requests and the **optimization** of current allocation. The admission control is made using some type of rank algorithm that allocates VMs to a server that will provide the smallest increase in power consumption. The optimization of VM allocation can be implemented using threshold-based heuristics.

[Chen et al. 2010] proposes an integrated solution that couples the management of IT resources and cooling infrastructure to reduce the energy consumption of both systems. While continually monitoring the performance and the temperature of groups of servers, the system reallocates VMs in order to shutdown servers with low usage to save power. The system also manages air conditioners in the room according to IT resources needs.

One interesting related problem is the application instance placement, defined in [Karve et al. 2006] as the problem of placing application instances on a given set of server machines to adjust the amount of resources available to applications in response to their resource demands. In [Steinder et al. 2008], authors developed an application placement controller that consolidates servers to achieve power savings without an unacceptable loss of application performance. To achieve this goal, two controllers work together: one controller **monitors** application performance information that will be used by the other one to place application instances, according to constraints such as CPU or memory capacity. The placement is managed by the starting and stopping of individual instances of applications.

Response Time Management

Response time management is an important concept related to how the resource allocation mechanism will treat the response time of the tasks to achieve good system performance. Ordinarily, this concept is used in Grid Computing. Makespan is a term

used to describe the amount of time taken from the initiation of the first task to the time the last task completes its execution [Lee and Zomaya 2010]. However, due to the dynamicity of grid resources usage, it is hard to guarantee that the makespan of a given schedule is reached. In this way, to ensure that tasks will finish their execution within the estimated completion times (in the presence of resource performance fluctuation) is seen as a major performance issue in Grid Computing [Lee and Zomaya 2010].

Since Grid applications may also be supported in Cloud, we also need to treat this challenge, because at first the Cloud is not concerned with execution time of applications, or in grid language, the response time of tasks. At same time, we can use the advantages of Cloud Computing to improve solutions. In [Lee and Zomaya 2010], the authors make use of Cloud resources to increase the reliability of task completion. Cloud resources are resorted to because authors have considered that they are often tightly coupled, homogeneous and reliable. Min-min, Max-min and XSufferage are static scheduling mechanisms, in which the rescheduling process is activated when a task completes its execution later than its estimated time. Authors propose the RC² algorithm, in which the scheduling mechanism is initiated together with the subsequent tasks that caused the delay in task completion. The RC² algorithm initially uses one of the static scheduling mechanisms mentioned previously to allocate tasks only on grid resources; and whether there is a significant amount of delay in tasks completion, waiting tasks would possibly be rescheduled onto Cloud resources.

Authors in [Kong et al 2010] focus on online and dynamic task scheduling in a virtualized datacenter scenario with the main goal being to achieve a trade-off between availability and performance. The proposed scheme makes use of the two Fuzzy predictors: one to calculate the availability level of the resources, and the other one to treat the load-balance requirement. The intersection set between availability and load-balance terms is then calculated. If the intersection is null, the algorithm gives priority to satisfying the availability requirement. Otherwise, the task with the minimum expected response time is selected.

In [Karve et al 2006], authors developed an application placement mechanism that dynamically configures the number and the location of application instances. Such a mechanism is based on three steps: the first step sorts nodes and applications according to the density and assigns them according to satisfaction constraints. The second step derives a new placement from the previous one in an attempt to minimize the number of placement changes. Ultimately, the third step is invoked to achieve a better load balance, considering the solution proposed by the second step.

Network Virtualization

The network virtualization area is a relatively new field in network research that is related to the research of Cloud Computing. Authors in [Chowdhury et al. 2009] describe the main problem of this area as the allocation of virtual networks over a substrate network. Resource allocation in Cloud Computing can also be seen with this point of view: the main goal is to allocate developers' virtual network requests on a substrate network according to some constraints and while aiming to obtain a clever configuration of the virtual network and underlying resources.

The **resource modeling** approach described next is based on [Chowdhury et al. 2009]. The Cloud operator infrastructure can be seen as a graph, called the substrate

graph. Virtual networks' requests are also modeled as graphs. Bandwidth, storage, CPU or memory requirements and restrictions can be modeled as values that each link or node has associated with them. Among other constraints that can be considered, there is the physical location of the virtual node, as modeled by [Chowdhury et al. 2009]. This can be used to model access delay requirements of the arriving virtual network nodes. Different versions of this model can also be created, for example, the model can neglect storage requirements when the storage is over-provisioned.

With a virtual network description, an allocation system can assign virtual networks. Such a virtual network assignment can be seen as a simple mapping from the nodes of the virtual network request to the substrate nodes, and from the links of the virtual network request to the substrate paths. Virtual network assignments consume the underlying resources onto which they are mapped.

Given this model, an algorithm that allocates virtual networks, which **selects and optimizes resources**, should consider the constraints of the problem (CPU, memory, location or bandwidth limits) and should have an objective function to optimize based on the algorithm objectives. In [Haider et al. 2009], authors describe possible objective functions to be optimized related to maximizing the revenue of the service provider while minimizing link and nodes stress, etc. It also provides a survey with some heuristic techniques used when allocating virtual networks. The authors also describe some basic strategies based on allocating the virtual nodes on substrate nodes first and then allocating the virtual links to substrate paths as not satisfactory.

In [Chowdhury et al. 2009], the objective function of the virtual network allocation algorithm is closely related to the cost and revenue of the provider. Authors reduce their problem to a mixed integer programming problem and then relax integer constraints to solve the problem with a polynomial time algorithm. Then an approximate solution to the initial problem is obtained based on this relaxed polynomial algorithm. There are two different algorithms for the approximation of this solution; one is the D-ViNE, a deterministic algorithm, and the other one is the R-ViNE, or randomized algorithm.

Another approach, taken by [Zhu and Ammar 2006] is based on an objective function that attempts to homogenize the distribution of hosts over the substrate network as much as possible by minimizing the maximum number of virtual nodes per substrate nodes and virtual links per substrate links. Thus, all resources from the substrate network are used in a balanced way. In order to achieve that, static and dynamic algorithms are presented.

[Haider et al. 2009] describes how some virtual network applications, mainly virtual network test-beds, implement their resource management and allocation. PlanetLab[34] allocation is generally chosen by the user, but some automatic mechanisms have been proposed in [Ricci et al. 2006]. Also, SWORD [Albrecht et al 2008], a service running in PlanetLab, can act on **resource discovery** and find a virtual network mapping with the lowest penalty function. Alternately, Emulab[35] uses a facility which is based on simulated annealing in order to **optimize the allocation**. Other techniques are

[34] PlanetLab, http://www.planet-lab.org/

[35] Emulab - Network Emulation Testbed, http://www.emulab.net/

also implemented to provide better performance to the user, such as selecting nodes that are in close proximity to already allocated neighbors with higher probability.

Furthermore, there are still several research challenges in the virtual network allocation area. Authors in [Haider et al. 2009] highlighted the development of dynamic allocation algorithms based on the dynamic behavior of the bandwidth consumption of the virtual networks. This could be implemented using game theory or adaptive feedback control theory. Also, the development of other heuristic approaches to overcome this NP-hard problem is essential. They suggest an investigation into the tradeoffs of the many characteristics of the already existing algorithms, as well as some studies on how static algorithms could deal with dynamic scenarios.

5.5. Open-source Mediation Systems

Building a Cloud often involves employing a Mediation System for the management of resources. The decision about which solution to use is very difficult since the range of solutions for Cloud management is broad and each solution has specific characteristics adapted towards specific scenarios. Such an ecosystem of solutions is also populated by proprietary and open-source solutions; however the scope of this section is restricted to the latter type.

Previous papers including ([Rimal et al. 2009], [Endo et al. 2010], [Cordeiro et al. 2010]) have surveyed open source Mediation Systems for Clouds and have compared features like architecture, virtualization platforms, and service type. Table 5.1 summarizes some of these solutions. Firstly, one can note the predominance of IaaS Mediation Systems and, secondly, that the open-source mediation systems support both proprietary and open-source hypervisors.

Table 5.1. Comparative between open-source Mediation Systems (adapted from [Endo et al. 2010]).

Solutions	Service	Main Characteristics	Infrastructure
XCP	IaaS	Automatic management	Xen
Nimbus	IaaS	Turns legacy clusters into Clouds	Xen and KVM
OpenNebula	IaaS	Policy-driven resource allocation	Xen, KVM, and VMWare
Eucalyptus	IaaS	Hierarchical Architecture	Xen and KVM
TPlatform	PaaS	Focus on web mining	TFS and MapReduce
Apache VCL	SaaS	Applications on the Internet	VMware
Enomaly	IaaS	Focus on small clouds	Xen, KVM, and VirtualBox

Particularly, this section follows the analysis made by [Cordeiro et al. 2010], which surveyed and compared in detail three open-source Mediation Systems. However, the paper showed that the Xen Cloud Platform (XCP) is intended to manage a virtualized infrastructure and do not offers a solution for resources negotiation, i.e., the XCP does not implement a Cloud Mediation System with all layers defined in the archetypal as shown in Figure 5.3. Therefore, this section will highlight the main characteristics and architectures of two solutions previously presented in this paper: Eucalyptus and OpenNebula, and will also discuss another solution: Nimbus. As well, this section follows the analysis made by [Sempolinski and Thain 2010].

5.5.1. Eucalyptus

Since its conception, Eucalyptus was designed to provide services compatible with Amazon EC2, i.e., it was designed for the rental of virtual machines in the Cloud. It is

interesting to note that all communications in Eucalyptus are made through Web Services standards. Also, Eucalyptus is intended to bridge the gap between open-source software for Clouds and enterprise solutions. This is clearly shown through the design of their external interface, which is fully compatible with the Amazon EC2 interface.

Architecture

The Eucalyptus architecture foresees two different user classes: administrators and developers. This once again reflects a business-oriented model view. The former are the users responsible for managing the entire Cloud who have access to all features of Eucalyptus. The latter are the developers that can request and make use of VM instances directly from Eucalyptus, without the need for administrators' intervention. In this scenario, developers can also be clients when using their VMs for own use, or can configure their VMs and provide a service to other clients.

As presented in [Nurmi et al. 2008], Eucalyptus has four components, as shown in Figure 5.6. VMs hosted on a given server are directly managed by a Node Controller (NC) running on the same server. This option is interesting because the NC offers a common layer that abstracts the hypervisor solution employed on each server. NCs are grouped in clusters managed by the Cluster Controller (CC). This CC gathers state information from each NC, schedules developers' requests to individual NCs, and manages the configuration of public and private networks. At the top of the architecture is the Cloud Controller (CLC), which processes requests and makes VM placement decisions. The Walrus is a data storage service compatible with Amazon's S3, which is responsible for manipulating VM templates and delivering them to NCs when a developer wishes to instantiate a VM. A clear advantage of the Eucalyptus architecture is it is a natural fit for existing enterprise processing environments, as a result of its hierarchical design.

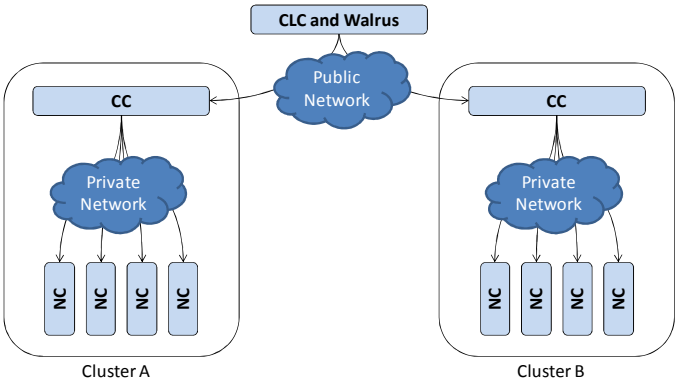

Figure 5.6. Components of the Eucalyptus Architecture [Nurmi et al. 2009]

Negotiation issues

Eucalyptus was developed entirely inside a Service-Oriented paradigm. Therefore, messages exchanged between Eucalyptus components as well as messages sent by developers requiring a resource are described through WSDL interfaces. In other words, Eucalyptus employs Web Services for communication on all layers of the Mediation System. The interfaces associated with the control of the internal components will be discussed later.

One must note that the negotiation or external interface is offered by the CLC and is responsible by translating developers' requests into operations for the internal components. Currently, for the sake of compatibility with Amazon's EC2, Eucalyptus supports both SOAP-based and HTTP Query-based interfaces [Nurmi et al. 2009]. The main operations are runInstances and terminateInstances, where an instance is a VM. Also for maintaining compatibility with Amazon EC2, developers on Eucalyptus may only instantiate VMs from classes previously configured. Such classes specify the main aspects of the target VMs including number of cores, memory size, disk size, operating system etc.

Resource Management and Control

For internal communication and control, Eucalyptus also uses Web Service interfaces. The NC implements a number of operations to manage VMs hosted on the server. Important operations supported are: runInstance, terminateInstance, and describeResource. The first two are used to send commands to start and to stop a specific VM, respectively. The last one reports current characteristics like memory and disk usage of the server (the physical node). The CCs also provide a WSDL interface; however, unlike the interface provided by the NC, each operation can be used to manage groups of VM instances as indicated by the operations' names: runInstances, terminateInstances, and describeResources.

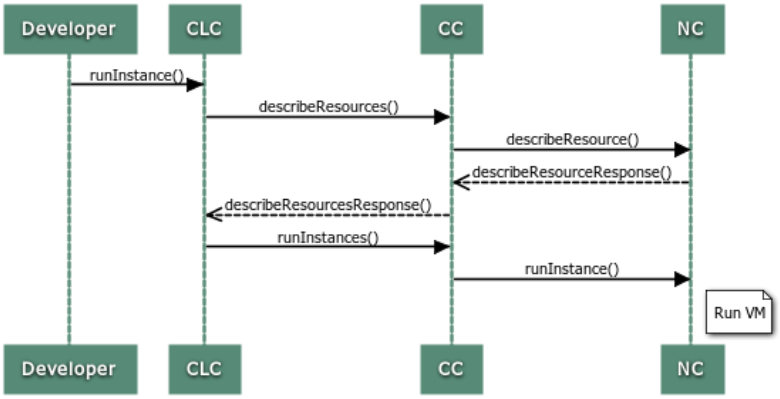

Figure 5.7. Sequence Diagram of messages to run a VM

Resource management in Eucalyptus is related to the allocation and deallocation of VMs on the servers. Such a process follows a peculiar algorithm whose message passing is shown in Figure 5.7. The resource allocation process is triggered by a developer request and finalized when the VM is actually running on a node [Nurmi et al. 2008]. When the negotiation interface on Eucalyptus receives a runInstances request from one developer, the CLC determines which CC can support the VM by querying CCs through the describeResources operation and then by choosing the first CC that responds with enough free resources. In the same way, when a CC receives a runInstances request from the CLC it queries each NC through the describeResource message and chooses the first NC that has enough free resources.

5.5.2. OpenNebula

OpenNebula is a flexible tool that orchestrates storage, network and virtualization technologies to enable the dynamic placement of services on distributed infrastructures. It has been designed to be modular to permit its integration into as many different

hypervisors and environments as possible. Such flexibility attracted a growing number of organizations including the European Space Astronomy Centre[36], the European Organization for Nuclear Research (CERN)[37], and Telefónica[38]. The entire list of organizations using OpenNebula or contributing in its development can be found on [OpenNebula Project 2011a].

Architecture

The physical infrastructure assumed for OpenNebula adopts a classical cluster-like (master and slave nodes) architecture with a front-end server and a set of servers for hosting VMs. Also, it requires at least one physical network joining all the cluster nodes with the front-end. The front-end server executes the main OpenNebula processes while the other servers on the cluster are hypervisor-enabled hosts providing resources to allocated VMs.

One distinct aspect of OpenNebula is that the software was developed following a layered architecture adequate extension, which is depicted in Figure 5.8. The OpenNebula architecture has three layers: Tools, Core and Drivers. The upper layer of the architecture is called Tools and contains all the modules for the manipulation of OpenNebula. This layer encompasses, for example, the interface's modules that are used by administrators and developers to interact with OpenNebula. Moreover, beyond negotiation layer issues, the Tools layer implements the Scheduler module, which makes runtime decisions about where VMs must be allocated. Other tools to provide extra functionalities for the OpenNebula can be implemented on this layer using the OpenNebula Cloud API which is based on an XML-RPC interface.

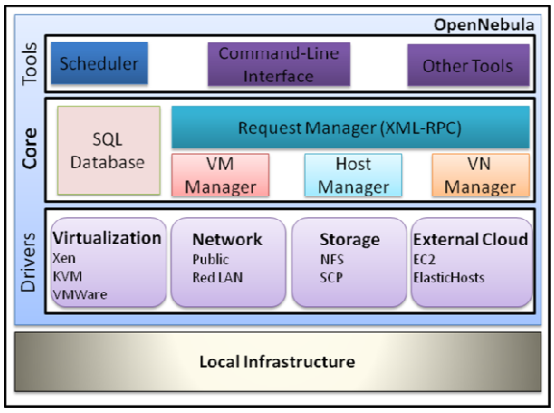

Figure 5.8. OpenNebula Architecture (adapted from [OpenNebula Project 2011b])

The Core layer has the components responsible for handling VM requests and controlling resources. The main component of this layer is the Request Manager, which handles developers' requests through an XML-RPC interface that calls the internal managers according to the invoked method. Physical servers and VMs are managed and monitored by the Host Manager and the VM Manager, respectively. The Virtual Network Manager (VN Manager) manages virtual networks by keeping track of IP and

[36] http://www.esa.int/esaMI/ESAC/index.html

[37] http://public.web.cern.ch/public/

[38] http://www.tid.es/en/Pages/default.aspx

MAC addresses and their association with VMs, acting like a DHCP for the virtual networks. Finally, the SQL Database stores internal data structures.

The third layer has modules called Drivers, which supports different underlying platforms. These Drivers run on separated processes, generally in the servers responsible for host VMs, and they communicate with the Core module through a simple text messaging protocol. There are drivers to deal with file transfers implemented by network protocols including NFS and SSH. Also, there are hypervisor-dependent drivers for managing VMs running on each server. Finally, there are drivers to request services from external Clouds like Amazon EC2 or ElasticHosts.

Negotiation issues

Similarly to Eucalyptus, OpenNebula works with administrative and client accounts, the latter being for Cloud developers. Administrators access OpenNebula as a private Cloud through the Command Line Interface (CLI) that allows for the manipulation of the overall infrastructure through intuitive commands. Developers launch and control their leased virtual infrastructure (computing, network, and storage) using Web Service-based interfaces. OpenNebula implements three different interfaces: one compatible with the EC2 Query API from Amazon, another compatible with the OCCI from the Open Grid Forum, and a third compatible with a subset of calls of the VMWare's vCloud API.

Resource Management and Control

The Scheduler is the OpenNebula module responsible for making VM placement decisions. By having access to all VM allocation requests received by OpenNebula, the Scheduler can keep track of VM allocations, make decisions about the allocation of incoming VM requests, and send appropriate deployment or enforcement commands to the Core. The default scheduler on OpenNebula provides a scheduling policy that places VMs according to a ranking algorithm, which relies on performance data from both the running VMs and physical servers. Moreover, following their modular principle, OpenNebula allows extension of their architecture with a new different scheduler.

Algorithm 1: Scheduling algortithm.

```
Inputs: requirements, rank, hostsList;
Outputs: selectedHost;
1  Begin
2  for each host in hostsList do
3      if (host meets requirements) then
4              candidates.new(host);
5      end
6  end
7  sorted = sortByRank(candidates, rank);
8  selectedHost = sorted(1);
9  end
```

Figure 5.9. Default scheduling algorithm (from [Cordeiro et al. 2010])

The basic operation of the default scheduling algorithm used in OpenNebula is shown in Figure 5.9. The algorithm requires two inputs that are sent in the VM request: requirements and rank. The former consists of a set of simple rules indicating minimum resource requirements (such as CPU or Memory) and the latter consists of a policy for resource allocation. The main objective of this policy is to provide an index for the ranking of candidate servers. It can be made in the form of an arithmetic expression encompassing server aspects including quantity of free CPU, number of VMs hosted,

temperature, etc. Thus, when a VM request arrives, the Scheduler, based on the requirements, filters those servers from hostsList that do not meet the minimum requirements requested and creates a new list of candidate servers (candidates). After that, the remaining hosts are sorted using the rank policy. Finally, the first host in the list sorted is chosen as the selectedHost which will be used to allocate the VM.

Despite being a simple algorithm for resource management, a careful adjustment of the inputs (Requirements and Rank) can permit different placement schemes. For example, using FREECPU as the rank indicates that servers with lesser loads must be prioritized. Such ranking will cause a spread of VMs among the servers of the cluster, which can be interesting when allocating VMs that demand high processing capacity. Other example is to rank candidates using RUNNING_VMS. Such a method can be used to allocate VMs in as few servers as possible, enforcing a server consolidation strategy.

5.5.3. Nimbus

Nimbus is an extensible open-source mediation system whose main purpose is to turn a grid of infrastructure resources into an IaaS Cloud for scientific computing. Nimbus enables providers to build private or community Clouds and enables developers to lease remote resources by deploying VMs, including the simple and automatic lease of large infrastructures. Nimbus is highly configurable and extensible, allowing providers to customize its solutions.

In the next section we describe the software components of Nimbus' architecture in more detail. Then we illustrate how the negotiation with developers works. Last there is some information about Nimbus' resource management.

Architecture

Similarly to OpenNebula, Nimbus runs over a cluster-like infrastructure with one server acting as a front-end for developers to access the Cloud and the other servers acting as VM hosts. Each hosting server must have one of either Xen or KVM hypervisors in order for the Nimbus' Virtual Machine Manager (VMM software, or workspace-control) to work.

The front-end server must also have Cumulus installed, which is the Nimbus storage solution for VM image repository. It comprises an open source implementation of the Amazon's S3 REST API[39]. Through Cumulus, developers are able to access and manage their storage quotas. Cumulus is designed for scalability and allows providers to configure multiple storage implementations.

Figure 5.10 shows the Nimbus software architecture. The most important component in this architecture is the workspace service, whose main role is to manage the virtual machines on the pool. Also, the workspace service is responsible for receiving developers' requests through different interfaces. Currently, Nimbus implements two interfaces: the WSRF-based (Web Services Resource Framework[40]) interface and the Amazon EC2-compatible interface. Thus, according to the interface,

[39] http://docs.amazonwebservices.com/AmazonS3/latest/dev/

[40] http://www.globus.org/wsrf/

developers must use proper client software for deploying, interacting with, querying, saving and terminating sessions with virtual machines.

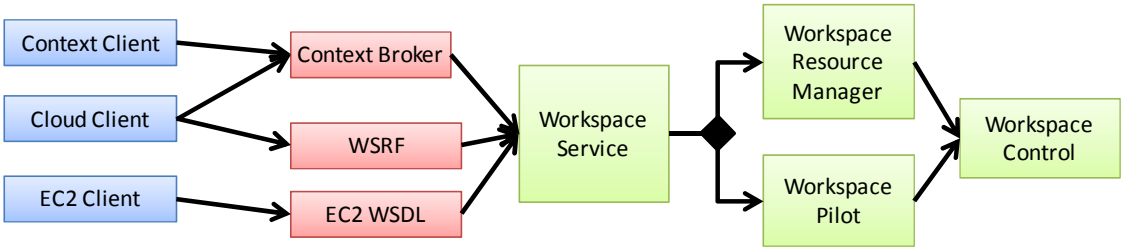

Figure 5.10. Nimbus software components [adapted from Keahey 2009]

The context broker allows developers to coordinate large virtual clusters to launch automatically and repeatedly. It allows for the creation of virtual networks, as opposed to launching a set of unconnected virtual machines like most VM-on-demand services.

The workspace control (or VMM) is the component of Nimbus that must be installed on each server in order to perform basic management and control operations of the VMs. Finally, the Workspace Resource Manager and Workspace Pilot components provide automatic management of VMs including deciding which servers are appropriate to run each VM.

Negotiation issues

To deploy their applications, developers can use a command line application provided by Nimbus which will communicate with the Cloud through WSRF (a specification that provides stateful operations to Web Services). Based on the client application, developers communicate with WSRF services which allows for data associated with resources to be stored and retrieved.

Another possible method of accessing the cloud infrastructure is using the Amazon EC2 interface. Such an interface will allow developers to use EC2 clients when dealing with Nimbus-based Clouds. Nimbus has a partial implementation of EC2's WSDL, and the EC2 Query API. Many EC2 operations are already supported [Nimbus Project 2011].

The developer also has the option of launching any number of VMs simultaneously (called a cluster of VMs). The type of VMs, their images, how many of each type should be launched, and the layout of the VMs can be specified through an XML file. The cluster deployment is implemented by the context broker and using information coming from the context agent that should boot on each VM node.

Resource Management and Control

The resource management in Nimbus is implemented on two components: the Workspace Resource Manager and the Workspace Pilot. The former implements a strategy where the developer has direct control of a pool of VMM nodes. The latter allows developers to submit regular jobs on the infrastructure and makes use of already configured batch schedulers like TORQUE[41]. One interesting feature for the scientific

[41] http://www.clusterresources.com/pages/products/torque-resource-manager.php

community offered by Nimbus is that, due to their grid-based nature, users must inform the amount of time for which the VM is running.

The Nimbus resource control is implemented mainly using the Workspace Control. It should be installed on each server and permits operations such as starting VMs, pausing VMs, stopping VMs, connecting VMs securely to the network, etc. The workspace control uses Libvirt over a Xen/KVM hypervisor. The front-end server uses SSH to access the hosting servers.

5.5.4. Comparisons

In this section, we compare the open source systems for Cloud management as presented earlier using various criteria. The main aspects and features studied here were chosen because of their believed fundamental importance on the decision of which solution an operator should use. At the end of this section, some useful scenarios for each mediation system are discussed based on the differences identified herein.

Through the analysis of OpenNebula, Nimbus and Eucalyptus, we can see that these mediation systems are focused on implementing an actual Cloud by offering virtualized elements over a pool of physical resources. They implement, in fact, an IaaS that can be used for private or public purposes. Although Eucalyptus can already handle two different virtualization platforms, OpenNebula distinguishes itself by having a native ability – provided by its modular architecture – to deal with different underlying virtualization platforms supported by the use of its Drivers, which makes it much more flexible. Alternately, Nimbus employs the Libvirt as a common layer for communication with different hypervisors.

Architecture

The architecture of a mediation system for Cloud management indicates how that platform works and how it was built. It is also an important decision parameter when choosing an environment in which to implement a Cloud. Though a completely centered system is, of course, easier to implement and manage, it can suffer from performance and availability problems. Conversely, hierarchical approaches are more reliable when considering control and management.

Eucalyptus has a truly hierarchical approach. The Cloud is divided into different clusters, each with its own cluster controller that is responsible for all nodes under that cluster. On the upper level of the hierarchy, the Cloud controller is responsible for all the cluster controllers.

OpenNebula and Nimbus work with a front-end server and several hypervisor-enabled servers, creating a hierarchy of two levels only. In terms of software architecture, OpenNebula offers a completely modular and layered architecture making it easily integrated with different technologies. It has three main layers: Tools, Core and Drivers. Among the Tools, there is the Scheduler, responsible for assigning new VMs to hosts. Nimbus also has clear modular software architecture to provide support of both VMs allocation and regular job scheduling.

Negotiation issues

This aspect concerns the problem of how Cloud clients are going to access their leased resources. Public Cloud systems should provide this interface insomuch that the clients

can operate their resources transparently. Note that this is a concept that hopefully will be standardized in the future.

On OpenNebula, Nimbus and Eucalyptus, an external user is able to use their resources through a Web Service interface provided by the mediation system. It is interesting to note that all these are also compatible with Amazon interfaces, primarily Amazon's EC2. A further advantage of OpenNebula is that it can easily be extended to implement other interfaces, as well as Nimbus.

Resource Management and Control

The placement algorithm of the platform is important to guarantee a clever and perhaps optimal use of the resources. Poorly used resources always result in inefficiency. The publically available Eucalyptus version has a simple algorithm that seems to pay little attention to VM placement on the Cloud.

Nimbus offers the Workspace Resource Manager and the Workspace Pilot that are responsible for deploying nodes and jobs on the Cloud, respectively. The main advantage of these components is that their underlying implementations are based on well-accepted solutions for scheduling in grids.

OpenNebula brings an interesting approach for VM placement based on its defined Rank value, which can be used to implement useful policies. As such, it may be more attractive to developers who see resource usage and optimization as a critical requirement.

Scenarios

With the main differences now identified, we are able to discuss where and when each Cloud mediation system should be used.

In the first scenario, suppose we want to offer a public Cloud service to a community (more specifically, a simple IaaS). In this case, Eucalyptus will fit very well, however it is not able to provide some important services like VM migration. Moreover, its allocation algorithms are simple and may not be efficient in some cases.

Now, consider that we have a pool of resources, possibly with heterogeneous virtualization platforms, and we want to create a private/hybrid Cloud over it. OpenNebula or Nimbus can easily support this. Choosing one or the other will depend on the intended audience of the Cloud. OpenNebula is more adapted to VM-oriented scenarios, i.e., scenarios where developers want VMs to try and test new software in a sandbox environment. On the other hand, Nimbus is more driven towards scientific computing developers and the sharing of excess time between them [Sempolinski and Thain 2010].

5.6. Final Considerations

As was highlighted in this short course, Cloud Computing is a good solution for users interested in using or developing services provided on demand. We differentiated two types of users that providers interact with: clients and developers.

Beyond this separation, we divided open research challenges into three areas: negotiation, resource management, and resource control. This grouping was implemented with the goal of separating challenges according to the individualities of

each layer defined in the Archetypal Cloud Mediation System (presented in Section 5.2.4). It is interesting because this Archetypal system was conceived considering that resource management is the main service of any Cloud Computing provider. In this way, we directed the entire paper towards resource allocation aspects.

In this section we will finalize the short course, review the challenges addressed previously and make some considerations about the future prospects of Cloud Computing.

5.6.1. Review of the matter

Section 5.2 described important definitions regarding Cloud Computing, including references that answered the question: "What is Cloud Computing?". Several authors made their contributions, however we emphasize that there is no consensus about a unique definition, but rather a collection of functionalities that characterize the Cloud.

Beyond the questions about the definition of Cloud, there are many open research problems that were addressed in Section 5.3. Divided into three layers (negotiation, resource management, and resource control), these problems describe aspects related to interface and resource allocation.

Since the negotiation layer is responsible for the interface between the Cloud and the developers, here we find problems associated with interaction mechanisms with human users, as well issues surrounding interoperability. The two other layers are strongly associated with achieving their main goal: the automated provisioning of resources. In this way, problems of these layers are also associated. Energy management is an important aspect in Cloud Computing due to the high need for electricity to power and to cool the computing infrastructure. VM consolidation was described as one strategy that makes use of VM migration and cloning to save energy. Other aspect related to VM consolidation is the development of protocols which should be used to manage and control all types of Cloud resources, which would allow for interoperability between them.

Considering these concepts, we defined a Resource Allocation System (RAS) as a resource allocation mechanism responsible for controlling Cloud resources and for attending to applications' requirements. To do this in an optimized way, RAS needs to know the current status of each resource. In this short course, we described challenges related to this issue as divided into four categories (resource modeling and description, resource offering and treatment, resource discovery and monitoring, and resource selection), and grouped into two phases (conception and operational). Essentially, we may say that the challenge is all about to how properly describe resources for the purpose of offering them to users when posed with a request. Then, the challenge is to discover the resources on the Cloud and identify the optimal resource that satisfies all the requirements of the request.

5.6.2. Future Trends

As we could notice in this short course, Cloud Computing is still in its maturation phase and there are many challenges to be solved or improved. In this Section, we present some considerations about future perspectives of Cloud Computing.

We think that resource modeling and description, and resource offering and treatment areas will be concerned with answers of questions like “How is the ideal level of resource modeling abstraction?” or “How different Cloud providers can consent about service offering, considering scenarios with heterogeneous physical infrastructure?”. Here, we believe that standardization efforts will keep as a trend with the main goal of creating compatible and integrated Cloud providers.

About resource discovery and monitoring, we see the need of automating discovery and allocation processes, and make the monitoring process more active, being able to re-allocate resources according to demand or to the current status of the Cloud (in order to optimize resource usage). In this way, we could think in an adaptative Cloud that is able to self-manage its resources and self-adapt its configurations according to different circumstances.

Other important aspect treated in resource selection and optimization challenges is the question about energy management. We guess that the effort in this area will become increasingly concrete due to importance of current questions about energy efficiency. Together with this, still with the goal of improving resource usage, we can also think about Clouds in which underused and non-dedicated resources can be clustered and availed in an opportunistic way.

References

Albrecht, J., Oppenheimer, D., Vahdat, A., and Patterson, D. A. (2008) “Design and Implementation Trade-offs for Wide-area Resource Discovery”. In ACM Transactions on Internet Technology, Vol. 8 , Issue 4, Article 18, September.

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M. (2009) “Above the Clouds: A Berkeley View of Cloud Computing”, Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A. (2003) “Xen and the art of virtualization”. In: 19th ACM symposium on Operating systems principles, New York, NY, USA.

Beloglazov, A. and Buyya, R. (2010) “Energy Efficient Resource Management in Virtualized Cloud Data Centers”. In IEEE/ACM International Conference on Cluster, Cloud and Grid Computing.

Broberg, J. and Buyya, R. and Tari, Z. (2009) “MetaCDN: Harnessing Storage Clouds for high performance content delivery”. In Journal of Network and Computer Applications, pp. 1012-1022, Elsevier.

Buyya R., Beloglazov, A., Abawajy, J. (2010) “Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges”. In International Conference on Parallel and Distributed Processing Techniques and Applications.

Buyya, R. et al. (2008) “Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities”. In 10th IEEE international conference on high performance computing and communications.

Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I. (2009) "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility". In: Future Generation Computer Systems, Elsevier B. V.

Charlton, S. (2009) "Model-Driven Design and Operations for the Cloud". In Workshop on Best Practices in Cloud Computing: Designing for the Cloud.

Chen, Y., Gmach, D., Hyser, C., Wang, Z., Bash, C., Hoover, C., Singhal, S. (2010) "Integrated Management of Application Performance, Power and Cooling in Data Centers", In IEEE Network Operations and Management Symposium (NOMS), 2010, pp. 615-622, Osaka, Japan, April.

Chen, Y., Paxson, V., Katz, R. H. (2010) "What's New About Cloud Computing Security?", Tech. Rep. UCB/EECS-2010-5, EECS Department, University of California, Berkeley.

Chowdhury, N. M. M. K., Rahman, M. R., and Boutaba, R. (2009) "Virtual Network Embedding with Coordinated Node and Link Mapping," IEEE INFOCOM.

Chowdhury, N.M. M. K. and Boutaba, R. (2010) "A survey of network virtualization". In Computer Networks, Vol. 54, issue 5, pp. 862-876. Elsevier.

Cordeiro, T., Damalio, D., Pereira, N., Endo, P., Palhares, A., Gonçalves, G., Sadok, D., Kelner, J., Melander, B., Souza, V., Mångs, J. (2010) "Open Source Cloud Computing Platforms". In: 9th International Conference on Grid and Cloud Computing, Nanjang, China, pp.366-371.

Cully, B., Lefebvre, G., Meyer, D., Feeley, M., Hutchinson, N. e Warfield, A. (2008) "Remus: High availability via asyncronous virtual machine replication". In: 5th USENIX Symposium on Networked Systems Design and Implementation, April.

Dean, J. and Ghemawat, S. (2004). Mapreduce: simplified data processing on large clusters. In OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, pages 10–10, Berkeley, CA, USA. USENIX Association.

Dillon, T., Wu, C., and Chang, E. (2010) "Cloud Computing: Issues and Challenges", In IEEE International Conference on Advanced Information Networking and Applications, pp. 27-33.

Edmonds, A. (2010) "OCCI Implementations". Open Cloud Computing Interface Blog. Available at: http://occi-wg.org/2010/11/17/occi-implementations/.

Endo, P. T., Gonçalves, G. E., Kelner, J., Sadok, D. (2010) "A Survey on Open-source Cloud Computing Solutions". Workshop em Clouds, Grids e Aplicações, Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos.

Erl, T. (2009) "SOA design patterns", Prentice Hall, Boston.

Ghemawat, S., Gobioff, H., and Leung, S. (2003) "The Google file system". In 19th Symposium on Operating Systems Principles, pages 29–43, Lake George, New York.

Greenberg, A., Hamilton, J., Maltz, D. A., and Patel, P. (2008). "The cost of a cloud: research problems in data center networks". SIGCOMM Comput. Commun. Rev. 39, n. 1, pp. 68-73.

Haider A., Potter, R., Nakao, A. (2009) "Challenges in Resource Allocation in Network Virtualization". In 20th ITC Specialist Seminar, Hoi An, Vietnam, May.

Houidi, I., Louati, W., and Zeghlache, D. (2008) "A distributed virtual network mapping algorithm". In Proceedings of IEEE International Conference on Communications, pp. 5634–5640.

Houidi, I., Louati, W., Zeghlache, D., and Baucke, S. (2009) "Virtual Resource Description and Clustering for Virtual Network Discovery" In Proceedings of IEEE ICC Workshop on the Network of the Future.

Karve, A., Kimbrel, T., Pacifici, G., Spreitzer, M., Steinder, M., Sviridenko, M., and Tantawi, A. (2006) "Dynamic Placement for Clustered Web Application". In International World Wide Web Conference Committee.

Keahey, K. (2009) "Nimbus: Open Source Infrastructure-as-a-Service Cloud Computing Software". In Workshop on adapting applications and computing services to multi-core and virtualization, CERN, Switzerland.

Khan, S., Maciejewsk, A., and Siegel, H. (2009) "Robust CDN Replica Placement Techniques", In IEEE International Symposium on Parallel&Distributed Processing.

Khoshafian, S. (2007) "Service Oriented Enterprises". Auerbach Publications.

Kong, X., Lin, C., Jiang, Y., Yan, W and Chu, X. (2010) "Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction". In Journal of Network and Computer Applications, Elsevier.

Kong, X., Lin, C., Jiang, Y., Yan, W and Chu, X. (2010) "Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction". In Journal of Network and Computer Applications, Elsevier.

Lagar-Cavilla, H. A.; Whitney, J. A.; Scannell, A. M.; Patchin, P.; Rumble, S. M.; Lara, E.; Brudno, M.; Satyanarayanan, M. (2009) "SnowFlock: rapid virtual machine cloning for cloud computing". In Fourth ACM European Conference on Computer Systems.

Lee, Y. and Zomaya, A. (2010) "Rescheduling for reliable job completion with the support of clouds". In Future Generation Computer Systems, vol. 26, pp. 1192-1199, Elsevier.

Llorente, I. M. (2010) "Key Research Challenges in Cloud Computing". Presented at 3rd EU-Japan Symposium on Future Internet and New Generation Networks.

Lo Presti, F et atl. (2007) "Distributed Dynamic Replica Placement and Request Redirection in Content Delivery Networks". In 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems.

Marshall, P., Keahey, K., and Freeman, T. (2010) "Elastic Site: Using Clouds to Elastically Extend Site Resources", 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 43-52, Australia, June.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. and Turner, J. (2008) "OpenFlow: enabling innovation in campus networks", ACM SIGCOMM Computer Communication Review.

Mell, P. and Grance, T. (2009). "The NIST Definition of Cloud Computing". National Institute of Standards and Technology, Information Technology Laboratory.

Metsch, T., Edmonds, A., Nyrén, R. (2010) "Open Cloud Computing Interface – Core". Open Grid Forum, OCCI-WG, Specification Document. Available at: http://forge.gridforum.org/sf/go/doc16161?nav=1.

Murphy, M, and Goasguen, S. (2010) "Virtual Organization Clusters: Self-provisioned clouds on the grid". In Future Generation Computer Systems.

Nelson, M. (2009) "Building a Open Cloud", Science, pp. 1656-1657, vol, 324, American Association for the Advancement of Science.

Nimbus Project. (2011) "Nimbus 2.7 Admin Guide", Available at: http://www.nimbusproject.org/docs/2.7/admin/index.html. Visited on March, 2011.

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L. and Zagorodnov, D. (2009) "The Eucalyptus Open-Source Cloud Computing System". In: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid.

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., and Zagorodnov, D. (2008) "Eucalyptus: A Technical Report on an Elastic Utility Computing Architecture Linking Your Programs to Useful Systems". University of California, Santa Barbara, October.

OpenCloud. (2011) "Cloud Computing Use Cases Whitepaper - Version 4.0". http://opencloudmanifesto.org/Cloud_Computing_Use_Cases_Whitepaper-4_0.pdf.

OpenNebula Project. (2011a) "OpenNebula 2.0 Documentation". Available at: http:// www.opennebula.org/documentation:documentation. Visited on February, 2011.

OpenNebula Project. (2011b) "OpenNebula.org – The Open Source Toolkit for Cloud Computing", Available at: http://www.opennebula.org/. Visited on February, 2011.

Padala, P. (2010) Automated management of virtualized data centers. Ph.D. Thesis, Univ. Michigan.

Patel, P., Ranabahu, A. and Sheth, A. (2009) "Service Level Agreement in Cloud Computing". In Cloud Workshop at OOPSLA.

Ramaswamy, L., Liu, L., and Iyengar, A. (2005) "Cache Clouds: Cooperative Caching of Dynamic Documents in Edge Networks". In Proceedings of the 25th IEEE International Conference on Distributed Computing System.

Ricci, R., Oppenheimer, D., Lepreau, J. and Vahdat, A. (2006) "Lessons from Resource Allocators for Large-Scale Multiuser Testbeds". In ACM SIGOPS Operating Systems Review, Vol. 40 , Issue 1, pp. 25-32, January.

Rimal, B., Choi, E., and Lumbi, I. (2009) "A Taxonomy and Survey of Cloud Computing Systems". In International Joint Conference on INC, IMS and IDC.

Sempolinski, P., and Thain, D. (2010) "A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus". In IEEE International Conference on Cloud Computing Technology and Science, pp. 417-426.

Sheth, A and Ranabahu, A. (2010) "Semantic Modeling for Cloud Computing, Part I". In IEEE Computer Society - Semantics & Services.

Sheth, R. (2009) “What we talk about when we talk about cloud computing”. Available at: http://googleenterprise.blogspot.com/2009/04/what-we-talk-about-when-we-talk-about.html.

Steinder, M., Whalley, I., Hanson, J.E., Kephart, J.O., Coordinated Management of Power Usage and Runtime Performance. 2008.

Teng, F. and Magouls, F. “A New Game Theoretical Resource Allocation Algorithm for Cloud Computing”, pp. 321-330, Advances in Grid and Pervasive Computing, 2010.

Urgaonkar, R., Kozat, U., Igarashi, K, and Neely, M. (2010) “Dynamic Resource Allocation and Power Management in Virtualized Data Centers”, In IEEE Network Operations and Management Symposium (NOMS).

Vaquero, L., Merino, L., Caceres, J., and Lindner, M. (2008) “A Break in the Clouds: Towards a Cloud Definition”, vol. 39, pp. 50–55, January.

Veras, M. (2009) “Datacenter: Componente Central da Infraestrutura de TI”. Brasport Livros e Multimídia, Rio de Janeiro.

Verdi, F. L., Rothenberg, C. E., Pasquini, R., and Magalhães, M. (2010). “Novas arquiteturas de data center para cloud computing”. In SBRC 2010 - Minicursos.

Vouk, M.A. (2008) “Cloud Computing – Issues, Research and Implementations”. Journal of Computing and Information Technology, pages 235–246. University of Zagreb, Croatia.

Wang, L., Tao, J., Kunze, M., Rattu, D. and Castellanos, A. (2008) “The Cumulus Project: Build a Scientific Cloud for a Data Center”. In: Workshop on Cloud Computing and Its Applications, Chicago, USA

Xhafa, F and Abraham, A. (2010) “Computational models and heuristics methods for Grid scheduling problems”. In Future Generation Computer Systems.

Yfoulis, C.A. and Gounaris, A. (2009) “Honoring SLAs on cloud computing services: A control perspective”. In Proceedings of EUCA/IEEE European Control Conference.

Youseff, L., Butrico, M., and Silva, D. (2008) “Toward a unified ontology of cloud computing”. In Grid Computing Environments Workshop (GCE08).

Zhang, Q., Cheng, L., and Boutaba, R. (2010) “Cloud computing: state-of-the-art and research challenges”. Journal of Internet Service Applications, Springer, pp. 7-18.

Zhu, Y. and Ammar, M. (2006) “Algorithms for assigning substrate network resources to virtual network components”. In Proceedings of IEEE INFOCOM.