UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

EDUARDO ROLOFF

Viability and Performance of High-Performance Computing in the Cloud

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Prof. Dr. Philippe Olivier Alexandre Navaux
Advisor

Prof. Dr. Alexandre da Silva Carissimi
Coadvisor

Porto Alegre, September 2013

CIP – CATALOGING-IN-PUBLICATION

Roloff, Eduardo

Viability and Performance of High-Performance Computing in the Cloud / Eduardo Roloff. – Porto Alegre: PPGC da UFRGS, 2013.

94 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2013. Advisor: Philippe Olivier Alexandre Navaux; Coadvisor: Alexandre da Silva Carissimi.

1. Cloud Computing. 2. High-Performance Computing. 3. Performance Evaluation. 4. Economic Evaluation. I. Navaux, Philippe Olivier Alexandre. II. Carissimi, Alexandre da Silva. III. Título.

“Do not go where the path may lead,
go instead where there is no path...
and leave a trail.”
— RALPH WALDO EMERSON

AGRADECIMENTOS

Este trabalho é dedicado a minha esposa Michele, por todo seu apoio durante esta jornada da Pós-Graduação. Agradeço também aos meus pais, Susana e Heinz, e ao meu irmão Daniel por todo o suporte dado por eles e pela paciência durante os períodos de ausência que se fizeram necessários durante o desenvolvimento de todo esse trabaho.

Agradeço também a empresa GetNet pelo apoio que me foi dado para a realização do mestrado, em especial ao Cristian e ao Givanildo que me apoiaram na realização do curso mesmo antes do mesmo ter sido iniciado.

A meus amigos do GPPD, obrigado pelas dicas e discussões, e também pelos momentos de descontração que vivemos juntos durante todo o mestrado. Menção a ser feita aos colegas Matthias, Marco e Francis que tiveram envolvimento direto com este trabalho, tanto em seu desenvolvimento como também no auxílio na revisão do mesmo. Um agradecimento especial ao meu orientador Professor Navaux e ao meu co-orientador Professor Carissimi, tanto pela oportunidade que me propriciaram como pelo constante orientação durante o mestrado, além do grande esforço despendido durante a revisão desse trabalho.

Aos demais familiares e amigos, muito obrigado por tudo.

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

API	Application Programming Interface
BEP	Break Even Point
BLAST	Basic Local Alignment Search Tool
CFD	Computational Fluid Dynamics
CRM	Customer Relationship Management
DLL	Dynamic Link Library
DNS	Domain Name System
EC2	Elastic Compute Cloud
ECU	Elastic Compute Unit
FTP	File Transfer Protocol
FPU	Floating Point Unit
GAE	Google App Engine
HPC	High-Performance Computing
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IT	Information Technology
LAN	Local Area Network
MPI	Message Passing Interface
NAS	Numerical Aerodynamics Simulation
NCBI	National Center for Biotechnology Information
NIST	National Institute of Standards and Technology
NPB	NAS Parallel Benchmarks
NUMA	Non-Uniform Memory Access
OS	Operating System

PaaS	Platform as a Service
QoS	Quality of Service
REST	Representational State Transfer
S3	Simple Storage Service
SaaS	Software as a Service
SGI	Silicon Graphics Incorporated
SLA	Service Level Agreement
SSH	Secure Shell
SOAP	Simple Object Access Protocol
URL	Uniform Resource Locator
VM	Virtual Machine
VPN	Virtual Private Network
VS	Visual Studio
XaaS	Everything as a Service

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Cloud computing is a new paradigm, where computational resources are offered as services. In this context, the user does not need to buy infrastructure, the resources can be rented from a provider and used for a period of time. Furthermore the user can easily allocate as many resources as needed, and deallocate them as well, in a totally elastic environment. The resources need to be paid only for the effective usage time. On the other hand, High-Performance Computing (HPC) requires a large amount of computational power. To acquire systems capable for HPC, large financial investments are necessary. Apart from the initial investment, the user must pay the maintenance costs, and has only limited computational resources. To overcome these issues, this thesis aims to evaluate the cloud computing paradigm as a candidate environment for HPC. We analyze the efforts and challenges for porting and deploy HPC applications to the cloud. We evaluate if this computing model can provide sufficient capacities for running HPC applications, and compare its cost efficiency to traditional HPC systems, such as clusters. The cloud computing paradigm was analyzed to identify which models have the potential to be used for HPC purposes. The identified models were then evaluated using major cloud providers, Microsoft Windows Azure, Amazon EC2 and Rackspace and compare them to a traditional HPC system. We analyzed the capabilities to create HPC environments, and evaluated their performance. For the evaluation of the cost efficiency, we developed an economic model. The results show that all the evaluated providers have the capability to create HPC environments. In terms of performance, there are some cases where cloud providers present a better performance than the traditional system. From the cost perspective, the cloud presents an interesting alternative due to the pay-per-use model. Summarizing the results, this dissertation shows that cloud computing can be used as a realistic alternative for HPC environments.

Keywords: Cloud Computing, High-Performance Computing, Performance Evaluation, Economic Evaluation.

Viabilidade e Desempenho de Processamento de Alto Desempenho na Nuvem

RESUMO

Computação em nuvem é um novo paradigma, onde recursos computacionais são disponibilizados como serviços. Neste cenário, o usuário não tem a necessidade de adquirir infraestrutura, ele pode alugar os recursos de um provedor e usá-los durante um certo período de tempo. Além disso, o usuário pode facilmente alocar e desalocar quantos recursos ele desejar, num ambiente totalmente elástico. O usuário só é cobrado pelo efetivo uso que for feito dos recursos alocados, isso significa que ele somente pagará pelo que for utilizado. Por outro lado, usuários de processamento de alto desempenho (PAD) tem a necessidade de utilizar grande poder computacional como uma ferramenta de trabalho. Para se ter acesso a estes recursos, são necessários investimentos financeiros adequados para aquisição de sistemas para PAD. Mas, neste caso, duas situações podem incorrer em problemas. O usuário necessita ter acesso aos recursos financeiros totais para adquirir e manter um sistema para PAD, e esses recusros são limitados. O propósito dessa dissertação é avaliar se o paradigma de computação em nuvem é um ambiente viável para PAD, verificando se este modelo de computação tem a capaciodade de prover acesso a ambientes que podem ser utilizados para a execução de aplicações de alto desempenho, e também, se o custo benefício apresentado é melhor do que o de sistemas tradicionais. Para isso, todo o modelo de computação em nuvem foi avaliado para se identificar quais partes dele tem o potencial para ser usado para PAD. Os componentes identificados foram avaliados utilizando-se proeminentes provedores de computação em nuvem. Foram analisadas as capacidades de criação de ambientes de PAD, e tais ambientes tiveram seu desempenho analisado através da utilização de técnicas tradicionais. Para a avaliação do custo benefício, foi criado e aplicado um modelo de custo. Os resultados mostraram que todos os provedores analisados possuem a capacidade de criação de ambientes de PAD. Em termos de desempenho, houveram alguns casos em que os provedores de computação em nuvem foram melhores do que um sistema tradicional. Na perspectiva de custo, a nuvem apresenta uma alternativa bastante interessante devido ao seu modelo de cobrança de acordo com o uso. Como conclusão dessa dissertação, foi mostrado que a computação em nuvem pode ser utilizada como uma alternativa real para ambientes de PAD.

Palavras-chave: Computação em nuvem, Processamento de alto desempenho, Avaliação de desempenho, Avaliação econômica.

1 INTRODUCTION

Cloud computing is a model of services that provides alternative solutions to the costs of acquisition and maintenance of computing environments. There are two main reasons. First, the cloud provides an elastic environment, which means that users can access, and acquire and release computing resources according to the current demand. Second, due to the pay-per-use billing model, the user is charged only for the time span in which the resources were used. Since the first appearances of studies related to cloud computing, it is enjoying a high level of interest from the scientific computing community (FOSTER et al., 2008) (BUYYA et al., 2009) (ARMBRUST et al., 2009).

Unlike traditional systems, there are no upfront investments in infrastructure and software licenses. Due to the elasticity and the pay-per-use billing model, the total cost of maintenance can be close to zero when resources are not in use. Also, the costs of facilities, hardware depreciation, energy consumption and cooling are eliminated or, at least, greatly reduced. Moreover the amount of available resources are virtually unlimited. Therefore, the motivation of moving applications into the cloud is to reduce these costs while increasing scalability and availability.

High-performance computing (HPC) demands a lot of computational power. To get access to this computational power, investments are needed. The big issue is the dimension of the investments. If the infrastructure is oversized, financial resources were misallocated, because the computing resources are being underutilized. On the other hand, if the infrastructure has been dimensioned only for the current demand any future increases in demand will not be met, requiring new investments.

Moreover, there are other costs involved in maintaining the infrastructure running. Costs for maintenance, energy consumption, depreciation of the hardware, cooling and facilities need to be considered. These costs are related to both HPC environments and for any general purpose computing environment.

The characteristics of cloud computing make the model an interesting alternative for the usage as HPC environments. Furthermore, cloud computing can be more environmentally friendly compared to traditional computing. Because the physical resources are not accessible by the users, the cloud platform can migrate the workload from one server to other to achieve virtual machines consolidation, with this it is possible to power off the unused machines.

On the other hand, cloud computing create concerns related to performance, reliability and security. All the security aspects change in a cloud environment. For example, in traditional systems, it is possible to control where the data is physically located and access and modification of that data can be controlled. In the cloud, the data is located according to the rules of the cloud provider and the user can usually not control access to it. Also the processing and network performance, the most relevant aspects for HPC

applications, can not be controlled by the user. These concerns must be taken into account when considering the use of cloud computing as an environment for running HPC applications.

1.1 Motivation

The capabilities offered by the cloud computing model attract attention of HPC users. The possibility to use a large set of machines without the need to purchase the machines has a tremendous advantage, especially for users without access to maintenance personnel, in temporary projects, or in applications without constant usage of the machines.

On the other hand, the focus of cloud computing services is normally on web servers and content delivery and not on HPC. Previous research in this area focuses on porting specific applications to clouds (FAN et al., 2012). More detailed performance analysis is restricted mostly to the Amazon EC2 provider and for private clouds (GUPTA; MILOJICIC, 2011). The economic evaluation was not performed using public cloud providers and, normally, cost models are defined but not evaluated in real usage (MASTELIC et al., 2012). An evaluation of cloud computing as a platform for HPC using the metrics proposed in this thesis was not performed yet. For this reason, there is a lack of research targeting the development and execution of HPC applications in the cloud.

HPC users are end-users or programmers, the end-users use standard applications without the need to customize them. The programmers adjust and customize scientific applications, to ensure that they execute more efficiently. While the first user group is interested just in the use of an application, the second group works with different applications and need a more flexible environment. This thesis covers both HPC user groups and they are treated as the user.

Due to the absence of comprehensive studies related to the use of cloud computing as an environment for HPC, this situation was identified as a research opportunity. During this study we performed an analysis of several aspects of cloud computing to evaluate if the model, or a subset of it, is feasible as an environment for HPC. The analysis was validated using three major cloud providers: Microsoft, Amazon and Rackspace.

1.2 Objectives

The main purpose of this thesis is to evaluate if cloud computing can be used as an environment for HPC, thus this work is focused in the evaluation of these aspects.

The major points of this evaluation are:

- Model analysis: cloud computing is an extensible model, with several possible configurations, in the model analysis the service and implementations models of cloud computing are analyzed to determine which combination of them are more suitable for our proposal.

- Technical evaluation: the technical evaluation consists of the evaluation if the chosen cloud providers can be used to deploy HPC environments and if they present acceptable performance. This is performed by the analysis of the deployment capabilities of each providers and also, with the performance evaluation of the same providers.

- Economic viability: the cost to use cloud computing as the environment for HPC is also evaluated in an economic perspective, to determine if the cloud providers present a competitive alternative to traditional machines.

With the analysis of these three aspects, it is possible to determine the feasibility of cloud computing for HPC.

1.3 Text Organization

In Chapter 2, the basic concepts of cloud computing and the architecture of a cloud are presented. Also, the main cloud providers are introduced with an explanation about their main features and benefits. The main issues for cloud adoption are addressed, and the state of the art of cloud computing for HPC are analyzed as well.

Chapter 3 presents the proposal of this thesis. An analysis is conducted to decide which service and implementation models are feasible for the proposed scenario of this study. The architecture used for the evaluation of the model is defined. The proposed cost-benefit model is presented.

In Chapter 4, the results of the evaluation criteria are shown. The deployment procedures are analyzed using major cloud providers. The performance and economic viability are also evaluated with the same providers. All results are compared to a traditional computer system.

Chapter 5 closes this thesis with the presentation of conclusions and the contributions made by this study. Also, ideas for future work are presented.

2 CLOUD COMPUTING

Cloud computing is a new paradigm of computing. It was developed with the combination and evolution of distributed computing and virtualization, with strong contributions from grid and parallel computing (BUYYA et al., 2009).

There are many efforts to provide a definition of cloud computing, such as the work of Grid Computing and Distributed Systems Laboratory (BUYYA et al., 2009) and the initiative from Berkeley University (ARMBRUST et al., 2009). In 2011, NIST has published its definition that consolidates several studies and became widely adopted. It is also the definition adopted as baseline in this thesis.

The purpose of this chapter is to present the basic concepts of cloud computing. The most outstanding cloud providers are introduced and their main characteristics are presented. A discussion about issues related to cloud computing adoption and the related work are presented.

The remainder of this chapter is structured as follows. The basic concepts, according with the NIST definition, of Cloud Computing are explained in Section 2.1. The architecture of a cloud platform is presented in Section 2.2. Then Section 2.3 shows the main cloud providers and analyze their characteristics. The open issues of cloud computing are explained in Section 2.4. In section 2.5 the related work regarding of the use of cloud computing for HPC is presented. Section 2.6 presents general discussions and considerations.

2.1 Basic Concepts

Cloud computing has two main actors that are defined as the user and the provider. The user is defined as the consumer and can be a single person or an entire organization. The provider is an organization that provides the services to the user.

According to NIST definition (MELL; GRANCE, 2011), cloud computing is a model that conveniently provides on-demand network access to a shared pool of configurable computing resources that can be provisioned and released quickly without large management efforts and interaction with the service provider. This model definition is composed of five essential characteristics, three service models and four implementation models, which will be discussed in this section.

2.1.1 Essential Characteristics

A cloud computing service needs to present the following characteristics to be considered adherent to the NIST definition.

- **On-demand self-service**: This characteristic means that it is possible to allocate and deallocate computational resources without the need of interaction with the provider's staff.
- **Broad network access**: All the services offered by the provider must be accessible over a network, a LAN or the Internet, and the access need to be available trough standard mechanisms.
- **Resource pooling**: The provider owns and manages a set of physical computing resources, and they serve multiple users according to their demand.
- **Rapid elasticity**: The provider allows the user to allocate and deallocate resources in any quantity, at any time. To the user, the resources available seems to be unlimited.
- **Measured Service**: The provider is responsible to control the resource usage and availability. This information needs to be made available to the user, to guarantee the service level agreement (SLA). They are also used for billing purposes.

The summary of these characteristics is that cloud computing needs to be accessible by the Internet, and should provide elasticity of the resources in a pay-per-use billing model.

2.1.2 Service Models

The services provided by a cloud provider are categorized into three service models (BADGER et al., 2011): Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Some providers denominates other service models such as Database as a Service, Framework as a Service, those models are commonly defined as XaaS, that means *everything as a service*, but it is possible to classify the XaaS models into one of the three defined by NIST.

2.1.2.1 Infrastructure as a Service (IaaS)

The infrastructure as a service model provides access to virtualized IT resources for computing, storage and networking. Commonly the resources are virtual machines (VM), network-accessible storage and network configurations.

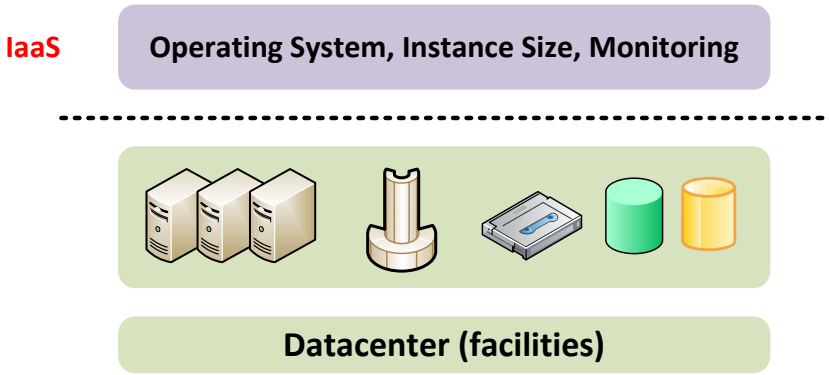

Figure 2.1: Infrastructure as a Service

The user of this service is a system administrator, which interacts with the provider to allocate, configure and deallocate the resources. These resources can be made available to the end user, so, we have two roles one for the system administrator and other for the end-user, but normally is just the administrator that interacts with the provider.

The provider is responsible to maintain the low level services, such as physical machines, network and hypervisor. Is provider's responsibilities also, to maintain an updated set of operating system images, defines the characteristics of the available VMs instances, resource allocation in the physical hardware.

Figure 2.1 illustrates an IaaS stack, the provider responsibilities are under the dotted line and the user has limited view to the interface and services defined by the provider. The provider needs to maintain the correctness of all the basic environment. The user needs to follow the configuration options defined, such as use one of the available operating systems images, or build his own image according to the procedures of the provider. When the user has a VM up and running is his responsibility to administrate it.

Some of the benefits of IaaS is that the user has total control over the resources, because the user has administrative access to the VMs. The capability to rent computational resources is an advantage, because the user can allocate and deallocate resources according to his demand. This model provides access to virtual machines, so the portability of legacy applications is guaranteed without modifications.

One big issue on IaaS is the network infrastructure that is shared between several VMs. They are allocated from a common pool and the components does not have the same level of isolation than physical servers, a loss of performance is possible because of this sharing. Another important issue is related to the security, for example the data erase policies, when the user deletes his data, the provider needs to ensure that they are physically removed.

2.1.2.2 Platform as a Service (PaaS)

The platform as a service model provides a complete toolkit that enables the user to develop and manages applications.

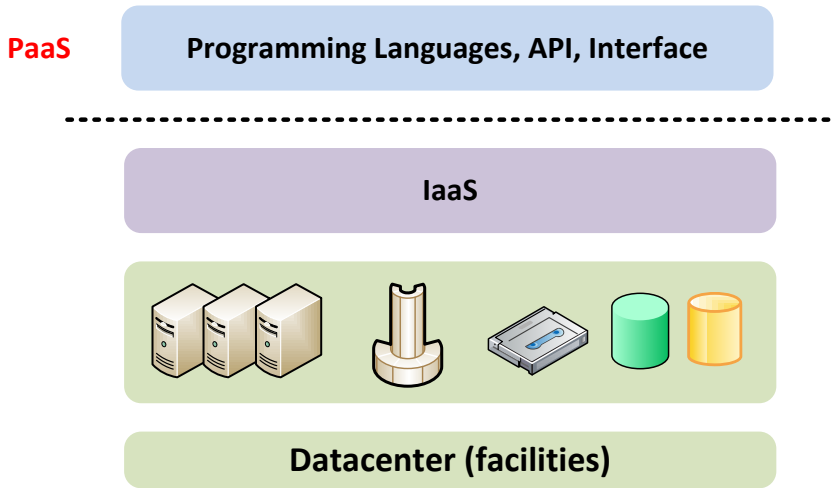

Figure 2.2: Platform as a Service

The user of this model of service is normally a technical user, such as application administrators, developers and testers. They can be single persons or employees of an organization. The user can be also an end-user of an application, but from the user perspective, the model applicable in this case is the SaaS, that will be explained in the next section.

The provider defines and supports a set of development tools. This definition covers the supported programming languages and their respective application programming interfaces (API), also additional libraries can be supported, such as Hadoop (SHVACHKO et al., 2010). All the physical resources are controlled by the provider, is its responsibility to maintain all the infrastructure evolved in the service. The provider is also responsible to maintain all the components of the platform up to date, for example, applying security patches or new releases of certain libraries.

Figure 2.2 illustrates possible layers of a PaaS model, in this case the provider is responsible for the physical resources and uses an IaaS layer to manage it. The user has access to a set of programming languages, APIs and interface definitions. The complete responsibility over all the hardware stack is on the provider, the maintenance of the basic software layer, such as operating system and libraries, including all the configurations required. The user responsibilities are to follow the guidelines of the provider regarding the use of programming languages and other defined tools. The development, deployment and administering of the application are also user's responsibilities.

The benefits of PaaS adoption for the user is the low-cost, in most of the cases no-cost at all, to develop an application and make it available to customers. In this case, the user will be charged just when his customer uses the application. The Google App Engine and Microsoft Windows Azure are examples of commercial providers where the developer does not have to pay anything for the development and deployment of an application, with certain limitations. The scalability of the platform is guaranteed by the provider, the user has no upfront costs to add more computing power to the application.

The user needs to follow the provider's guidelines to develop his application and this causes a situation where the application developed by the user is completely dependent on the provider. This situation causes a technological dependence of the provider, and the user can not use his application in other platform without modifications. This dependence on the provider is the major issue of this service model.

2.1.2.3 Software as Service (SaaS)

The Software as a Service model provides to the user the capability to use an application, the user has access to a software deployed as a web service over the Internet (CARRARO; CHONG, 2006). The user of SaaS can be a single person that wants to have access to certain software or an organization that provides access to their members. The typical SaaS application is: office productivity tools, such as Google Drive[1] and Microsoft Office 365[2]; collaboration systems, such as Yammer[3] and Box[4]; conference management, such as Cisco Webex[5] and Citrix GoToMeeting[6]; among others.

[1] https://drive.google.com
[2] http://office.microsoft.com/
[3] https://www.yammer.com/
[4] https://www.box.com/
[5] http://www.webex.com/
[6] http://www.gotomeeting.com/

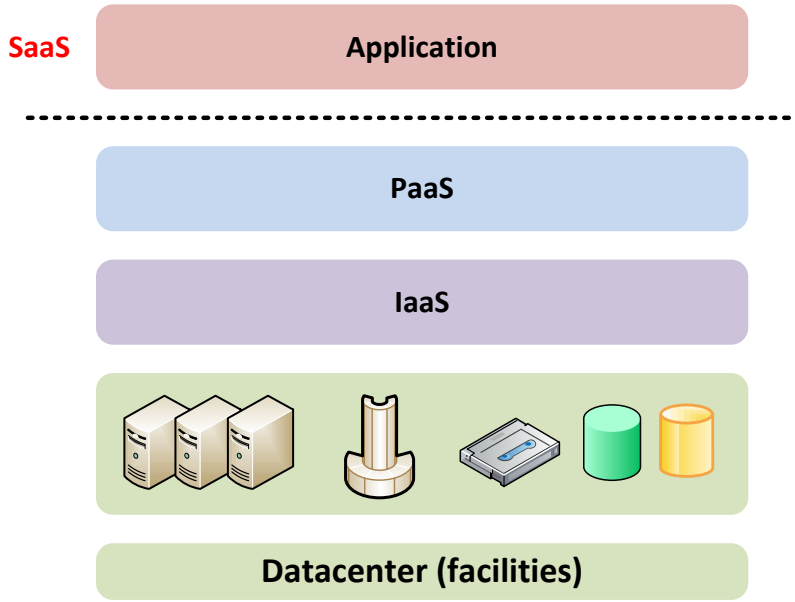

Figure 2.3: Software as a Service

The provider main responsibilities are to guarantee that the supplied software is supported and tested. The provider is also responsible to assure the data confidentiality, this means that the data from certain user can not be accessed by others. Security control of user access, disaster recovery sites and procedures and compliance check are the responsibilities of the provider too, also the platform issues, such as operating system, hardware devices and configuration choices.

Figure 2.3 shows the layers evolved in a SaaS scenario. There are several software and hardware stacks that must work to the service works properly. At the top of Figure 2.3 are the user responsibilities that are just to use the application and, in some cases, the capability to do some limited administrative configuration as a user level control. The provider is responsible for all the other layers. Specifically for the SaaS the provider has the responsibility to maintain the software up to date with all the security patches applied and for licensing control, in case that the application is provided by another organization.

The key benefits of SaaS adoption is that the user does not need to have a big software footprint on his side, the access is regularly made using a web-browser. In general no installation and configuration procedures are required, this decrease the time needed for software distribution. The efficient use of software licenses is also a benefit. There are many software license models that can be used, for example, it is possible that a company with 150 employees decides to use a cloud based project management application, and all of the employees need to have access to it, but not necessarily at the same time. In this situation is possible to the company acquire a limited number of licenses, like 50, and it can provide simultaneous access to 50 employees, but all the 150 employees have access to the application. The number of concurrent users is controlled by the provider.

The issues of this service are mainly related to security, even regarding of user access and for data confidentiality. The access is made typically using a web browser over the Internet, this is a non-secure environment and potential data leaks can occur. The Internet

access is not under control of the provider neither the user, in a real scenario it is possible that the user can not access the service even when it is up and running.

2.1.2.4 XaaS

The models IaaS, PaaS and SaaS are very well defined by NIST and cover all the services that can be offered as cloud services. However, cloud computing emerged as a good marketing strategy, so lots of companies renamed their products to cloud products. Each company created its own product as a service and this generates a lot of new acronyms.

Recently the term XaaS was defined by the community as "Everything as a Service" and concentrates all specific cloud services. Some examples of Xaas are Network as a Service (NaaS), Unified Communication as a Service (UCaaS), Database as a Service (DaaS), Monitoring as a Service (MaaS) among others. However, the XaaS is not part of the NIST definition of cloud computing.

2.1.3 Implementation Models

The NIST cloud definition (MELL; GRANCE, 2011) lists four implementation models: private, community, public and hybrid. Each one of these models has its particularities regarding to aspects such as network dependency, security, quantity of resources, among others. In the public model the user rents the resources from a provider, the private and community models can be used in two configurations, outsourced, where the user rents exclusive resources from a provider, and on-site, where the resources are owned by the user. The hybrid model is a combination between any of the other three models.

The resources are controlled by a cloud platform, that represents an abstraction of all the components which form a cloud provider architecture. This concept will be detailed in Section 2.2. One important aspect of Cloud Computing is the access control. In this thesis we assume that an entire organization is a security perimeter (GASSER, 1988), this means that users who are within the organization can freely access the resources within it. The users that are outside the organization need to be allowed by a security controller, such as firewalls and VPNs.

2.1.3.1 Private Clouds

A private cloud is a model where the physical resources are controlled by the user, a single user or an organization. The most common scenarios of using this model is an organization that decides to virtualize its data center (on-site configuration), or if an organization wants to rent all the resources from a provider (outsourced scenario). In both scenarios, the user has exclusivity over the physical resources.

In terms of cost, this model is the most expensive of all four. This occurs because the user needs to buy and maintain the resources by itself or rent exclusive resources from a provider. The resources are dedicated to the organization, then there are no other users accessing it. This model achieves the same level of isolation and security than the user of traditional computational resources. Those characteristics make this model the most secure of all.

However, this model is controversial, because it is not completely adherent to the essential characteristics of cloud computing. Due to the limited quantity of physical resources, the elasticity of this implementation cannot be considered totally consistent. The pay-per-use billing model is also not complete, because the resources are reserved to the

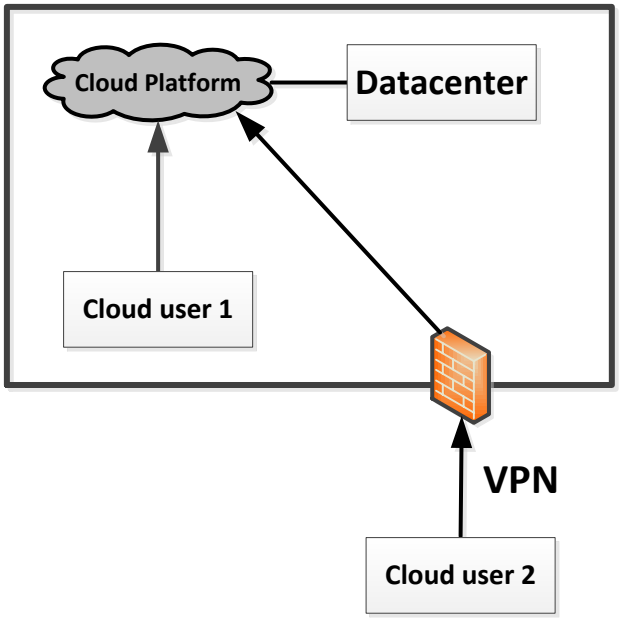

Figure 2.4: Private Cloud Configuration

user, for both on-site and outsourced scenarios. And in this situation, the user is charged for the total amount of resources, even if they are not in use.

A possible configuration of a private cloud implementation is shown in Figure 2.4. This figure represents the on-site scenario, the organization is represented by a rectangle with a bold line. Inside the organization there is a data center facility in which all the resources are installed. These resources are controlled by a cloud platform. The security policies and user access control are defined and implemented by the organization.

In this example, there are two users with access to the cloud platform, the first user is inside the organization, the security perimeter, and the access is granted by the organization's LAN. The second user is outside the organization, and it is necessary to access the security perimeter then the cloud can be accessed. In this case the access is granted by using a VPN.

2.1.3.2 Community Clouds

A community cloud can be considered as a derivation of a private cloud. The major concerns related to both are the same, such as security policies and resource management. The difference between the two models is that the cloud resources are shared among organizations, in other words, the users of a community cloud are a group of organizations, known as members. Usually the members share interests, they can be different companies owned by the same economic group or different departments of a public institution.

The same considerations of the private model about the adherence of this implementation to the essential characteristics of cloud computing need to be considered. The elasticity and pay-per-use model presents similar limitations as the private clouds.

The costs of this model are analogous to the private, but in this model they are shared among all the members. There are three possible scenarios to deploy this model. One of them is when just one of the community members maintains all the resources and provides

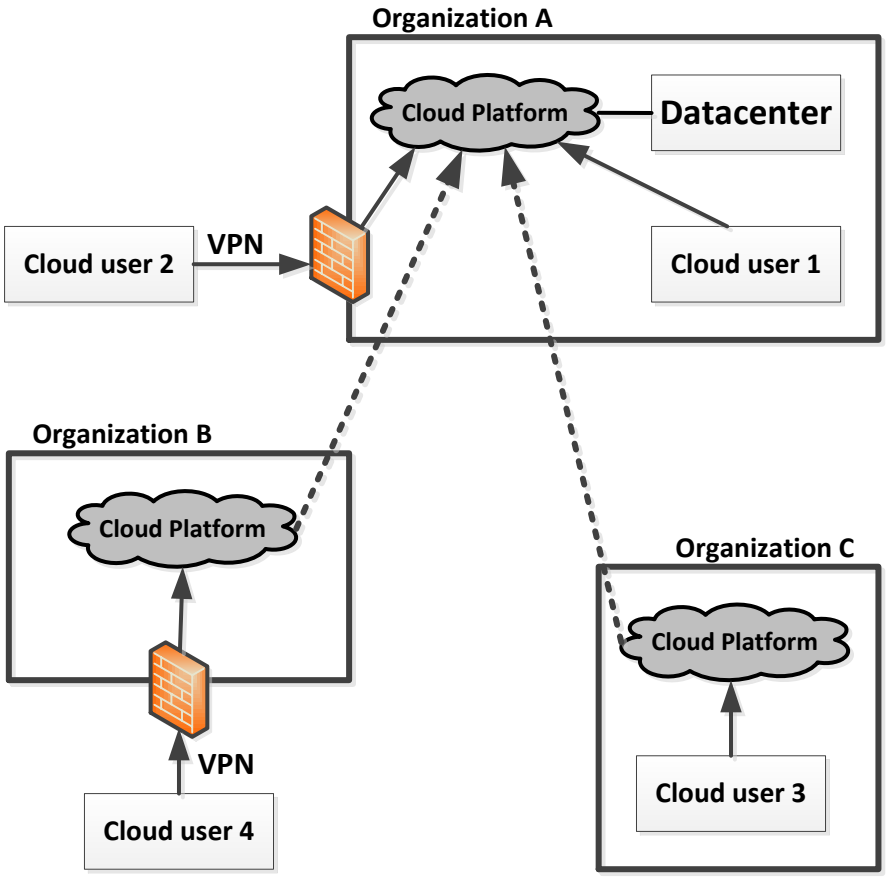

Figure 2.5: Community Cloud Configuration

the service to all members. In this case all the other members are just users of the service. An example of this scenario is shown in Figure 2.5. Another possible scenario is when more than one member maintain resources and provide the service to all members. The third scenario is where none of the members has resources and the community rents this service from an outsourced provider, similar to the outsourced private scenario.

In Figure 2.5, three different organizations are represented by the rectangles with bold lines. All of them have access to the same cloud platform, but only the Organization A has physical resources inside it own data center. The access to the resources are made through the cloud platform and the users from the three members have their access granted by each member security policies. In the figure, the user 1 is inside the organization A and the user 3 is inside the organization C, their access is granted by the organizations' LANs, because they are inside the security perimeters. However, the users 2 and 4 are outside the security perimeter from organizations A and B respectively, and their access is granted by the use of VPNs as boundary controls.

2.1.3.3 Public Clouds

This implementation model is often treated as the real definition of cloud computing, but this assumption is not true, because all the four are considered valid implementation models by the NIST definition of cloud computing. The resources always belong to a

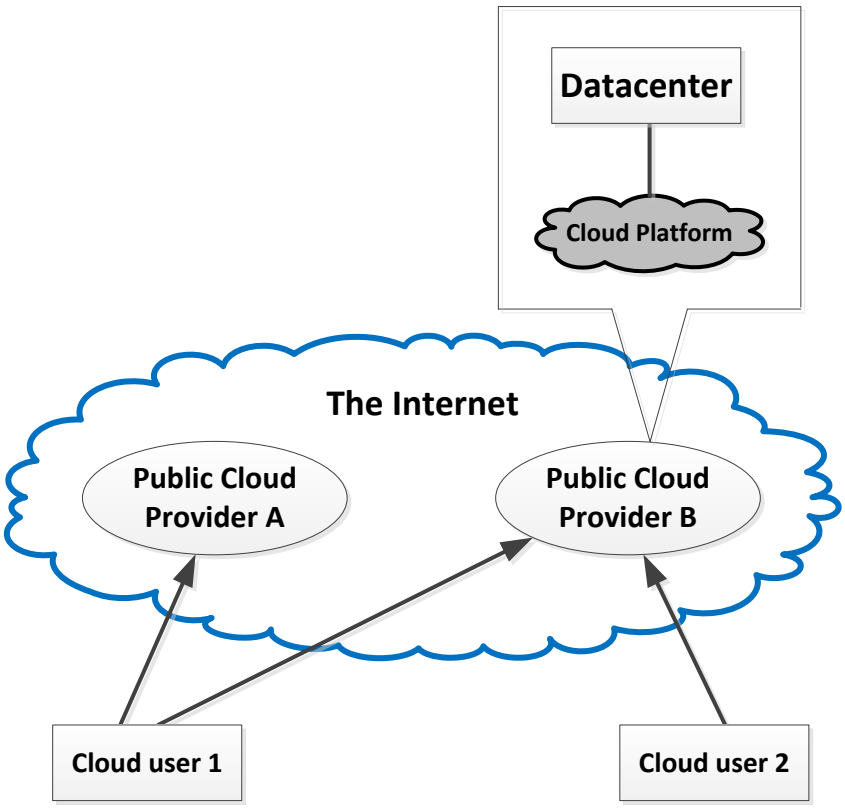

Figure 2.6: Public Cloud Configuration

provider and they are shared within its various users. The provider defines its policies and prices and the users need to agree with them to have access to the resources. The user of this model can be an entire organization or just a single person, because it is possible to allocate just one small, and cheap, machine.

This model is adherent with all the essential characteristics. Mainly the pay-per-use is totally fulfilled, because there are no upfront costs to the user. The elasticity is granted by the provider infrastructure and, for the user, it seems to be unlimited. The access to the cloud is made through an Internet connection. The user control is defined by the provider, and the user needs to use his credentials to have access to the resources. Usually each user has unique credentials, even when they are users of an organization with a corporate account.

Figure 2.6 shows an implementation of a public cloud model. In this case there are two users and two providers, one of the users is a customer from the two providers and the other user uses services from just one of the providers. Each provider needs to maintain all the physical resources. Normally the providers are large companies with distributed datacenters across a country or the entire world. The provider needs to configure the cloud platform that is responsible to manage all the cloud resources. This platform can be an existing one or the provider can build its own.

The data confidentiality is the main concern in this model. Due to the provider controls the resource allocation, one possible scenario is a physical machine running several applications from different users. The provider has the capability to migrate the workload of the user, at any time, to another machine and also to another location. In most

public providers, the user can define in which location the application is executed, but the level of physical machine isolation cannot be defined. Another important aspect is the data deletion, the user cannot verify if the data are physically deleted from a provider's storage.

2.1.3.4 Hybrid Clouds

In theory, a hybrid cloud is a model that is a combination of two or more private, public and also community models. However, these outcomes a tremendous complexity in the configuration, because it can change as the providers come and leave the cloud.

A more realistic scenario is an organization that owns a private cloud connected to a public cloud provider. In this case, the public part is an extension of the private one. When the dedicated resources of the organization are fully used, the organization can allocate new resources in the public provider. To this scenario be possible, the cloud platform used internally by the organization need to be compatible with the public provider. Usually this model is used by an organization that has seasonal demand, for example retailers during Christmas sales, which do not want to have underused resources during most part of the year.

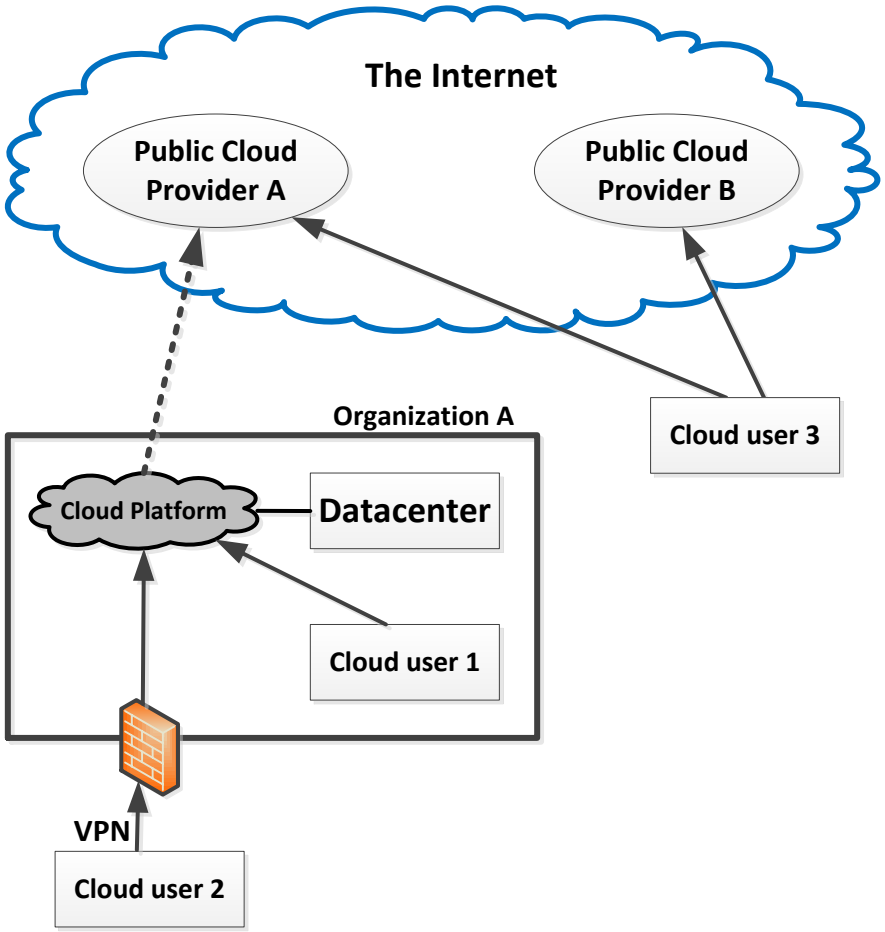

Figure 2.7: Hybrid Cloud Configuration

The cost to maintain the private part of this model is the cost to maintain the computational resources by themselves and when a public part is used the cost is based in the pay-per-use model. The elasticity is granted by the public provider and it is unlimited in the user perspective. Due to this, this model achieves all the characteristics of cloud computing model.

Figure 2.7 illustrates one possible configuration. In this example, the organization A owns a datacenter and some or all of the computational resources are controlled by a cloud platform that is compatible with the platform used by the provider A. The cloud platform is configured to access the provider and has the capability to allocate resources from it. The user access control has the same characteristics as the other implementation models.

The resources allocated from the provider can be dedicated, this means that the hybrid cloud is a combination of an on-site cloud and a public or outsourced private cloud. In the example from Figure 2.7 the hybrid cloud is a combination between an on-site private cloud and a public cloud. The public provider has other users who do not belong to the organization. In the end-user point of view, there is only a private cloud controlled by his organization. The access is controlled by the organization policies. The user can not access the public provider directly and always need to be authenticated by the organization.

The data security can be controlled by the organization, configuring sensitive data to never migrate to the public provider. In this case the sensitive data manipulation is done over a secure environment. The major issue to the adoption of this model is the interconnection bandwidth and confidentiality between the user and the provider. In applications that are communication intensive this connection can be a bottleneck. And if this connection is not available, the organization lost the access to all the resources allocated into the provider.

2.2 Cloud Architecture

Generally we can say that a cloud service is controlled by a cloud platform, which is responsible for all procedures related to the service. A cloud platform is a very abstract term. In a practical approach it is made up of several components that are responsible for its operation. The purpose of this section is to explain the forming components of a cloud platform.

Figure 2.8 presents the cloud platform layers. The base of any cloud service is the *hardware*. By hardware we mean servers, storage and networking equipment. In a public cloud provider this hardware is maintained in a datacenter facility. The hardware is normally grouped into a container, that holds thousands of physical machines and storages interconnected with a high-speed network. This strategy is used by providers to optimize energy consumption. The containers are switched on and off according to user demand, each container has its own cooling system that is only used when the container is turned on.

In a cloud service normally the machines and storages offered to the users are a virtualization of real hardware. The component that performs this virtualization is the *hypervisor*. Basically the hypervisor controls the underlying hardware and provides VMs to the upper layers of the cloud platform. The hardware is a group of machines composed by different sizes and configurations and also a different type of processor architectures and operating systems. The purposes of the use of a hypervisor are to take care of all

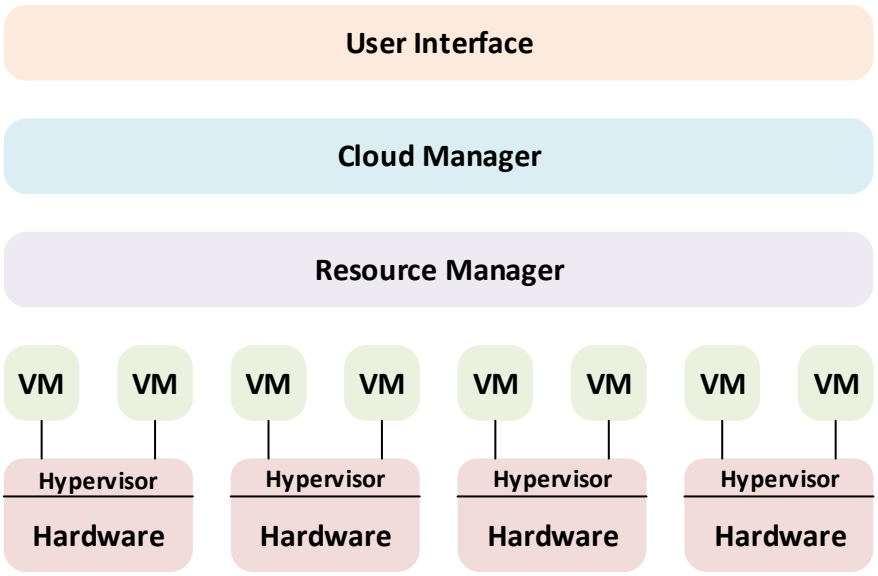

Figure 2.8: A typical cloud platform architecture

this heterogeneity and provide a standard interface to the cloud platform. Example of hypervisors that can be used are: Xen[7], KVM[8], Virtualbox[9], Hyper-V[10] and VMware[11].

The hypervisor layer delivers basically *virtual machines (VM)* to the other layers. The VMs are abstractions of real hardware and can be used for general purposes. For example, a physical server has 8 cores processor and uses GNU/Linux as its native operating system. From this server is possible to create 4 VMs each one with 2 cores and using different operating systems.

The *resource manager* is responsible for providing the interface between the resources and the cloud platform. It controls the VM allocation and deallocation, also the VM migration between different servers is controlled in this layer. The security of resources is defined in this layer too. For example, the policies of VMs interconnection are defined by the resource manager. This is the main component regarding the energy consumption. Because this is the layer that decides when to power up a new machine, or an entire container, according to the demand. The resource manager has the responsibility to perform the VM consolidation to be possible turn off a server.

The main part of a cloud service is the *cloud manager*. This component performs the entire administrative tasks of the cloud platform. The user authentication is performed by the manager, that has a complete user record system controlling each user rights. The consumption of resources that each user makes of the system and the pricing mode are also controlled by this component. This control is used for billing purposes, in this way it can be stated that the pay-per-use charging model is implemented here. The instance sizes of VMs and the standard operating systems are also defined in the cloud manager. All the capabilities of customization of the images, size changing, multiple creation, user

[7] http://www.xenproject.org/
[8] http://www.linux-kvm.org/
[9] https://www.virtualbox.org/
[10] http://www.microsoft.com/hyper-v-server
[11] http://www.vmware.com/

access security, among others are controlled in this layer too. The user reports are generated and provided here. Examples of cloud managers are: OpenStack[12], Eucalyptus[13], OpenNebula[14] and Nimbus[15]. Several providers implement their own proprietary cloud managers.

The *user interface* is the front-end layer of the cloud platform. All the user-provider interaction is made through this layer. The user interface is normally a web page or a smart phone application, from the point of view of the user this layer is the entire cloud platform. Commonly the cloud managers provide a standard user interface, but each provider customizes it.

2.3 Main Cloud Providers

The number of cloud computing providers is increasing quickly, because many companies are interested in selling their products and services into the cloud. At the time of this study, some of the biggest companies in the computer world, such as HP, IBM and DELL, just offers solutions to a customer create its own private cloud, when they sell the infrastructure and provide consulting services and management software. And these companies do not offer services of public cloud.

The focus of this section is to present the most prominent public cloud providers and provide a brief description of their services.

2.3.1 Microsoft

Microsoft started its initiative in Cloud Computing with the release of Windows Azure[16] in 2008, which initially was a PaaS to develop and run applications written in the programming languages supported by the .NET framework. At these days, the company owns products that covers all types of service models. Online Services[17] is a set of products that are provided as SaaS, while Windows Azure provides both PaaS and IaaS.

Windows Azure PaaS is a platform developed to provide to the user, the capability to develop and deploy a complete application into Microsoft's infrastructure. To have access to this services, the user needs to develop his application following the provided framework. The Azure framework has support to a wide range of programming languages, including all .NET languages, Python, Java and PHP. A generic framework is provided in which the user can develop in any programming language that is supported by Windows OS.

Windows Azure IaaS is a service developed to provide to the user access to VMs running in Microsoft's infrastructure. The user has a set of base images of Windows and Linux OS, but other images canbe created using Hyper-V. The user can also configure an image directly into the Azure and capture it to use locally or to deploy to another provider that supports Hyper-V.

[12] http://www.openstack.org/
[13] http://www.eucalyptus.com/
[14] http://opennebula.org/
[15] http://www.nimbusproject.org/
[16] http://www.windowsazure.com/
[17] http://www.microsoftonline.com/

2.3.2 Amazon

Amazon web services[18] is one of the most widely known cloud providers. It offers services on many different models, including storage, platform and hosting services. Two of the most used services of Amazon are the Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3).

Amazon EC2 is an IaaS model and may be considered the central part of Amazon's cloud platform. It was designed to make web scaling easier for users. The interaction with the user is made using a web interface that permits to obtain and configure any desired computing capacity with little difficulty. Amazon EC2 does not use regular configurations for the CPU of instances available. Instead it uses an abstraction called Elastic compute units (ECU). According with Amazon, each ECU provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.

Amazon S3 is also a IaaS and consists of a storage solution for the Internet. It provides storage through web services interfaces, such as REST and SOAP. There is no particular defined format of the stored objects, they are simple files. Inside the provider, the stored objects are organized into buckets, that is a proprietary method of Amazon. The names of these buckets are chosen by the user and they are accessible using a HTTP URL, with unmodified HTTP client. This means that Amazon S3 can be easily used to replace static web hosting infrastructure. One example of an Amazon S3 user is the Dropbox service, provided as SaaS for the final user, where the user has a certain amount of storage into the cloud to store any desired file.

2.3.3 Rackspace

Rackspace[19] was founded in 1998 as a typical hosting company with several levels of user support. They developed the cloud services offered during the company growth and in 2009 they launch the Cloud Servers that is a service of VMs and cloud files, an Internet based service of storage.

The provider has datacenters distributed in several regions: USA, Europe, Australia and Hong Kong. Is one of the major contributors of Open Stack cloud project. Rackspace is one of the biggest cloud providers in the world.

The product offered is the Open Cloud, that is an IaaS model. They provide several computing instances that the user can launch and manages using a web-based control panel.

2.3.4 Google

Google App Engine (GAE)[20] is a service that enables users to build and deploy their web applications on the Google's infrastructure. The service model is PaaS and the users of it is commonly developers. The users need to develop their application using the framework provided.

Currently the languages supported are Python, Java and Go. However, the provider publish that the service was designed to be language independent and in the future other languages shall be included.

The user develops and deploys the application using some of the available toolkits, and all the execution of it is managed by Google staff. The high-availability and loca-

[18] http://aws.amazon.com/

[19] http://www.rackspace.com/

[20] https://developers.google.com/appengine/

tion distribution are automatically defined. Google is responsible for the elasticity that is transparent to the user, this means that if one user application receives a lot of requisitions, the provider increases the resources, and the opposite also happens.

2.3.5 Salesforce.com

Salesforce.com[21] was founded in 1999 as a company specialized in SaaS and today is a leader of cloud computing CRM software. It offers SaaS and PaaS models.

In the SaaS model, the company provides three different services for the users. The first one is the Sales Cloud, that is a complete platform for sales teams, providing access to the user's information using Internet.

The Service Cloud is focused in customer support, using the concept of social networks where the customer can decide how to interact with the company and also can interact with other customers. The last product is the Chatter that is a collaborative platform to be used by the employees from the same company.

The PaaS provided by Salesforce.com is the Force.com platform. It supports the .NET family, Java, and PHP as programming languages, it also has a variety of proprietary tools that can be used to create front-end applications to interact with major back-office applications, such as Oracle and SAP, that is already in use by customers.

The main development that the user can do is to extend the functionalities of the Salesforce.com platform and customize it. The applications supported is focused on office productivity and automation and the user does not have full control of the development.

2.4 Open Issues in Cloud Computing

Cloud computing offers good advantages to clients specially in an economic perspective, due to the characteristic of elasticity and no upfront costs. However, it is still an emerging paradigm and has some open issues. These issues have particular importance in a public or outsourced cloud scenario and the user needs to pay attention during the decision process of which kind of application and data will be moved to the cloud.

Certain types of application are strongly recommended to not move to the cloud, because its behavior is incompatible with the cloud technology. For example, industrial automation and real-time processing applications require a high degree of predictability that cloud providers can not achieve. Applications that manipulate sensitive data, such as military simulations, are also not encouraged to be executed in a cloud environment.

We have grouped the main issues into four: security, portability and interoperability, performance and reliability.

2.4.1 Security

Information security is the process to protect the data confidentiality, integrity and availability. In a private cloud scenario, an organization has the same concerns that in traditional systems, such as administrative and physical controls to authorize access to the data. Otherwise in a public or outsourced cloud environment the organization has different security concerns. Because in this case, all the data are physically stored in the provider infrastructure. In this case, the user needs to ensure that the provider offers equivalent security controls.

[21] http://www.salesforce.com/

The data confidentiality needs to be assured by the provider's procedures and policies. Data privacy and disclosure is an important point that the user needs to take into account. The privacy of the data needs to be guaranteed by the provider applying mechanisms and techniques to isolate the data from different customers. Due to avoid that unauthorized access to the data is performed. The correct data disclosure should be assured by using a kind of tool, such as an identity manager, that only provides access for authorized users. Even the provider personnel cannot have full access to the customer data. This can be achieved by the provider using a mechanism that protects data at rest, such cryptographic storage. As a benefit of the protection of data at rest, the provider also prevents the physical data leaks, such as the loss of a backup tape.

Data integrity is another concern in a public scenario. The provider needs to ensure that all the data is replicated into two different sites, to be possible the restoration. In case of a failure of one of the sites, the provider needs to evoke the disaster recovery procedures to restore the service and the data access as soon as possible. When choosing a provider, it is necessary to verify if these procedures are implemented. Another issue is related to the data transfer and user interface. In an on-site environment all the data is transferred in the organization's network, but in a public cloud, the data transfer is over the Internet. The user interface is commonly the client's web browser, the provider must use all the available security mechanism for protection, of the data transfer. All the data transfer between the user and the provider must be encrypted and authenticated using industry standard protocols, such as https.

The data availability means that the data needs to be available when the user wants to access it. The entire provider integrity is a concern regarding protection against intentional subversion or sabotage of a cloud. The provider needs to able to identify and defend itself from malicious attacks. Attacks like distributed denial of service (DDoS) where the provider receives a huge amount of network traffic flooding the provider, needs to be avoided. Otherwise the provider will get a service outage for its customers.

2.4.2 Portability and Interoperability

Portability is the capability to move an application from one provider to another, if the cost to move is less than to create a new application, the application is considered portable. The SaaS portability is practically inexistent, because each provider has its own application and formats. The portability in PaaS is possible, but in most of cases it is necessary to adapt the application interfaces, such as user interface, I/O calls, to be compatible with the provider. In the case of IaaS providers, the portability is the capability to copy a VM instance from one provider, move it to another provider, then load and use it. Another consideration is related to the data movement from one provider to another, there are costs to transfer the data between them and these costs need to be considered.

Interoperability means the capability to apply an existing scenario from one provider to another, and is always a concern for the user adoption of cloud computing. There are some industry proposed standards available, such as *Open Virtualization Format* (DMTF, 2013) proposed from Distributed Management Task Force and the *Cloud Data Management Interface* (SNIA, 2012) from Storage Networking Industry Association. However, these standards are very specific and are not adopted in the major of cloud providers. IEEE also has a working group (IEEE, 2012) discussing the interoperability of cloud providers, is expected that when this study is finished the defined standard will be broadly adopted by the providers.

These standards are mostly applicable to IaaS providers, due to interoperability of fully customized VMs. Some of the SaaS providers had tools that help the migration process of the client workload, for example a tool to convert the entire set of office productivity documents from one format to another. In terms of PaaS providers, the migration process is much more difficult due to all specific platform concerns and a full automatically conversion may be impossible, in this case the client need to port all the source code by himself or hire a supplier to perform this migration.

2.4.3 Reliability

The reliability is the probability of the provider will be available during a certain period of time (SIEWIOREK; SWARZ, 1998). Normally this value is represented as a percentage and the considered period of time is a year. In regular systems, such as a server, the only variable involved in the calculation of the reliability is the time where the machine is unavailable.

In terms of cloud services, there are several components to determine this value. Normally is difficult to have correctness in this metric in a cloud environment, due to the combination of the components of a cloud service. It is necessary to measure all the underlying infrastructure, also the maintenance personnel and the Internet connection availability.

In general terms, the reliability of cloud computing is the probability of the provider being with all the service components up. Normally this value is defined in a SLA contract. For example, Amazon EC2 and Microsoft Windows Azure have a defined SLA of 99.95%. This percentage means that both providers will be out of service for less than 4h30min in a year.

Otherwise the cloud providers can be located in a different country than the user, in this manner can be complicated the execution of the contract. Thus mitigation techniques are recommended for the user. One example of this kind of mitigation is the off-line synchronization available in some providers. Dropbox and Microsoft Office 365 offers this service, when the user has his files into the cloud, but a local copy is made automatically. Clearly this is not possible for all types and sizes of cloud services, and the user needs to analyze each situation.

2.4.4 Performance

The first issue related to the performance is the Internet service that the user has access. A typical public cloud computing environment is a hosted service available in the Internet. The user's Internet connection speed and availability are an issue even for performance and reliability with a cloud computing service. The performance in this case is related to the management procedures.

A cloud computing service uses virtualization of the physical resources. The major issues are the computing and network interconnection performance. If the hypervisor does not have a good resource management it is possible that the physical resources are over used, causing a performance decrease in the computing performance. The network interconnection of the VM is also a concern, as the network resources are pooled among all the users, the network performance is not guaranteed.

The scalability of applications executing into the cloud is also an issue. Due to the characteristic of computing and communication virtualization of cloud, a good scalability can not be achieved. For example, when the user needs to improve the overall computing power of the application, more VMs are allocated. If the new VMs are allocated in re-

sources that has less computing power or worst communication performance the user will not have the desired performance and will need to allocate more resources.

2.5 State of The Art: Cloud Computing for HPC

We grouped the state of the art regarding to cloud computing used as a platform for HPC into into three categories. One of them contains studies related to deployment procedures of application into the cloud. Another one is dedicated to performance evaluations researches. The final group concentrates studies related to economic viability.

2.5.1 Deployment Procedures

Mao and Humphrey performed a study of deployment procedures of Azure (PaaS), EC2 and Rackspace (MAO; HUMPHREY, 2012). They conducted a systematic study on the cloud VM startup time across these three cloud providers. Five aspects were considered: the time of day, the OS image size, the VM instance type, the VM location and multiple instance allocation. Moreover, the VM allocation time was the only study performed. They conclude that the time of the day, the size of the VM instance and the location does not consist a problem, because the startup times are very similar in different scenarios where this factor were analyzed. Also, they observed that when multiple instances are allocated, there is no significant variation in Azure and EC2, Rackspace does not provide multiple instance creation. However, the variability is very high in EC2 when multiple instances are allocated, in Azure just when they reach 15 instances allocated at the same time a higher level of variation is observed.

Fan et all (FAN et al., 2012) proposes a framework to help users to deploy scientific applications into the cloud based onto the application topology. They propose an automatic topology detection method, which uses pre-execution and multi-scale clustering to discover the topology of a scientific application. And then, based on topology information, they propose a deployment method that can improve the performance of a scientific application. They evaluated their methodology porting NAS to a simulated cloud environment. The focus of this study is on the application side and does not take into account the procedures in the cloud side. They compare their method with other available and the configuration proposed reaches up to 13% better performance when multiple applications are deployed.

More relevant studies related to the deployment procedures for cloud computing can be found in these works (CHIEU et al., 2010), (LIU et al., 2012) and (FAN et al., 2011).

2.5.2 Performance Evaluation

The work of Walker (WALKER, 2008) evaluates the performance degradation of Amazon EC2 instances. He compares the performance of the NAS Parallel Benchmarks with their performance on a real cluster, with InfiniBand interconnection, and analyzes the network efficiency by executing the *mpptest* (GROPP, 1999) MPI benchmark. Both computing and network performance proved to be challenges for the cloud solution. He concludes that HPC applications can be delivered to the EC2, but the performance is lower than compared to a traditional cluster, specially the network performance.

In the work of Gupta and Milojicic (GUPTA; MILOJICIC, 2011) they conduct a study to compare a set of commodity machines using standard network interconnection in a scenario of private cloud. They compared the performance of a cluster using Infiniband as interconnect with the Open Cirrus testbed (AVETISYAN et al., 2010) and with a private

cloud using Eucalyptus (NURMI et al., 2009) as cloud manager. For the evaluation they use the NAS benchmarks and also two applications, NAMD (BHATELE et al., 2008) and NQueens. They conclude that cloud computing is a viable platform to be used for some applications and not for all general HPC applications. However, the authors do not evaluate the cost-benefit of using a cloud, just a performance evaluation was made, besides they just evaluate a testbed and a private cloud.

Li et al. (LI et al., 2010) compare four commercial cloud providers: Amazon EC2, Windows Azure, Google AppEngine and Rackspace. They measure the performance of several components of the cloud solutions, including computing, network, database and storage. They conclude that no cloud solution is best overall in terms of performance, but that each cloud excels in different areas.

The analysis of high performance applications running on cloud systems has been explored by Ekanayake and Fox in (EKANAYAKE; FOX, 2010). In these works, different implementations of the MapReduce (DEAN; GHEMAWAT, 2008) model were investigated, in order to offer evaluations of Hadoop (SHVACHKO et al., 2010), Dryad (ISARD et al., 2007) and their new implementation, CLG-MapReduce. The purpose of the paper is to demonstrate the overhead between the map reduce implementations against a classic MPI implementation. The paper's evaluations were made using algorithms that have little or no communication between processes (such as EP from NAS). In this way, the results are an unfair comparison between the cloud technologies and MPI, since the entire point of such a comparison is that MPI has a much lower communication latency.

The performance of a Xen-based virtual cluster environment is evaluated by Kejiang in (YE et al., 2010). The authors consider resource consumption and introduce a model for measuring the performance overhead for network latency and bandwidth. They use the HPC Challenge Benchmark suite (HPCC) (LUSZCZEK et al., 2006) to compare the performance of a physical machine, paravirtualized VM and Full virtualized VM. The focus of this work is only in virtualized environments and does not take into account full cloud computing environments. They conclude that there exists an overhead when executing an application in the cloud, but this overhead is in an acceptable range, also they conclude that paravirtualized VM is totally compatible with the full virtualized machines, and they identify the network performance as the bottleneck of cloud computing.

Other studies about the performance evaluation of cloud computing can be found in (THANAKORNWORAKIJ et al., 2013), (STANTCHEV, 2009), (TUDORAN et al., 2012), (YANG et al., 2009), (GUPTA et al., 2013) and (IOSUP et al., 2011).

2.5.3 Economic Evaluation

Deelman et al. (DEELMAN et al., 2008) evaluate the cost of Amazon EC2 by porting a real-life astronomy application to the cloud and execute it using different resource provisioning plans. They conclude that the cloud is a cost-effective option since the scientific application provider does not need to buy an entire cluster for a few runs of the application. Many clusters are underused as the hardware quickly becomes obsolete. The cloud solves this problem as it is a responsibility of the cloud provider to keep upgrading the hardware and provide an up-to-date service to the users.

Angabine et al. compare costs and performance issues when using a cloud versus physical infrastructure (ANGABINI et al., 2011). They made an empirical evaluation of the challenges of developing a data-intensive application and compare realistic costs of cloud development versus using local servers. They also present their method to calculate the local and cloud resources costs. However, their work focuses only on data-intensive

applications using Google App Engine, so they do not consider HPC applications. This study was done in an empirical way, and the authors do not present formal conclusions, they just cite that each cloud provider has mechanisms to perform the procedures.

Mastelic et al. create a methodology for trade-off analysis when moving applications to the cloud (MASTELIC et al., 2012). They compare different computing infrastructure and provides a systematic approach for calculating costs of using a certain infrastructure. All the comparison was made using local resources. The focus is to compare four aspects: energy, resource usage, performance and setup complexity. The energy is the consumption of a given infrastructure. Resource usage is regarding to CPU, memory and disk of the hardware. The performance evaluation is to compare different scenarios of infrastructure. And the setup complexity is related to create a complete environment from nothing, installing OS, configure the network. However the authors proposed the methodology and do not evaluate it yet.

Other studies related to cloud economics can be found on (MACH; SCHIKUTA, 2011), (WANG et al., 2009) and (MARTENS; WALTERBUSCH; TEUTEBERG, 2012).

2.6 Concluding Remarks

This chapter has introduced a number of concepts related to cloud computing. Both abstract, in the definition provided by NIST used as guideline, and practical concepts, in the explanation regarding to cloud architecture.

The cloud computing model presents two main features: elasticity and pay-per-use. Elasticity is the capability to increase and decrease a computational environment. The pay-per-use is the billing model used, where the user pays just for the real usage of the resources. These characteristics are interesting for HPC users. Because this group of users, sometimes, does not have a permanent demand of resources, since they can work in temporary projects.

There are several studies related to the use of cloud computing for HPC. They conclude that cloud computing provides an interesting approach to be used as an HPC environment. However there are some concerns, such as the performance degradation, that need to be considered by the users. However, none of them performs a full analysis of the relevant aspects needed for HPC.

Due to this, it was possible to identify new opportunities of research. There is a lack of comprehensive studies regarding to the cloud computing model as an environment for HPC development. The service and implementation models are not evaluated to identify which of them are the best alternatives to be used in HPC. Also, most of the studies were performed taking into account only one provider, and do not perform a comparison between several providers. The economic viability was not addressed in the HPC scenario. All these aspects are analyzed in the following chapters.

3 EVALUATING CLOUD COMPUTING FOR HIGH-PERFORMANCE COMPUTING

Cloud computing presents a paradigm that attracts attention of the scientific community due to the possibility to have instant access to unlimited computational power. At the same time, it is not clear if the cloud computing model is a feasible environment to execute scientific applications, due to the broad scope of its definition.

Our goal in this thesis is to evaluate the use of cloud computing for general purpose High-Performance Computing (HPC) applications. For the evaluate, two analyses must be performed: a technical evaluation and an economic evaluation.

The technical evaluation consists of validating the cloud computing model as an environment for HPC. This means, checking whether HPC applications execute properly in the cloud. Then, if HPC applications work correctly in the cloud, it is necessary to determine the cost-benefit of this model compared a traditional system. This verification is performed in the economic evaluation.

In this chapter, we present HPC concepts. An analysis of the available service and implementation models is conducted to identify which of the models will be evaluated. Then, we define the configurations used for the technical and economic evaluation. We also introduce the cost model used for the economic evaluation.

This chapter is structured as follows. In Section 3.1, the concepts of HPC are presented. In Section 3.2, we analyze the service and implementation models of cloud computing, to determine which of them will be used. The configurations used for the evaluation are presented in Section 3.3. Section 3.4 introduces the cost model used for the economic evaluation. The concluding remarks of this chapter are presented in Section 3.5.

3.1 High-Performance Computing (HPC)

HPC applications are designed to solve complex problems that require a large amount of calculation. Normally, the problems require much more resources than available in regular workstations, or their execution is slower than the user needs, for example the weather prediction for tomorrow needs to be finished today. In these cases, the users need to use high-performance computing to solve these problems.

To achieve their computational power, HPC systems use large scale parallelism. Parallelism means that many tasks, also called processes, can be performed simultaneously. HPC systems provide the capability to execute a large amount of tasks (hundreds or thousands) efficiently. However, in order to utilize the full capacity of HPC systems, it is

required that the applications are built with this intention, they are called parallel applications.

There are two paradigms used to create parallel applications: shared memory and distributed memory. In the shared memory paradigm, all the processes have access to the same memory, and they can communicate with each other using it. This paradigm is used in supercomputers, the most common technique to program using shared memory is through the use of OpenMP[1]. OpenMP is a programming interface that provides the ability to exploit the parallelism inside a computer node, with the use of shared memory.

In the distributed memory paradigm, the processes execute in distinct machines, each one with its own memory. To share data with each other, the processes need to communicate between them using another mechanism, such as network messages. This paradigm is used in clusters, and the standard way to program is with the use of MPI[2]. MPI consists of a standard that exploits the parallelism between computer nodes, distributing the task among several nodes using a message-passing system, with the use of distributed memory. OpenMP exploits the processing power only and MPI exploits both processing power and network.

HPC users can split into two groups, the application users and the developers. Application users are interested in the use of HPC as a tool to perform their work, executing their applications faster. Normally, they are researchers from several areas, such as physics and biology, and their main research is related only to their area. The developers normally are computer scientists that use HPC as their main field. They are interested in improve the performance of an application applying different programming techniques, sometimes they are engaged with the applications users to improve their applications. The developers need a flexible HPC environment to perform their work. This thesis considers both user groups.

3.2 Service and Implementation Models for HPC

Cloud computing has three different service models: SaaS, PaaS and IaaS, as presented in Chapter 2. These service models can be used in four different implementation models: private, community, public and hybrid. This section analyzes them with the intention to identify which are most suitable to be used as an environment for HPC.

3.2.1 Analysis of the Service Models

The *SaaS* model is regularly used to build e-science portals (HOFFA et al., 2008), this kind of portal is used to run standard scientific applications, and no customization is allowed to the user. Normally they are built by a provider porting an application to its environment and then providing access for the users to use the applications on a regular pay per use model. The user of this model is the end-user, such as a biologist, and there is no need to modify the application.

One example of a provider porting a scientific application and then providing the service to the community is the Azure BLAST project (LU; JACKSON; BARGA, 2010). In this project, Microsoft ports the Basic Local Alignment Search Tool (BLAST) of the National Center for Biotechnology Information (NCBI) to Windows Azure. BLAST is a suite of programs used by bioinformatics laboratories to analyze genomics data. Another

[1] http://openmp.org/

[2] http://www.mpi-forum.org/

case of this usage is the Cyclone Applications[3] that consists of twenty applications offered as a service by Silicon Graphics (SGI). SGI provides a broad range of applications that cover several research topics, but there is no possibility to customize and adapt them. The big problem of SaaS as the environment for HPC is the absence of the capability of customization. Research groups are constantly improving applications, adding new features or improving the performance of them and they need an environment to deliver the modifications. There are several applications that are used just for a few research groups, but this kind of applications does not attract interest of the cloud providers to port.

The *PaaS* model presents more flexibility than SaaS. Using it, it is possible to develop a new fully customized application and then executed in the provider's environment. It is also possible to modify an existing application to be compatible with a provider's model of execution, in the major of cases this is a realistic scenario of HPC. The major providers of this service model offer an environment to execute web based applications. This kind of application processes a large number of simultaneous requests, coming from different users. The regular architecture of these applications is composed of a web page, that interacts with the user, a processing layer, that implements the business model, and a database, used for data persistence. Each user request is treated uniquely in the system and has no relationship with others requests.

In PaaS, the provider defines the programming languages and the operating system that can be used, this is a limitation for general purpose HPC development. Moreover, the concept of virtual machine (VM) is not present in this service model, because all the applications are web based. Due to this, it is impossible to create a system to perform distributed computing. However, in the processing layer of this model the shared memory can be used.

The *IaaS* model is the most flexible service model of cloud computing. The model delivers raw computational resources to the user, normally in the form of virtual machines. It is possible to choose the size of the VM, defining the number of cores and the amount of memory. Even the operating system can be chosen by the user. The user can install any desired software in the VM, and can build a fully customized environment. The user can allocate any desired quantity of VM and build a complete parallel system. With this flexibility, it is possible to use IaaS for applications that use shared and distributed memory.

3.2.2 Analysis of the Implementation Models

In terms of implementation there are four models available, each one has advantages to users of scientific high-performance applications, as well as limitations and concerns.

A *private* cloud is basically the same as owning and maintaining a traditional cluster, where the user has total control over the infrastructure, and can configure the machines according to his need. One big issue in a private scenario is the absence of instant scalability, as the capacity of execution is limited to the physical hardware available. Moreover, the user needs to have access to facilities to maintain the machines and is responsible for the energy consumption of the system. Another disadvantage is the hardware maintenance, for example if a machine has physical problems the user is responsible to fix or replace it. A case in which private cloud is recommended is if the application uses confidential or restricted data, in this scenario the access control to the data is guaranteed by the user

[3] http://www.sgi.com/products/hpc_cloud/cyclone/

policies. The weakness of this model is the absence of elasticity and the need of upfront costs. To build a private cloud to be used for HPC can be considered the same as buying a cluster system.

In a *community* cloud the users are members of one organization and this organization has a set of resources that are connected to resources in other organizations. An user of the organization can use the resources of all other organizations. The advantage of this model is to provide access to a large set of resources without charging, because the remote resources belong to other organizations of the community and not to a provider. In other words, the pay-per-use model may be not applicable to this type of cloud. One disadvantage of the model is the limited number of resources, they are limited to the number of machines that are part of the community cloud. The interconnection between all the members constitutes a bottleneck for the application's execution, if the application needs a number of machines bigger than the available in single part of the cloud (a single member), the machines need to be allocated among two or more members.

All the community members need to use the same cloud platform, this demands an effort to configure all the machines, and is necessary to have personnel to maintain the machines. The community model is recommended for research groups that are geographically distributed and want to share the resources among them. In this case, the members can use existing resources to join the community and get access to other members' resources. The weakness of this model is the same of the private, absence of elasticity and pay-per-use model, as well as the interconnection latency between members.

In a *public* cloud, the infrastructure is provided by a company, the provider. The advantage in this case is the access to an unlimited number of computational resources, where the user can allocate and deallocate them according to his own demand. The pay-per-use billing model is also an advantage, because the user just spends money while using the resources. Access to an up-to-date hardware without upfront costs and the absence of maintenance costs completes the list of advantages of the public model. The main disadvantages are related to the data privacy, because in this model the underlying hardware belongs to a provider and all the maintenance procedures are made by the provider's personnel. The data privacy issue can be addressed by a contract regarding data access, but for certain types of users, such as banks, this is insufficient.

The user has access to a virtualized hardware controlled by a hypervisor and does not have control over the underlying resources, such as physical machines and network infrastructure. In this model, the user has access only to a virtual environment and, sometimes, this can be insufficient. Certain applications need specific hardware configurations to reach acceptable performance levels, and these configurations can not be made in a public cloud environment. The recommended scenario to use this model is in cases where the user needs to execute an application during a limited period. Moreover, in case of an application executing few hours a day, the user can allocate the machines, execute the application, and deallocate the machines, and the user just needs to pay for the time used. Even in cases where the application will run during almost the entire day, without a predefined end date, it is necessary to determine the cost-benefit of using a public cloud instead of buying physical machines.

A *hybrid* cloud can be used to extend the computational power available on a user-owned infrastructure with a connection to an external provider. This model is recommended in cases where the user needs to increase the capacity of his infrastructure without the acquisition of new hardware. The main advantage of it is the instant access to computational power without upfront costs. In certain scenarios, it is possible to configure the

system to allocate resources in the cloud automatically, with the system allocating and deallocating machines according to the demand. This model is well applicable in cases where the user already has a set of machines and needs to increase them temporarily, for example for a specific project.

The weakness of this model is related to data transfer, because the local cloud is connected to the public cloud trough a remote connection, normally an Internet connection, and in this case the bandwidth is limited by this connection. In an application that has a large amount of communication, the connection between the user and provider will be the bottleneck and can affect the overall performance. Another important issue is the cloud platform used by the cloud provider. It is necessary that the user system uses the same platform, or at least a compatible one. This means that the user needs to reconfigure all the local machines to follow the cloud model. The concerns about data confidentiality are the same as in the public model.

Summarizing the characteristics presented in this section, we can conclude that all deployment models can be used to create HPC environments in the cloud. The appropriate model depends on the needs of the user, and also his availability of funds. All the models has advantages and disadvantages, and is clear that there is not an ideal model for all the scenarios of use.

3.3 Cloud Computing for HPC: Architecture Proposal

Our goal is to define a cloud architecture to be used for HPC. To achieve this objective, it is necessary to define the service model and the implementation model. The definition needs to take into account the behavior of HPC applications. Due to this, the environment need to be as flexible as possible and capable to use the standard HPC tools and libraries.

3.3.1 Service model for HPC

After the analysis of the service models, the SaaS model was discarded because it cannot be customized sufficiently. It can be used just as an HPC application, but the users considered in our proposal need a general purpose environment. The PaaS model also presents concerns, but still has potential to be used. The IaaS model is the most flexible of the three and was shown to be totally adequate for HPC. Due to these reasons, our proposal will consider the PaaS and IaaS models.

In the PaaS execution model, the application processes several requests separately. The scalability in this scenario is related to increasing the number of requisitions that can be treated at the same time. This is achieved by replicating the application among several isolated machines. In this way, this model cannot be used for distributed computing, which is using MPI. However, due to the processing capability of a single machine, this model can be suitable for exploiting shared memory. This can be performed in the computing layer of the PaaS model, using the OpenMP paradigm. The scenario that will be used in our evaluation is the PaaS with the use of OpenMP programs.

IaaS service model is the most flexible model of cloud computing, due to this, it can be used to create several different configurations. A parallel machine can be configured to connect several VMs, and this configuration can be used to execute MPI programs. Moreover, a single node can be used to evaluate OpenMP applications. For these reasons, the IaaS will be used for both OpenMP and MPI evaluations.

3.3.2 Implementation Model for HPC

The four implementation models of cloud computing have the potential to be used as environments for HPC. All of them have disadvantages, such as the lack of scalability in the community model, and advantages, such as data security in private model. The major reason to discard models was the similarity with traditional systems.

The private model was discarded because it is the same as owning a traditional system, with the same advantages and disadvantages, and does not adhere completely to the cloud computing concepts. The community model has the same disadvantages than the private, with only little advantages over it. This model provides the possibility to have access to shared resources, but it was also discarded due to the non-adherence to the cloud computing concepts. The hybrid model in a practical HPC scenario is unlikely to be used, because the connection between the local system and the provider will be the bottleneck. When the user needs more computing power, an entire HPC system will be allocated in the cloud. When the user allocates an entire HPC system in the cloud, even when using the hybrid model, he is really using the public model. Therefore, the public model is the only new approach to be used for HPC and we will focus our evaluation on this model.

3.3.3 Proposed Architecture

The proposed scenarios for the evaluation are public cloud providers of PaaS and IaaS. In the PaaS execution model, the application processes several requests separately. In this way, this model does not provide the capability to connect different VMs, and it is only possible to use an independent VM instance to execute the application. Due to the processing capability of a single machine, this model will be used for the shared memory paradigm. The scenario that will be used in our evaluation is the PaaS with the use of OpenMP programs.

IaaS is the most flexible service model and can create any kind of environment. This high level of flexibility is achieved because the model delivers raw computing resources. For this reason, it is possible to increase the computational power by interconnecting several machines. With a parallel machine, it is possible to use the distributed memory paradigm through the use of MPI applications. It is also possible to use a single machine to exploit the shared memory paradigm, using OpenMP. We will evaluate IaaS with OpenMP and MPI applications.

3.4 Cost Evaluation Model

To provide a complete evaluation of cloud computing for HPC, it is necessary to perform an economic evaluation of cloud providers comparing the cost of using them against the use of traditional machines. An important aspect of cloud computing is that all the computational power consists of virtualized hardware. The virtualization layer introduces more overhead in the entire system. It is expected that cloud computing VMs has less performance than equivalent traditional machines.

For this reason, the economic evaluation of cloud computing cannot be done using just the price as criteria, and needs to consider also the performance. To provide a fair comparison, it is necessary to combine the performance with the cost of cloud and traditional solutions. This metric was defined during the development of this thesis and is called *cost efficiency factor* (CEF).

To calculate the usage behavior with which a cloud system can have a better cost-benefit than an equivalent traditional system, we introduce another metric called *break even point* (BEP). This metric, expressed in number of days, determines when a cloud is cheaper than a traditional system. This section introduces these two metrics.

3.4.1 Cost Efficiency Factor

To calculate the *cost efficiency factor* for different systems, two values are required. The first one is the cost of the cloud systems. This cost, in the great majority of cloud providers is expressed as cost per hour. The second value is the overhead factor. To determine this factor, it is necessary to execute the same workload in all the candidate systems and in the base system. The overhead factor O_F is the execution time in the candidate system ET_{Ts} divided by the execution time in the base system ET_{Bs}. The following equation represents this calculation.

$$O_F = \frac{ET_{Ts}}{ET_{Bs}} \tag{3.1}$$

As an example, we want to compare a traditional server against a machine in the cloud. We define that the traditional server is the base system. We need to execute the problem on both systems and then calculate the overhead factor. For the workload, we can take the calculation of π with 42 decimal digits. Assuming that the server takes 30 minutes to calculate and the cloud takes 45 minutes, applying Equation 3.1 ($\frac{45}{30} = 1.5$), the resulting overhead factor is 1.5 for the cloud. As the traditional system is the base system, its overhead factor is 1.

Using the overhead factor O_F, it is possible to determine the cost efficiency factor CE_F. The cost efficiency factor is defined as the product between the cost per hour C_{HC} and the overhead factor, resulting in the following equation.

$$CE_F = C_{HC} \times O_F \tag{3.2}$$

For example, using the calculated overhead factor, 1.5, and assuming a cost per hour of \$50.00 of a cloud machine, the resulting cost efficiency is $1.5 \times 50.00 = 75.00$.

The cost efficiency gives the price to perform the same amount of work in the target system, that the base system performs in one hour, because the cost used in our equation is the cost per hour. If the result is less than the cost per hour of the base system, the candidate system presents a higher cost-benefit than the base system. The cost efficiency factor also can be used to verify the scalability of the candidate system. If the number of machines increases and the cost efficiency factor is constant, the candidate system has the same scalability rate than the base system.

3.4.2 Break Even Point

The *break even point* is the point where the cost to use both the base and candidate systems are the same, on a yearly base. In a cloud computing environment, with its pay-per-use model, this metric is important. It represents the number of days in a year, where is cheaper to use a cloud instead of buying a server. In Figure 3.1, the break even point is represented by the vertical bold line. If the user needs to use the system for less days than the break even point (left side of the line), it is better to use a cloud, but if the usage is higher, it is more cost efficient to buy a server.

To calculate the break even point, is necessary to obtain the yearly cost of the base system. The yearly cost $Y_\$$ is composed of the acquisition cost $Acq_\$$ of the machines

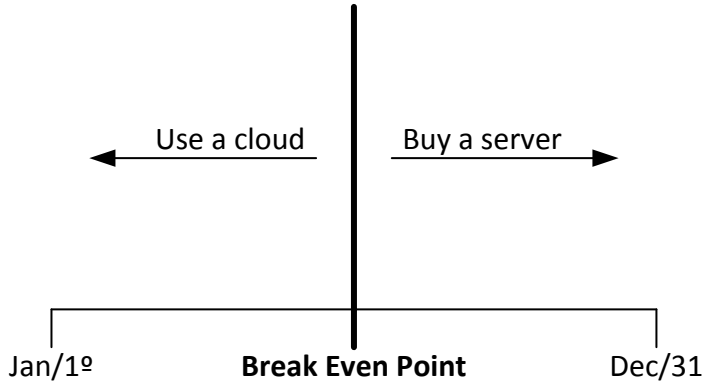

Figure 3.1: Break Even Point (BEP)

themselves plus the maintenance costs $Ymn_{\$}$. To obtain the cost of the machines on a yearly base, it is necessary to determine the usage lifetime LT of the machine, normally three to five years. It is necessary to divide the acquisition costs of the machines by the usage time, this results in the cost per year of the machines. In the yearly cost, also there is necessary to include the maintenance, personnel and facilities costs of the machines. The following equation calculates the yearly per year.

$$Y_{\$} = \frac{Acq_{\$}}{LT} + Ymn_{\$} \tag{3.3}$$

For example, the acquisition cost of a supercomputer is $300,000 and it will be used during ten years. To maintain the supercomputer up and running it is necessary to contract a maintenance specialist for $1,000 per month, or $12,000 per year. Moreover, the energy consumption of this machine is $200 per month, $2,400 per year. Applying the equation, it results in a yearly cost of

$$\frac{\$300,000}{10} + \$14,400 = \$44,400$$

As the cost efficiency factor represents the cost on an hourly base, to obtain the number of days the yearly cost is divided by the cost efficiency factor times 24. The following equation represents this calculation.

$$BEP = \frac{BS_{YC}}{CE_F \times 24} \tag{3.4}$$

Where BEP represents the break even point, BS_{YC} represents the yearly cost of the base system, CE_F represents the cost efficiency factor, and 24 is the number of hours in a day. The result of this equation is expressed in number of days, after which it becomes more cost efficient to use a server cluster instead of a cloud. It is important to remark that the number of days expressed by this equation is for a continuous usage, 24 hours per day, but real world usage is normally less than that. In a practical approach, if the server is used for less days per year than the break even point, it is cheaper to use the cloud instead.

Using the values of the examples calculated, and applying the break even point equation we have the following result

$$\frac{44,400}{75 \times 24} = 24,66$$

this result is interpreted as: if we use the supercomputer for less than 24 days in a year, it is better to use the cloud machine instead to buy the supercomputer.

3.5 Concluding Remarks

The service and implementation models were analyzed in this chapter with the purpose to identify which of them are suitable to be used as an environment for HPC. Due to their characteristics, and taking into account the target user groups, we concluded that the service models to be used are PaaS and IaaS, both using the public implementation model. This decision was made considering the use of the standard parallelization models of HPC, OpenMP and MPI.

To evaluate the proposed cloud architecture, we developed a cost evaluation model. This model calculates the break even point, which determines for how much time of continuous usage it is cheaper to use a cloud service instead of buying a traditional system. There are other aspects, some subjective and other practical, that need to be considered, such as the deployment procedures. In the next chapter, we will evaluate the use of cloud computing for HPC using practical examples.

4 EXPERIMENTAL EVALUATION

In this chapter, we conduct a comprehensive evaluation of the major cloud computing providers to determine if they can be used as a competitive alternative for HPC. In Chapter 3, we selected the PaaS and IaaS service models as candidates for evaluation and performed the analysis and experiments on both of them. We selected the most relevant cloud providers for the evaluation, and we analyze their deployment procedures, performance and cost efficiency.

This chapter is structured as follows. Section 4.1 presents the methodology used to conduct the experiments. In Section 4.2, the providers used in the evaluation are chosen. Section 4.3 explains the environments used as platforms for the experiments. Section 4.4 presents the evaluation of the deployment procedures. Sections 4.5, 4.6 and 4.7 presents the results of the experiments. Section 4.8 presents the general conclusions of this chapter.

4.1 Methodology

The study conducted in Chapter 3 has defined three different scenarios for the evaluation of cloud computing for HPC, they are.

- **Scenario 1**: PaaS and OpenMP
- **Scenario 2**: IaaS and OpenMP
- **Scenario 3**: IaaS and MPI

The scenario of PaaS and MPI is not possible due to restrictions of the model. The evaluation of these scenarios will be conducted using a benchmark suite as the workload and the results in the cloud need to be compared to the results of the same workload in a physical system Moreover, the evaluation criteria need to be defined.

4.1.1 Definition of the Workload

To perform a fair evaluation of the cloud providers against the traditional system, it is necessary to execute the same workload on all of them. HPC applications are normally developed to solve a specific problem and their configuration is complex. These characteristics make the executions difficult to be reproduced with the same conditions. The most common way to evaluate HPC environments is with the use of small applications. These applications are designed to simulate the behavior of real applications, and their execution conditions are easily reproducible. They are known as benchmarks. The term benchmark can be used to identify any kind of testing workload used in performance

evaluation studies. They can be classified into five different types: addition, instruction, synthetic, application and kernel (JAIN, 1991). The addition and instruction benchmarks execute just basic operations, such as multiplication and addition. Synthetic benchmarks are very small applications developed to solve classical problems, such the N-Queens problem. Application benchmarks are small but complete applications with all the steps for program initialization and finalization being performed. Normally, these steps are related to the data input and output procedures. The kernel benchmarks are similar to application benchmarks, but they just calculate the main part of it, the kernel, without considering the input and output steps.

For the evaluation performed in this work, we selected the Numerical Aerodynamic Simulation Parallel Benchmarks suite of benchmarks (NAS-NPB). It is composed of several applications that calculate numerical methods based on applications of Aerodynamic Simulation. The simulation performed by the NAS reproduces the behavior of real Computational Fluid Dynamics applications.

NAS has eight different problem classes, called S, W, A, B, C, D, E and F in ascending order. Class S is used just for configuration purposes and is not used for performance evaluation. Class W represents a problem size to be executed using a workstation from 1990, and is a very small size today. Classes A, B and C are the standard test problems. Classes D, E and F are classified as large test problems.

The NAS suite has a version for OpenMP and a version for MPI. There are eleven applications in total, ten of which available for OpenMP and nine for MPI. The characteristics and behavior of the applications are as follows.

- **BT** - *Block Tridiagonal*: It uses an implicit algorithm to solve the 3-D compressible Navier-Stokes equations. The x, y and z dimensions are decoupled using factorization, the resulting systems are Block-Tridiagonal.

- **CG** - *Conjugate Gradient*: This benchmark computes an approximation to the smallest eigenvalue of a large matrix using the inverse power method. The linear equations are the algorithm is solved by the conjugate gradient method. It uses unstructured matrix vector multiplication and tests irregular long distance communications.

- **DC** - *Data Cube*: This benchmark performs data mining operations, with the use of a data cube algorithm. It is the only benchmark that uses disk accesses to perform its calculation.

- **DT** - *Data Traffic*: It is an application that performs data communication between processes, very little computation is performed. It is used to measure the communication performance of a system.

- **EP** - *Embarassingly Parallel*: In this kernel, the floating point performance is tested by the generation of gaussian random deviates and the tabulation of them in successive square rings. The measurements are made without significant interprocess communication and only a reduction is performed at the end of execution.

- **FT** - *Fast Fourier Transform*: It uses forward and inverse FFTs to solve a 3-D partial differential equation. It tests the long distance communication performance.

- **IS** - *Integer Sort*: It performs an integer parallel sorting operation. It tests both integer computation speed and communication performance.

Table 4.1: Overview of the NAS Benchmarks used in the Evaluation.

Name	Description	Focus	Language	Version
BT	Block Tridiagonal	Floating point performance	Fortran	OpenMP, MPI
CG	Conjugate Gradient	Irregular communication	Fortran	OpenMP, MPI
DC	Data Cube	Data mining	C	OpenMP
DT	Data Traffic	Data Movement	C	MPI
EP	Embarrassingly Parallel	Floating point performance	Fortran	OpenMP, MPI
FT	Fast Fourier Transform	All to All communication	Fortran	OpenMP, MPI
IS	Integer Sort	Integer performance	C	OpenMP, MPI
LU	Lower and Upper Triangular	Regular communication	Fortran	OpenMP, MPI
MG	Multigrid	Regular communication	Fortran	OpenMP, MPI
SP	Scalar Pentadiagonal	Floating point performance	Fortran	OpenMP, MPI
UA	Unstructured Adaptive	Irregular communication	Fortran	OpenMP

- **LU** - *Lower and Upper triangular system*: In this application, the purpose is to solve a sparse, lower and upper triangular system.
- **MG** - *Multigrid*: This is a simplified multigrid kernel that uses the V-Cycle algorithm. This calculation requires structured communications, and it tests long and short distance data communication.
- **SP** - *Scalar Pentadiagonal*: It solves 3-D Navier-Stokes equations using factorization to decouple the x, y and z dimensions. The resulting system has scalar pentadiagonal bands of linear equations.
- **UA** - *Unstructured Adaptive*: It uses dynamic and irregular memory accesses to solve a heat transfer problem in a cubic domain.

Table 4.1 summarizes the properties of the NAS benchmarks used in our evaluation. We can observe that the NAS suite comprehends a very complete set of benchmark that covers the majority of the HPC programs ad is adequate to be used as the workload in this thesis.

4.1.2 Evaluation Criteria

To provide a comprehensive evaluation of cloud computing as an environment for HPC, for both technical and economic criteria, it is necessary to evaluate three aspects.

- **Deployment**: This aspect is related to the deployment capability of providers to build HPC environments in the cloud and the capability to execute the workload.

- **Performance**: This is the performance evaluation of the cloud compared to a traditional machine.
- **Economic**: The economic evaluation is performed to determine if it is better to use a cloud or to buy regular machines.

The deployment capability of cloud computing is related to the configuration procedures needed to create an environment for HPC. The setup procedures to create, configure, and execute an application, and then deallocate the environment are important aspects of cloud computing for HPC. The characteristics that should be evaluated are related to procedures and available tools to configure the environment. Features related to network configuration, time needed to create and configure VMs and the hardware and software flexibility are also important. Criteria related to configuration procedures defined in our study are:

- **Setup procedures**: They consist of the user procedures to create and configure the environment in the cloud provider.
- **Hardware and software configurations**: These configurations are the available VMs size, number of cores and memory, and the capability to run different operating systems.
- **Network**: This criteria are related to the features offered by the provider to user access, as well as the interconnection between the VMs in the cloud.
- **Application porting procedures**: This consists of the adaptation that needs to be performed in the application for it to be executed in the cloud. The evaluation covers both changes in the source code and in the execution environment.

To evaluate the performance of the cloud, it is necessary to compare it with a traditional system. The traditional system is considered the *Base System* and all the performance results need to be compared to it. For a fair comparison, both the base and cloud systems need to present similar characteristics, mainly the number of cores of each system. The purpose is to have a direct comparison between a known system, the base system, and a new system, the cloud.

To achieve a high level of accuracy, each test is executed 50 times, and then we calculate the arithmetic mean. To analyze the variability of the results we calculate the confidence interval for a confidence level of 95% in a Student's t-distribution.

After the deployment and performance analysis, we perform the economic evaluation of the cloud. To determine the cost efficiency, we applied the cost model, defined in Section 3.4, on the same systems used in the performance evaluation. The purpose of this evaluation is to determine when it is cheaper to use a cloud instead of a traditional system for HPC.

4.2 Choosing Public Cloud Providers

After the analysis of the service and implementation models conducted in Section 3.2. This section defines the providers that will be used to conduct the evaluation experiments. The major criteria used in the selection process are: the relevance of the providers and the capability to execute the chosen workload. In the evaluation of relevance, we selected

the providers that are well-known in the cloud computing community, both in academic and commercial context. The candidate providers must have been used or analyzed in previous works, and they have to have an active user community.

4.2.1 PaaS Providers

To select the providers to be evaluated, a study was conducted to identify the main PaaS providers. The possible platforms were *Force.com* from Salesforce.com, *App Engine* from Google and *Windows Azure* from Microsoft.

Force.com is a platform to extend the functionalities of the Salesforce.com applications and cannot be used to execute general purpose applications. For this reason, this platform was discarded. The most used languages in HPC applications (such as NAS) are Fortran and C. For this reason, the platform needs to provide support for these languages. However, both App Engine and Windows Azure do not support these languages as standard. Windows Azure has a special kind of application, called *Native Code Application*, that supports all the languages supported by the Windows OS. The applications can not execute natively and must be converted to the Windows Azure execution model. For these reasons, the only provider used in our experiments for the PaaS service model is Windows Azure.

4.2.2 IaaS Providers

There are several providers that offer IaaS. The majority of them are large traditional companies. All of them present solid market share and a large user group. For this reason, no provider was eliminated by the relevance criteria.

In the analysis of the related work, the majority of the evaluation studies was conducted using Microsoft Windows Azure, Amazon Elastic Compute Cloud (EC2) and Rackspace. The next step was to verify that all the three providers offer similar VM configurations. They also have the capability to execute the workload selected to perform the evaluation. For these reasons, the three providers were selected.

4.3 Definition of the Platforms and Environments

As presented in Section 4.1, the defined criteria for the evaluation were: configuration procedures, the performance analysis and the economic evaluation. These criteria would be applied in all of the defined test environments, to provide a comprehensive understanding of them as viable environments for HPC.

The configuration procedures are evaluated by themselves, without comparing them to the base system. However, for the performance evaluation and the economic evaluation, it is necessary to use the base system to compare them. We select a traditional system with similar configuration as the cloud for the base system. This section describes the platforms and environments used in the experiments.

4.3.1 Microsoft Windows Azure

Windows Azure was used for the PaaS and IaaS evaluation. For both service models, the VM configurations were the same. Azure provides seven configurations for VMs, from the *Extra Small* instance, using a shared core with 768 MB of RAM, to the *A6* instance, with eight cores and 56 GB of RAM. For the experiments, we selected the *Extra Large* instance because it has eight cores running at 1.6 GHz and the amount of memory,

14 GB, is sufficient to execute the selected workload. The hourly cost of this instance is $0.64.

The only supported operating system in Azure PaaS is the Windows 2008 Server. To obtain a fairer comparison, the same OS was installed in the base system. For the IaaS evaluation of Azure, we select Ubuntu GNU/Linux distribution in the version 12.04 as the operating system. This operating system was used for all the IaaS evaluations.

4.3.2 Amazon EC2

Amazon EC2 offers eleven configurations for single machines, with a large variety of settings. The smallest configuration is a single core machine with 613 MByte of main memory. There are three large configurations, each one with eight cores and with a different focus: I/O performance, high memory size, or CPU performance. EC2 also offers two cluster instances, one with four nodes and other with eight nodes, but the use of these instances is the same as renting a cluster and no elasticity is provided. For the experiments, we selected the *Hi-CPU Extra Large On-Demand* configuration, which is the configuration that matches the other systems best.

Amazon does not provide a detailed description of the machines. Amazon uses their own metric called *Compute Units* to describe the performance of a hardware configuration. One EC2 compute unit is the equivalent of a 2007 Intel Xeon core running at 1.2 GHz, or a 2006 Intel Xeon core running at 1.6 GHz. The selected configuration consists of eight cores, and consists of 20 compute units. The cost of this configuration is $0.66 per hour.

4.3.3 Rackspace

Rackspace has eight configurations for VM instances, the smallest one is a machine that shares a single core among four instances and has 256 MByte of main memory, the largest is an eight core machine with 30 GByte of memory.

Rackspace does not identify the instances by name. We selected the largest configuration available in Rackspace, this configuration consists of a 8 cores machine running at 2.4 GHz and 30 GByte of main memory. The cost of this configuration is $1.80 per hour.

4.3.4 Base System

All the three providers offer a very complete set of OS images, covering both several GNU/Linux distribution and Windows versions. Table 4.2 summarizes the properties of the available VM instances of the three providers.

To compare the cloud providers and evaluate the performance overhead and the economic viability, it was necessary to define a traditional system which will serve as the base

Table 4.2: Hardware and Software Properties of a VM instance.

Property	Azure	EC2	Rackspace
VM instances	7	11	8
Memory size (GB)	0.768 – 14	0.613 – 68	0.256 – 30
Number of cores	1 to 8		
Supported OS	GNU/Linux, Windows		

Table 4.3: Configuration of the systems used in the performance evaluation.

Parameter	Azure	EC2	Rackspace	Base System
Instance Name	Extra Large	High-CPU Extra Large	n/a	n/a
Processor model	n/a	2006/2007 Intel Xeon	n/a	Intel Xeon E5530
Processor Speed (GHz)	1.6	$\sim$ 2.5 – 3.0	2.4	2.33
Number of cores	8	8	8	8
Memory (GB)	14	7	30	12
Price per hour	$0.64	$0.66	$1.80	n/a

system. Among the available platforms in GPPD-UFRGS[1], the system that is closest to cloud providers is the set of machines used for simulation, called Viking machines. This system consists of four machines with two Intel Xeon E5530 processors, each containing four cores running at 2.33 GHz with one thread per core and 12 GByte of main memory. The four machines are interconnected with a standard Gigabit Ethernet network. Table 4.3 summarizes the characteristics of all the environments used for the evaluation.

For the IaaS evaluation, all systems were configured to use the Ubuntu GNU/Linux distribution in the version 12.04 as the operating system. The benchmarks were compiled using GCC 4.6.3. For the MPI experiments, we used mpich2 as the runtime environment.

4.4 Evaluating the Deployment Procedures

The deployment procedures are the capabilities offered to create an environment to be used for HPC. This evaluation shows the features that the providers offer to perform this task. The criteria used was defined in Section 4.1. The criteria was applied individually on each provider and the boot and release times were evaluated comparing the three providers

4.4.1 Microsoft Windows Azure

Windows Azure was used to evaluate both PaaS and IaaS service models, for this reason the procedures for these two scenarios are presented. The PaaS procedures include the application porting process and the IaaS procedures are related to the configuration of the environment only.

4.4.1.1 Windows Azure PaaS

The big efforts for deployment in the PaaS platform is related to the application porting process. It is necessary to adapt the application to the execution model of the provider. There is also necessary to configure a VM instance and a web address to execute the application.

The first step is to adapt the application execution model to the model supported by the provider. Windows Azure has a very high level of integration with the Microsoft Visual Studio (VS). Due to this, the evaluation was made using this programming environment. It is necessary to install a VS extension to build Azure applications. VS does not support Fortran as a native programming language, it is necessary to choose a compatible

[1] http://www.gppd.inf.ufrgs.br

Fortran compiler. After some research we chose the Intel Visual Fortran Composer XE for Windows, which integrates well with Visual Studio and compiles the source code of the NAS benchmarks written in Fortran correctly. The only operating system available is Windows.

After the programming environment be ready, it is necessary to convert the application. Figure 4.1 summarizes the necessary steps to port an application written in a language that is not standard supported by Windows Azure PaaS. Basically there are three steps that need to be performed.

The first one is to convert the application to the Win32 programming model, assuming it was not originally developed for Windows. It is necessary to convert the build system, use a Windows compatible Fortran compiler and adapt the system calls. The second step is related to adapt the application to the Windows Azure itself. The execution model of Windows Azure is web based, then the application needs to be encapsulated into a library that will be invoked from a web page. The final step is the deployment procedures, where the user interface is created and the application is uploaded to the provider's infrastructure. The details of this conversion procedures can be found in (ROLOFF et al., 2012).

To configure an environment in Azure PaaS the user needs to use the Azure console, that is a web interface. Each user has a unique Microsoft account that consists of a username and password, this account can be used to access other Microsoft's services, such as Office 365 or Hotmail.

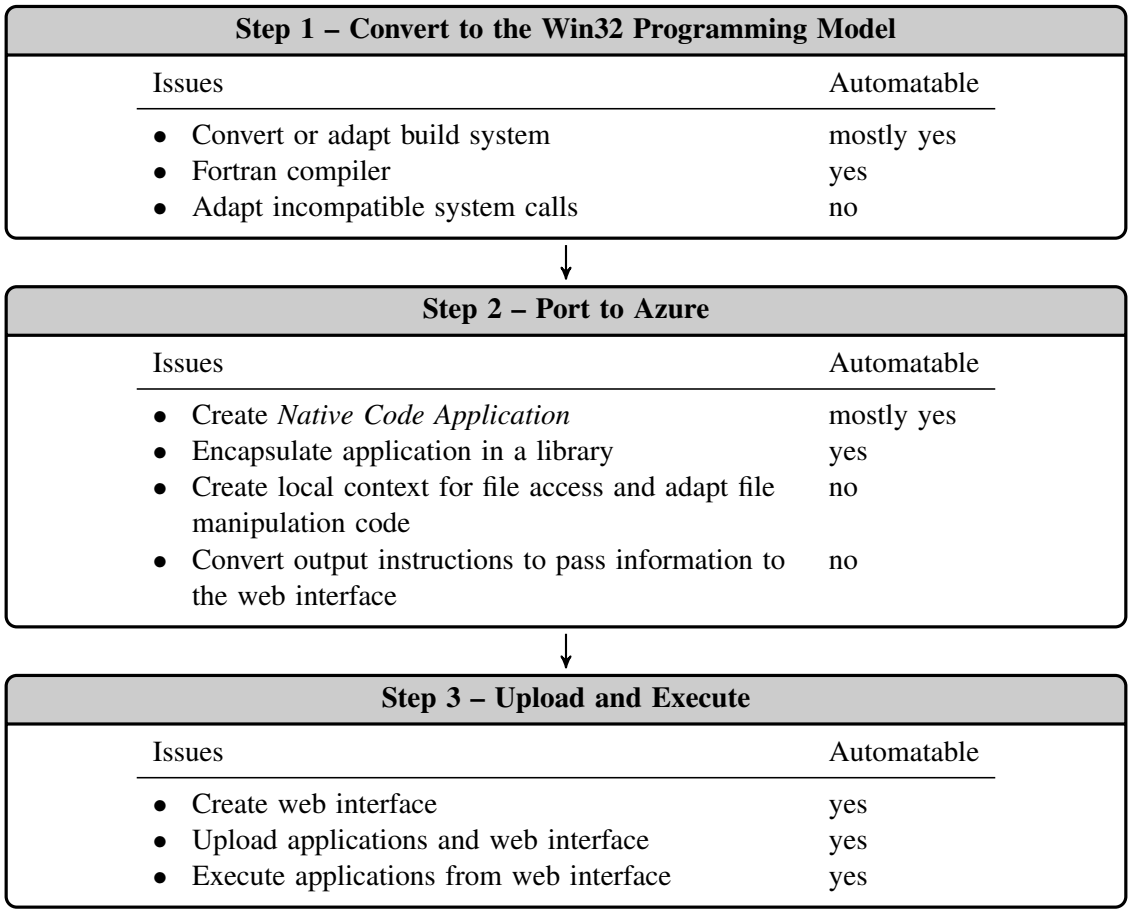

Figure 4.1: Summary of porting process of an application to Windows Azure PaaS.

To create a cloud service, the user needs to configure some parameters. The first one is the name of the website; then the parameters of the web server; and the size of the machine. After this configuration, the environment is ready to receive the application.

The setup of the size of the instance can be made using the Windows Azure SDK installed in the Visual Studio. All the configuration of the execution environment can be made without being connected to the provider. There are several parameters related to the configuration of the web server, such as caching mechanisms, cookies, among others, but in our tests we kept them in the standard.

Inside the VS environment the user can packs the application to be delivered to the provider. It is possible to execute the deploy directly by VS. To perform that, it is necessary to create, in the provider, and install, in VS, proprietary certificates that allow this procedure.

Due to the characteristic that OpenMP exploits only the shared memory paradigm, the evaluation was performed using just one node. Because of this, the evaluation of the network capabilities of Azure PaaS was not executed.

The conclusion is that is possible to use Azure PaaS as an environment for HPC, but due to extensive efforts needed to port the application it is not the best solution at all.

4.4.1.2 Windows Azure IaaS

To configure an environment in Windows Azure IaaS the user needs to use the Azure console, that is a web interface. As in the PaaS model, each user has a unique Microsoft account that consists of a user-name and password.

To configure a VM the user needs to follow four steps. These steps consist of the selection of the desired operating system; the configuration of credentials to access the machine; configuration of the external access to the machine and its location; the last step is optional and is related to configure the high availability scheme that the user wants to use. Then the machine is ready to be launched.

Azure offers 6 standard images of GNU/Linux and 2 standard images of Windows. Also, 11 images of Windows with customized software, such as SQL Server and Share Point, are available. The user can capture a running VM, then this captured image will be available in the personal images, and create a customized image to launch new instances with the same configuration. It is possible to upload a customized image to be used to launch instances.

Each machine created in Azure receives its own Internet name. Through this name, the user can connect to the machine using SSH of FTP, using the credentials defined during the configuration procedures. If the user is not using a customized image, the application can be deployed in a standard VM using FTP. In our case, the workload used was developed to be executed using a Unix-like operating system then no customization in the source code was necessary.

To configure the parallel machine in Azure it is necessary to create an *Affinity Group*, that is a group where the machines that belongs to it are visible among them. During the configuration of a VM, one of the steps is to choose the affinity group, but this field is not mandatory. In terms of user interaction, just the label of this group is informed by the user, all the configurations are made by the provider. Azure documentation mentions that the provider delivers the machines from the same group physically as close as possible. The features presented in Azure makes it a viable environment to create HPC environments.

4.4.2 Amazon EC2

The configuration procedures for the EC2 are performed using a console, that is a web interface. The user has access to an Amazon account and this account is exclusive to the Amazon cloud services.

The user needs to follow 7 steps to create an instance. The first step is to choose an operating system; then the user selects how many instances will be created at the same time and the size and location of these instances. The next step is to choose the kernel version for GNU/Linux instances; the fourth step is related to connection to an external storage; then tags for management purposes only can be defined The next step is related to the creation or uploading of an SSH key pair, all the machines are accessed using an SSH key pair, there is no possibility to use user-name and password authentication; the last step is the configuration of a security group. A security group is a configuration of a virtual network group, where only the machines that belongs to this group can access each other.

EC2 has 10 standard images of GNU/Linux and 3 standard images of Windows. Also, 6 images of Windows with specific software, such SQL Server, are available. In EC2 when the user configures a VM using the provider's tools it is possible to save the image, and use it to create new instances with the same configuration. It is possible to import customized images created both in the user's local system than in other providers. The formats supported are VMware ESX, VMware Workstation VMDK, Citrix Xen VHD and Microsoft Hyper-V VHD.

The machine created in EC2 receives a unique Internet name. Using this name, the user can connect to the machine using SSH and FTP. The user does not have a user-name to access it, then the access is granted by using the SSH key defined during the configuration procedures. It is possible to use a customized image, but if the user wants to use a standard image, the application can be deployed using FTP. Due to the use of the same operating system than in Azure, no customization in the source code was necessary.

To create a parallel machine in EC2, it is necessary to define a *Security Group*. This is a mandatory field in EC2, and the machines that belong to the same security group can connect between each other. After the analysis of EC2, it is possible to conclude that it is a viable environment for HPC.

4.4.3 Rackspace

The configuration procedures in the Rackspace are performed using the web console. The user has access to a Rackspace account and this account is exclusive to the Rackspace system. In Rackspace, only two steps are needed to create an instance. The user needs to choose the operating system image in the first step, and in the second the name of the machine and its size. Then the machine is ready to be launched.

Rackspace provides 16 standard images for the GNU/Linux and 3 standard images of Windows. Also, 6 images of Windows with specific software, such SQL Server, are available. The Rackspace has the capability to clone an existing VM instance, then it is possible to create new instances using this as the base image. There is no possibility to upload an image configured outside the provider.

The machine created in Rackspace receives a real IP address in the Internet. The user can connect to the machine using this IP, through the use of SSH and FTP. The user has the root password, the access is granted using this user-name and password. If the user does

Table 4.4: Time (min:sec) needed to manage the cloud instances.

Provider	Boot 1 instance	Boot 4 instances	Delete 1 instance	Delete 4 instances
Azure	3:06	12:43	3:07	12:26
EC2	2:34	2:34	0:33	0:33
Rackspace	12:37	12:57	0:23	0:23

not have a customized image, the application can be deployed by FTP. The application customization is also not needed, due to the same reasons of Azure and EC2.

All the machines configured in Rackspace have a real IP address, then they can be accessed by any other machine. The configuration of security groups are inexistent in this provider. The feature presented in Rackspace makes it a feasible provider to build HPC environments.

4.4.4 Manage Time of the VM Instances

The time to manage VM instances are an important factor to HPC users, because if the time needed to boot an instance is large, the provider is not viable to be used to create large environments. There were analyzed the boot and deletion times for one and four instances. The boot time includes the time to configure a machine using the provider's front-end and the time that this configured machine is accessible as a VM. The booting and deleting times for one and four instances are presented in Table 4.4.

The Azure presents higher times because it needs exclusive access during the creation and deletion time, this means that only one instance can be created and deleted at the same time. However, Microsoft publishes that this will be corrected soon. When Azure has this issue corrected, all the providers will be feasible alternatives to create HPC environments.

4.5 Evaluating PaaS and OpenMP Applications

For the performance and economic evaluation of Windows Azure PaaS, the OpenMP version of NPB-NAS was used with the problem class A. The FT test contains many procedure calls in the source code and its execution was terminated by the Azure platform due to insufficient stack size in the web server. We found no way to change the maximum stack settings of the Azure instance. Due to the problems with the stack size, we did not include FT in our experiments. Also, as DC focuses on the performance of file system access, creating and modifying large data files, and the focus of this thesis is in the computing part, the DC was not included as well.

Because the PaaS model has the capability to run just the OpenMP version, only one instance from Azure and base system were used.

4.5.1 Performance Evaluation

The performance evaluation was executed according with the guidelines defined in Section 4.1.2. After the 50 executions on both systems, the arithmetic mean and the confidence interval were calculated.

For the graphical representation, the mean and the confidence interval were normalized, taking into account that the time of execution in the base system represents 100%. This representation was chosen, because the observation of the overhead is more evident.

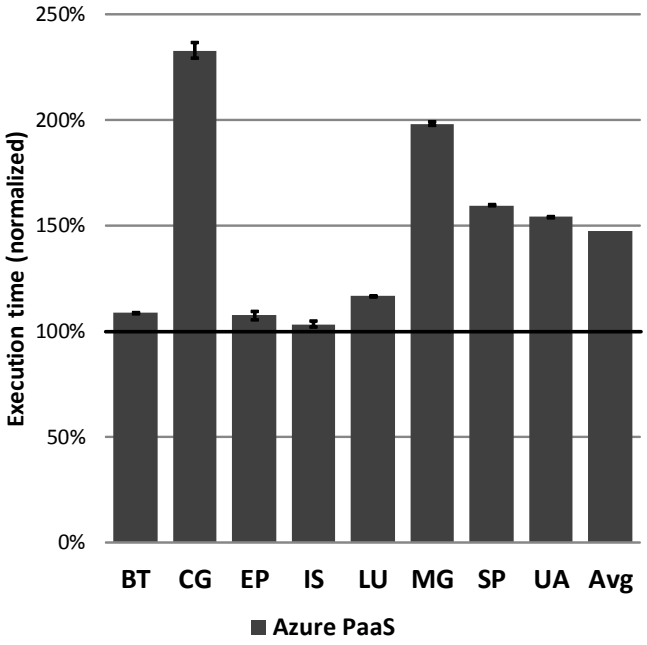

Figure 4.2: Performance results for PaaS with OpenMP

The graphical representation is shown in Figure 4.2. The absolute values of mean and confidence interval, for both systems, are shown in Table 4.5.

It is possible to group the results into two different groups, according with the performance degradation. The first group, consisting of CG, MG, SP and UA shows a large degradation from 50% to about 100%. The second group, consisting of EP, IS, BT and LU, show a much smaller degradation of 3% to 15%. The reasons for these results are as follows.

The IS, BT and LU benchmarks focus on integer and floating point performance and also have a medium amount of communication, compared to the other tests in the workload. Their performance on Azure is close to the performance on the base system.

EP shows different behavior from all other benchmarks. As it uses an embarrassingly parallel algorithm with very little memory usage and communication, its performance depends to a large extent on the FPU performance of the processors and not on the memory performance. From the EP result we can therefore conclude that the Azure cloud has a very similar processor than the base system.

Table 4.5: Mean and confidence interval for PaaS with OpenMP (seconds).

Test	Azure	Base system
BT	17.65 ($\pm$ 0.049)	16.26 ($\pm$ 0.085)
CG	1.13 ($\pm$ 0.041)	0.48 ($\pm$ 0.012)
EP	2.64 ($\pm$ 0.048)	2.46 ($\pm$ 0.013)
IS	0.25 ($\pm$ 0.003)	0.24 ($\pm$ 0.006)
LU	15.83 ($\pm$ 0.020)	13.61 ($\pm$ 0.059)
MG	1.54 ($\pm$ 0.013)	0.77 ($\pm$ 0.024)
SP	23.33 ($\pm$ 0.079)	14.65 ($\pm$ 0.219)
UA	25.19 ($\pm$ 0.078)	16.36 ($\pm$ 0.113)

The CG, MG, SP and UA benchmarks present a large performance degradation. Three of these benchmarks, CG, MG and UA, focus on communication between processes. With these results we can conclude that the memory access performance in Azure PaaS is much slower than in the base system, a possible explanation for this behavior can be the virtualization that is inherent in cloud computing.

The SP benchmark, although focusing on FPU performance and being similar in structure and operation to BT, has a much larger amount of communication between the threads(COARFA et al., 2004), this explains its large degradation compared to BT.

These results lead to two conclusions for the OpenMP experiments: the performance of the memory interconnections are worst performance in Azure than in the base system, and they are very relevant for the performance of HPC applications.

Another important result is the variability of the execution time of the benchmarks. Low variability is an important factor for HPC, as it leads to a higher predictability of experiments, both for the time it takes and the cost. From the confidence interval shown in Figure 4.2, and in Table 4.5, it is easy to note that performance is very predictable on the base system as there is practically no variability for all benchmarks. On Azure, the results are very similar, showing up that the variability in the execution is not an issue for Azure. All benchmarks have a negligible variability on these architectures.

Using an average between all the tests of the workload, the overhead observed in of Azure is about 50% compared with the base system. However, this metric cannot be used alone, it needs to be combined with the economic evaluation seen in the next section.

4.5.2 Economic Evaluation

To perform the economic evaluation, first we need to calculate the cost of the base system using the desired usage scenario. To provide a comprehensive evaluation, we consider two scenarios, one assuming that the base system will be used for three years and the other considering the usage time of five years.

Considering a purchase price of $12,000 for the base system and that the machine will be used for three years, we arrive at a hardware cost per year of

$$\frac{\$12,000}{3} = \$4,000$$

and considering the usage time for five years, the cost per hour is

$$\frac{\$12,000}{5} = \$2,400$$

The average power consumption was measured being 275W during execution of the benchmarks. Assuming a price of $0.0991 per kWh, the hourly cost is $275W \times \$0.0991 = \$0.0273$ and the annual cost of power is

$$\$0.0273 \times 24 \times 365 = \$239.148$$

We cannot objectively measure the price for facilities and personnel and do not include it in our experiments. However, if the user can calculate these costs, they can be included. The total cost per year is therefore $4,239.148 considering the three years usage and $2,639.148 considering the usage for five years. To compare the cost efficiency with the cloud, the price per hour, in the three and five years scenarios, needs to be known. These values are $0.484 for the three years usage and $0.301 for five years usage. These costs

Table 4.6: Cost efficiency factor and Break even point for IaaS with OpenMP.

Parameter	Azure
Overhead factor	1.47
Cost efficiency factor	0.94
Break even point - 3 years	187
Break even point - 5 years	117

are for each machine and they are the same for all the three evaluations (PaaS, IaaS with OpenMP and IaaS with MPI).

To calculate the cost efficiency factor, it is necessary to use the overhead factor and the price per hour of the Azure and apply the Equation 3.2 from the Chapter 3. Because the NPB-NAS suite is a set of the most common problems in HPC, the average between all the tests are considered as the overhead factor, this value was calculated as 1.473 for Azure PaaS The price per hour of the instance used in Azure is $0.64. The result of the cost efficiency factor is 0.942 and is also shown in Table 4.6. Comparing this value with the price per hour of the base system, it is higher than the cost per hour of the base system for both usage scenarios. We can conclude that the Azure PaaS costs more to perform the same work than the base system in one hour, in other words it presents less cost efficiency than the base system.

The provider may also be considered as an alternative for use in a usage scenario where the application does not execute for 24 hours a day. To determine the number of days in which the provider has better cost-benefit than the base system, we use the break even point. To calculate the break even point, it is necessary to use the cost efficiency factor and the yearly cost of the base system. The yearly cost of the base system is $4,239.148 and $2,639.148, for the three and five years usage scenarios respectively. With these values, the Equation 3.4 from the Chapter 3 is used and the results are shown in Table 4.6.

Figure 4.3 shows the break even point. It is possible to note that the initial cost of the base system, for both usage scenarios, is much higher than the provider, but its growth factor is slower. The intersection between the Azure line and the base system line represents the break even point. The results are interpreted as: if we will use the HPC environment for less than 187 days per year, considering the continuous use for 24 hours a day, during

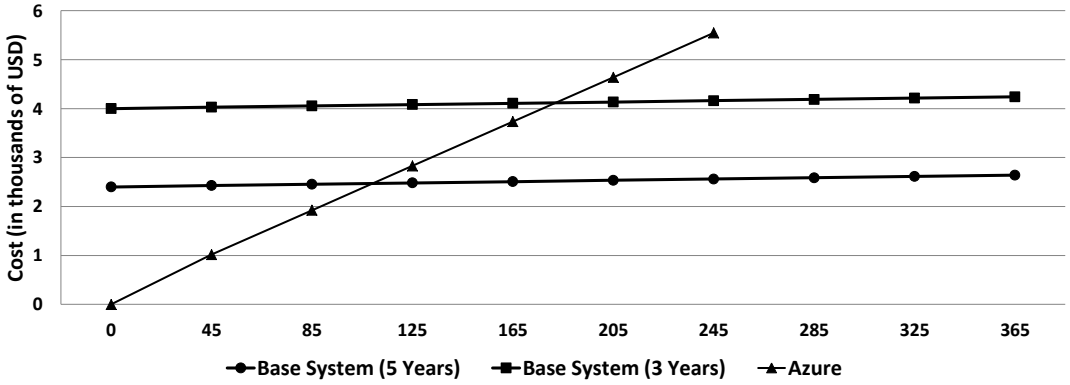

Figure 4.3: Break Even Point for PaaS with OpenMP

a three year period, it is better to use the Azure PaaS instead to buy the base system. When considering the five year scenario, the usage of the environment needs to be higher than 117 days to present more cost-benefit.

4.6 Evaluationg IaaS with OpenMP Applications

The analysis of the IaaS providers regarding to OpenMP applications was performed using Microsoft Windows Azure, Amazon EC2 and Rackspace. The performance and economic evaluation were conducted using the OpenMP version of NPB-NAS with the problem size B. The DC test performs data mining operations and manipulates large disk files. As the focus of this thesis is not the disk access, we did not include DC in the evaluation. Due to the shared memory characteristic of the OpenMP applications, only one instance from the cloud providers and base system were used.

4.6.1 Performance Evaluation

After uploading the workload to the providers, we execute the performance tests to compare the execution time in the IaaS providers with the execution time in the base system. The experiments were conducted using the same methodology as presented before. As in the PaaS evaluation, the arithmetic mean was normalized to the base system. Figure 4.4 contains the performance results for a single node running the OpenMP version of the benchmarks. It shows the normalized mean and confidence interval. Table 4.7 contains the absolute values.

Azure and Rackspace have a similar performance with a slight advantage for Azure, except in the case of IS and LU when Rackspace was better. For BT, FT and IS, the performance of EC2 is similar to the other two cloud providers as well. For CG, LU, MG, SP and UA, the performance is much worse however. Interestingly, the EP benchmark shows a better performance compared to the other clouds. This can be attributed to the higher processor frequency in EC2, as this benchmark only focuses on raw computation. Benchmarks with a higher amount of communication present worse results. Summarizing

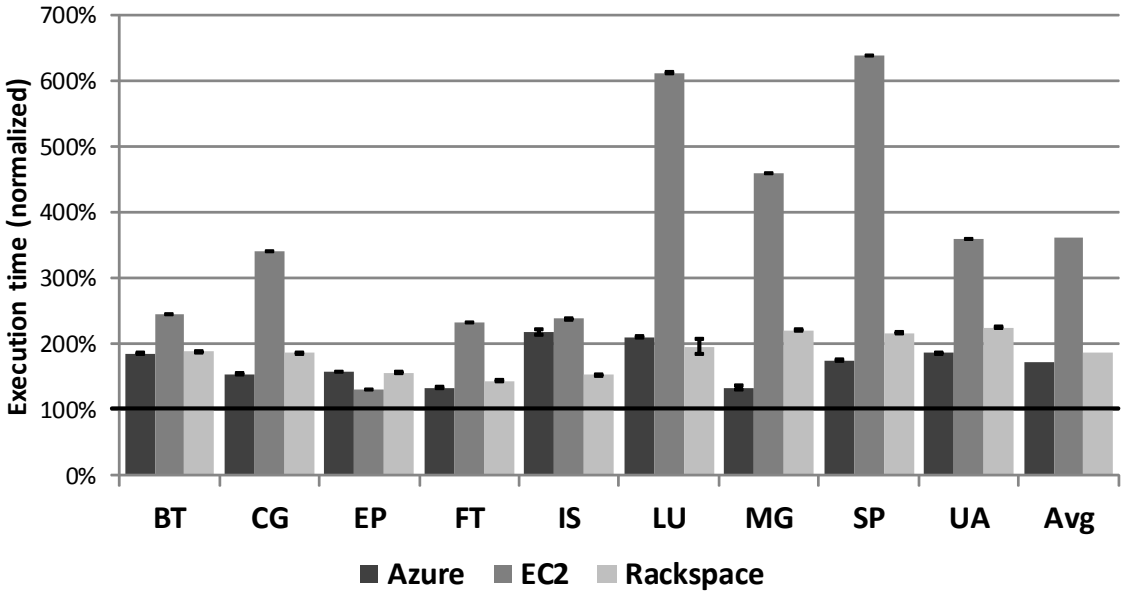

Figure 4.4: Performance results for IaaS with OpenMP

Table 4.7: Mean and confidence interval for IaaS with OpenMP (seconds).

Test	Azure	EC2	Rackspace	Base system
BT	87.92 (± 0.142)	116.64 (± 1.000)	89.47 (± 0.982)	47.62 (± 0.1807)
CG	30.03 (± 0.074)	66.12 (± 0.397)	36.28 (± 0.419)	19.47 (± 0.0259)
EP	17.68 (± 0.042)	14.55 (± 0.037)	17.52 (± 0.187)	11.22 (± 0.0340)
FT	17.85 (± 0.046)	31.20 (± 0.172)	19.25 (± 0.230)	13.41 (± 0.1270)
IS	1.44 (± 0.011)	1.58 (± 0.062)	1.01 (± 0.016)	0.66 (± 0.0154)
LU	83.06 (± 2.527)	241.37 (± 1.185)	77.21 (± 8.745)	39,51 (± 0.2715)
MG	3.82 (± 0.050)	13.16 (± 0.099)	6.33 (± 0.064)	2.87 (± 0.0461)
SP	96.84 (± 0.197)	353.57 (± 0.945)	119.73 (± 0.612)	55.37 (± 0.5535)
UA	91.37 (± 0.397)	176.02 (± 0.759)	110.26 (± 1.212)	49.10 (± 1.1305)

the results of a single instance, Azure and Rackspace perform better than EC2, but all clouds have a lower performance compared to the base system.

Another important result is the relatively low variability, as can be observed in the confidence interval in Figure 4.4 and in Table 4.7. With these experiments, we observed a very low variability in all experiments. The only remark is the execution of LU in Rackspace, but still this rate may be acceptable.

For all benchmarks, the base system presented the fastest execution. However, the results differ greatly for different providers and benchmarks, and to conclude the analysis it is necessary to perform the economic evaluation using these results.

4.6.2 Economic Evaluation

The base system used to compare the IaaS OpenMP was the same than the used in the PaaS OpenMP evaluation. The economic evaluation was conducted considering the three and five years usage scenarios.

To calculate the cost efficiency factor, it is necessary to use the overhead factor and the price per hour of the providers, then the Equation 3.2 is applied. The overhead factor used is the calculated average of all the NPB-NAS tests executed in the performance evaluation. These values are shown in Table 4.8 and are 1.72 for Azure, 3.61 for EC2 and 1.87 for Rackspace. The other component of the cost efficiency factor is the price per hour of the cloud providers, these values were presented in Table 4.3 and are $0.64 for Azure, $0.66 for EC2 and $1.8 for Rackspace. The resulting values are in Table 4.8. Comparing these values with the price per hour of the base system, $0.484 for the three years and $0.301 for five years, all the providers present higher values than the base system. We can conclude that the three cloud providers have less cost efficiency than the base system, because they

Table 4.8: Cost efficiency factor and Break even point for IaaS with OpenMP.

Parameter	Azure	EC2	Rackspace
Overhead factor	1.72	3.61	1.87
Cost efficiency factor	1.10	2.38	3.37
Break even point - 3 years	160	74	52
Break even point - 5 years	100	46	33

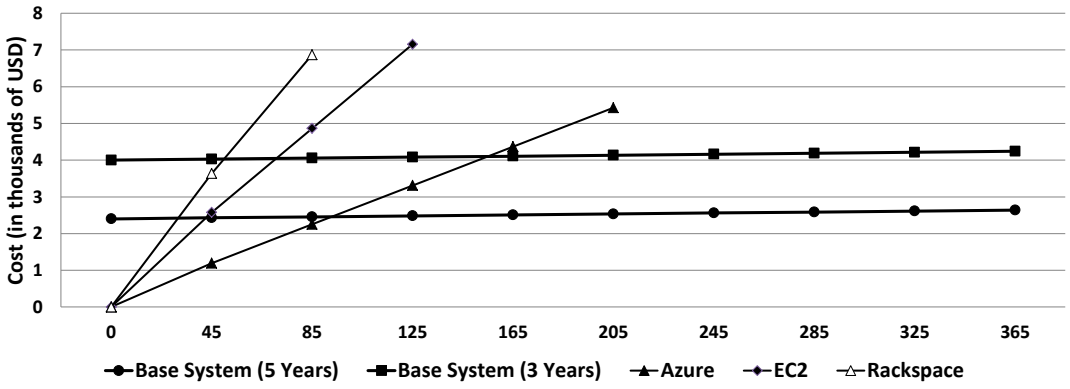

Figure 4.5: Break Even Point for IaaS with OpenMP

cost more to perform the same work than the base system in one hour. Azure has a higher cost efficiency than EC2 and Rackspace, while EC2 is more cost efficient than Rackspace.

However, the providers may also be considered as an alternative for HPC in a usage scenario where the application is not executed during 24 hours a day. To determine the number of days in which the providers have better cost-benefit than the base system, we calculate the break even point. To calculate it, it is necessary to use the cost efficiency factor of all the providers, also, the yearly cost of the base system. The yearly cost of the base system is the same $4,239.148 and $2,639.148 for the three and five years usage scenarios respectively. Applying the Equation 3.4 results the values shown in Table 4.8.

Figure 4.5 shows the break even point for IaaS using the OpenMP applications. Due to the higher cost efficiency factor, the Rackspace provider is the first provider to intersect the base system lines, the next provider that intersects these lines is EC2 and the last one is Azure. The best result is Azure in the three years usage scenario. In this case, the base system needs to be used continuously for more than 160 days to be more cost efficient than Azure. Even in the worst case, Rackspace in the five years scenario, it is necessary to use the base system during an entire month for the base system became to be more attractive.

4.7 Evaluationg IaaS with MPI Applications

The performance and economic evaluation of the IaaS providers regarding to MPI applications was performed using the same providers than the IaaS OpenMP evaluation. It was conducted using the MPI version of NPB-NAS with the problem size class B, and the DC test was not included for the same reasons.

The performance evaluation for IaaS using MPI was executed with three different configurations, with a single node, using two nodes and using four nodes. The main purpose was, apart from performance degradation, to observe the scalability of the cloud providers. The system is considered scalable if it will remain effective when the number of resources and users are increased (COULOURIS et al., 2011). In this evaluation the number of processes represents the users and the resources are the number of machines and cores available. For example, if we have a machine with eight cores, we use eight MPI processes, then when we have two machines with eight cores each, we use sixteen processes. The purpose is to verify if the cloud providers present an adherent behavior than the base system.

Table 4.9: Mean and confidence interval for IaaS with MPI using one node (seconds).

Test	Azure	EC2	Rackspace	Base system
BT	163.19 (± 0.538)	205.97 (± 0.218)	162.27 (± 0.194)	70.77 (± 0.300)
CG	70.12 (± 0.306)	91.98 (± 0.217)	40.73 (± 0.049)	14.08 (± 0.100)
DT	15.38 (± 0.157)	11.04 (± 0.183)	17.44 (± 0.256)	3.97 (± 0.061)
EP	17.18 (± 0.076)	14.17 (± 0.019)	16.21 (± 0.016)	11.35 (± 0.032)
FT	38.37 (± 0.088)	59.75 (± 0.069)	30.68 (± 0.018)	16.36 (± 0.024)
IS	2.68 (± 0.051)	3.40 (± 0.011)	1.64 (± 0.009)	0.79 (± 0.015)
LU	102.76 (± 0.251)	136.91 (± 0.650)	82.75 (± 0.062)	43.65 (± 0.154)
SP	234.05 (± 0.145)	392.53 (± 0.016)	228.02 (± 0.009)	95.12 (± 0.016)
MG	7.55 (± 0.453)	12.64 (± 0.227)	5.52 (± 0.237)	2.65 (± 0.664)

4.7.1 Performance Evaluation

There were executed 50 runs for each benchmark in all the three configurations. The mean and confidence interval results for one node are represented in a normalized form in Figure 4.6, and the absolute values are shown in Table 4.9. All the benchmarks using eight MPI processes, excepting BT and SP that need the number of processes being a power of two, in this case the number of processes for both are nine (3^2).

We can observe similar behavior related to the performance overhead compared to the IaaS OpenMP experiments. However, when considering the average execution time, the Rackspace was the best cloud provider.

The variability is low again, as seen in the confidence interval in the graph and in Table 4.9. The biggest variability was observed in the base system during the execution of the DT test. It presents a variability rate of 5.47% but this value can be considered as accepted.

When we move to the next configuration using two nodes, the behavior of the experiments vary greatly. In this configuration all the tests were executed using sixteen MPI

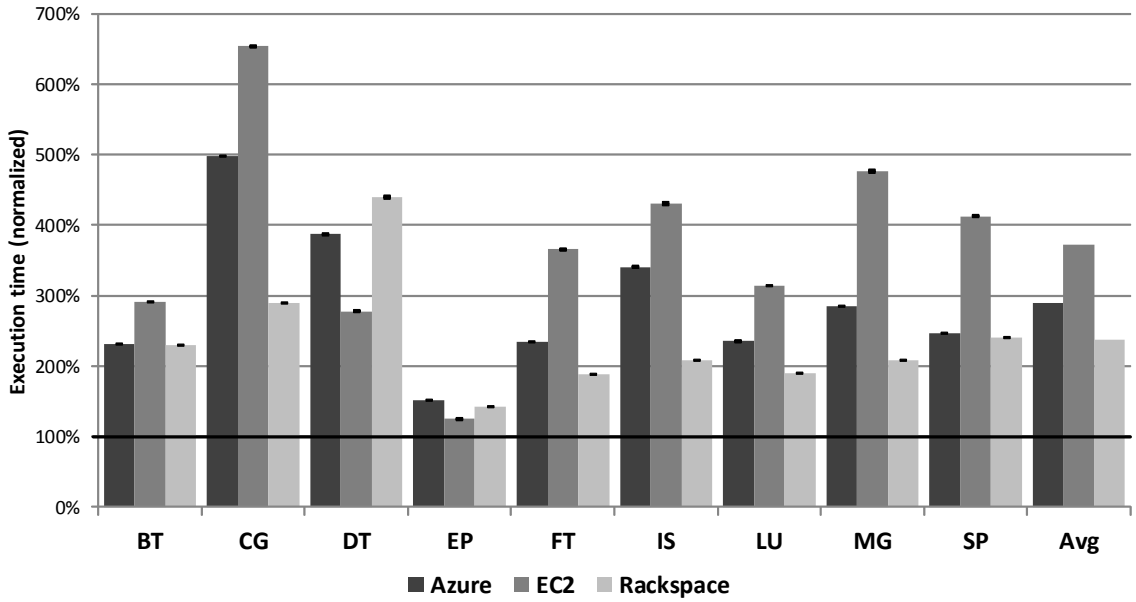

Figure 4.6: Performance results for IaaS with MPI using one node

Table 4.10: Mean and confidence interval for IaaS with MPI using two nodes (seconds).

Test	Azure	EC2	Rackspace	Base system
BT	69.16 (± 0.705)	127.13 (± 0.689)	181.17 (± 1.693)	44.94 (± 0.183)
CG	47.74 (± 1.525)	144.06 (± 0.843)	231.15 (± 4.683)	39.27 (± 0.102)
DT	11.46 (± 0.128)	9.00 (± 0.166)	12.26 (± 0.142)	4.63 (± 0.045)
EP	9.34 (± 0.035)	7.41 (± 0.048)	8.17 (± 0.028)	5.79 (± 0.053)
FT	36.78 (± 0.788)	62.56 (± 0.147)	74.85 (± 0.865)	36.77 (± 0.114)
IS	3.62 (± 0.257)	6.69 (± 0.037)	15.86 (± 0.265)	3.23 (± 0.011)
LU	45.81 (± 0.501)	101.15 (± 0.264)	88.00 (± 0.950)	30.16 (± 0.286)
MG	3.02 (± 0.181)	10.39 (± 0.029)	12.02 (± 0.357)	2.23 (± 0.051)
SP	85.51 (± 0.902)	236.19 (± 0.615)	326.41 (± 2.876)	62.50 (± 0.578)

processes, with eight executing on each machine. These results are shown, in a normalized view, in Figure 4.7 and the absolute values are shown in Table 4.10.

Compared with the behavior of two nodes configuration with the one node configuration, the performance of Azure became closer than the base system, presenting a degradation of 46%. In the first configuration the performance degradation of Azure was 189%, so we can conclude that Azure presents very good scalability. Amazon also presents an improved result, in the first configuration it has a degradation of 271% and in this configuration the degradation was 180%, but this rate still is an unacceptable rate. Rackspace presents an unexpected behavior, in the first configuration it presents the best rates with a performance degradation of 137%, and in this configuration its degradation was 282% that is the biggest degradation observed in all our experiments.

These results mean that the Azure and EC2 have better scalability than the base system, in other words, the performance of the network used in these providers seems to be better performance than the Gigabit Ethernet used in the base system. Because in this configuration there are communication between the processes using the network.

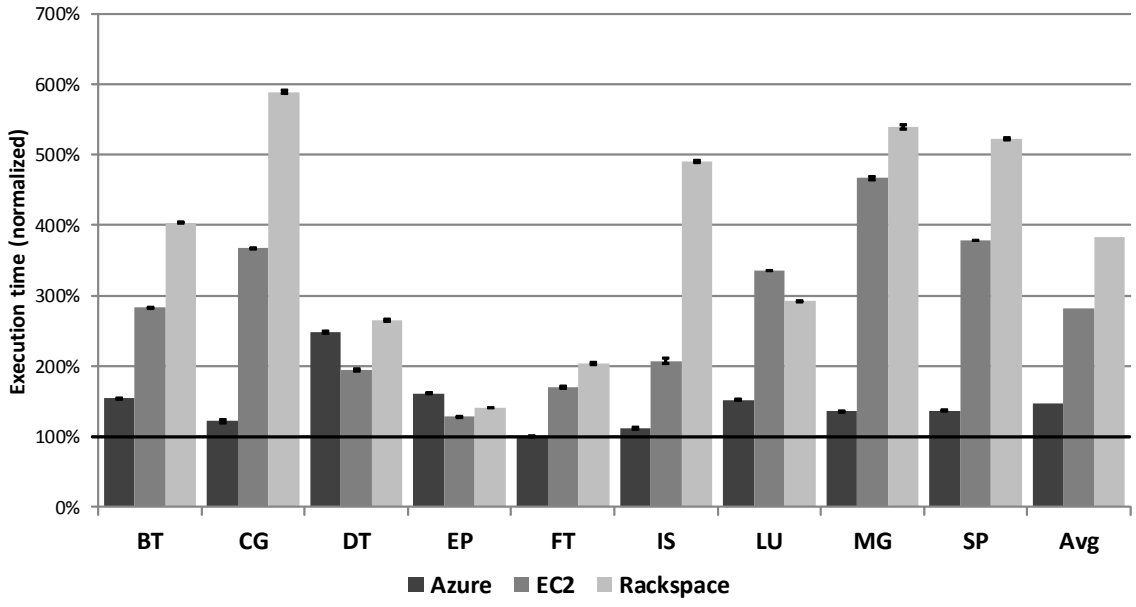

Figure 4.7: Performance results for IaaS with MPI using two nodes

Table 4.11: Mean and confidence interval for IaaS using OpenMP using four nodes (seconds).

Test	Azure	EC2	Rackspace	Base system
BT	70.02 (± 0.571)	152.66 (± 0.833)	186.91 (± 2.156)	124.90 (± 0.493)
CG	54.38 (± 0.932)	213.76 (± 1.147)	362.96 (± 3.913)	254.89 (± 2.524)
DT	11.52 (± 0.156)	9.82 (± 0.260)	11.69 (± 0.224)	5.78 (± 0.046)
EP	4.94 (± 0.029)	3.85 (± 0.067)	4.33 (± 0.043)	3.05 (± 0.070)
FT	38.95 (± 0.272)	70.77 (± 0.457)	60.75 (± 0.687)	61.81 (± 0.476)
IS	3.53 (± 0.136)	10.84 (± 0.193)	19.86 (± 0.358)	18.16 (± 0.366)
LU	40.79 (± 0.545)	110.70 (± 0.415)	89.74 (± 0.880)	60.00 (± 1.509)
MG	3.42 (± 0.293)	17.18 (± 0.089)	17.70 (± 0.395)	14.93 (± 0.351)
SP	114.60 (± 0.604)	272.45 (± 1.221)	370.74 (± 4.373)	246.99 (± 0.836)

The variability observed in the graph from Figure 4.7 and in Table 4.10 is low for all the experiments. And is totally acceptable for all the providers.

The last configuration is the execution using four nodes. The results of the this configuration are seen in Figure 4.8 that represents the normalized mean execution time and the confidence interval for all the benchmarks using 32 MPI processes, excepting BT and SP that need the number of processes being a power of 2, for these two, the number of processes are 36 (6^2). Also, the absolute results are shown in Table 4.11.

All the tests, except DT and EP, executes faster in Azure than in the base system. The CG, IS and MG executes about five times faster in Azure than the base system.

CG and MG have their focus on communication and IS has the focus in integer and floating point performance with a medium amount of communication. Due to this, we can conclude that the network used in Azure performs better than the Gigabit Ethernet used in the base system. EC2 also presents better performance than the cluster for CG and IS

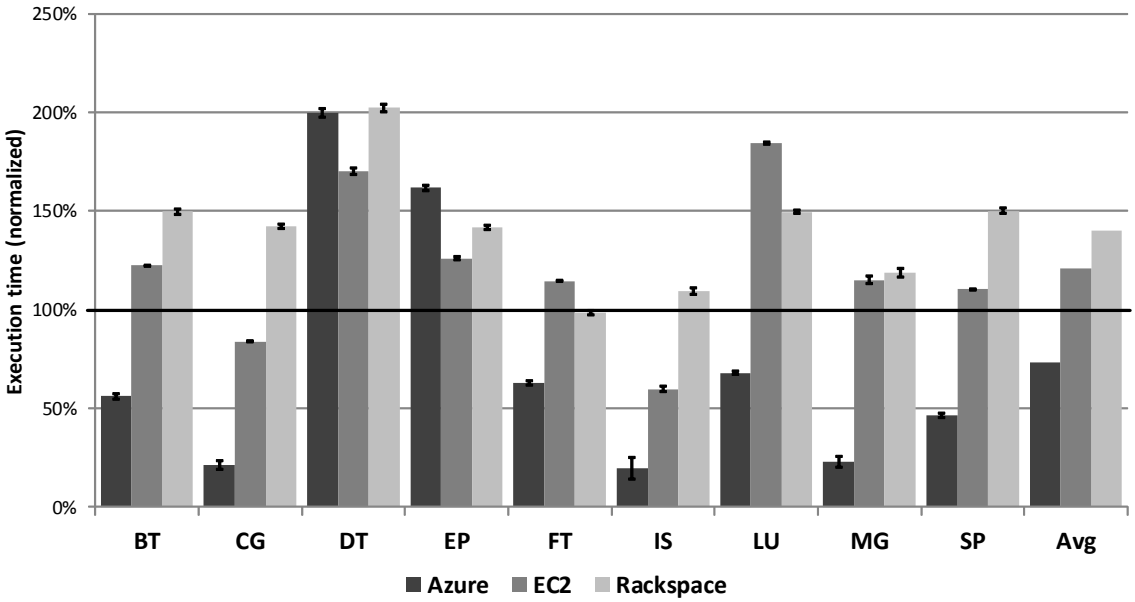

Figure 4.8: Performance results for IaaS with MPI using four nodes

and the same assumptions can be made. The Rackspace has about the same performance than the base system in the FT test.

In terms of scalability all the cloud providers presents very good rates, even Rackspace that has presented the worst overhead using two nodes. On average, Rackspace and EC2 present acceptable degradation rates, of 40% and 20% respectively. However, Azure presents better performance than the base system in average, Azure has 30% better performance than the cluster. This means that the execution made in Azure is faster than in the base system.

The variability is low again and does not represent a problem in any of the environments, as seen in the performance graphs from Figure 4.8 and in Table 4.11. We can conclude that all the clouds scale better than the base system, at least in the configurations used in this evaluation.

4.7.2 Economic Evaluation

We calculated the economic evaluation taking into account the three configurations. The base system used to compare the IaaS with MPI was also the same, than in the other two evaluations, the difference in this evaluation is that more than one node was considered. When we are using more than one node, we need to multiply the cost per hour by the number of instances in use. We conducted the economic evaluation taking into account the same three and five years usage scenarios.

The overhead factor used is the average of all the tests executed in the performance evaluation, for the three scenarios. These values are shown in Table 4.12. The cost of the clouds are \$0.64, \$1.28 and \$2.56 for Azure; \$0.66, \$1.32 and \$2.64 for EC2 and \$1.8, \$3.6 and \$7.2 for Rackspace, in the one, two and four nodes configurations respectively. Using these values and the Equation 3.2, the cost efficiency factor is calculated for the three providers in the three configurations, the results are shown in Table 4.12. The costs per hour of the base system are \$0.484 for one node, \$0.968 for two nodes and \$1.936 for four nodes in the three years usage scenario, and \$0.301 for one node, \$0.602 for two nodes and \$1.204 for four nodes in the five years usage scenario. Comparing the cost efficiency factor with the price per hour of the base system, we can conclude that in the one

Table 4.12: Cost efficiency factor and Break even point for IaaS with MPI.

Parameter		Azure	EC2	Rackspace
Overhead factor	1 node	2.89	3.71	2.37
	2 nodes	1.46	2.81	3.82
	4 nodes	0.73	1.20	1.40
Cost efficiency factor	1 node	1.85	2.45	4.27
	2 nodes	1.88	3.71	13.78
	4 nodes	1.87	3.19	10.09
Break Even Point - 3 years	1 node	95	72	41
	2 nodes	188	95	26
	4 nodes	377	222	70
Break Even Point - 5 years	1 node	59	45	26
	2 nodes	117	59	16
	4 nodes	235	138	44

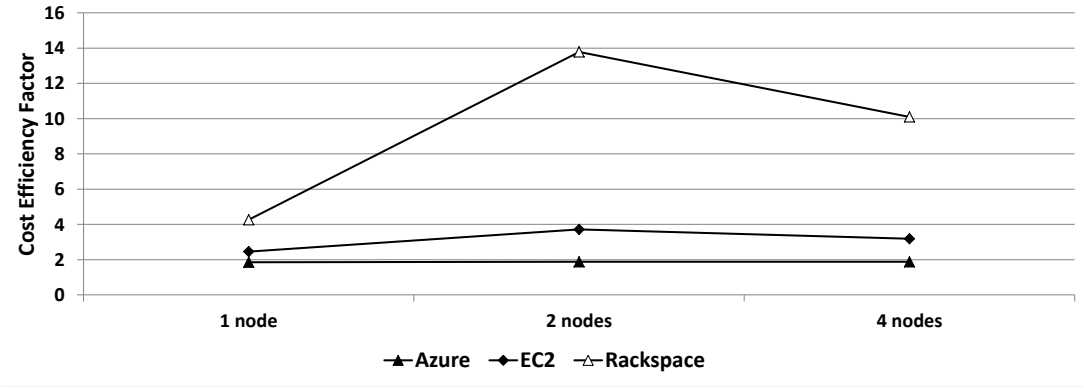

Figure 4.9: Cost Efficiency Factor for MPI

node configuration, the three providers present less cost efficiency than the base system and the Azure is the best provider for this configuration, followed by EC2 and Rackspace. In the two nodes configuration, again the providers have less cost efficiency than the base system, also, the cost efficiency factor of the EC2 and Rackspace present a higher increase, this means that these two cloud providers do not have a constant scalability rate, however, Azure presents a constant rate. In the four nodes configuration, the EC2 and Rackspace presented less cost efficiency than the base system, but in the three years usage scenario, Azure presents a cost efficiency factor smaller than the cost per hour of the base system, this means that Azure in this scenario has better cost-benefit than the base system, also, the scalability of Azure is constant and the EC2 and Rackspace have a slightly better rate.

Figure 4.9 shows the cost efficiency factor for the three scenarios, we can observe that the Azure behavior is a constant meaning that Azure presents constant scalability, EC2 behavior is not so constant than Azure, but still may be accepted and the scalability of Rackspace is totally indeterministic.

The three providers may also be considered as an alternative for HPC according with the usage scenario. To determine when to use a cloud instead the base system we need to calculate the break even point. To calculate the break even point, it is necessary to use the cost efficiency factor of the providers and the yearly cost of the base system for the three configurations. The yearly costs of the base system are $4,239.148 and $2,639.148 for the three and five years in the one node configuration, $8,478.296 and $5,278.296 in the two node configuration and 16,956.592 and $10,556.592 in the four node configuration. Using the Equation 3.4 the resulting values are shown in Table 4.12. Figure 4.10 shows the break even point for the IaaS using MPI in the one node configuration. The first provider that intersects the base system lines is Rackspace, due to its high cost efficiency factor, the next provider that intersects these lines are the EC2 and the last one is Azure. Figure 4.11 shows the break even point for the IaaS using MPI in the two nodes configuration. The base system lines are intersected first by Rackspace, then EC2 and Azure. The bad scalability of Rackspace can also be observed in these results, both Azure and EC2 presented better results for break even point, but Rackspace has worst results, this can be explained by the combination of the worst performance in the two node configuration and the higher price of Rackspace instance. Figure 4.12 presents the results of the four nodes configuration. Rackspace and EC2 presented the same behavior, intersecting the base system lines in the same order, but Azure in the three years usage scenario does not

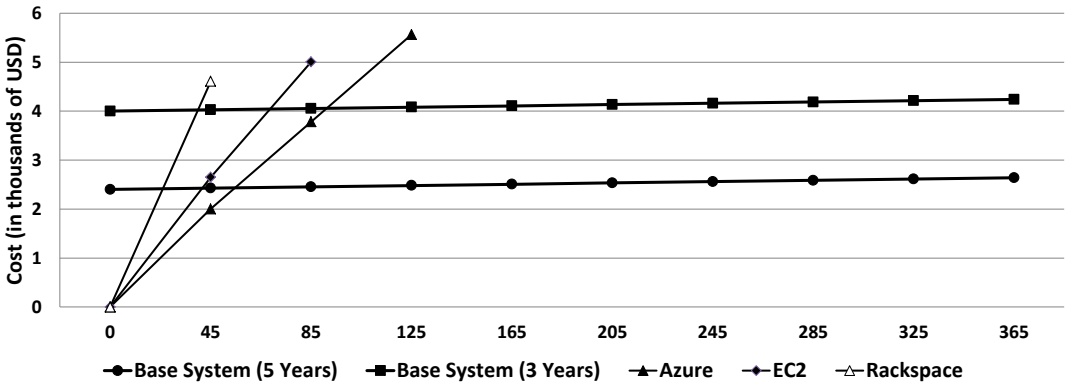

Figure 4.10: Break Even Point for IaaS with MPI using one node

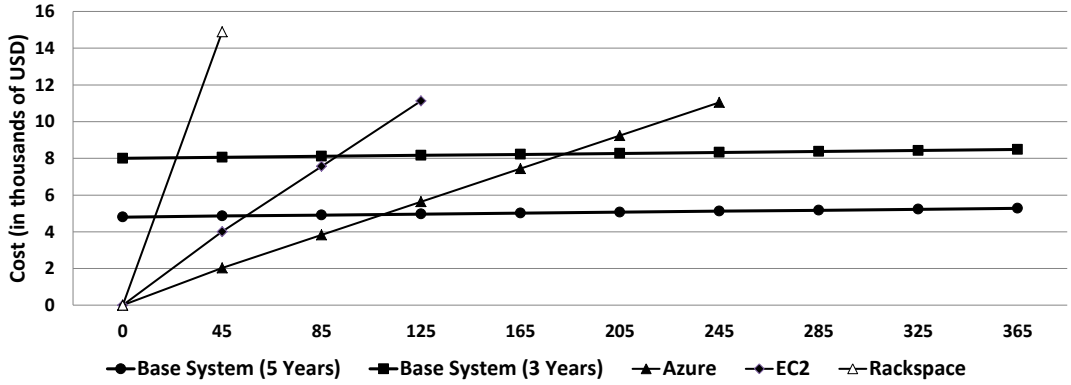

Figure 4.11: Break Even Point for IaaS with MPI using two nodes

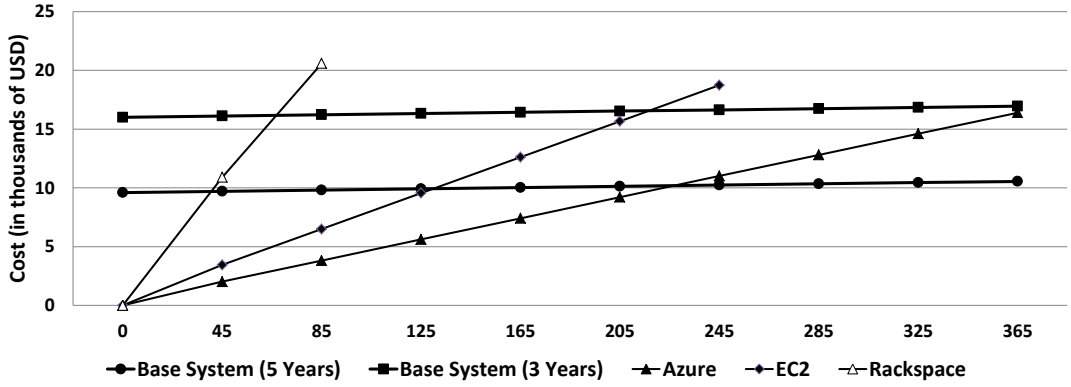

Figure 4.12: Break Even Point for IaaS with MPI using four nodes

intersect the base system line at all, this means that is always better to use Azure instead the base system in this usage scenario.

4.8 Concluding Remarks

Based on the study conducted in this chapter, we can conclude that PaaS is not a recommended environment for HPC. Despite the fact that it has acceptable performance rates and is attractive in terms of cost, there are two main problems with this model. The first problem is the porting process of an application, the process is long and depending on

the application behavior can be complex. However, the main problem is the incapability to execute MPI programs in more than one node. This is a crucial requirement for a general purpose HPC environment.

IaaS is the recommended model to be used to create a general purpose HPC environment. It is possible to create HPC environments into the IaaS providers without huge efforts. All the providers analyzed presents adequate machine sizes and operating systems support. The variability of the executions is not a problem in all the providers, because it is about the same than in the base system. The cost-benefit to use a cloud instead of the base system is attractive in many scenarios. There is performance degradation in the major of scenarios, but still they are good alternatives because they present a good cost-benefit.

5 CONCLUSIONS

This thesis has presented a study regarding to identify if cloud computing is a feasible environment to be used for HPC. The general conclusion is that it is indeed possible to use cloud computing for HPC, in certain conditions. The majority of the HPC applications are programmed using the OpenMP or MPI models, and the cloud shows the capability to execute both of them.

Cloud computing defines three service models, SaaS, PaaS and IaaS, and four implementation models, private, public, community and hybrid. The SaaS service model cannot be used because it does not permit customizations. The service models that can be adopted in this case are both PaaS and IaaS, but just IaaS provides the desired flexibility and it is the model to be adopted. In terms of the implementation model, the only model that offers unlimited elasticity and no upfront costs is the public model. During the analysis of the Windows Azure PaaS provider, it presented acceptable, in some cases, performance degradation about 50% and also acceptable cost-benefit. But due to the impossibility of use this type of service to configure parallel machines and the amount of efforts needed to port an application, this service model was not considered a feasible environment to be used for HPC. So, the conclusion is that the cloud configuration to be used for HPC is the IaaS in a public implementation.

This configuration was evaluated using the major cloud providers, Microsoft Windows Azure, Amazon EC2 and Rackspace. There were evaluated the deployment procedures and the three providers presents adequate capabilities to configure environments to be used for HPC. The performance of the three providers was evaluated using the well know NPB-NAS suite of benchmarks, the results were compared with a base system to measure the overhead. Four configurations were used, the OpenMP version was evaluated using one node, also the MPI version was evaluated in three different configurations, with one, two and four nodes.

The cloud provider that presents the best results in terms of performance was the Windows Azure, in a scenario using four nodes executing the MPI version of the benchmarks, in this case the average execution time of Azure was 30% better than the base system. The economic evaluation was performed applying the cost model created during this thesis. These results show that cloud computing is a real alternative for HPC environments, because it was observed that, according with the usage pattern, is cheaper to use a cloud instead of the base system. Again, the provider that delivers the best cost-benefit was Windows Azure. Considering a scenario where the base system will be used for five years, Windows Azure has better cost-benefit than the base system, if the base system will be used for less than 327 days for a year, those considering a 24 hour a day usage.

The general conclusion is that cloud computing can be used for HPC, it provides better performance, in case of Azure, and presents an interesting alternative in economic perspective.

5.1 Contributions

The major contribution of this thesis was the methodology to perform the evaluation of cloud computing as an environment for HPC (ROLOFF et al., 2012). This methodology involves the analysis of deployment procedures, performance evaluation and the economic evaluation. The economic evaluation is performed using the cost model developed during this thesis. This model is composed of two metrics, the cost efficiency factor and the break even point.

The cost efficiency metric represents the cost to realize the same amount of work using a cloud instead of a real machine. This information is useful to determine the cost-benefit of a cloud provider. Moreover, it can be used to determine the amount of money needed to perform a research. This can be applied in estimates for a project request for proposals. This metric helps the users to decide if they need to buy a real machine or use the services of cloud computing.

The break even point determines the number of days of continuous usage where is cheaper to use a cloud instead to buy a real machine. This metric is especially useful in scenarios where the usage will be temporary. Our experiments show that in many cases, is cheaper to use a cloud instead to buy a real machine. Even in the worst case, use a cloud during an entire month is cheaper than use a traditional system. Then, this metric helps the decision process of choosing a cloud or a machine.

5.2 Future Work

The economic evaluation can be improved, because the current cost model was simplified to make it possible to validate the methodology. It is necessary to adapt the cost model to make it compatible with standard accounting rules.

Due to the results obtained in this thesis, we will port a real scientific application to the cloud to perform the evaluation using it. With this porting process, we can validate the methodology proposed and also obtain new data to be analyzed.

APPENDIX A SUMMARY IN PORTUGUESE

In this appendix, we present a summary of this master thesis in the portuguese language, as required by the PPGC Graduate Program in Computing.

Neste apêndice, é apresentado um resumo desta dissertação de mestrado na língua portuguesa, como requerido pelo Programa de Pós-Graduação em Computação.

A.1 Introdução

A computação em nuvem é um novo modelo de computação que apresenta alternativas interessantes para problemas recorrentes a aquisição e manutenção de sistemas computacionais. As principais razões para isso, são a capacidade de criação de ambientes elásticos em provedores públicos e o modelo de *pague por uso*, onde o usuário só é cobrado pelo uso efetivo que for feito dos recursos. Desde as primeiras aparições de computação em nuvem, esse modelo tem atraído a atenção da comunidade científica (FOSTER et al., 2008) (BUYYA et al., 2009) (ARMBRUST et al., 2009).

Ao contrário de sistemas tradicionais, não há a necessidade de investimentos iniciais para se ter acesso a recursos computacionais. Devido também as características de elasticidade e do modelo de cobrança, os custos de manutenção tendem a ficar próximos a zero quando as máquinas não estiverem em uso. Também os custos relativos as instalações, como *datacenter*, eletricidade, resfriamento, são drasticamente reduzidos. Além dos recursos disponíveis serem virtualmente ilimitados, a grande motivação de mudar os ambientes tradicionais para a nuvem é a grande redução de custos com o aumento da escalabilidade.

Processamento de alto desempenho (PAD) exige a utilização de um grande número de recusrsos computacionais, e para ser ter acesso a essa quantidade de recursos, são necessários investimentos. O grande problema é o planejamento de tais investimentos, pois caso os mesmos sejam superestimados, haverá ociosidade de recursos e caso sejam sub-avaliados, haverá a situação de escassez de recursos. Além dos custos de manutenção e instalações que são necessários para se ter um ambiente de alto desempenho. Cabe ressaltar que esses custos são tanto para ambientes de alto desempenho como para ambientes tradicionais de computação.

As características de computação em nuvem, tornam esse modelo uma alternativa interessante para ser usada para a construção de ambientes de processamento de alto desempenho. Mas ainda existem problemas que devem ser estudados antes da sua adoção para esse tipo de ambiente, além das capacidades de criação desse tipo de ambiente em si que devem ser possíveis de serem reproduzidas em ambientes de computação em nuvem. Dessa maneira, essa dissertação pretende estudar os conceitos de computação em nuvem para identificar se o modelo, ou parte dele, pode ser utilizado para alto desempenho.

A análise será feita levando-se em conta três aspectos principais que são: capacidades de criação de ambientes de alto desempenho; análise do desempenho desses ambientes comparados com um sistema tradicional e análise da viabilidade econômica do uso de computação em nuvem ao invés de sistemas tradicionais.

A.2 Computação em Nuvem

Computação em nuvem é um novo paradigma de computação que foi desenvolvido com a combinação e evolução de computação distribuída e virtualização com contribuições de computação em grade e também de computação paralela. Houveram várias esforços para a definição desse paradigma, mas somente em 2011 com a publicação da definição do NIST [1] que consolidou essas diversas iniciativas e vem sendo adotada como padrão desde então. A definição de computação em nuvem do NIST compreende cinco características essenciais, três modelos de serviço e quatro modelos de implementação. O objetivo dessa seção é apresentar essa definição, bem como elencar alguns dos problemas que ainda estão abertos nesse novo paradigma.

A.2.1 Características Essenciais

Um serviço de computação em nuvem deve apresentar as seguintes características para ser considerado aderente a definição do NIST.

- Auto serviço sob demanda.
- Acesso através da rede.
- Conjunto de recursos ilimitado na visão do usuário.
- Rápida elasticidade.
- Serviço controlado e medido.

O sumário dessas cinco características pode ser interpretado como a capacidade da criação de ambientes elásticos, onde o usuário pode criar ambientes de qualquer tamanho instantaneamente e o modelo de pagar para usar onde somente os recursos que foram utilizados efetivamente são cobrados.

A.2.2 Modelos de Serviço

A definição do NIST abrange três modelos de serviço que são: infraestrutura como serviço (*IaaS*), plataforma como serviço (*PaaS*) e software como serviço (*SaaS*). Cada um desses modelos de serviço oferece um diferente nível de abstração e de serviços para o usuário.

O modelo de IaaS representado na Figura 2.1 é o modelo que oferece ao usuário o acesso a maquinas virtuais, tais máquinas são totalmente controladas pelo usuário. É possível que se escolha o sistema operacional a ser instalado na mesma, bem como todas as configurações pertinentes a ele, além de total liberdade na instalação de softwares que o usuário julgue que sejam necessários. O provedor é responsável por manter o parque de hardware atualizado bem como pode monitorar o serviço para garantir os níveis de qualidade de serviço definidos em contrato entre as partes.

[1] National Institute of Standards and Technology

O modelo de PaaS representado pela Figura 2.2 é um modelo no qual o usuário tem acesso a um ambiente de execução de aplicações, ambiente esse que é totalmente definido pelo provedor no que tange ao modelo de execução, linguagens de programação suportadas e bibliotecas e ferramentas disponíveis. O usuário pode desenvolver uma aplicação, seguindo as regras do provedor, ou pode também adquirir uma aplicação pronta dentro desse modelo para que utilize essa plataforma somente para a execução da mesma.

No modelo de SaaS que está representado na Figura 2.3 o usuário simplesmente tem acesso a uma aplicação já pronta e mantida por um provedor, nesse modelo o usuário tem somente a capacidade de utilização dessa aplicação. Não há a capacidade de customização dessa aplicação, somente alguns parâmetros pré-definidos podem ser ajustados de acordo com a necessidade do usuário. Uma grande vantagem desse modelo é que não há qualquer necessidade do usuário em possuir e manter infraestrutura para execução dessa aplicação.

A.2.3 Modelos de Implementação

Os três modelos de serviço apresentados podem ser implantados utilizando-se um dos quatro modelos de implantação definidos: privado, comunitário, público e híbrido. Cada um desses modelos apresenta características relativas ao controle de exeução e também relativas a escalabilidade disponível.

No modelo privado representado pela Figura 2.4 uma organização possui os recursos computacionais e decide controlá-los através da utilização de uma plataforma de nuvem. Nesse modelo o usuário tem total controle sobre a execução da aplicação e também sobre os dados, já que ambos são manipulados dentro do ambiente da organização o nível de controle atingido é total. O problema com esse modelo é a ausência de escalabilidade, pois a quantidade de recursos disponíveis para os usuários é limitado pelo hardware disponível.

O modelo comunitário, representado na Figura 2.5, é semelhante ao modelo privado, mas nesse caso ao invés de somente uma organização ser usuária dos recursos privados existe um conjunto de organizações que são usuárias desse recursos. As questões de controle de dados e de escalabilidade são as mesmas do modelo privado.

No modelo público é introduzido o papel do provedor de serviços e está representado na Figura 2.6, nesse modelo o usuário não possui qualquer recurso computacional e se utiliza dos recursos do provedor no modelo de pague para usar. Nas questões de segurança, como o usuário não tem controle físico sobre o ambiente no qual estão sendo executadas suas aplicações não há a garantia de que esse ambiente é totalmente seguro e isolado, fazendo com que seja necessário tomar um certo cuidado na hora de decidir quais dados serão enviados para a nuvem. Em termos de escalabilidade ela é virtualmente ilimitada e garantida pelo provedor.

O modelo híbrido (Figura 2.7) é uma combinação entre quaisquer dos outros três modelos, mas em termos práticos ele será utilizado quando uma organização tiver um conjunto de recursos privados sendo controlados por uma plataforma de nuvem interligada a um provedor público Quando houver necessidade de maior poder computacional ao invés dessa organização adquirir mais recursos privados, sua plataforma de nuvem aloca recursos do provedor público de maneira transparente ao usuário.

A.2.4 Problemas Existentes

Apesar do modelo de computação em nuvem demonstrar ser uma alternativa interessante para ser utilizado para substituir o modelo tradicional de computação, existem alguns pontos em abertos nesse modelo que devem ser resolvidos antes de sua adoção

em massa por parte de organizações. Os principais pontos em aberto são: segurança, portabilidade e interoperabilidade, continuidade e desempenho.

Em termos de segurança, quando se utiliza um provedor de nuvem pública não é possível que se garanta o mesmo nível de segurança do que se obtém em um ambiente tradicional. Dessa maneira a manipulação segura dos dados não é garantida e pode apresentar vazamentos. Outro ponto de atenção também é referente a disponibilidade dos dados, pois como normalmente os provedores públicos são acessados através da Internet, possíveis falhas nesse acesso podem trazer problemas na disponibilidade dos dados. A replicalção de dados entre locais distintos também é uma preocupação dentro de ambientes públicos, pois em caso de desastres devem existir procedimentos para a rápida disponibilização desses dados.

Questões de portabilidade e interoperabilidade são relativas a portabilidade de aplicações, ou configurações de ambientes inteiros, entre provedores distintos. Quando se optar pelo uso da computação em nuvem, deve-se tomar o cuidado de escolha de um provedor que siga padrões públicos de configuração, pois dessa maneira a migração de ambientes entre provedores distintos é facilitada. Questões de interoperabilidade são relativas ao uso de diversos provedores públicos ao mesmo tempo, sendo estes compatíveis entre si possibilitando que o usuário utilize os mesmos procedimentos em qualquer deles, novamente deve-se optar por provedores que implementem os padrões atualmente definidos de interoperabilidade.

A continuidade é relativa a capacidade do provedor em garantir um determinado número de falhas físicas, seja em algum servidor ou mesmo em um local inteiro de suas instalações. Por fim, como ambientes de nuvem utilizam-se muito de virtualização é esperada uma perda de desempenho frente a máquinas físicas, tanto em termos de processamento como em termos de desempenho de comunicação, portanto o usuário deve ter esse conhecimento e procurar utilizar provedores que utilizem mecanismos de virtualização que causem o mínimo impacto nas máquinas físicas, oferecendo um melhor desempenho geral.

A.3 Proposta para Avaliação de Computação em Nuvem para uso em Processamento de Alto Desempenho

Processamento de alto desempenho apresenta demanda por grandes conjuntos de recursos computacionais para executar suas tarefas de maneira mais rápida, tradicionalmente mais poder computacional é obtido através da aquisição de mais recursos físicos. Os dois paradigmas mais utilizados para a programação de aplicações paralelas de alto desempenho são o paradigma de memória compartilhada e o paradigma de memória distribuída, as ferramentas utilizadas para esses paradigmas são OpenMP e MPI respectivamente. As linguagens de programação mais utilizadas em PAD são Fortan e C/C++, portanto nossa proposta deve levar em conta tanto as ferramentas OpenMP e MPI, bem como as linguagens de programação Fortran e C/C++. O objetivo desse trabalho é avaliar o paradigma de computação em nuvem quanto a sua capacidade de criação de ambientes para processamento de alto desempenho, avaliando aspectos de *deployment*, desempenho e custo-benefício.

Após a apresentação dos modelos de serviço, é necessário que seja verificado quais dos três, SaaS, PaaS e IaaS, possuem as características necessárias para que possam ser avaliados como alternativas para PAD. O modelo SaaS apresenta aplicativos prontos para serem usados pelos usuários sem a capacidade de customização, como usuários de PAD,

além de executarem aplicações, trabalham no desenvolvimento de suas aplicações e neste modelo esse desenvolvimento é impossível, o modelo de SaaS foi descartado como um possível modelo para utilização para PAD e dessa maneira não foi incluído dentro de nossa análise. Já o modelo de PaaS apresenta mais flexibilidade do que o SaaS, mas ainda assim apresenta suas limitações. Com ele não é possível se criar uma máquina paralela, já que ele utiliza o modelo de execução de páginas *web*, onde cada requisição é tratada de maneira separada não podendo se utilizar de uma máquina paralela, onde se juntam diversas máquinas para processar uma grande tarefa. Mas devido ao grande poder computacional oferecido por somente uma instância, é possível que se utilize o OpenMP para a programação de aplicações PAD nesse modelo, dessa maneira ele foi incluído em nossa avaliação com essa restrição. O modelo de IaaS oferece recursos de computação na forma de máquinas virtuais, sob as quais podem ser feitas quaisquer configurações, inclusive a criação de máquinas paralelas, portanto o modelo de IaaS foi incluído na avaliação tanto para validação de OpenMP como de MPI.

Após a escolha dos modelos de serviço a serem avaliados, é necessário que se escolham quais dos modelos de implementação devem ser avaliados. Os modelos privado e comunitários são semelhantes, pois em ambos os recursos computacionais são de propriedade de uma organização ou de um conjunto de organizações, tais recursos são utilizados por usuários que possuem algum vínculo com essas organizações. Nesses casos, os recursos são limitados pela quantidade de recursos disponíveis e não há elasticidade por conta dessa limitação, em uma simplificação, podemos assumir que ambos os modelos são o mesmo do que possuir recursos tradicionais, clusters e grades computacionais, e não representam efetivamente um novo modelo de computação, devido a isso os dois modelos foram descartados de nossa proposta de avaliação. O modelo público apresenta as características de elasticidade e do modelo de pegue para usar em sua totalidade, dessa maneira esse modelo de implementação é considerado realmente um novo modelo de computação e por isso ele foi incluído em nossa avaliação. O modelo híbrido é utilizado para estender um conjunto de máquinas que já existem dentro de alguma organização, evitando assim que sejam feitas novas aquisições para esse propósito, mas em um cenário realístico a conexão entre as máquinas locais e o provedor público se configuraria como um gargalo, por isso, quando uma máquina maior do que a quantidade de recursos locais for necessária ela possivelmente será alocada inteiramente no provedor puúblico, o que acaba se torando o modelo público, portanto o modelo híbrido também é descartado de nossa avaliação.

Dessa maneira nossa avaliação será feita em provedores públicos de PaaS e IaaS. As avaliações de procedimentos de *deploymente* e desempenho são largamente conhecidas pela comunidade, dessa maneira elas serão somente aplicadas dentro do contexto desse trabalho. Por outro lado, para a avaliação de custo-benefício, não há um modelo padrão a ser seguido, dessa maneira faz parte do escopo desse trabalho a criação de um modelo de avaliação de custo, tal modelo é apresentado em seguida. O modelo de custo que foi criado busca criar uma relação entre o desempenho e o custo dos provedores de nuvem em relação a sistemas tradicionais, para dessa maneira fazer uma comparação justa entre os dois paradigmas. Tal modelo é composto de dois cálculos, o primeiro é chamado de *Cost Efficiency Factor* (CEF) e o segundo de *Break Even Point* (BEP), ambos os cálculos comparam um sistema em nuvem contra um sistema tradicional que é chamado de sistema base.

O cálculo do CEF busca definir o custo de se realizar o mesmo trabalho computacional em um provedor de nuvem que um determinado sistema tradicional, sistema base, execute

em uma hora. Para realizar esse cálculo ele necessita de duas informações: o custo por hora do provedor de nuvem e o sobrecusto de se executar determinada tarefa no provedor de nuvem comparado com o sistema base, o cálculo do sobrecusto é feito através da equação 3.1 que resulta em um fator de sobrecusto. Com essas duas informações é feito o cálculo do CEF com a utilização da equação 3.2, o resultado desse cálculo pode ser interpretado da seguinte maneira: se o resultado for menor do que o custo por hora do sistema base, custo esse que deve ser calculado pelo usuário, o provedor de nuvem apresenta melhor custo-benefício do que sistema base. O CEF também pode ser utilizado como uma medida de escalabilidade, pois como ele é um fator entre o desempenho e o custo relacionados ao sistema base, com o aumento da quantidade de máquinas se ele se manter constante nos indica que sua escalabilidade é a mesma do sistema base.

O BEP representa, em número de dias equivalentes, quando o custo de se utilizar o provedor de nuvem é o mesmo do que comprar um sistema tradicional para executar a aplicação. Para se calular o BEP, são necessários o CEF do sistema atual, calculado anteriormente e também o custo anual do sistema base, esse custo anual deve ser calculado utilizando-se a equação 3.3, essa equação utiliza o custo de aquisição do sistema, dividido pelo tempo de uso do mesmo expresado em quantidade de anos, e considera também o custo de manutenção anual desse sistema, como valor resultante temos o custo anual do sistema base. O valor do BEP, então, é calculado utilizando-se a equação 3.4 que nos resulta a quantidade de dias equivalentes nos quais utilizar um provedor de nuvem é mais barato do que adquirir um sistema computacional tradicional.

A.4 Validação Experimental

A validação experimental consiste em definir os cenários de testes bem como as aplicações que serão executadas sob eles, após isso os procedimentos de *deployment* são analisados para todos os cenários, por fim as avaliações de desempenho e econômicas são realizadas.

A.4.1 Metodologia

Os cenários definidos para avaliação foram: (1) PaaS público utilizando OpenMP; (2) IaaS público utilizando OpenMP; e (3) IaaS público utilizando MPI. Para que essa avaliação seja efetuada é necessário que se defina qual será a carga de trabalho que será executada em todos esses ambientes, como nossa avaliação se trata de avalição de ambientes de nuvem para PAD, a escolha da suíte de *benchmarks* do NAS foi feita, pois essa carga de trabalho possui diversas aplicações que avaliam diversos aspectos inerentes ao processamento em ambientes computacionais e são frequentemente utilizados para esse tipo de avaliação. O NAS possui versões tanto para OpenMP como para MPI, ambas as versões foram utilizadas em nossa avaliação.

A escolha dos provedores foi feita com base na capacidade dos mesmos executarem o conjunto de aplicações do NAS, bem como a participação do mercado dos provedores. Para a avaliação de PaaS, o único provedor encontrado que possui capacidade para execução de aplicações escritas em Fortran e C foi o *Microsoft Windows Azure*, portanto a avaliação de PaaS foi feita somente utilizando-se esse provedor. Para a avaliação de IaaS, verificou-se que praticamente todos os provedores disponíveis possuem a capacidade de executar tanto as versões OpenMP quanto MPI do NAS, dessa maneira os critérios de seleção para esses provedores foram a participação no mercado e também os grupos de

usuários/desenvolvedores ativos, com base nesses critérios, chegou-se a três provedores: *Microsoft Windows Azure*, *Amazon EC2* e *Rackspace*.

Após a escolha dos provedores é necessário que sejam definidos as configurações dos ambientes de teste, bem como o sistema tradicional (sistema base) que será utilizado para comparar os provedores de nuvem. A máquina utilizada como sistema base é um conjunto de quatro nós que forma um dos clusters do Grupo de Processamento Paralelo e Distribuído (GPPD), cada nó desse cluster consiste em uma máquina com processador de 8 núcleos, bem como 12 Gigabytes de memória principal, a interconexão de rede entre os quatro nós é feita com uma rede Giga-Ethernet padrão. Para comparação com esse sistema base, foram escolhidas as configurações mais próximas disponíveis nos provedores de nuvem, em todos foi possível configurar uma instância de máquina com 8 núcleos e quantidade de memória suficiente para executar todas as aplicações do NAS.

A.4.2 Procedimentos de *Deployment*

Os procedimentos de *deployment* foram feitos em separado para os dois modelos de serviço analisados, PaaS e IaaS. Para se portar as aplicações do NAS para o Azure, foram necessários uma série de alterações e adaptações tanto nos códigos de entrada e saída como no modelo de execução da mesma, o sumário das etapas pode ser visualizado na Figura 4.1 que divide o processo de porte em três etapas distintas. A primeira etapa é referente ao porte das aplicações do modelo de execução do Unix para o modelo de execução do Windows, pois o Azure PaaS se utiliza somente de Windows como sistema base. Foram necessárias três grandes modificações nessa etapa, sendo a primeira a adaptação do projeto dos tradicionais *makefiles* do Unix para o modelo de projeto do Visual Studio, essa adaptação não é automatizada e teve que ser feita manualmente. Após o porte do projeto, foi necessário adaptar as chamadas de sistema que eram incompatíveis entre os dois sistemas operacionais, por fim a interface de linha de comando do Unix teve que ser adaptada para a interface de linha de comando do Windows.

A segunda etapa foi referente a adaptação do modelo de execução de linha de comando das aplicações para o modelo de execução do Windows Azure. Essa etapa consistiu de duas grandes tarefas que foram a adaptação dos acessos a disco, utilizados pelas aplicações para entrada e saída, como no Azure não existe um disco local, foi necessária a adaptação das aplicações para acessarem um disco no contexto do servidor do Azure. A outra adaptação foi necessária para a execução de código Fortran e C no Azure, pois essas linguagens não são suportadas nativamente pelo sistema. Dessa maneira foi necessária a criação de uma aplicação especial do Azure que é uma aplicação de código nativo, com esse tipo de aplicação se podem executar quaisquer aplicações do Windows no Azure.

A terceira etapa é relativa a interface com o usuário, onde no Azure obrigatoriamente deve ser feita utilizando-se uma interface *web*, essa adaptação foi necessária tanto para passagem dos parâmetros de entrada como para apresentação da saída dos resultados. Outra tarefa dessa etapa é executar a entrega da aplicação dentro do ambiente de produção do Azure, mas essa tarefa é feita de maneira automatizada pelo próprio ambiente do Visual Studio. Em resumo, foi possível efetuar o porte das aplicações para o Azure PaaS, mas o processo é longo e bastante trabalhoso.

Na análise do modelo IaaS, para os três provedores selecionados, os procedimento de porte das aplicações foi realizado sem maiores problemas, como há a possibilidade de se utilizar um sistema operacional compatível com o modelo de execução do Unix, não há a necessidade de modificações na aplicação, bastando que a mesma seja enviada para os provedores e em seguida executada. Para a avaliação da versão MPI das aplicações, foi

necessária a configuração de um ambiente semelhante a um *cluster* onde mais de uma máquina deve poder se conectar as outras, em nenhum dos provedores houve dificuldade para criação desse ambiente.

A.4.3 Análise de desempenho e econômica

A avaliação de desempenho do PaaS com as aplicações OpenMP foi executada utilizando o tamanho A do NAS e cada aplicação foi executada 50 vezes, as aplicações DC e FT não foram utilizadas devido as características das mesmas. Houve um sobrecusto de cerca de 50% em se considerando a média de todas as execuções feitas, tendo aplicações que apresentaram baixo sobrecusto, BT, EP, IS e LU, essas aplicações tiveram desempenho muito próximo do sistema base, mostrando que para alguns tipos de aplicações a execução utilizando a nuvem é totalmente equivalente a execução em máquinas tradicionais. Os resultados normalizados ao sistema base podem ser vistos na Figura 4.2, essa figura mostra também o intervalo de confiança de 95%, como pode-se perceber pelo intervalo de confiança a variabilidade das execuções foi bastante baixa, o que que indica que o ambiente do provedor é bastante estável para esse tipo de execução. Na Tabela 4.5 apresenta os resultados absolutos tanto para o Azure como para o sistema base, como podemos verificar os tempos de execução e variabilidade são compatíveis entre os sistemas. A avaliação econômica do Azure PaaS foi feita utilizando o valor médio, pois ele representa um caso médio de aplicações de alto desempenho. A Tabela 4.6 apresenta os resultados econômicos calculados utilizando todas as fórmulas definidas anteriormente, como tempo de vida do sistema base foram considerados dois cenários distintos, três e cinco anos, e são apresentados os resultados para ambos. Pode-se observar que no cenário de depreciação de três anos, a utilização da nuvem é mais barata se for feita durante 187 dias equivalentes, ou seja, se utilizarmos uma aplicação no Azure PaaS durante 24 horas por seis meses durante um ano, é mais barato utilizar o Azure do que comprar uma máquina física.

A avaliação de IaaS utilizando as aplicações OpenMP foi realizada utilizando-se o tamanho B do NAS e cada aplicação foi executada também 50 vezes, a aplicação DC, devido a sua característica, não foi utilizada devido a característica de acesso a disco da mesma. A Figura 4.4 apresenta os resultados das execuções nos três provedores normalizados em relação a execução no sistema base, pode-se perceber ao analisar o tempo médio de execução nos três provedores que tanto o Azure como o Rackspace tiveram pouco menos de 100% de sobrecusto em relação ao sistema base e o tempo de execução no EC2 teve cerca de 250% de sobrecusto em relação ao sistema base. O comportamento das aplicações seguiu o mesmo padrão em todos os provedores, com nenhum ponto a ser destacado. Em termos da variabilidade, pode-se notar no gráfico da mesma figura que ela é baixa para todos os provedores e totalmente aceitável para os padrões esperados em PAD. A Tabela 4.8 apresenta os dados relativos a avaliação econômica do cenário de IaaS com OpenMP, os dados foram calculados utilizando-se o tempo de execução média dos três provedores. Pode se perceber que o break even point apresentou melhores resultados no Azure, seguido pelo EC2 e então pelo Rackspace, mesmo os resultados de desempenho do EC2 tendo sido muito piores do que no Rackspace, o maior custo do Rackspace acaba fazendo com que o mesmo apresente pior desempenho econômico. O gráfico do BEP, apresentado na Figura 4.5 mostra o comportamento do mesmo, sendo que as intersecções nos gráficos representa o BEP, pode-se observar que quanto menor o ângulo da linha do provedor, melhor são os seus resultados econômicos.

Enquanto as avaliações utilizando o OpenMP foram feitas com a utilização somente com um nó devido as características dessas aplicações, as avaliações utilizando as aplicações MPI foram feita utilizando-se cenários de um, dois e também quatro nós. Os resultados de desempenho para um nó estão mostrados no gráfico na Figura 4.6, os resultados de dois nós estão na Figura 4.7 e os resultados com a utilização de 4 nós estão apresentados na Figura 4.8. Percebe-se que quando houve o aumento de um para dois nós, houve uma piora do desempenho do Rackspace, passando de um sobrecusto de 120% para mais de 250%, enquanto tanto o Azure como o EC2 tiveram uma melhora em seu desempenho. Na avalição utilizando quatro nós, os três provedores apresentaram resultados muito próximos ao do sistema base, tendo o Azure apresentado inclusive um desempenho melhor do que o sistema base em cerca de 30%, isso significa que ao se executar as aplicações do NAS, utilizando a média delas, o Azure executa mais rapidamente do que a máquina física. A variabilidade nos três cenários é totalmente aceitável em todos os provedores e não representa um problema para aplicações de PAD em nenhum deles. A análise econômica para os três cenários é apresentada na Tabela 4.12, que apresenta os resultados para todos as fórmulas definidas nesta dissertação. Pode-se perceber que o Rackspace apresente os piores resultados relativos ao BEP, tanto devido ao seu pior desempenho em relação ao outros provedores, bem como o seu maior custo de máquinas. Um resultado que merece destaque é o do Azure em um cenário de tempo de vida de três anos para o sistema base no cenário do MPI com a utilização de 4 nós de processamento, esse resultado de 377 dias equivalentes significa que dentro de um cenário de uso de um ano, o Azure é sempre mais barato de ser utilizado do que o sistema base. Isso pode ser observado também no gráfico do BEP para 4 nós MPI, na Figura 4.12, nessa figura pode-se notar que a linha referente ao Azure nunca faz intersecção com a linha referente ao sistema base na depreciação de três anos, o que significa que ele sempre é mais barato nesse cenário.

Nossa métrica do CEF ainda pode ser utilizada para análise da escalabilidade no cenário de MPI com o aumento de máquinas, como ele representa um fator entre o custo da solução e o desempenho da mesma, com o aumento de máquinas, se ele manter esse fator constante isso nos indica que ele tem escalabilidade igual ao do sistema base. Na Figura 4.9 temos a representação do CEF para os três cenários de MPI representando os três provedores analisados. Pode-se perceber que o Rackspace não apresenta um comportamento previsível tendo uma enorme variação entre os três pontos, mostrando que sua escalabilidade não é aderente ao sistema base. Já o EC2 apresenta um melhor comportamento, mas ainda assim possui uma variação de 100% entre os pontos, isso significa também que sua escalabilidade não pode ser considerada a mesma do que o sistema base. Por fim o Azure apresenta uma excelente escalabilidade pois sua representação é praticamente uma linha reta, o que nos diz que sua escalabilidade é totalmente aderente ao sistema base.

A.5 Conclusões

Essa dissertação apresentou um estudo que buscou identificar se o modelo de computação em nuvem é uma alternativa viável para ser usada para a construção de ambientes de processamento de alto desempenho. A conclusão geral é que esse modelo pode ser usado para processamento de alto desempenho, desde que se utilize de certas condições. Os dois modelos de programação mais utilizados para processamento de alto desempenho,

OpenMP e MPI, foram utilizados sem maiores problemas nos provedores de nuvem analizados.

O paradigma de computação em nuvem define três modelos de serviço, SaaS, PaaS e IaaS, e também quatro modelos de implementação, privado, público, comunitário e híbrido. Durante nossa análise o modelo SaaS foi descartado devido a ausência da capacidade de customização, sendo que os modelos que foram adotados em nossa avaliação foram tanto PaaS como IaaS. Em ralação aos modelos de implantação, somente o modelo público apresentou-se como uma nova alternativa de computação e foi o único avaliado. A análise de PaaS foi efetuado utilizando somente o provedor Windows Azure a a análise de IaaS foi feita utilizando três provedores: Windows Azure, Amazon EC2 e Rackspace.

A análise de PaaS mostrou que o Windows Azure apresenta uma degradação de performance de cerca de 50%, mas que pode ser aceita em determinados cenários, além de apresentar um custo-benefício interessante. A grande restrição desse modelo é a incapacidade de se configurar máquinas paralelas e o grande esforço que deve ser feito durante o processo de portar uma aplicação para o seu modelo de execução, portanto esse modelo, não é um modelo totalmente viável para ser usado para PAD. Dessa maneira o modelo que se apresentou totalmente adequado foi o IaaS, e dentre os três provedores avaliados, foi possível criar um ambiente dedicado ao processamento de alto desempenho em todos eles, nesses ambientes foram usadas quatro configurações diferentes para avaliação: um nó utilizando OpenMP e um, dois e quatro nodos para a avaliação de MPI. O provedor que apresentou os melhores resultados foi o Windows Azure, inclusive no cenário de avaliação do MPI utilizando-se quatro nodos, ele apresentou um desempenho 30% melhor do que o cluster físico utilizado para comparação. A conclusão é que computação em nuvem, no modelo de IaaS, pode ser utilizada para processamento de alto desempenho, apresentando desempenho compatível com a máquina física analizada, além de apresentar-se como uma alternativa interessante em termos econômicos.

A maior contribuição desse trabalho foi o modelo de custo criado, com o qual se pode avaliar objetivamente quando é melhor, em termos de custo, se utilizar uma máquina física ou se utilizar de recursos de um provedor público de computação em nuvem, dessa maneira o usuário pode avaliar qual o melhor cenário para a sua realidade. Como trabalhos futuros pretende-se melhorar o modelo de custo, utilizado-se regras de contabilidade e também padrões nacionais de depreciação. Também se pretende realizar o porte de aplicações para a nuvem para verificar o comportanto de aplicações reais e para validar o modelo de avaliação proposto nesse trabalho.

REFERENCES

ANGABINI, A. et al. Suitability of Cloud Computing for Scientific Data Analyzing Applications; An Empirical Study. In: P2P, PARALLEL, GRID, CLOUD AND INTERNET COMPUTING (3PGCIC), 2011 INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2011. p.193–199.

ARMBRUST, M. et al. **Above the Clouds**: a berkeley view of cloud computing. [S.l.]: EECS Department, University of California, Berkeley, 2009. (UCB/EECS-2009-28).

AVETISYAN, A. et al. Open Cirrus: a global cloud computing testbed. **Computer**, [S.l.], v.43, n.4, p.35–43, 2010.

BADGER, L. et al. DRAFT Cloud Computing Synopsis and Recommendations. **NIST Special Publication**, [S.l.], n.800-146, May 2011.

BHATELE, A. et al. Overcoming scaling challenges in biomolecular simulations across multiple platforms. In: PARALLEL AND DISTRIBUTED PROCESSING, 2008. IPDPS 2008. IEEE INTERNATIONAL SYMPOSIUM ON. **Proceedings...** [S.l.: s.n.], 2008. p.1–12.

BUYYA, R. et al. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. **Future Generation Computer Systems**, [S.l.], p.599–616, 2009.

CARRARO, G.; CHONG, F. **Software as a Service (SaaS)**: an enterprise perspective. Microsoft Corporation, http://msdn.microsoft.com/en-us/library/aa905332.aspx.

CHIEU, T. et al. A Cloud Provisioning System for Deploying Complex Application Services. In: BUSINESS ENGINEERING (ICEBE), 2010 IEEE 7TH INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2010. p.125–131.

COARFA, C. et al. Co-array Fortran performance and potential: an npb experimental study. **Languages and Compilers for Parallel Computing**, [S.l.], p.177–193, 2004.

COULOURIS, G. et al. **Distributed Systems: concepts and design (5th edition)**. 5.ed. [S.l.]: Addison Wesley, 2011.

DEAN, J.; GHEMAWAT, S. MapReduce: simplified data processing on large clusters. **Communications of the ACM**, [S.l.], v.51, n.1, p.107–113, 2008.

DEELMAN, E. et al. The cost of doing science on the cloud: the montage example. In: HIGH PERFORMANCE COMPUTING, NETWORKING, STORAGE AND ANALYSIS, 2008. SC 2008. INTERNATIONAL CONFERENCE FOR. **Proceedings...** [S.l.: s.n.], 2008. p.1 –12.

DMTF. **Open Virtualization Format Specification**. http://www.dmtf.org/standards/ovf.

EKANAYAKE, J.; FOX, G. High performance parallel computing with clouds and cloud technologies. **Cloud Computing**, [S.l.], p.20–38, 2010.

FAN, P. et al. Toward Optimal Deployment of Communication-Intensive Cloud Applications. In: CLOUD COMPUTING (CLOUD), 2011 IEEE INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2011. p.460–467.

FAN, P. et al. Topology-Aware Deployment of Scientific Applications in Cloud Computing. In: CLOUD COMPUTING (CLOUD), 2012 IEEE 5TH INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2012. p.319–326.

FOSTER, I. et al. Cloud Computing and Grid Computing 360-Degree Compared. In: GRID COMPUTING ENVIRONMENTS WORKSHOP, 2008. GCE '08. **Proceedings...** [S.l.: s.n.], 2008. p.1–10.

GASSER, M. **Building a secure computer system**. [S.l.]: Van Nostrand Reinhold Co., 1988.

GROPP, W. Reproducible measurements of MPI performance characteristics. **Recent Advances in Parallel Virtual Machine and**, [S.l.], 1999.

GUPTA, A. et al. Improving HPC Application Performance in Cloud through Dynamic Load Balancing. In: CLUSTER, CLOUD AND GRID COMPUTING (CCGRID), 2013 13TH IEEE/ACM INTERNATIONAL SYMPOSIUM ON. **Proceedings...** [S.l.: s.n.], 2013. p.402–409.

GUPTA, A.; MILOJICIC, D. Evaluation of HPC Applications on Cloud. In: OPEN CIRRUS SUMMIT (OCS), 2011 SIXTH. **Proceedings...** [S.l.: s.n.], 2011. p.22–26.

HOFFA, C. et al. On the Use of Cloud Computing for Scientific Workflows. In: SCIENCE, 2008. ESCIENCE '08. IEEE FOURTH INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2008. p.640–645.

IEEE. **IEEE P2302 Working Group**. http://cloudcomputing.ieee.org/standards/standards-guidance-p2302.

IOSUP, A. et al. Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. **Parallel and Distributed Systems, IEEE Transactions on**, [S.l.], v.22, n.6, p.931–945, 2011.

ISARD, M. et al. Dryad: distributed data-parallel programs from sequential building blocks. **ACM SIGOPS Operating Systems Review**, [S.l.], v.41, n.3, p.59–72, 2007.

JAIN, R. **The Art of Computer Systems Performance Analysis**: techniques for experimental design, measurement, simulation, and modeling. [S.l.]: Wiley, 1991. (Wiley Professional Computing).

LI, A. et al. CloudCmp: comparing public cloud providers. In: INTERNET MEASUREMENT, 10. **Proceedings...** [S.l.: s.n.], 2010. p.1–14.

LIU, B. et al. Deploying Bioinformatics Workflows on Clouds with Galaxy and Globus Provision. In: HIGH PERFORMANCE COMPUTING, NETWORKING, STORAGE AND ANALYSIS (SCC), 2012 SC COMPANION:. **Proceedings...** [S.l.: s.n.], 2012. p.1087–1095.

LU, W.; JACKSON, J.; BARGA, R. AzureBlast: a case study of developing science applications on the cloud. In: ACM INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING, 19., New York, NY, USA. **Proceedings...** ACM, 2010. p.413–420. (HPDC '10).

LUSZCZEK, P. R. et al. The HPC Challenge (HPCC) benchmark suite. In: ACM/IEEE CONFERENCE ON SUPERCOMPUTING, 2006., New York, NY, USA. **Proceedings...** ACM, 2006. (SC '06).

MACH, W.; SCHIKUTA, E. A Consumer-Provider Cloud Cost Model Considering Variable Cost. In: DEPENDABLE, AUTONOMIC AND SECURE COMPUTING (DASC), 2011 IEEE NINTH INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2011. p.628–635.

MAO, M.; HUMPHREY, M. A Performance Study on the VM Startup Time in the Cloud. In: CLOUD COMPUTING (CLOUD), 2012 IEEE 5TH INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2012. p.423–430.

MARTENS, B.; WALTERBUSCH, M.; TEUTEBERG, F. Costing of Cloud Computing Services: a total cost of ownership approach. In: SYSTEM SCIENCE (HICSS), 2012 45TH HAWAII INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2012. p.1563–1572.

MASTELIC, T. et al. Methodology for trade-off analysis when moving scientific applications to cloud. In: CLOUD COMPUTING TECHNOLOGY AND SCIENCE (CLOUDCOM), 2012 IEEE 4TH INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2012. p.281–286.

MELL, P.; GRANCE, T. **The NIST Definition of Cloud Computing**. 2011. (NIST Special Publication 800-145).

NURMI, D. et al. The Eucalyptus Open-Source Cloud-Computing System. In: CLUSTER COMPUTING AND THE GRID, 2009. CCGRID '09. 9TH IEEE/ACM INTERNATIONAL SYMPOSIUM ON. **Proceedings...** [S.l.: s.n.], 2009. p.124–131.

ROLOFF, E. et al. Evaluating High Performance Computing on the Windows Azure Platform. In: IEEE 5TH INTERNATIONAL CONFERENCE ON CLOUD COMPUTING (CLOUD 2012), 2012. **Proceedings...** [S.l.: s.n.], 2012. p.803–810.

ROLOFF, E. et al. High Performance Computing in the cloud: deployment, performance and cost efficiency. In: CLOUD COMPUTING TECHNOLOGY AND SCIENCE (CLOUDCOM), 2012 IEEE 4TH INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2012. p.371–378.

SHVACHKO, K. et al. The hadoop distributed file system. In: MASS STORAGE SYSTEMS AND TECHNOLOGIES (MSST), 2010 IEEE 26TH SYMPOSIUM ON. **Proceedings...** [S.l.: s.n.], 2010. p.1–10.

SIEWIOREK, D. P.; SWARZ, R. S. **Reliable computer systems (3rd ed.)**: design and evaluation. Natick, MA, USA: A. K. Peters, Ltd., 1998.

SNIA. **Cloud Data Management Interface**. http://snia.org/sites/default/files/CDMI

STANTCHEV, V. Proc. Performance Evaluation of Cloud Computing Offerings. In: ADVANCED ENGINEERING COMPUTING AND APPLICATIONS IN SCIENCES, 2009. ADVCOMP '09. THIRD INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2009. p.187–192.

THANAKORNWORAKIJ, T. et al. A reliability model for cloud computing for high performance computing applications. In: PARALLEL PROCESSING WORKSHOPS, 18., Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2013. p.474–483. (Euro-Par'12).

TUDORAN, R. et al. A performance evaluation of Azure and Nimbus clouds for scientific applications. In: INTERNATIONAL WORKSHOP ON CLOUD COMPUTING PLATFORMS, 2., New York, NY, USA. **Proceedings...** ACM, 2012. p.4:1–4:6. (CloudCP '12).

WALKER, E. Benchmarking amazon EC2 for high-performance scientific computing. **Usenix Login**, [S.l.], v.33, n.5, p.18–23, 2008.

WANG, L. et al. In cloud, do MTC or HTC service providers benefit from the economies of scale? In: WORKSHOP ON MANY-TASK COMPUTING ON GRIDS AND SUPERCOMPUTERS, 2., New York, NY, USA. **Proceedings...** ACM, 2009. p.7:1–7:10. (MTAGS '09).

YANG, B. et al. Performance Evaluation of Cloud Service Considering Fault Recovery. In: JAATUN, M.; ZHAO, G.; RONG, C. (Ed.). **Cloud Computing**. [S.l.]: Springer Berlin Heidelberg, 2009. p.571–576. (Lecture Notes in Computer Science, v.5931).

YE, K. et al. Analyzing and Modeling the Performance in Xen-Based Virtual Cluster Environment. In: HIGH PERFORMANCE COMPUTING AND COMMUNICATIONS (HPCC), 2010 12TH IEEE INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2010. p.273–280.