Exploring the limits of cloud computing

VICTOR DELGADO

KTH Information and Communication Technology

Master of Science Thesis
Stockholm, Sweden 2010

TRITA-ICT-EX-2010:277

Exploring the limits of cloud computing

Victor Delgado
vdel26@gmail.com

Masters Thesis
October 4, 2010

Kungliga Tekniska Högskolan (KTH)
Stockholm, Sweden

Supervisor and examiner: Gerald Q. Maguire Jr.

Abstract

Cloud computing is a new computing paradigm that, just as electricity was firstly generated at home and evolved to be supplied from a few utility providers, aims to transform computing into an utility. It is being forecasted that more and more users will rent computing as a service, moving the processing power and storage to centralized infrastructures rather than located in client hardware. This is already enabling startups and other companies to start web services without having to invest upfront in dedicated infrastructure. Unfortunately, this new model also has some actual and potential drawbacks and remains to be seen whether concentrating computing at a few places is a viable option for everyone. Consumers are not used to renting computing capacity. The question of how to measure performance is already a major issue for cloud computing customers. This thesis demonstrates that current metrics for the performance of offerings by cloud providers are subject to imprecision and variability. Several surveys show that customers are concerned with the difficulty of predicting how the services that they have contracted for will behave. Moreover, switching from the traditional own and operate model to a service model involves replacing existing licenses with new license that include service level agreements (SLAs). However, existing SLAs can not succesfully guarantee performance levels.

This thesis will try to clarify concerns about performance in cloud computing, analyzing the factors that make the performance of clouds unpredictable and suggesting ways to solve this problem. The performance degradation due to virtualization and the lack of isolation between virtual machines were empirically evaluated in an Eucalyptus testbed based on the KVM virtualizer. Drawing upon previous research, all the parts of the problem, from the behaviour of specific application types when hosted in clouds to a proposal for a new generation of SLAs with performance guarantees, will be discussed.

The findings led to the conclusion that clouds will have difficulties to meet the needs of specific types of workloads, while succesfully adapting to others. This thesis argues for the formulation of cloud offerings and SLAs that feature performance parameters more familiar and useful to the customer, such as response time, thus facilitating the process of selecting cloud provider or deciding whether to move an application to the clouds.

Sammanfattning

Cloud computing är en ny computing paradigm som, precis som elektricitet var först genereras hemma och utvecklats som skall levereras från ett fåtal verktyg leverantörer, syftar till att omvandla data till ett verktyg. Det är förutspått att fler och fler användare kommer att hyra datorer som en tjänst, att flytta processorkraft och lagringskapacitet till centraliserad infrastruktur i stället ligger i klientens hårdvara. Detta är redan möjligt nystartade företag och andra företag att starta webbtjänster utan att behöva investera i förskott i särskilda infrastruktur. Tyvärr har den nya modellen också några faktiska och potentiella nackdelar och återstår att se huruvida koncentrera computing på ett fåtal platser är ett hållbart alternativ för alla. Konsumenterna är inte vana att hyra datorkapacitet. Frågan om hur man kan mäta prestanda är redan en viktig fråga för kunderna cloud computing. Denna avhandling visar att nuvarande mått för utförandet av erbjudanden genom moln tjänsteleverantörer som omfattas av oklarheter och variabilitet. Flera undersökningar visar att kunderna handlar om svårigheten att förutsäga hur de tjänster som de har beställt kommer att uppföra sig. Dessutom byter från den traditionella äga och driva modellen till en tjänst modell handlar om att ersätta befintliga licenser med nytt körkort som innehåller servicenivåavtal (SLA). Däremot kan befintliga SLA inte framgångsrikt garantera prestanda.

Denna uppsats kommer att försöka klargöra oro prestanda i cloud computing, analysera de faktorer som gör utförandet av moln oförutsägbar och föreslå sätt att lösa detta problem. Den prestandaförlust på grund av virtualisering och bristen på isolering mellan virtuella maskiner empiriskt utvärderades i en Eucalyptus testflygplan baserat på KVM virtualizer. Ritning på tidigare forskning, skall alla delar av problemet, från beteendet för särskild tillämpning typer när värd i molnen för att ett förslag till en ny generation av servicenivåavtal med fullgörandegarantier kommer att diskuteras.

Resultaten ledde till slutsatsen att molnen kommer att få svårigheter att tillgodose behoven av särskilda typer av arbetsbelastningar, medan framgångsrikt anpassat till andra. Denna avhandling argumenterar för utformningen av moln erbjudanden och SLA som har prestandaparametrar mer bekant och användbar för kunden, till exempel responstid, vilket underlättar processen för att välja moln leverantör eller fattar beslut om att flytta ett program till molnen.

Acknowledgements

I would like to sincerely thank my supervisor and examiner Professor Gerald Q. Maguire Jr. He introduced me to an interesting topic on which I have enjoyed so much to work. Gerald's guidance has also been essential during some steps of this thesis and his quick invaluable insights have always been very helpful.

I would also like to thank my colleagues in the department, the people I got to know during my stay in Stockholm which I can now call friends, my friends in Barcelona, and my familiy and my girlfriend, Laia, all of whom have always been encouraging me during all this time.

Thank you all.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Cloud computing is an emerging area within the field of information technology (IT). It is turning upside down the way we realize computation by enabling the use of storage, processing, or higher level elements such as operating systems or software applications, not by owning them and having them installed on computers that we own - but rather to use these resources simply as a service. The term cloud computing causes confusion due to the multiple aspects of service that it may include. From a generic point of view, it could be said that cloud computing is a kind of computing where scalable, adaptable, and elastic IT capabilities are provided as a service to multiple users.

In a pure cloud computing model, this means having all the software and data hosted on a server or a pool of servers, and accessing them through the internet without the need for very much (if any) local hard disk, memory, or processor capacity, allowing the use of very light weight client computers by the end user. In some cases the client is simply a device equipped with a minimal OS and running a web browser. We want to understand if this is a feasible solution and if there are any limitations on what types of applications or data that such a computing model can be applied to.

1.2 Problem statement

The introduction of cloud computing changes our thinking as what is considered to be "our system" and "our data" is no longer physically stored on a specific set of computers and disks, but rather both the concept of system and the locus of our data have evolved into something diffuse and geographically distributed. A logical deduction is that this makes it harder to have everything under your control. So, as in most major technologic developments, there is concern among potential customers of cloud computing services of the details of the limitations and potentials that cloud computing may offer.

To find what these limitations are we must first look at what cloud computing means from several different perspectives, specifically in this thesis we will consider the economic, legal, and technical perspectives. We will identify some of the questions that the customers are going to ask to the cloud providers before signing a service agreement and entrusting them with confidential data.

This thesis specially addresses the problems that might arise related to the performance of applications running in clouds. The analysis is based upon previous research and our own experimentation in a cloud testbed. The goal was to discern the factors affecting performance and, when possible, provide some solutions or guidelines to cloud users that might run into performance problems.

Chapter 2

General Background

2.1 What is Cloud Computing?

It seems that everyone in this industry, from experts to cloud providers, has their own definition about what cloud computing is. Today there is not yet a consensus for what exactly this term means. Examining some of the existing definitions helps to clarify the term and what it involves (or might involve). Here we quote four definitions for cloud computing:

> "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." - U.S. National Institute of Standards and Technology (NIST) [48]
>
> "A pool of abstracted, highly scalable, and managed compute infrastructure capable of hosting end-customer applications and billed by consumption" - Forrester Research, Inc. [64]
>
> "A style of computing where massively scalable IT-enabled capabilities are delivered as a service to external customers using Internet technologies." - Gartner, Inc. [55]

> "A Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers" - R. Buyya, C.S Yeo, and S.Venugopal [11]

As we can see, and will be explained later in more detail, the definitions of cloud computing include different classes of services. For example cloud computing can supply remote storage space; but could also supply processing power to supply applications as a service over the internet. Reading these definitions there is a noticeable pattern, this pattern enables us to extract the main features of a cloud computing system. These features are described in the following subsections.

Finally, it should be noted that there are two major technologies that have led to the development of the cloud computing paradigm: virtualization and grid computing. The former will be discussed in detail in section 2.4.1 so nothing more will be added here. The latter, grid computing, refers to applying the resources of many computers in a network simultaneously to solve a single problem, and was first introduced by Foster and Kesselman in the early nineties and formally presented by them in a book in 1999 [30]. Grid computing is typically used to tackle scientific or technical problems that require a great number of computer processing cycles or that involve large amounts of data. The difference between this paradigm and cloud computing is that grid systems are architected so that a single user can request and consume large fractions of the total resource pool, whereas in cloud systems individual users' requests are limited to tiny fractions of the total system capacity. As a consequence of users having very small fractions of the total capacity, cloud computing has focused on scaling to handle large numbers of users.

2.1.1 On-Demand

A basic condition that a cloud computing provider must fulfill is the ability to deliver computing resources whenever the customer needs them. From the customer's point of view the available computing resources are nearly infinite (i.e., the customer is not limited the set of servers located at one site and it is the responsibility of the cloud computing provider to have sufficient resources to satisfy the requirements of all their customers).

Utilizing computing resources on-demand is one of the most desired capabilities for a large number of enterprises because it eliminates the need for planning ahead, purchasing, and installing the resources they will require at some point in the future. This enables the customer to avoid making an unnecessary upfront investment in servers. Furthermore, when comparing cloud computing with the traditional model of owning the servers, cloud computing will helps avoid the costs of having underused resources. Effectively the cloud computing vendor is doing what firms such as EDS did when it started to run service bureaus - by combining the needs of multiple firms the service bureau is able to take advantages of the effects of resource pooling. (See for example [71]).

Consequences of this feature of on-demand computing resources are a lowering of the entry barriers to some business models, as software vendors can develop applications without worrying beforehand of provisioning for a specific number of customers and then bearing with the risk of greater success than planned, leading to the service not being available or, worse, having very few users and a large capital expense caused by purchasing resources that are very underutilized.

2.1.2 Pay-per-use

Another new aspect of cloud computing is application of an usage based billing model. The customer pays only for short term use of processors or storage, for example this usage could be metered in increments of hours or days; converting what would have been capital expenses (CAPEX) into operational expenses (OPEX).

We can see that the concept of cloud computing is strongly related to the idea of utility computing. In both cases the computing resources are being provided on-demand, much as electricity, water, or gas are supplied by a utility company; but in the case of computing resources the waste product is largely heat and after some time scrap computing equipment - hence the customer is essentially renting these computing resources. However, unlike a traditional rental agreement where the resources would be physically located at the

customer's premises, in the case of cloud computing the resources are simply some where in the cloud - rather than in a single physical location. Further note that unlike the case for water and gas, which when they are not used are available for later use - not using processor cycles of a computer does in fact waste these cycles - since they will not be available for usage later. Therefore it is advantageous for a cloud computing provider to accept business to utilize all (or nearly all) of these cycles.

2.1.3 Rapid elasticity

Based upon the specific of a service level agreement, the cloud provider scales up or down the resources that are provided to meet the customer's changing needs. This service level agreement must define the response time for the cloud provider to adapt to the customer's needs. Such an agreement is needed by the cloud provider, because the cloud provider does not in fact have infinite resources, so depending upon the service level agreement the cloud provider has to find a set of allocations of resources that satisfy the current demands of the aggregate of their users *while* meeting the various service level agreements of these costumers - otherwise the service level agreement may specify a penalty that the cloud provider has to pay to each customer for not meeting the relevant service level agreement.

2.1.4 Maintenance and upgrading

Because the cloud provider rather than the customer maintains the computing resource, there is an effective outsourcing of maintenance tasks. Thus the cloud provider maintains and updates the resources, whether the resource is hardware or software. Therefore all repairs and replacement of the underlying hardware resources are transparent to the customer, as they do not affect the customer's experience. While this might be true in the ideal case, there may be short intervals when a customer's image is migrated from one hardware platform to another in order to perform maintenance or repair of a given physical platform, during this period of time the customer might not have any of the resources associated with this image available.

2.2 Cloud computing service models

Cloud computing can be classified by the model of service it offers into one of three different groups. These will be described using the XaaS taxonomy, first used by Scott

Maxwell in 2006, where “X” is Software, Platform, or Infrastructure, and the final "S" is for Service.

It is important to note, as shown in Figure 2.1, that SaaS is built on PaaS, and the latter on IaaS. Hence, this is not an excluding approach to classification, but rather it concerns the level of the service provided. Each of these service models is described in a following subsection.

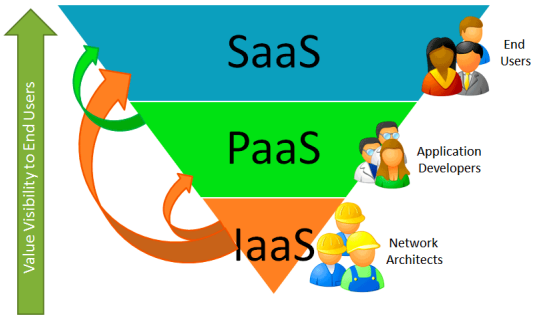

Figure 2.1: Cloud computing service models

2.2.1 IaaS (Infrastructure as a Service)

The capability provided to the customer of IaaS is raw storage space, computing, or network resources with which the customer can run and execute an operating system, applications, or any software that they choose. The cloud customer is not able to control the distribution of the software to a specific hardware platform or change parameters of the underlying infrastructure, but the customer can manage the software deployed (generally from the boot level upward).

2.2.2 PaaS (Platform as a Service)

In the case of PaaS, the cloud provider not only provides the hardware, but they also provide a toolkit and a number of supported programming languages to build higher level services (i.e. software applications that are made available as part of a specific platform). The users of PaaS are typically software developers who host their applications on the platform and provide these applications to the end-users.

2.2.3 SaaS (Software as a Service)

The SaaS customer is an end-user of complete applications running on a cloud infrastructure and offered on a platform on-demand. The applications are typically accessible through a thin client interface, such as a web browser. The customer does not control either the underlying infrastructure or platform, other than application parameters for specific user settings.

Figure 2.2 shows the difference in the number of parts of the whole server stack that a customer of an IaaS or PaaS provider is able to control compared to a private on-premises server.

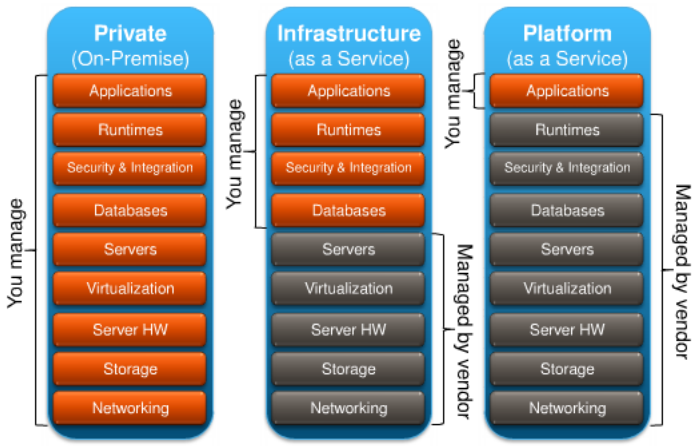

Figure 2.2: Server stack comparison between on-premise infrastructure, IaaS, and PaaS.

2.3 Deployment models

Clouds can also be classified based upon the underlying infrastructure deployment model as Public, Private, Community, or Hybrid clouds. The different infrastructure deployment models are distinguishing by their architecture, the location of the datacenter where the cloud is realized, and the needs of the cloud provider's customers (for example, due to regulatory, legal, or other requirements).

2.3.1 Public clouds

A public cloud's physical infrastructure is owned by a cloud service provider. Such a cloud runs applications from different customers who share this infrastructure and pay for their resource utilization on a utility computing basis.

2.3.2 Private clouds

A pure private cloud is built for the exclusive use of one customer, who owns and fully controls this cloud. Additionally, there are variations of this in terms of ownership, operation, etc. The fact that the cloud is used by a specific customer is the distinguishing feature of any private cloud.

A private cloud might be owned by the customer, but built, installed, and managed by a third party rather than the customer. The physical servers might be located at the customer's premises or sited in a collocation facility.

A recently introduced alternative to a private cloud is a 'virtual private cloud'. In such a virtual private cloud a customer is allocated a private cloud within the physical infrastructure of a public cloud. Due to the allocation of specific resources within the cloud the customer can be assured that their data stored on and processing is done only on dedicated servers (i.e., these servers are not shared with any other customer of the cloud provider).

2.3.3 Community clouds

When several customers have similar requirements, they can share an infrastructure and might share the configuration and management of the cloud. This management might be done by themselves or by third parties.

2.3.4 Hybrid clouds

Finally, any composition of clouds, be they private or public, could form a hybrid cloud and be managed a single entity, provided that there is sufficient commonality between the standards used by the constituent clouds.

2.4 Technology review

There is a feature which has not yet been commented in this report. This feature may be what sets cloud computing apart from earlier computational styles. This feature is multi-tenancy, the ability to host different users and allow them operate in the same physical resource. In cloud computing multi-tenancy is realized using virtualization technologies. Using virtualization to implement multi-tenancy in cloud computing is a return to what IBM did in 1972 with their VM/370 system, introducing time-sharing in a mainframe computer of the System/370 line. IBM was able to provide its multiple users with seemingly separate System/370 computing systems [20].

2.4.1 Virtualization

In the cloud model what customers really pay for, that is what they dynamically rent, are virtual machines. This enables the cloud service provider to share the cloud infrastructure located in a datacenter between multiple customers. The level of virtualization of what is offered depend on which of the three (SaaS, PaaS, or IaaS) service models the user requires.

Virtualization strictly refers to the abstraction of computer resources using virtual machines: software implementations of machines that execute programs as if there were separate physical machines. Virtualization allows multiple operating systems to be executed simultaneously on the same physical machine. Virtualization and the dynamic migration of virtual machines allows cloud computing to make the most efficient use of the currently available physical resources.

Virtualization is achieved by adding a layer beneath the OS, between the OS and the hardware. This additional layer makes it possible to run several OS instances on top of the same underlying resources. Two different options for this virtualization layer exist:

- Type-1: This kind of virtualization layer is called a hypervisor. It is installed directly onto the system, and has direct access to the hardware. For this reason it is the fastest, most scalable, and robust option.

- Type-2: Hosted architecture. The virtualization layer is placed on top of a host operating system.

In either case, the virtualization layer manages all the virtual machines, launching a virtual machine monitor (VMM) for each one.

Today the hypervisor technique is the used in all cloud computing datacenters as it is the most efficient option in terms of hardware utilization. See Figure 2.3.

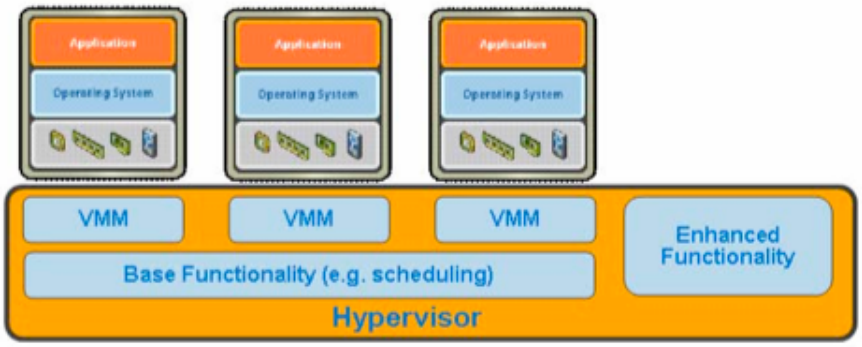

Figure 2.3: Diagram of an hypervisor virtualization layer with 3 VMMs running

Both options for virtualization are applicable to x86 architecture systems. This platform will be used for the examples in this thesis as it is by far the most common architecture nowadays. Due to its dominance in the PC market most operating systems are designed to be compatible with this architecture. The x86 architecture offers four levels of privilege named "rings". In this architecture ring 0 is the most privileged level. The operating system (OS) is usually located in ring 0. User applications commonly run in ring 3. To provide the appropriate virtualization and security the virtualization layer must be placed at ring 0 in order to create and manage the virtual machines that actually deliver the shared resources. There are some processor instructions that have different semantics when they are not executed in ring 0, therefore some translation mechanism is needed so that when these instructions are executed outside ring 0 in a virtual machine that the correct semantics are applied. There are three different techniques to perform this transformation: full virtualization, paravirtualization, or hardware-assisted virtualization.

Full virtualization

Full virtualization uses a combination of direct execution and binary translation. Binary translation is used in order to adapt the non-virtualizable instructions by replacing these

instructions with other instructions that realize the same effect on the virtualized hardware. This approach is called full virtualization because it completely abstracts the guest OS from the underlying hardware, without the guest OS noticing. The hypervisor traps and translates all these special instructions and replaces these instructions with a new sequence of instructions (which are cached for future use). User-level instructions run directly on the hardware - hence full virtualization has no impact on their execution speed.

Paravirtualization

Paravirtualization involves modification of a guest OS. Today this method is only supported for open source operating systems, limiting its applicability. However, paravirtualization offers higher performance than full virtualization in performance because it does not need to trap and translate every OS call.

Hardware assisted virtualization

Hardware assisted virtualization is an alternative approach. In recent years vendors have added virtualization support to their processors due to the widespread use of virtualization. Since 2006, hardware assisted virtualization has been available in products employing Intel's VT-x and AMD's AMD-v technologies. This hardware solution is based on a new CPU execution mode that allows the VMM to run below ring 0. In this approach sensitive OS requests are automatically trapped by the hypervisor, so there is no need for paravirtualization or the binary translation required in full virtualization.

2.4.2 Current alternatives in the cloud computing market

This section presents the current cloud computing offers, distinguishing them basing on the level of abstraction (i.e. the level of service) presented to the programmer and the level of management of the resources.

Amazon Web Services (AWS)

AWS [25] refers to the services offered by Amazon to cover the entire service spectrum. Amazon is the only provider to the date with products in all three classes. AWS includes a number of components:

- Amazon Elastic Compute Cloud (EC2): The IaaS product of Amazon is the leader in its class. It supplies customers with a pay-as-you-go resource that can include storage

or computation. EC2 has a web interface for requesting virtual machines as server instances. An EC2 instance seems like physical hardware and its relatively low level of abstraction (i.e. by definition, IaaS have low levels of abstraction when compared to PaaS or SaaS. See Figure 2.2) lets the customer control settings of nearly the entire software stack. Customers have the chance to increase or decrease the number of server instances, then AWS reacts by scaling the number of instances up or down. Server instances are available in three different sizes; each one having a different amount of memory, computing power, and bandwidth.

- Amazon Simple Storage Service (S3) implements a dynamically scalable storage service which can be used to host applications that are subsequently offered to end-users.
- Amazon SimpleDB realizes a database (DB) and provides it as a web service. Developers store and query data items via web services requests. Amazon liberates these developers from worrying about the database's internal complexity.

Rackspace

Rackspace [16] offers infrastructure as a service, named Cloudservers, or a platform as a service, Cloudsites, to host web applications with scaling needs. Rackspace also provides Cloudfiles, a storage service, which can be combined with a content delivery network (CDN) service. This latter service competes directly with the CDN from Amazon, called Cloudfront, but Rackspace, unlike Amazon, does not charges for bandwidth consumption between the storage service and the CDN.

GoGrid

GoGrid [34] provides infrastructure as a service, standing as a direct competitor to Amazon or Rackspace. GoGrid offers a competitive service consisting on dedicated hosted servers in their cloud facilities. Thus they are a provider of virtual or *physical* infrastructure on-demand, unlike Amazon (who only supplies virtual infrastructure on-demand). Additionally, GoGrid complements the offer of dedicated infrastructure with an hybrid environment that enables users of their dedicated hosting service to request virtual resources to handle usage spikes.

Salesforce

Salesforce[58] is one of the pioneers in cloud computing. Salesforce's first and still main product is a Customer Relationship Management (CRM) web service. Salesforce has focused on enterprise customers and has added new applications on top of its CRM. While earlier Salesforce only offered SaaS class products, in 2002 Salesforce shifted towards the PaaS market with the release of their Force.com platform that allows developers to develop applications that will execute natively on their Salesforce platform or be integrated with third party services. In the case of Force.com, Salesforce is responsible for scaling up or down the platform as needed, thus making the addition of new physical resources transparent to the user.

The Force.com development environment is based on the Eclipse integrated development environment (IDE) and uses a new programming language called APEX. APEX is closely related to C# and Java. Force.com also provides non-programmers with tutorials and models to enable them to compose business web applications in a visual way.

Google App Engine

Google's PaaS product [27] is a platform to develop and host web applications on Google's servers. The user can leverage Google's distributed and scalable file systems (BigTable and File System), along with technologies used by Google's wide range of web applications (e.g, Gmail, Docs, Google Reader, Maps, Earth, or Youtube).

Although in the beginning the only programming language supported was Python, presently there is also support for Java, and it is forecasted that other programming languages will be allowed in the future. In a move towards connecting both clouds, Google and Salesforce have recently provided libraries that allow the developer to access the other's web services application programming interface (API) from applications. Once installed, the application can seamlessly make web service API calls of the other service, hence integrating applications hosted on both clouds.

Microsoft Windows Azure

Microsoft's PaaS service is called Windows Azure [5]. This is a very new (commercially it became available starting in February 2010) cloud platform offering that provides developers with on-demand computing and storage to host, scale, and manage web applications on the Internet using Microsoft's datacenters.

The Azure Services platform currently runs only .NET Framework applications, but Microsoft has indicated that a large range of languages will be supported. Indeed, two software development kits (SDKs) have already been made available for interoperability with the Azure Services platform that enable Java and Ruby developers to integrate their application with .NET services.

Sun Cloud

Sun Microsystems (now Oracle) in March 2009 introduced a cloud service to compete against Amazon EC2 in the field of IaaS [17]. It is uncertain what the future of this service is today. After the merger with Oracle, it was announced that the Sun Cloud service will no longer be available, but it is unclear if another Cloud product will be released instead.

Eucalyptus

Eucalyptus [54] is not comparable in size or capacity with the previous offerings, but worth including because of its distinctive purpose. This is an open source cloud computing framework developed by the University of California at Santa Barbara as an alternative to Amazon EC2. The initial mission of Eucalyptus was, and continues to be, to enable academics to perform research in the field of cloud computing. In addition to the research market, it has also been positioned as a private cloud system offering (the Eucalyptus Systems' Private Cloud - see [54]). This initiative is unique in that no other cloud system combines support for open development with the goals of being easy to install and maintain. Its specific scope is the IaaS model where it is also fully compatible with Amazon's EC2, as Eucalyptus uses the same API as AWS. Additional information about Eucalyptus and a detailed analysis of the system and its components can be found in chapter 3 of this thesis. This platform was implemented in order to perform our own performance tests.

2.5 Limitations of cloud computing

Cloud computing is widely recognized as a revolutionary IT concept and with different offerings can fit the needs of very diverse customers, ranging from large enterprises, small start-ups, to end-users. Some cloud based applications, such as Gmail, have had great success; but as the diversity of the offerings grows so does the reluctance to trust some services or to trust more sensitive data to off-site computers. This is easily observed at the enterprise level when decision makers in the information technology departments of companies and

organizations keep rejecting a move to the cloud. At present most organizations are only willing to outsource applications that involve less sensitive information. According to a survey of more than 500 chief executives and IT managers of 17 countries they still "trust existing internal systems over cloud-based systems due to the fear about security threats and loss of control of data and systems" [57]. The ones that do agree to move to the cloud still demand third party risk assessments or at least ask the cloud providers questions such as:

- Who will have access to the data and applications and how will that be monitored?
- What security measures are used for data transmission and storage?
- How are applications and data from different customers are kept separate?
- Where, in terms of geographical location, will be the data stored? Could the choice of the location affect me?
- Can these measures and details be stipulated in a service level agreement?

All these customer worries can be translated into what can be identified as the main obstacles to the adoption and growth of cloud computing. Each of these obstacles are examined in the following subsections.

2.5.1 Availability of service

Outages of a service become a major worry when customers have deposited all their information in the cloud and might need it at anytime. Given that the customer management interfaces of public clouds are accessible via Internet, there is an increased risk of failure when compared to traditional services since there are more weak points in the chain of elements needed to access the information or application. For instance, web browser vulnerabilities could lead to service delivery failures. A feasible means to obtain a high degree of availability would be using multiple cloud computing providers.

Cloud providers are well aware of these risks and today provide more information about the current state of the system, as this is something that customers are demanding. Salesforce for instance shows the real-time average response time for a server transaction at Trust.salesforce.com. Amazon has implemented a service dashboard that displays basic availability and status history.

2.5.2 Data lock-in

As some people, such as GNU creator Richard Stallman have advised [38], the use of proprietary cloud-based applications could end up in situations where migration off the cloud to another cloud or to an in-house IT environment would be nearly impossible. The reason for the current poor portability and limited interoperability between clouds is the lack of standardized APIs. As a consequence migration of applications between clouds is a hard task.

An evolution towards standardized APIs would not only overcome this risk by allowing SaaS to develop software services interoperable in all clouds, but would provide a firm basis to progress towards hybrid computing models.

Google is the only cloud provider truly advancing to achieve a more standard environment and they even have an initiative, called Data Liberation Front [36], to support users moving data and applications in and out of their platform.

2.5.3 Data segregation

A direct consequence of the multi-tenant usage mode, where different customers' virtual machines are co-located in the same server or data is on the same hard disks, is the question of isolation. How should the cloud securely isolate users? This class of risks includes issues concerning the failure of mechanisms to separate storage or memory between different users (i.e., such a failure would enable information to leak from one customer's VM to another customer's VM). There are a number of documented vulnerabilities in different commercial hypervisors that have been exploited to gain access to one or more customers' virtual machines.

Another type of attack whose feasibility has been reported is a side-channel attack. A case study carried out by MIT and University of California at San Diego on the Amazon EC2 service considered this style of attack an actual threat, and they demonstrated this attack by successfully overcoming the following:

- Determining where in the cloud infrastructure a specific virtual machine instance is located.
- Determining if two instances are co-resident in the same physical machine.
- Proving that it is possible for an adversary to launch on purpose instances that will be co-resident with another user's instances.

- Proving that it is possible to take advantage of cross-virtual machine information leakage once co-resident.

They were able to successfully perform all the previous steps given that patterns can be found in the mapping of virtual machine instances into physical resources (for example, by examining internal and external IP addresses of a large number of different types of instances). In their tests they could launch co-resident instances with a 40% probability of success. They state that the only certain way to avoid this threat is to require exclusive physical resources, something that ultimately customers with high privacy requirements will begin to ask for.

2.5.4 Privilege abuse

The threat of a malicious insider with a privileged role (e.g. a system administrator) is inherent to any outsourced computation model. Abuse by insiders could impact and damage the customer's brand, reputation, or directly damage the customer. Note that these same type of attacks can be carried out by internal employees in a traditional (i.e., non-cloud) computing infrastructure.

Cloud customers should conduct a comprehensive assessment of any potential cloud provider, specifying human resource requirements (i.e. stating who will have access to their data and what level of access they will have) and requiring transparency measures. Additional trust systems that would not require the customer to blindly trust the provider would be useful.

2.5.5 Scaling resources

As noted earlier in section 2.1.3, the ability of scaling up or down resources to meet workload is one of the most desired cloud computing advantages. However, this great advantage can lead to service failures if it is not well implemented or if a maximum response time is not agreed upon beforehand. A web application developer who hosts its service on a cloud may see how the response time steadily increases when the usage of the application also increases - because the cloud does not scales up resources quickly enough.

On the other hand, scaling must be limited by some threshold. This threshold would stop the continuous increase in the allocation of resources to prevent the cloud provider from suffering a denial of service attack because the customer's application was malfunctioning. In either case the customer could be billed for service that they did not want.

Existing service level agreements determine quality of service requirements, but not in terms of response time in response to workload variations. There are proposed solutions in service level agreements (SLA) for scalability implemented through statistical machine learning.

2.5.6 Data security and confidentiality

The distributed nature of the cloud model necessarily involves more transits of data over networks, thus creating new challenging security risks. The confidentiality of the data must be assured whether it is at rest (i.e. data stored in the cloud) or in transit (i.e. to and from the cloud). It would be desirable to provide a closed box execution environment where the integrity and confidentiality of the data could be verified by its owner. While encryption is an answer to securely storing data in the cloud, it does not fit that well with cloud-based processing. This later problem occurs because generally the cloud both stores data and applications running on the cloud operate on this data. In most cases the data has to be unencrypted at some time when it is inside the cloud. Some operations would be simply impossible to do with encrypted data and, furthermore, doing computations with the encrypted data would consume more computing resources (and more money, in consequence).

There are recent steps towards dealing with this issue. One is the Trusted Cloud Computing Platform [59], which aims to apply the Trusted Computing model (developed in 2003 by Intel, AMD, HP, and IBM) to the cloud. However the scope of this initiative is to protect against malicious insiders, inside the cloud provider organization.

Another project of the Microsoft Cryptography Group is a "searchable encryption mechanism" introduced by Kamara and Lauter in [41]. The underlying process in this system is based on a local application, installed on the user's machine, composed of three modules: a data processor, a data verifier, and a token generator. The user encrypts the data before uploading it to the cloud. When some data is required, the user uses the token generator to generate a token and a decryption key. The token is sent to the cloud, the selected encrypted file(s) are downloaded, and then these files are verified locally and decrypted using the key. Sharing is enabled by sending the token and decryption key to another user that you want to collaborate with. The enterprise version of the solution consists of adding a credential generator to simplify the collaboration process. Other relevant projects are also being conducted. One example is a recently published PhD dissertation from Stanford University done by Craig Gentry in collaboration with IBM [33]. This research proposes

"A fully homomorphic encryption scheme". Using their proposed encryption method data can be searched, sorted, and processed without decrypting it. The innovation here is the refreshing mechanism necessary to maintain low levels of noise.

Although successful, both initiatives have turned out to be still too slow and result it very low efficiency. As a result, they are not commercially utilized yet.

2.5.7 Data location

In addition to the topology of the cloud network, the geographic location of the data also matters in some cases. Knowing data's location is fundamental to securing it, as there might be important differences between regulatory policies in different countries. A customer could be involved in illegal practices without even noticing, as some governments prosecute companies that allow certain types of data to cross geographical boundaries. Cloud computing customers must tackle this issue by understanding the regulatory requirements for every country they will be operating in. Not only the data's location, but the path the data follows may also matter. According to Forrester's "Cloud Privacy Heat map" [29], a possible conclusion is that it can be hard for an application operator to deploy applications at a minimum "distance" from the users (i.e., there may be locations where the data must travel to that require following a non-optimal path because the ideal path crosses countries with restrictive laws).

Currently there are cloud providers that leave the choice of the datacenter location to the user. For instance, Amazon offers two locations in the US and one in Europe. Very likely, other providers will add to Amazon's region choice offer as the location of data is an increasing important requirement of potential customers.

2.5.8 Deletion of data

Closely related with the isolation issues that the multi-tenant architecture can entail is the fact that the user can erase data upon request. A user of a public cloud may require his data to be deleted, i.e., completely removed from the cloud. As this can only be entirely done by erasing, repeatedly re-writing the disk sectors with random data, and possibly formatting the server's hard disk, this could turn out to be impossible to do at the service provider's environment. As noted earlier in the discussion of a side-channel attack, a malicious user could later take advantage of remaining data. Even with multiple cycles of re-writing the sectors which previously held the file it may be possible to access the "erased" data, but the

probability can be reduced - however, this is at a quite high cost in time and disk I/O and may not be completely successful.

In the latest report about cloud computing by the European Network and Information Security Agency (ENISA) [14] it has been suggested that if encryption were applied to data at rest, the level of this risk would be considerably lower.

2.5.9 Recovery and back-up

Cloud providers should have an established plan of data back-up in the event of disaster situations. This may be accomplished by data replication across different locations and the plan must be addressed in the service level agreement.

2.5.10 The "Offline cloud"

Being completely dependent upon an Internet connection might turn out to be impossible or highly risky for some users who need an application (or data) to be available at all times. This creates a bigger problem if the user is moving and the quality of the connection can change, hence in some situations relying on a Internet service provider is simply not an option.

The so-called "pure Cloud computing model" causes this impediment. This model is based on the fact that the most used software application nowadays is the web browser and that today complete applications can be delivered as a service through the Internet and all of the end-user's interaction can occur through a web browser. An obvious conclusion is to build a web based OS. In this approach the web browser acts as the interface to the rest of the system and hardware, such as hard disks or powerful processors, would not be needed locally anymore. Instead, a netbook or other thin-client with a low energy consuming processor (e.g. Intel Atom, Via Technologies C7, etc.) would suffice provided that most of the computation would take place in the cloud and that all the data would be stored there as well. This is the model that Google is pursuing with their Chrome OS [35]. In addition, other independent software vendors are developing web desktop offerings, such as the eyeOS [28]. In this pure Cloud model, loosing connectivity to the cloud is a major problem because it means that the local computer becomes almost useless.

In 2007, Google introduced Gears: a free add-on for the browser that enables data to be stored locally in a fully searchable database while surfing the Internet. Gears pretty much solved the "offline problem" enabling web applications to continue their operations

while offline and then synchronizing when the connection was available again. The Gears project has been officially abandoned in February 2010 because a better and more complete replacement has arrived with the updating of the HTML protocol and the provisional release of its fifth version, HTML5 [65].

The new version of the HTML protocol addresses the offline issue with a couple of elements: AppCache and Database. These elements provide methods to store application data locally on a user's computer in amounts beyond what can be stored in an HTTP cookie. Among a long list of new features there are some other HTML5 elements that are worth a detailed description because of their close relation with new Cloud application opportunities:

Canvas: Provides a straightforward and powerful way to draw arbitrary graphics on a web page using Javascript. (e.g. Mozilla's BeSpin: an extensible web-based code editor with an interface written in Javascript and HTML. It allows collaboration between coders accessing a shared project via web browser).

Video: Aims to make it as easy to embed video on a web page as it is to embed images today. It makes unnecessary the currently used Flash plug-ins. (e.g. Youtube and Vimeo are already using it as an optional feature).

Webworkers: A new mechanism to undertake on background threads tasks that otherwise would slow down the web browser.

2.5.11 Unpredictable performance

One of the main features of any cloud computing service is the level of abstraction from the underlying physical infrastructure it is supplied with. The cloud's end customer does not know how are or where the computers where its application is running are located. The end customer might not even know the number of physical machines that their application is currently running on. The only source of information the user has about these servers are the hardware specifications provided by the cloud provider for each type of service. Moreover, these metrics do not have the same meaning in a cloud server as they did in a traditional server, as in the cloud server several users may be sharing computing and I/O resources on a given instance of a physical processor. Users expect always the same performance for the same money, but this could simply not be true as the performance depends on various factors - many of which the end customer has no control over. In fact, this is

currently one of the three main concerns enterprise customers have about cloud computing, according to a survey by IDC in the last quarter of 2009 [60]. Cloud computing's economic benefits are based on the ability to increase the usage level of the infrastructure through multi-tenancy, but it is not clear that one user's activity will not compromise another user's application performance. On top of that, the latency to the datacenter where the server is hosted, along with other network performance parameters, could vary as a function of the time of day, the particular location of the current servers, and the competing traffic in the communication links. Therefore, the performance might not be as expected and furthermore could fluctuate. This variance in performance may cause a problem if the customer is unable to predict these variations, their magnitude, and duration - as the price remains deterministic (or at least current SLA are based upon measurements at the cloud's servers and not at the end customer's interfaces or computers).

The remainder of this thesis will focus on this particular area of concern about cloud computing. Using a set of experiments this thesis will try to clarify if the performance of a cloud is indeed non-deterministic and, if that is the case, analyze what are the main factors that cause this, what are its consequences, and present existing or potential solutions to address this problem.

Chapter 3

Performance study in an Eucalyptus private cloud

3.1 Overview

As it has been pointed out before, varying performance is among the most worrying characteristics of cloud providers for enterprise customers and as such it has been studied before. Previous studies have focused on studying the performance of public clouds, specially Amazon's EC2. [32], one of the first benchmarks of Amazon's cloud, found that there are inconsistencies between the speed of a first request and a second one, and that between 10% and 20% of all queries suffer decreased performance that were at least 5 times slower than the mean. Tests of Microsoft's Windows Azure platform [39] show results that indicate that when the number of concurrent clients increases the performance drops. The greatest degradation is seen in the networking performance, where the variability sometimes makes the TCP bandwidth decrease to a quarter of its mean value.

The fact that these previous tests have been done on public clouds is logical as these are the most popular and pure instances of cloud computing. Additionally, the major advantages of the cloud computing new paradigm are associated with such public clouds. However, accurate benchmarking was difficult due to the lack of a controlled environment where the load in the server and network are exactly known at all times. Although these earlier experiments have been very insightful, they lack a higher degree of repeatability. Therefore, in order to identify and pin down with exactitude the effects of resource sharing between virtual machines, experiments must be performed in a controlled environment

without background loads other than the load added for the purposes of benchmarking. This is the reasoning behind the choice of a private cloud setup for my test environment. For this research I selected the de facto standard for private clouds: the Eucalyptus open source private cloud.

In the following sections a thorough explanation of the essential software that comprise the whole private cloud used will be presented, along with a practical configuration guide that addresses the problems that arose during the setup process and the necessary steps that must be taken to realize a running cloud.

3.2 Software components

3.2.1 Eucalyptus

Eucalyptus was the software platform of choice, but it is not the only private cloud offering today. Similar software can be found from other vendors, among which Open Nebula stands out. Eucalyptus as open-source software, differs from Open Nebula primarily in non-fundamental features for the purpose of this research: more specifically Open Nebula offers an API to extend the core capabilities and the instruction interface. On the other hand, this could be one of the best reasons for the adoption of Eucalyptus, as although the API does not enable extension of core capabilities the API used to interact with the cloud is the same as in the Amazon cloud, making easier the process of building an hybrid cloud that in low or moderate usage would operate as a private cloud, but that could expand to utilize a public cloud during peaks in load. The choice of Eucalyptus was based on the superior quantity of documentation available, as this greatly eased my learning curve.

The installation package chosen was the Ubuntu version, supplied with the Ubuntu Server 10.04 LTS. This package facilitates the installation of the Eucalyptus' platform core components, and implements a few add-on features on top of them.

3.2.2 Euca2ools

Euca2ools is the open-source version of the set of management utilities and command-line tools used with Amazon's EC2 and S3 services, called Amazon's EC2 API tools. They implement a large list of image, instance, storage, network, and security management features including:

- Query of availability zones (i.e. called "clusters" in Eucalyptus)

- SSH key management (add, list, delete)
- VM management (start, list, stop, reboot, get console output)
- Security group management
- Volume and snapshot management (attach, list, detach, create, bundle, delete)
- Image management (bundle, upload, register, list, deregister)
- IP address management (allocate, associate, list, release)

3.2.3 Hybridfox

Hybridfox is an open source extension for the Mozilla Firefox web browser that helps manage both Amazon EC2 or Eucalyptus user accounts from a single interface. It is an alternative for a cloud user to the command-line tools, and although it also implements administrator tools it does not cover all the functionality of these tools. It was used my experiments as the interface for hypothetic end user running on a Mac OS X environment. The main capabilities of Hybridfox are:

- Creating instances of a VM with a Private IP address.
- Support for Eucalyptus 1.5.x as well as 1.6.x
- Other usability enhancements

3.2.4 KVM

Kernel-based Virtual Machine (KVM) [44] is a full virtualization solution for Linux. It is based upon CPU virtualization extensions (i.e. extending the set of CPU instructions with new instructions that allow writing simple virtual machine monitors). KVM is a new Linux subsystem (the kernel component of KVM is included in the mainline Linux kernel) that takes advantage of these extensions to add a virtual machine monitor (or hypervisor) capability to Linux. Using KVM, one can create and run multiple virtual machines that will appear as normal Linux processes and are integrated with the rest of the system. It works on the x86 architecture and supports hardware virtualization technologies such as Intel VT-x and AMD-D.

Eucalyptus supports running on either Xen or KVM virtualization. Because Xen (which appeared first in 2007) has been around for longer than KVM, and also is the underlying

virtualization system of the biggest cloud vendor, Amazon, there is much more research regarding Xen. This fact and the promising features of KVM's integration with the Linux kernel caused me to choose to run KVM over Xen in my cloud testbed.

KVM has built in support for live migration, which refers to the ability to migrate a virtual machine from one host to another one without interruption of service. This migration is performed transparently to the end-user, without deactivating network connections or shutting down the applications running in the virtual machine.

Detailed information about KVM is provided in section 3.5, along with the performance analysis of the testbed private cloud.

3.3 Eucalyptus modules

The Eucalyptus cloud platform is composed of the five software building blocks. Details of this software are described below.

3.3.1 Node controller (NC)

An Eucalyptus node is a VT-x enabled server capable of running an hypervisor, in our testbed this was KVM. A Node Controller (NC) runs on each node and controls the life cycle of virtual machine instances running on the node, from the initial exectution of an instance to the termination of this instance. Only one NC is needed in each node, and it is responsible for controlling all the virtual machines executing on a single physical machine. The NC interacts with the OS and the hypervisor running on the node, and it interacts with a Cluster Controller (CC). The NC is also responsible for querying the OS running on the node to discover and map the node's physical resources (CPU cores, memory size, available disk space) and reporting this data to a CC.

3.3.2 Cloud controller (CLC)

The Cloud controller (CLC) is at the top of the hierarchy in a private cloud, and represents the entry point for users to the entire cloud infrastructure. Each Eucalyptus cloud needs one and only one CLC, installed in the physical server that acts as a front-end to the whole infrastructure. It provides an external web services interface, compliant with Amazon's Web Services' interfaces, and interacts with the rest of Eucalyptus components on the other side. The CLC is responsible for authenticating users, monitoring instances

running in the cloud, deciding in which cluster a requested instance will be allocated, and monitoring the overall availability of resources in the cloud.

3.3.3 Cluster controller (CC)

One or more physical nodes, each with its NC, form a cluster, managed by a Cluster Controller (CC). The CC can be located either on a dedicated server that is able to access the nodes and the cloud front-end simultaneously or, if in the case of a single node cluster, directly in a node. Its main tasks are:

- Deploying instances on one of its related nodes upon request from the CLC.
- Resource arbitration: deciding in which physical node a given instance will be deployed.
- Manage the networking for the running instances and control the virtual network available to the instances.
- Create a single virtual network or grouping instances on virtual networks depending on the Eucalyptus networking mode established.
- Collection of resource information from NCs and reporting this information to the CLC.

3.3.4 Walrus storage controller (WS3)

The Walrus storage controller provides a persistent simple storage service using REST and SOAP (i.e. different style architectures for web services[1]) APIs compatible with Amazon's S3 APIs. Persistent means that it is not exclusively linked to some instance and thus the contents of this storage persist even when instances are terminated. Therefore the Eucalyptus Machine Images (i.e. templates used to launch virtual machine instances. More on this in section 3.4.3) are stored in this stable storage.

3.3.5 Storage controller (SC)

The Storage controller provides persistent block storage for use by the instances, in a similar way to Amazon's Elastic Block Storage (EBS) service. The main purpose of the

[1]In simple terms, in REST each URL is a representation of some object, while SOAP is a protocol for web services

block storage service is providing the instances with persistent storage. This storage can be used for the creation of a snapshot: capturing a instance's state at a given moment for later access.

3.4 System and networking configuration

3.4.1 System design

The latest version of Eucalyptus is prepared to run on very slim resources, for example it can be run on a single physical machine. A single physical machine configuration of Eucalyptus is limited in a lot of ways though (e.g. it cannot be used to create isolated virtual networks, therefore being useless for testing network isolation between virtual machine instances. See Table 3.1), due to the fact that Eucalyptus currently limits the networking modes that can be used in a single machine configuration only to SYSTEM mode. Therefore, although it is not a system requirement, a multiple machine setup is needed to fully test Eucalyptus functionality. For this research a two machine configuration was used:

- Front-end server, running the cloud controller, cluster controller, Walrus storage service, and storage controller.
- Node, running the node controller.

3.4.2 Network design

The networking as implemented is outlined in the text that follows. An overview of this network is shown in Figure 3.1.

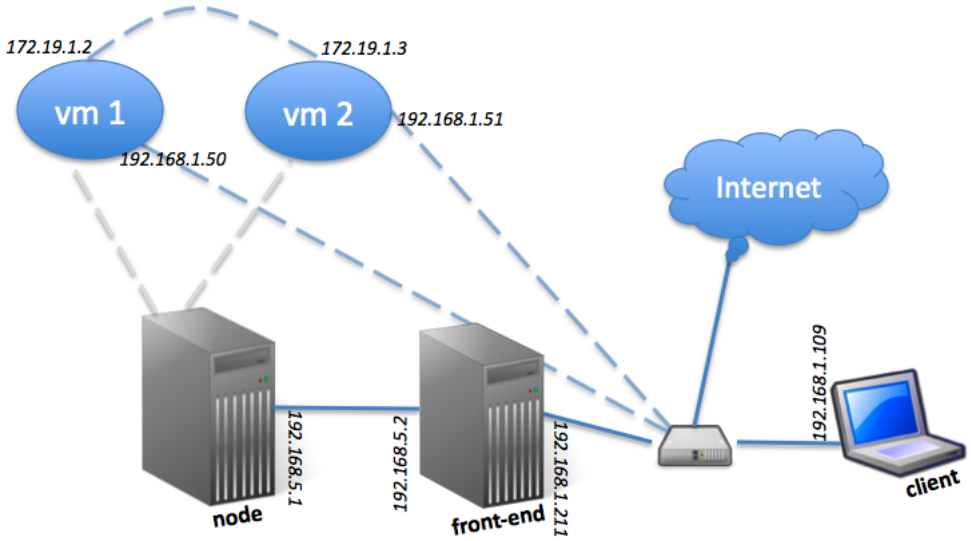

Figure 3.1: Networking outline of the private cloud

The two servers (marked node and front-end in the Figure) were interconnected using an ethernet crossover cable, creating a private network. The front-end had two ethernet interfaces so that it was able to simultaneously access the private network and the local area network of the lab (this later network will be called the public network). This public network in turn has access to the internet, but it is important to highlight that the node itself does not directly access to the internet. By configuring the routing tables of both machines appropriately, the node directs all of its traffic to and from the internet through the front-end. During the experiments, the clients of the cloud were located generally on the public network. Additionally, we succesfully tested the ability of a machine connected to the internet to access the cloud, launch virtual machines, and communicate with them from hosts attached to the internet.

The networking configuration and the resulting connectivity can be quite different depending on which Eucalyptus mode is used. There are four such modes and the main distinguishing feature among them is the level of control and management features offered to the cloud administrator (see Table 3.1). In increasing order of configuration features

these modes are: SYSTEM, STATIC, MANAGED-NOVLAN, and MANAGED. Only the two most feature rich modes, MANAGED and MANAGED-NOVLAN, were used in the experiments. Additional information about both of these modes is given in the following sections, as their details are important in order to understand the experiments. It is also worth saying that, although this four level classification is inherent and particular to Eucalyptus, most cloud platforms offer very similar management techniques and methods.

Table 3.1: Eucalyptus networking modes

Networking Type	DCHP server running on the network?	CC runs own DCHP server?	Instance isolation	Private IPs	Ingress filtering
SYSTEM	Required	No	No	No	No
STATIC	No	Yes	No	No	No
MANAGED-NOVLAN	No	Yes	No	Yes	Yes
MANAGED	No	Yes	Yes	Yes	Yes

MANAGED-NOVLAN

The MANAGED-NOVLAN mode is set by default during the initial configuration of Eucalyptus. In this mode, the Eucalyptus administrator specifies in the cloud controller configuration file a network from which VM instances will draw their IP addresses. As with every mode (but the SYSTEM mode), the cloud controller provides a DHCP server (in our case this will be located in the front-end server) with static mappings for each instance that is launched. The DHCP server allocates IP addresses when the request is sent to the NC to raise an instance.

This mode also features what in Eucalyptus is called security groups: a named set of rules that the system applies to the incoming packets for the instances. The intention of security groups is to provide the instances with ingress filtering. Each security group can have multiple rules and only those incoming packets that match the rules will be let in. These rules are composed of fields (such as protocol, destination, or source port). The security group of an instance is specifyied prior to its launching. Security groups only apply to incoming traffic, there is no egress filtering so all outbound traffic is allowed.

MANAGED

This mode is nearly identical to the MANAGED-NOVLAN mode but adds the ability to create multiple VLANs through tagging of packets. According to the Eucalyptus documentation, this VLAN capability provides the cloud with "instance isolation" but, as it will be clarified by the network experiments (section 3.5.4), this is *not* a very accurate term and can lead to a misunderstanding of what is offered.

The security group concept is more powerful in the MANAGED mode thanks to its multiple VLAN capability. In this mode, the cloud controller divides the entire range of IP addresses available to executing instances, thus creating subsets that are allocated to each security group created by the user. When launching an instance, the user specifyies a security group to which this new instance should be associated. As a result, all instances that are part of the same security group use IP addresses from the same subset. Therefore, there is a difference in the use of security groups as compared with the former mode; here a cloud user is capable of creating instances that are in different virtual networks with each VLAN applying different sets of security rules. Experiments exemplifying this characteristic are described in section 3.5.4.

3.4.3 Configuration process

Before undertaking experiments in the cloud some configuration and tuning must be performed. Some notes about this process are provided in this subsection. The main tasks to be commented are: creating new cloud client accounts via a web interface and building an Eucalyptus Machine Image (EMI).

Web Interface

To access and begin using the private cloud the client uses a web interface, located at http://192.168.1.211:8443/. This is the IP address of the front-end server. Prospective users request an account that will have to be approved by the cloud administrator. Should the user be granted access, the user utilizes one of two types of credentials: an RSA key to bring up instances and a query interface key to retrieve information from the cloud controller via Euca2ools or Hybridfox.

Building an Eucalyptus Machine Image (EMI)

An EMI is a special type of virtual appliance used to instantiate a virtual machine within the Eucalyptus private cloud. It has three main components: a virtual disk image, a ramdisk image, and a kernel image. The virtual disk image is a filesystem image which includes an operating system and any additional software and data files required to deliver a service or a portion of it. This virtual disk image is bundled along with the ramdisk and kernel images, and then compressed, encrypted, signed, and split into a series of smaller chunks that are uploaded into the WS3. Together with these three main pieces of the machine image is an XML manifest file that stores essential information about the EMI (such as name, version, architecture, and the decryption key).

Building an Eucalyptus machine image is not strictly necessary to begin using the private cloud as there are already several different EMIs available for downloading for free. However, creating an EMI is very useful as it automates some tasks that otherwise would need be done everytime a virtual machine instance was raised. Some benchmarking software that will be commented later on was used for performance analysis, by including these programs into a customized EMI they are available in every instance, offering a great time saving.

There are several ways to create a customized EMI, all of which boil down to either creating a completely new EMI or modifying an existing one. As I only needed to add a few software packages I chose to resize and modify an existing Ubuntu server 10.04 EMI [40]. The process to do so was:

- Create a 5GB space for the new image: dd if=/dev/zero of=new.img bs=1M count=4096
- Creating temporary mount points for the old (i.e. the unmodifyied Ubuntu image) and new images, associating them with free loop devices, creating a filesystem on the new 5GB image and populating it with the Ubuntu server from the old one.
- Temporarily changing the root to the mount point of the new image using the "chroot" command.
- Install the desired software packages there.

Once the EMI is customized, then the only remaining step is to upload it to the WS3 located in the front-end server. That is achieved using the following set of Eucalyptus commands to bundle, upload, and register the parts of the machine image.

- euca-bundle-image

- euca-upload-bundle
- euca-register

Each of the three parts (ramdisk, filesystem, and kernel), when uploaded, returns a code that must be used later to register the three parts as a complete set. If the process was completed succesfully, then the resulting machine image should appear in the list of available EMIs after executing “euca-describe-images”.

3.5 Testing performance isolation in cloud computing

This section presents the results along with introducing the methodology used to test the performance isolation capability of an KVM-powered Eucalyptus private cloud. The main issue is the impact on the performance of a particular virtual machine instance when another VM is making intensive use of one or more physical, and therefore shared, resources. The physical resources being considered are the CPU, memory, disk, and network interface. For each of these, the results of an experiment will be presented followed by a discussion of the feasible causes for these results, supported by additional references to previous studies when necessary. As it has been said earlier, a customized EMI was created to run these tests. This EMI was composed of a clean installation of Ubuntu Server 10.04 and a number of benchmarking and testing programs. In the following subsections, a brief explanation of how each of these testing programs works will be given. If not expressed different, every experiment was performed with two running virtual machine instances of the customized EMI, with: 15 GB disk, 512 MB RAM, and 1 CPU core.

3.5.1 CPU test

To test the effects of processor interference due to background load (i.e. another instance resident on the same server), if any, it should be analyzed how each instance uses the processor. This is a consequence of how Eucalyptus assigns the processor to each instance. In order to learn this, some sort of practical test must be performed as there is not any reference to processor allocation techniques in the Eucalyptus documentation. A simple program in C was created to find out whether there is processor sharing between instances or not. This program consists of a while loop that calls the function gettimeofday() and saves the output in a text file. By graphing the timestamps collected one should see a continuous increasing line. Otherwise, if there is any processor sharing, some gaps will appear along the graph. In this case the result shows no gaps, so it must be assumed that each instance has a single CPU core available all of the time.

To test the CPU performance without involving other resources, the benchmark must be designed to be strictly CPU-bound, without disk or memory accesses. Therefore, an arithmetical only task is the perfect fit for this kind of test. There is a well known example of this: the calculation of digits of the number Pi [53]. Measuring the completion time for a specific number of digits will provide direct information about the performance of the processor. With the test instance, that will be called from now on the well-behaving

instance, idle otherwise, the benchmark is run 100 times, each time calculating 524288 digits of the number Pi and obtaining 100 samples of the completion time.

Afterwards, the whole process is repeated but with a co-located instance running a CPU intensive task, modeled using the program "Lookbusy" [13]. This is an application that can generate and maintain for a desired amount of time a predictable fixed amount of load on a CPU. Lookbusy is also a strictly CPU-bound task but, to ensure it actually is, a simple verification is done: we measure the completion time of a run of the program. It can be observed that the real time equals the user time, which means that there was no I/O involved in the process. Given this check, the program is set to generate a 100% CPU load. This will use all of the processor resources allocated to the background instance, the "misbehaving" instance, to its maximum. Following this, another 100 samples of the completion time for the calculation of the number Pi to the same number of digits as before are collected and compared to the first set of samples. The results can be graphed in Figure 3.2:

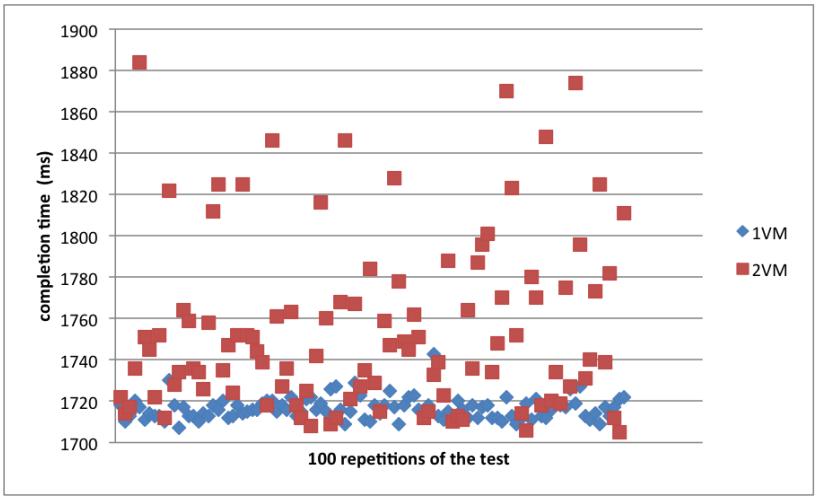

Figure 3.2: Completion time for the calculation of the number Pi (in miliseconds)

Table 3.2: Numeric results of the cpu test

	1 VM	2 VM
mean (ms)	1716,48	1754,90
stdev (ms)	5,39	41,47
stdev (%)	0,31	2,36

As shown in Figure 3.2 and summarized in Table 3.2, the mean completion time for the idle background case reports that a vast majority of samples are grouped around 1720 ms. Whereas, with a misbehaving processor hogging the CPU resources operating in the same server, this mean time increases slightly. The absolut increase, in mean, does not suppose a heavy worsening of the performance. What is remarkable though, is how the variability increases. As can be better seen in the graph: there are several outliers, a great number of outliers for a set of samples of this size, leading to a standard deviation eight times greater than the original.

This result is significant because, as noted earlier, each instance receives an entire core. This is not always achievable, depending on the available physical processor resources. The amount of processor assigned to a VM is called the Virtual CPU (VCPU). In this experiment each VCPU equals exactly one physical core, without sharing. However, it is common to have a VCPU that is half or less than one core of a multiple core CPU. Given the measurements from this experiment, it is reasonable presuming that when processor sharing exists the jitter would be far higher.

3.5.2 Memory test

The memory performance stress test is based upon a bandwidth test, as this is what distinguishes between types of memories. To measure the memory bandwidth the STREAM memory benchmark [47] has been used, following recommendations of previous studies in this area [4]. In this measurement a data set of 100 samples, from 100 consecutive runs of the benchmark, were collected. Every run consists of 10 trials selecting the best result, which directly translates into the sample. The size of the RAM memory allocated to both instances is 512 MBytes.

In this case the misbehaving instance simulates a RAM intensive task of another cloud user's application resident in the same server. This is modeled also using the STREAM memory benchmark, although the data is always gathered from the well-behaving instance. This is shown in Figure 3.3 and summarized in Table 3.3.

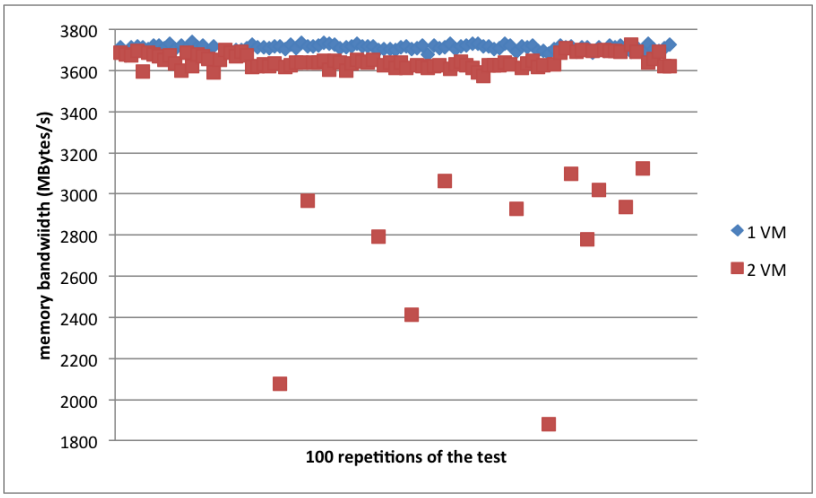

Figure 3.3: Memory bandwidth (in MB/s)

Table 3.3: Numeric results of the memory bandwidth test

	1 VM	2 VM
mean (MB/s)	3711,70	3540,68
stdev (MB/s)	13,83	321,69
stdev (%)	0,37	9,09

As can be extracted from the results, this problem has a similar result as the case of the processor interference. There is again a decrease in the average throughput of the memory, in this case of less than 5%; which in most cases would not be a determinant factor. A bigger worry is the points far from the mean. The standard deviation, in presence of a second virtual machine instance, is more than 20 times (or more than 9 percentage points) higher. This fact could be seen as a much worse tolerance of the memory than the processor to a background load. Because, while it is true that the glitches in memory throughput reach very low values, they go to even half of the mean bandwidth, they happen much more rarely than for the processor.

Currently, memory management techniques that apply to KVM-based virtual machines are the same as those the Linux kernel that applies to any other process. The memory of a virtual machine is stored in memory as any other Linux process, managed by the standard Linux memory allocator. This is due to the fact that KVM is a Linux kernel extension. The kernel has a feature called Kernel Same-page Merging (KSM), that scans the memory of each virtual machine, detects where virtual machines have identical memory pages and

merges these into a single page shared between the virtual machines. This results in space saving given as it allows storing only a single copy of each page. If at anytime a guest wants to change this shared page it will be provided with a private copy of its own. Due to the characteristics of the test, with the misbehaving instance running exactly the *same* benchmark as the well-behaving one, it is assumable that there can be a signficant number of replicated memory pages. Therefore there is a chance that the KSM feature may be interceding frequently, in turn could leading to lower memory throughput. This is only a hypothesis, and should be proved by rerunning this benchmark with a different memory load generator for the misbehaving instance, but it was not given much importance because of the relatively low number of outliers that appeared in this test.

3.5.3 Disk I/O test

The I/O access from KVM virtual machines is different than in other hypervisors such as Xen or VMware. KVM leverages hardware assistance to virtualization provided by the newest chipsets to its full potential, creating a third additional CPU operating mode besides the existing two (user and kernel): the guest mode. CPU and memory are managed directly from the guest mode, using the standard Linux procedures and scheduling policies (the current standard Linux process scheduler is the Completely Fair Scheduler, CFS). Disk I/O and network devices are managed from user mode. The normal code is run inside the virtual machine by the processor in guest mode until an I/O instruction is encountered or an external event occurs (i.e. a common external event is the arrival of a network packet). When any of that two situations happens the processor switches to kernel mode where it evaluates the nature of the interruption. If it is an I/O instruction the processor exits to user-level, where the I/O is performed, ultimately returning to guest mode, to the point in the code just after the I/O instruction.

In a virtualized server, the physical server's disk capacity and bandwidth is typically shared between competing VMs. The capacity is shared in a straightforward way: each virtual machine has a virtual disk image of a determined size that is allocated at the virtual machine starting time. It does not change until the termination of the virtual machine execution. On the other hand, the bandwidth of the disk is shared between all the resident VMs and there is currently no method of dividing this bandwidth or imposing limitations on its consumption by VMs. Therefore, the assumption was that the disk I/O performance of one user would be interferred by another user's VM with intensive disk I/O behaviour.

Two types of experiments were performed to evaluate quantitatively the level of perfor-

mance degradation that occurred. The first was the Iozone benchmark [51]. This is one of the most widely used tools for filesystem benchmarking. It allows benchmarking any possible disk operation, with selectable block size, file size, and number of threads running the program. It is also capable of expressing results in throughput per operation or time per operation. Three different kinds of measurements were obtained with Iozone. In each of which results in throughput and time were calculated:

1. Small read/write operations. Block size of 4KB, file size of 100MB. One Iozone thread on VM1 against varying number of threads in VM2. (1,2,3,4,5,6)

2. Large read/write operations. Block size of 10MB, file size of 500MB. One Iozone thread on VM1 against varying number of threads in VM2. (1,2,3,4,5,6)

3. Small read/write operations. Block size of 4KB, file size of 50MB. Two Iozone threads on VM1, against varying number of threads in VM2. (1,2,3,4,5,6,10,12)

In the case of the writing benchmark, with small operations (experiment 1) the performance dropped 70% with only one thread running in the misbehaving VM. Although when the number of threads increased, up to 6 threads, the degradation kept stable at that value, reaching a maximum of 73%. The behaviour is nearly the same with larger block and file sizes (experiment 2) but slightly more progressive, dropping a 64% when the misbehaving VM features a single thread and keeping around a 77% drop from there onwards.

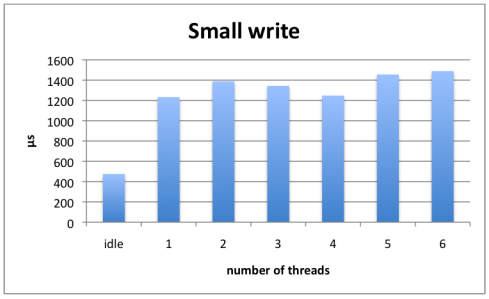

Figure 3.4: Small write operation

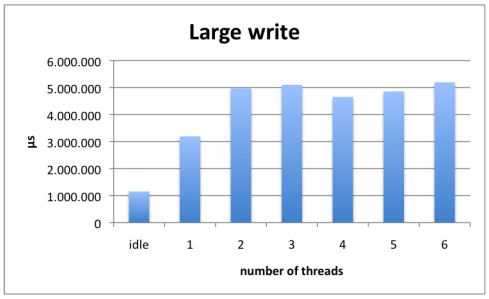

Figure 3.5: Large write operation

A similar situations is observed reading in small blocks (4KB) from a relatively small file size (100MB). This operation's latency drops to the 45% of its peak performance (i.e. the latency measured when there is not any other VM performing disk I/O operations), from 6 to an average of 11 microseconds. A completely different thing happened when reading in large chunks from a larger file. Very little degradation is experienced, what is shocking

after having seen the previous read or write measurements. As it seems an exception this may be signalling some error in the procedures of this particular test, so a second set of experiments was performed.

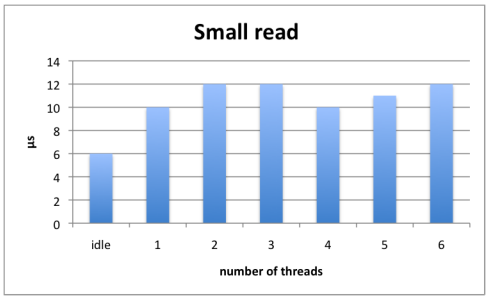

Figure 3.6: Small read operation

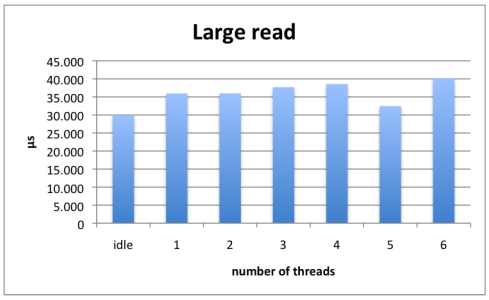

Figure 3.7: Large read operation

The second part of the disk I/O performance experiments was aimed at correcting a possible flaw in the first part of the measurements. Admittedly, using Iozone to create an intensive read/write background load is completely valid, but can hinder the repeatability of the measurements. This is due to the fact that Iozone always benchmarks the different operations in the same order and, when the test being performed in the well-behaving VM has about the same duration as the benchmark being run in the misbehaving one to simulate disk activity, this could lead to some of the tests being always performed against the same I/O operation. Avoiding this situation requires the misbehaving VM to switch rapidly between read and write operations. This behaviour was implemented with a script that combines the following write and read instructions in a loop:

- dd if=/dev/zero of=w_temp count=50 bs=1M > output.out
- cat r_temp > /dev/null

The caches were cleared ar each step of the loop to make sure that the reading/writing is being done, *without* using memory buffers.

The data yielded by the Iozone test shows a decrease in the performance of 60% in the throughput of the writing operation and 68-70% in its latency. Whereas, the read operation shows better isolation of the interference: the throughput drops 20% and the latency an average of 30%. These results remain stable through multiple repeatitions of the experiments, so the methodology is considered valid. What was suggested before is now confirmed: co-resident VMs affect disk I/O operations of other VMs dramatically degrading

its performance. It is important to note that, while that is true (i.e. disk I/O interference happens), there appears to be a ceiling for the amount of degradation that it is not surpassed by adding further threads of I/O interference. For instance, the write throughput decreases to 41% of the baseline value when a misbehaving VM runs one thread of alternate I/O operations, but it only drops about 15% more when there are four threads.

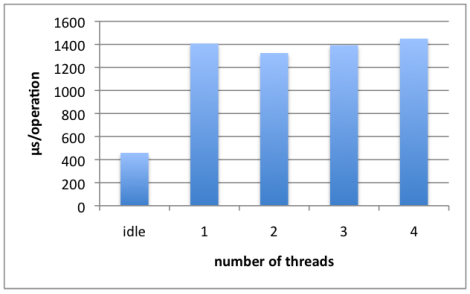

Figure 3.8: Iozone write test (time)

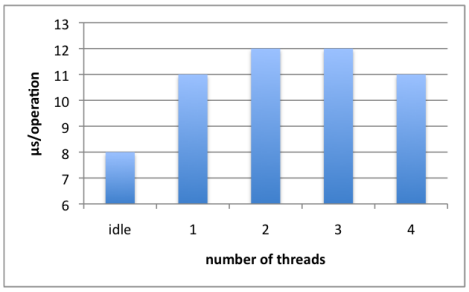

Figure 3.9: Iozone read test (time)

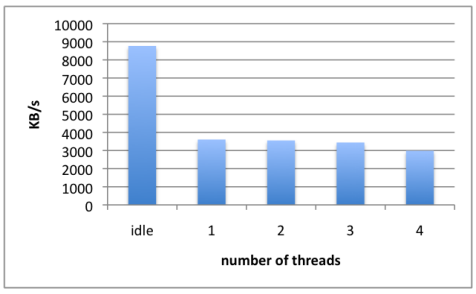

Figure 3.10: Iozone write test (throughput)

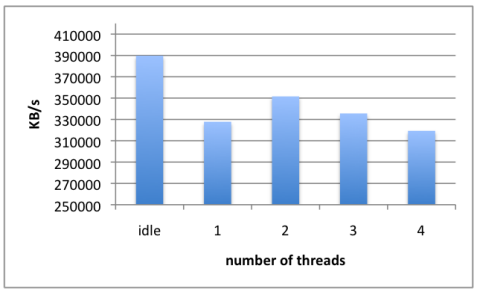

Figure 3.11: Iozone read test (throughput)

For a better grasp of what this interference means, a last experiment is conducted using Postmark [43]. This is a benchmark that recreates the I/O workload of a large e-mail server, creating a large pool of continually changing small files. It is configured to generate 3000 files ranging in size from 1 to 20KB and perform 15000 read/write transactions over this set of files. The elapsed time of this experiment is measured, first with the co-located VM being idle, and then progressively adding from one to four threads of the disk I/O intensive script. Each run of the experiment is repeated twenty times, with the following outcome.

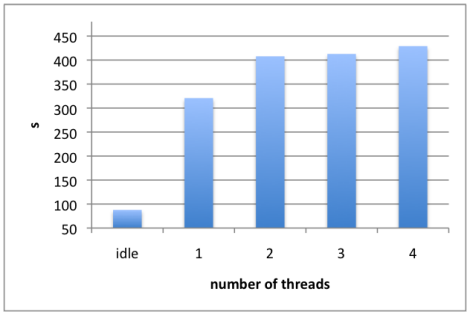

Figure 3.12: Postmark elapsed time

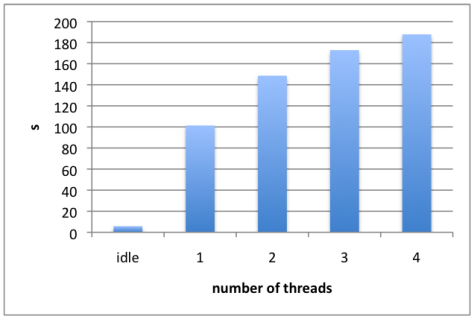

Figure 3.13: Postmark standard deviation

As seen in figures 3.12 and 3.13, the degradation can be a major problem for applications that make intensive use of disk I/O, similar to this e-mail server simulation. It takes 454% of the baseline elapsed time to complete the same amount of transactions when there is any other disk intensive VM using the same physical disk. Not only the total performance is degradated, but it can be noted that the variability of the results increases proportionally to the number of threads in the misbehaving VM. The latter is essential, because the scheme of accesses that the disk perceives in a server with various consolidated VMs is much more random. This is the main underlying problem with disk I/O in virtualized environments as current mechanical disk technologies show a performance orders of magnitude slower when accessed in a random manner. Because, besides the physical disk itself, the disk scheduler works poorly under such a load. Previous research found that current disk scheduling methods are not appropriate to work with the random latencies that virtual disks of guest OSs have, limiting their ability to provide isolation and fair utilization of the VM's share of I/O resources. This same research shows the need to modify the current scheduling in virtualized environments, that happens at the same time at the VM and at the hypervisor level: the VM level scheduling should leverage the knowledge of application information, whereas the hypervisor level scheduler should make best use of the underlying disk technology. VM level disk schedulers should not be built with strict rules about the disk behaviour. At the same time, new semiconductor based disks could help out solving this issue [4]. Reportedly, semiconductor disk technologies like Solid-State Drives (SSD) have faster random access latencies as there is no seeking motion required because they do not have moving mechanical parts and the physical location of data is irrelevant. This could allow more VMs to be placed succesfully in the same physical server, this way serving as a factor to lowering costs of cloud computing.

3.5.4 Network test

Having seen the disk I/O interference problems, it is expected to find similar issues in the process of sharing another resource: the network adapter. What would be fitting to the purpose of cloud computing is a network sharing methodology that divides the available bandwidth and provides network isolation maintaining network latency values when multiple VMs share the link.

To test the network isolation capability announced by Eucalyptus, the system was switched to the MANAGED mode. Now, Eucalyptus creates a virtual network for every new instance of VM launched, the same way as described in previous sections. The VM instances maintain connectivity via the LAN through bridges that share the network interface of the node.

The network bandwidth measurements were performed using Iperf [66], an open source network testing tool that can create TCP and UDP data streams and measure the throughput of the network that is carrying them. It implements a client and server scheme to measure the network performance between two ends. Experimentally, it is easily checked that the KVM virtualized system has no built-in system of bandwidth fair-sharing between VMs: every time concurrent TCP connections to clients in the LAN are started from two VMs, each of them gets a different share of the link bandwidth and has the ability to starve the other depending on which connection begins first. Besides KVM, the most popular hypervisor nowadays and the one being used by Amazon EC2, Xen, does not feature any bandwidth sharing mechanism either. Therefore, in current clouds, the network sharing system must be implemented by the network administrator. However, Eucalyptus does make it easier to implement some sort of solution thanks to its VLAN separation of VMs. One alternative is to create a traffic shaping mechanism taking advantage of this Eucalyptus' feature, using the Linux traffic control tool (tc) [3].

The Linux traffic control tool allows attaching queuing disciplines, called "qdisc", to the network devices. These qdiscs can either have multiple child classes or not, in which case they have only a single parent class. These classes form a system of channels or flows in which the outgoing packets are classified, and over each flow some rules to limit the bandwidth or other parameters, such as priority, can be specified. Each class, in turn, has a default queuing discipline that can be changed. Generally, the tc tool filters are used to classify packets among all the flows but, as in the system of this research Eucalyptus has already facilitated this by separating different VMs in different VLANs, hence no filters will be needed. The queuing discipline chosen is the Hierarchical Token Bucket (HTB) [22],

whose design suits this experiment since it facilitates guaranteeing bandwidth to classes. This is only for the outgoing traffic as the incoming traffic utilizes another methodology called ingress filtering or policing (i.e. basically dropping packets when the number of them received exceeds the bandwidth specified in the filter). To check the current traffic control discipline of one of the virtual networks using tc you use the command:

```
tc -s -d qdisc show dev vnet0
```

A short script was created to configure each virtual interface shaping and policing discipline. This script is:

```
#! /bin/bash
DEV=vnet0
#outgoing traffic
sudo tc qdisc add dev $DEV root handle 1: htb default 12
sudo tc class add dev $DEV parent 1: classid 1:12 htb rate 47mbit
#incoming traffic
sudo tc qdisc add dev $DEV handle ffff: ingress
sudo tc filter add dev $DEV parent ffff: protocol ip prio 50 u32 match ip /
src 0.0.0.0/0 police rate 50mbit burst 10k drop flowid :1
```

This script sets the outgoing bandwidth share of one VM instance to 47 Mbps, and the incoming share to 50 Mbps. These values where found experimentally by choosing the highest possible value that did not cause packet loss when testing with iperf in UDP mode (that is because in TCP mode Iperf does not allow choosing the bandwidth used in each test, instead by choosing the highest value possible, without showing the packet loss results). In practice, this values turn out to be 45 Mbps both in the uplink and in the downlink. Before moving on, some testing was done to be sure that the traffic shaping setup works. Starting simulataneous connections from two VMs shows that now each one obtains its fair-share of bandwidth every time. Therefore, traffic shaping solves the problem of limited bandwidth while leading to fair sharing pretty well.

Although bandwidth is now being fairly divided, the isolation of latency and packet loss remain to be proved. To test the network performance isolation, two clients located on the laboratory LAN (the external network of the cloud) were used. One of them was connected with the well-behaving VM instance and the other one with the misbehaving one. Since the goal was to examine both transmitting and receiveing by the server, to see if the traffic

shaping could enforce the desired isolation, two types of stress test were designed. In the first, the misbehaving VM acts as the sender while in the second one it acts as the receiver. Either when it is sending or when the client that connects with it sends, the background traffic load is created as concurrent UDP connections at different data rates.

Initially, a large set of baseline round-trip time (RTT) measurements between a client and a VM were collected by sending 9000 pings in three sets of 3000 with an idle interval of several minutes between each set. This operation was repeated with the misbehaving VM acting as if it hosted a network intensive application, in two phases: first sending and then receving. The methodology of the stress latency test is to collect RTT measurements between one client and the well-behaving VM while another client is connected to the misbehaving one.

For both the sending and receiving stress tests the latency results have average values that grow in direct proportion to the data rate of the misbehaving connection (see table 3.4). Although the increase is not very significant the fact that there is any difference given that the bandwidth is being fairly-shared correctly was surprising. What is more worrying is that the maximum RTT values of the baseline measurements are nearly the same in every set of samples collected, but, when there is background network activity, there are consistently higher maximum values that reach as much as ten times the baseline maximum value. This can be seen in the deviation of the ping times (see table 3.4): while the deviations were the same for the three sets of samples in the case of the baseline measurement, the deviations were more than double this value for every set of samples collected with background network activity.

Table 3.4: Latency stress test results (3 sets of 3000 samples)

SENDER (Mbps)	avg (ms)	max (ms)	dev (ms)	RECEIVER (Mbps)	avg (ms)	max (ms)	dev (ms)
baseline	0,688	0,792	0,044	baseline	0,688	0,792	0,044
baseline	0,688	0,787	0,043	baseline	0,688	0,787	0,043
baseline	0,687	0,788	0,043	baseline	0,687	0,788	0,043
40	0,847	1,933	0,093	40	0,796	2,601	0,129
40	0,851	2,870	0,102	40	0,799	6,455	0,111
40	0,847	1,112	0,101	40	0,798	2,002	0,116
45	0,924	6,804	0,220	45	0,823	3,292	0,126
45	0,900	4,772	0,167	45	0.828	3,213	0,219
45	0,919	1,375	0,099	45	0.824	8,103	0,117

Since this test showed the variation in the RTT that appears when there is a co-located network intensive VM, further experiments in this direction were performed. Using Iperf, a connection to the well-behaving machine was maintained while the other VM sent or received flows of UDP packets, packet loss and jitter results were obtained. The test involves incoming and outcoming packets and during each part of the experiment both VMs were transmitting in the same direction as this is the case of most interference.

Figure 3.14 depicts the amount of jitter (which refers to the variability of the packet latency across a network) for various combinations of data rates of both VMs. Overall, jitter was quite proportional to the throughput of the competing VM. Although increasing slightly, jitter remains low when the competing VM is sending or receiving at a fraction of its fair-share of bandwidth, but when it uses all of its share, the jitter doubles in the outgoing case and is almost two orders of magnitude higher in the incoming case. Both cases show levels of jitters above what is normal for a small ethernet LAN. Nonetheless, the fact that the incoming case reaches 2.5 ms of jitter at 40 Mbps is clearly an important issue.

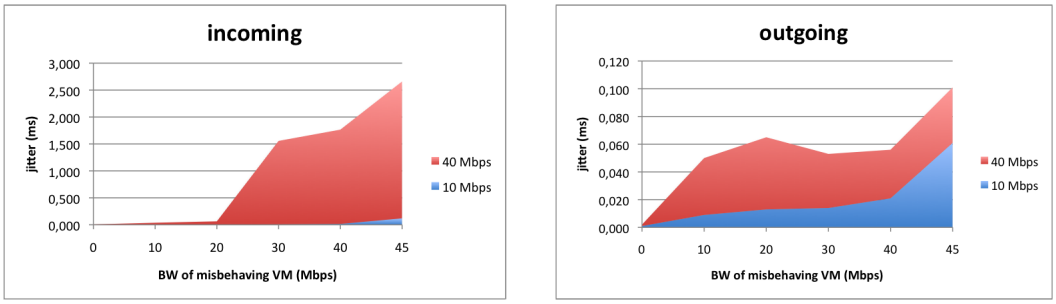

Figure 3.14: Jitter stress test results

Likewise, packet loss levels remain around zero when the misbehaving VM sends or receives at 10 Mbps, but is higher when it uses all its available bandwidth (see Figure 3.15). In contrast with the jitter case, packet loss is worse (i.e. higher) in the *outgoing* situation. Overall the levels of packet loss are not critical, but for a cloud service, with probably packets being dropped along the way to the end-user, the levels of packet loss should be kept minimum (preferably zero) inside the datacenter. Further discussion about this point will be presented in the following chapter.

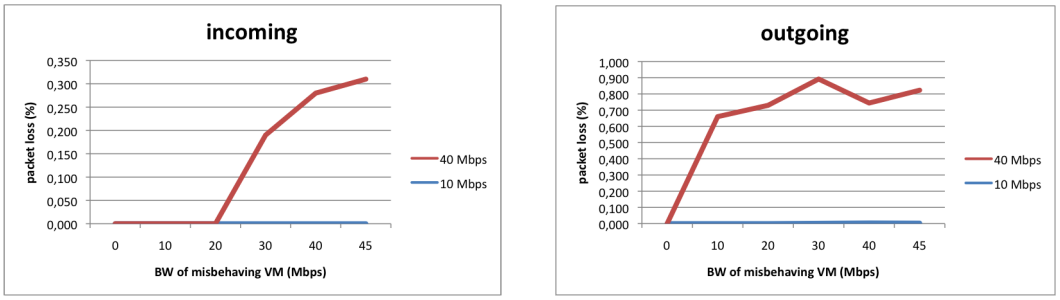

Figure 3.15: Packet loss stress test results

Chapter 4

Cloud performance factors and Service Level Agreements

4.1 Determining performance behaviour parameters

Drawing from the previous chapter's results, it has been proved that the environment where an application executes determines its performance. Not only does virtualization cause some degradation in the performance of the application itself, but this application can also cause interference with (and receive interference from) co-resident VMs. Current hypervisors either do not have performance isolation mechanisms or apply methods that have not been designed with a virtualized environment in mind (e.g. traditional disk scheduling mechanisms are used inside VM and at the VMM level at the same time, as was discussed in subsection 3.4.4). In the previous chapter, inter-VM isolation issues were analyzed. This is one of the factors that cause the performance of a cloud-based application to not be deterministic and reliable, but this is not the only factor. To pin down all the factors that lead to unpredictable performance in cloud computing it is useful to divide what is perceived as the cloud into two parts. Looking at the whole cloud application delivery chain, a basic distinction can be made: what happens inside the cloud (i.e. within a single datacenter) and what happens along the path to and from the end-user.

4.1.1 Inside the cloud

Inside the cloud refers to the servers and networking gear within the datacenter. Typically these datacenters utilize x86 architecture servers and networking equipments to in-

terconnect these servers: the physical layout inside datacenters consists of 20 to 80 nodes within a rack connected through a top-of-rack switch to a second level aggregation switch and at a third level connected via routers to storage area networks and the Internet or inter-datacenter WANs [4]. Thus, possible sources of variation in the performance of this portion of the cloud application delivery chain are the servers and the connections between them. The only factors that can contribute to variations in the performance of an application hosted on one of these servers, besides what is normal for any physical server, must be consequences of virtualization or other sources related to the datacenter configuration. The effects of virtualization over these servers' performance have been discussed in the previous chapter through hands-on experimentation with an Eucalyptus private cloud, and more of its consequences will be discussed when analyzing the problems with current Service Level Agreements (SLAs) for cloud providers.

With regard to the networking inside the datacenters, a few comments should be made. While the amount of networking inside the datacenter may seem limited it can greatly affect application performance because of the details of how these applications are deployed in the cloud. One of the key advantages of the cloud paradigm is its scalability. The earlier approach to providing performance was by scaling up, meaning adding more computing power to a same server, now increasing the available computing resources is done scaling out, adding another VM instance. Every additional instance created to run an application can be deployed in one physical server or in many physical servers. In the latter case, the communication between the different instances has to be carried over the networking within in the datacenter, this traffic can experience queuing at any of the switchs due to high network load, this delay (and potentially packet drops) could affect the interaction between application components. Figure 4.1[1], extracted from Hill et al. [39], shows the TCP bandwidth between two instances running in Microsoft's Windows Azure cloud.

[1]Reproduced here with the permission of the copyright holder

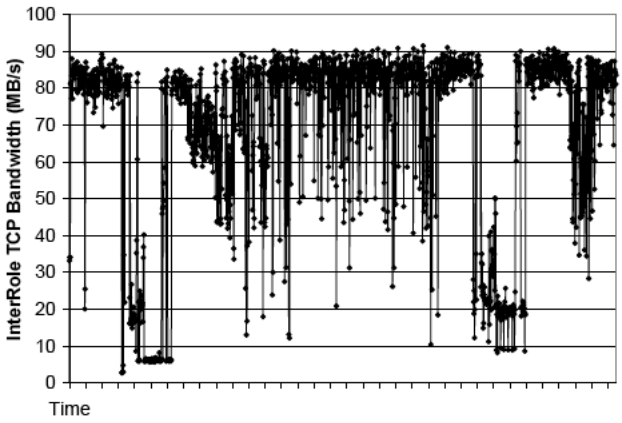

Figure 4.1: TCP bandwidth between two small instances in Windows Azure (Source: [39])

The physical network connection in the Azure cloud is a gigabit ethernet link with a theoretical maximum data rate of 125 MB/s, in a cloud environment. As can be noted in the figure, the actual bandwidth never reaches this maximum and is mostly around 80 to 90 MB/s. However, there are a considerable number of samples below 50 MB/s, some of them are even below 10 MB/s. Hill et al. attribute this highly variable performance to interference caused by neighboring instances located in the same physical platform. This agrees with the results of my own network tests (as described in subsection 3.5.4), where the jitter increased in the presence of co-resident instances making intensive use of the network I/O interface. Although it could not be determined whether Microsoft's Azure cloud uses traffic shaping or some other sort of network sharing techniques, there is a need to implement a mechanism to allow bandwidth sharing between virtual machines without increasing jitter. The same situation, with slightly better performance than Azure's but showing a similar amount of variability, is apparent in Dave Mangot's measurements of Amazon's EC2 service as presented in Figure 4.2[2].

[2]Reproduced here with the permission of the copyright holder

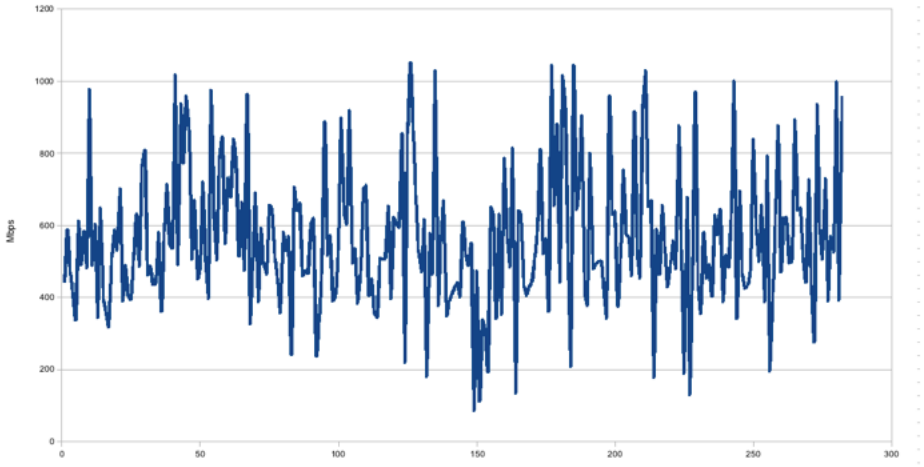

Figure 4.2: TCP bandwidth between two m1.small instances in Amazon EC2 (Source: [46])

Implications of these findings impact most severely distributed architecture applications, such as many scientific computing applications. In distributed applications where messages are sent very often, for instance when using the Message Passing Interface (MPI) when a number of worker processes have to exchange results in order to proceed to the next task. Another example is MapReduce-based applications [21]. MapReduce is a framework introduced by Google and supported in their cloud computing offering App Engine, where a master process divides a bigger problem in smaller parts that are passed to workers (the "map" step), each of which computes on its set of data, and then the results are collected to generate the outcome (the "reduce" step). In both MPI and MapReduce applications if network connections have low throughput and high jitter, then worker processes may have to wait for other workers leading to an accumulation of delays that degrade the overall performance.

4.1.2 From the datacenter to the end-user

Some cloud computing providers have as many as five or six datacenters where users can request virtual machine instances to be hosted. For cloud-based applications with many users distributed intercontinentally, thus many of these users are located very far away from where their application is hosted. The location of the datacenter where an application is hosted matters for legal reasons about data privacy regulations in different countries as has been discussed in section 2.5.7. However, the location also matters for the performance that the end-user perceives, and the variations in these perceptions depending on where the user

is located and what other traffic is competing for the links on the path between the user and their instance(s).

In this part of the delivery chain, the performance parameter that is most important is latency. Although details of the communication path between the datacenter and the users are not directly related to cloud computing, it is important to briefly study what happens along this network path because clouds are widely used to host consumer oriented applications that must compete against traditional applications installed in the user's computer. Therefore each source of performance degradation is worth examining. From this point of view it is useful to look in a broader sense at the sources of web latency, specifically:

Network latency: the time spent travelling across a network.

Processing latency: the time it takes to the server to prepare the content that will be sent to the user. This is the part discussed before, that happens inside a datacenter in a cloud model.

Client-side latency: the time the web browser needs to prepare the received content to be presented.

The most relevant factor in this discussion is the network latency (since we have considered the processing latency in the previous section). This network latency is actually a function of several parameters: the round-trip time between the browser and the server, the number of round-trip times it takes to completely load a web page, the protocol's flow & congestion control properties, and competing traffic. The main source of delay in web applications is often a consequence of the properties of the TCP protocol. This delay is caused by the flow control and congestion control properties of TCP. TCP was not designed to match the characteristics of today's web applications, but instead was optimized for high throughput of large files while preventing congestion in the network. TCP uses the slow start algorithm, which increases the size of the congestion window by the number of data segments acknowledged for each received acknowledgement, until packet loss occurs. This point is interpreted by TCP as the maximum capacity of the connection, and from this point on the TCP's congestion avoidance mechanism comes into play. As of this year, the average size of a web page is 320 KB [56], thus the majority of connections do not reach the maximum bandwidth. Because the initial congestion window is 4KB, most of today's web pages take several round-trips to download while at the same time not exiting the slow-start phase, making an inefficient use of the network link. A common workaround implemented

by almost every popular web browser is opening up to six TCP connections. According to Dukkipati et al. [23], this solution is inefficient because each TCP connection incurs slow-start problem, but initiating multiple TCP connection tricks TCP's congestion avoidance mechanisms. They argue for an increase in the initial congestion window to adapt TCP to the web applications of today.

The experiments by Dukkipati et al. show that this change would favour specially high latency or low bandwidth networks. These two particular cases are the two most common use cases for cloud computing. Low bandwidth networks are characteristic of mobile communications, a large pool of users that could benefit from cloud computing as most mobile phones do not have a large amount of storage space and due to battery power limitations have limited spare local processing capacity. Cloud-based applications delivered through mobile networks match this requirement very well, and there are an increasing number of products following this trend. Secondly, networks with high round-trip times are common in the cloud computing enviroment because, as has been said before, when an application with global reach is deployed in one or a few cloud datacenters it is likely that a large fraction of its users are located far away. Therefore changes to the TCP protocol and modifications to HTTP (such as those proposed by Google's SPDY [37]) would be very advantageous for the growth of cloud computing and improved performance of web browsing.

The impact of the difference in the geographical distribution of users and datacenters is hard to quantify beforehand, without actually deploying a complete infrastructure and measuring the perceived performance. Experimental test to measure the effect of geographical deployment of applications in actual clouds can be expensive. Buyya et al., in [12] and [70], have developed a toolkit that enables modelling cloud environments in order to test different application configurations. This toolkit offers two tools: Cloudsim and CloudAnalyst. Cloudsim [12] is a more complete and fully configurable tool that allows modelling at the level of schedulling and allocation policies inside the datacenter, whereas CloudAnalyst [70] is aimed at the simulation of large-scale parameters such as geographic distributions. In section 3.5 the server-level behaviour of cloud environments was tested with the Eucalyptus private cloud hence in this section the features of CloudAnalyst are used to examine the impact of the network distance between the datacenter and the end-user of cloud computing. As the CloudAnalyst creators have targeted this simulator to cloud-based social networks, the case that will be briefly studied here will be the micro-blogging social network named Twitter [67]. Concretely, the simulation is performed using

data about this network collected around February 2008. The reason for choosing this date is that Twitter, as nearly all social networks, does not provide much official data of its user base and there are few reports or thorough analysis of it. An additional reason is that Twitter was still hosted in the Joyent cloud at this time (i.e. it was still a public cloud based application). Later Twitter switched to a dedicated datacenter provided by NTT America, due to the enormous growth experienced in the latest year that made latencies grow too high. The sources of parameters for the simulation that I have used for this study of Twitter are from [45] and [72] (although the latter source is from 2010, only the peak hours of use were extracted from this source and I have assumed these were the same peak hours as ocurred in 2008). For a simplier simulation, the inside-of-datacenter setup used in the simulation is the same as that used by Buyya et al. in their CloudAnalyst whitepaper. This setup is modeled after Amazon's EC2 infrastructure offering. The geographic locations of users were estimated by comparing the time offset each user sets in his account with the global time zones. For simplicity users of the same region (North America, South America, Europe, Asia, Africa, or Oceania) are assumed to be in a single time zone. Finally, the worlwide latencies were extracted from Verizon's IP statistics web page on October 10th 2010, which is updated on a daily basis [68]. Three geographic deployment scenarios are considered: one datacenter located in the US, one datacenter located in Europe, or two datacenters (one each in the US and Europe). This is the worldwide distribution of the user base of a total 1.400.000 users, according to [45], with the peak hours (i.e. 20-22h, local time) translated to GMT reference, and the number of simultaneous users in the peak and low hours (it is worth noting that the difference between peak and low hours, ranges from 6% to 3% of concurrent users and is not as much of a difference between extremes as in other web services):

Table 4.1: Distribution of Twitter user base (February, 2008)

	Time zone	#users (%of total)	GMT peak hours	peak concurrent users (6%)	non-peak concurrent users (3%)
N. America	-6	700.000	14:00-16:00	42.000	21.000
S. America	-4	112.000	16:00-18:00	6.720	3.360
Europe	+1	308.000	21:00-23:00	18.480	9.240
Asia	+6	280.000	2:00-4:00	16.800	8.400

The regions not included in Table 4.1 did not have a significant amount of users or had none at all during February 2008. Table 4.2 shows the latency results of the CloudAnalyst simulation when serving each user using the nearest datacenter.

Table 4.2: Delays experienced by Twitter's end-users (all values are in miliseconds)

	1 DC (US)	1 DC (EU)	2 DC (US&EU)
overall avg response time	217,29	309,54	143,51
overall min response time	41,00 (North America)	40,34 (Europe)	39,94 (Europe)
overall max response time	650,88 (Asia)	805,68 (Asia)	392,25 (Asia)
avg response time in N. America	80,54	262,36	101,83
avg response time in S. America	203,51	503,40	203,44
avg response time in Europe	269,91	60,54	68,84
avg response time in Asia	504,91	624,96	305,87

Rather than providing accurate and reliable approximations to what Twitter's response times were, the purpose of this simulation was to gain some insight into the effect of the choice of datacenter location (or cloud provider, if it only has one possible location) in deployment of a cloud-based application. Table 4.2 shows the magnitude of the differences that an end-user might experience depending on where the application is hosted. To put these values into context, Twitter considers that the threshold for user satisfaction of a page request is 500 ms, although the goal is to ensure a maximum page load time between 250 and 300 ms for any user [1]. Based upon this point of view, the setups with one datacenter located either in North America or Europe would not be acceptable and only the third deployment, with two datacenters, would achieve the desired maximum latency, while users in Asia would still experience worse performance some of the time. Generalizing from these simulations, leads to the conclusion that it is of major importance to know your application's requirements in terms of end-user experience because the response time requirements of your application and the distribution of its users will be sufficient to compute an aproximation to the number of locations you may need to meet this requirement for these users.

4.2 SLA problem and application models

4.2.1 The problem with current SLAs

The abstracted infrastructure, intrinsic feature of cloud computing, has the side effect that customers have no way to truly know the type of computers their applications are running on. To circumvent this, cloud providers supply customers with a number of performance metrics. For example Amazon's EC2 Compute Unit (ECU) is a metric provided by Amazon to compare the processing resources of the different types of virtual machines they offer. It is equivalent to a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor [25]. This abstraction of the underlying computing resource is indeed inherent to the utility model on which cloud computing is based. This enables a cloud provider to supply computing resources as if they were supplying electricity, on-demand, while the customer only pays for the resources used. When a customer signs up to use a cloud the customers is not renting a particular machine, but rather is renting capacity by the hour.

As has been discussed previously, performance metrics such as Amazon's ECU in a multi-tenant cloud environment are not directly related to the CPU capacity of a specific processor. Due to the absence of succesful mechanisms to enforce perfomance isolation the performance measurement of a instance many not be constant nor reproducible. There is a need for some sort of agreement between the cloud provider and the customer ensuring that the price paid for a specific amount of resources will lead to a performance that fulfills the user's expectations. This is the reason to explicitely a proper service level agreement (SLA). SLAs are the contracts negotiated between the two parties that contain a number of conditions for the delivered quality of service, referred to as service level objectives (SLO), and the corresponding compensations if these specified service levels are not met.

At present, the SLAs of all the major cloud providers follow the same pattern of conditions and compensations, and focus, nearly exclusively, on *availability of service*, referring to the amount of time the service is up and responsive. However, there are some differences in the definition of downtime and the method for compensation. For example, Amazon EC2's SLA [61] specifies 99.95% service uptime (i.e. about 4.3 hours per year of non-scheduled downtime), one of the highest levels of service uptime among cloud providers. In contrast, Rackspace Cloud Server's SLA [63] guarantees 100% uptime (this is a typical selling technique, which basically means they will compensate the customer for any downtime). Unfortunately, if a business depends upon the 100% uptime guarantee it is likely to be disappointed, since the failure to provide this level of uptime simply leads

to a payment by Rackspace which might not compensate the business for their loses due to not being able to operate for the period of time that the cloud is not available. There are also differences in the time required to resolve a claim of unmet SLA, here Rackspace is the only cloud provider that specifies a time limit. The compensation for failing to meet the terms of the SLA is usually proportional to the amount of non-scheduled downtime suffered, but while customers of Amazon EC2 have to prove an uptime of less than 99.95% of availability during the whole year preceding the claim, Google App Engine's [62] or Rackspace's customers need only to prove the uptime was less than the specified amount (99.9% and 100%, respectively) during the previous month. Last but also important is who has the responsibility for both detecting and notifying the SLA violation. Today this is something that all cloud providers agree unanimously upon, this is the responsibility of the customer. This may be one of the biggest issues concerning the current state of cloud SLAs, the burden is put on the customer who, besides suffering because of the SLA violation, has to administratively prove that the violation of the SLA occurred. This is not only a burden, but is sometimes in feasible. Google App Engine or Amazon's S3 cloud storage SLAs define the availability percentage based upon user requests to a cloud customer's application that result in an error. From the position of a cloud-customer/developer that offers an application hosted in the cloud to a large community of users, the complexity of monitoring the parameters that are specified in the SLA is very high, considering that the requests usually come directly fom the users' web browser, making the number of requests that result in errors difficult to determine and difficult to report as an aggregate number. A way to avoid this problem would be for cloud providers to take responsiblity for fully monitoring the service and automatically paying or offering credit when outages occur.

4.2.2 Performance SLA

In the previous subsection the problem of deterministically assessing the performance of cloud offerings was introduced, followed by a review of current cloud SLAs' weaknesses. There is currently no cloud provider that includes in their cloud services' SLA any condition, any SLO, to ensure a given level of performance. Therefore there is no compensation to the customer if the service provided (e.g. an instance in the IaaS case) does not behave as expected. This may be due to the relative immaturity of the cloud computing paradigm, at least with respect to its mass adoption by larger audiences, but it is also a barrier for its growth as this is among customers' top concerns [60] [18]. This problem is exacerbated by the difficulty customers experience in monitoring cloud providers. Current methods range from simple semaphore-like indicators of the service status (i.e. up or down) to monitoring the use of resources (e.g. Amazon's Cloudwatch). There is a clear need for a more thorough monitoring system than those that cloud providers currently offer in order for the customer to understand what the cloud provider's actual performance was (or perhaps even what the current performance is). Some third-party companies have begun filling that gap, most notably CloudSleuth [19]. As of September 2010, CloudSleuth's response time monitoring system shows a huge variation in Rackspace Cloud Server's, with a worldwide 30-day average of around 6 seconds of response time but varying from 850 milliseconds for users in Dallas, USA, to the extremely bad 20 seconds experienced by users in Mumbai, India, or 35 seconds in Beijing, China. While the latter values clearly indicate some exceptional problems, the variation in response time over the world strongly supports the earlier suggestion about thoroughly studying the location of your application's user base(s) and the datacenters that will be hosting your application.

Future cloud services' SLAs should take on the problem of specifying performance, in such a way that the customer the knows limits between which its application performance may vary. A basic requirement for this is that the performance should be mapped in measurable factors. For a cloud service these factors could be:

availability: The status of the service is defined as available when it can be reached and is operational and responsive, and is measured as a percentage of time.

response time: The response time is the delay between when a request is sent by the user and when the response is received, as measured in seconds to some stated precision.

throughput: The amount of work a cloud instance can do in a given time, measured in served requests or transactions per unit of time.

Establishing bounds to these three parameters makes it straightforward for the customer to be sure that their requirements will be met, thus succesfully transferring the responsibility of mantaining performance levels to the cloud provider. In order to achieve an agreement in terms of performance, the cloud customer should know and be able to map the requirements of their application or service to be to hosted in the cloud onto these three factors. This seems an obvious situation, but it is not. According to Durkee [24], customers are inexperienced in purchasing computing resources as a commodity and they are overwhelmed with the complexity of selecting and determining the cost of service, thus many current cloud end customers use price as their primary decision criteria. In part, they could not do otherwise due to the lack of reliable benchmarking and performance guarantees. A new type of SLA which covered performance would enable customers to use factors such as response time as additional decision criteria, elaborating a value for money idea of the offering. Unfortunately, today this requires actually deploying the application in each cloud and testing it yourself.

If performance levels can be included in the negotiation, then the existing difficulty in selecting cloud provider can be reduced to an optimization problem (as most buying decisions can): the customer wants to obtain the maximum value at the least possible cost. In economics, the value that something represents to the customer is called utility. A buying choice between different products can be represented using a utility function which describes the amount of utility provided by every possible combination of the products involved in the choice. Menascé and Ngo [49] proposed the use of utility functions using the three parameters of performance defined earlier would be suitable for selecting cloud providers depending on application requirements. Their proposal aims to obtain a combination of service levels in availability, throughput, and response time that satisfy a given cloud customer, requiring that the customer models this utility functions for each of these parameters (i.e. a function expressing how much satisfaction an "additional unit" of availability, throughput, or response time would bring the customer). In turn, the cloud provider should adapt their pricing model to these performance factors, but here the proposal collides with the current pricing models.

What pricing model is needed to make delivering quality of service guarantees commercially viable is an ongoing discussion in the cloud computing world. The metered usage of service model being used right now is similar to other utility services, such as water or electricity. But the actual difference resides in the process of metering: is easier to measure water or electricity consumption than consumption of computing resources.

Experimentation has shown that variable performance results from other users resource use, but with the current pricing model the customer's usage would have the same cost despite the fact that the customer actually got less performance for a given expenditure in some cases. This does not happen with electricity or water supply. So, either computing is not suitable for being provisioned as a utility or new pricing strategies have to be developed. As more and more performance critical business processes are implemented in the cloud, new methods of determining and monitoring performance have to be deployed to avoid the current problems of metered usage. Prior to the cloud computing concept becoming widely popular, there was research about how to control the performance of three-tiered websites; this research could be adapted to cloud computing. For example, Karma et al. [42] developed a controller that is able to maintain a stable response time without significant degradation in the web server's throughput, and in addition was able to use the feedback obtained by measuring the response times of each HTTP request to adapt to changes in workload.

Further study should consider a performance SLA based on response time, as response time is the customers' top concern. This research could be organized as follows. A customer could either: (1) request an statistical bound on their application's response time, such as demanding that 90% of the time the response time should be lower than a specifically stated threshold, or (2) establishing limits for the average response time and the maximum response time. Any of these options are better than focusing only on average response time, because they address variability more effectively. As the client perceived response time is what really matters and this metric is not available at the server, there are two possible ways to collect this data. The first method is to aproximate the client perceived response time by using server-side measurements (specifically measuring connection drop, accept, and completion rates) and a TCP model, such as the "Certes" mechanism presented by Olshefski et al. [52]. The second method involves a third party who would undertake the process of collecting the client response time measurements, while simultaneously collecting information from the cloud provider's monitoring of the resource usage by the customer's application. In the second method, this third-party would be in charge of enforcing the agreed performance at a maximum hourly cost specified by the customer and, in the event that the SLA is violated, this third party would claim the corresponding financial compensation on behalf of the customer. The first method is more agile because the cloud provider would be able to automatically compensate the customer in the case of SLA violations. Additionally, server-side approximations of client perceived response time would enable the cloud provider to

use this as a threshold in the auto scaling process, rather than the resource consumption metrics currently being used (e.g. Amazon only allows CPU consumption, network activity, or disk utilization to be used as threshold parameters to invoke auto-scaling). This would make auto scaling much more straightforward to set up given the application response time requirements.

4.2.3 Application workload models

Having stated that what matters the most is the end-to-end performance, and particulary the client perceived response time, previous sections of this thesis have focused on indentifying the factors which these metrics depend on. Initially, the problems that exist when sharing hardware resources between virtual machines were reviewed and evaluated through measurements with a private cloud. Next, the effects of networking inside the cloud and between the cloud and the end user of the cloud-based applications were discussed. Additionally other researchers have shown there are other factors that impact cloud performance. For example, there exist substantial differences in the underlying platform technologies that enables some cloud provider to outperform others in some metrics, while being worse in others. Closely linked to this is the fact that the requirements for different types of applications to be hosted in cloud environments can be quite diverse.

When choosing a cloud platform, even among those that offer the same class of service (namely IaaS, PaaS or SaaS), most of cloud platforms have different target audiences. In other words, these platforms can be seen as packages with different emphasis, meaning that there are cloud providers whose architectures give priority to certain aspects of performance at the sacrifice of others. For example, one platform might emphasize fast access to databases and high I/O performance, whereas the competitive advantage of another platform might be focused on raw processing power. Previous research has tried to answer the following question: What are the differences across clouds for specific functions? In July 2010, Bitcurrent and Webmetrics presented an study that covers a great part of this question [6]. Although they highlight that due to the multi-tenant and the variable nature of clouds their testing was not scientifical or repeatable, the results are meaningful enough to extract some conclusions. Similar to the methodology of the chapter 3 but in five public clouds, they executed four tests to measure the latency of different resource-specific operations: a disk I/O specific task, a CPU-intensive calculation, the retrieval of a 2 MB image, and finally fetching a 1-pixel image. They collected measurements from a range of diverse global locations (see Figure 4.3[3]).

[3]Reproduced here with the permission of the copyright holder

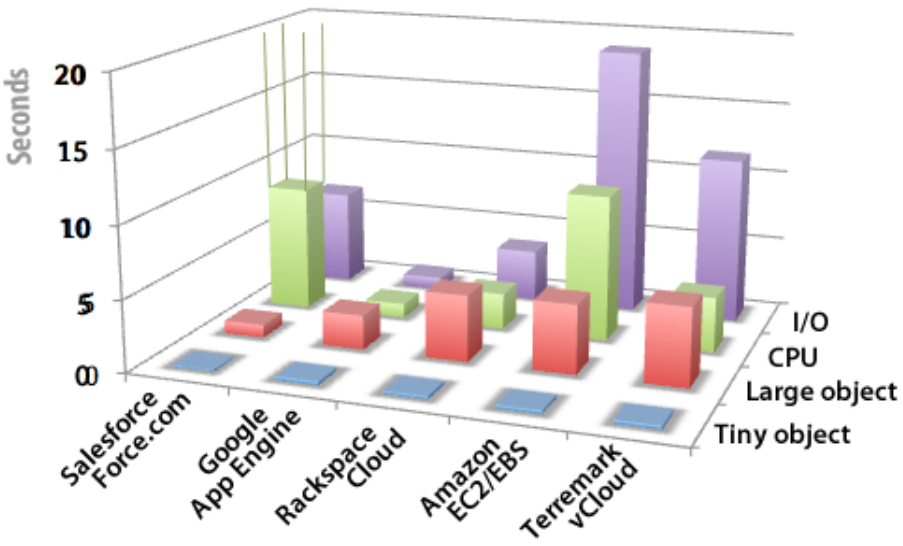

Figure 4.3: Average performance by test type and cloud (Source: [6])

Overall, PaaS providers offer better network I/O that enables them to deliver the large object faster. This is probably because of their ability to distribute workload out to caching tiers better than an individual virtual machine can, according to Bitcurrent and Webmetrics. The Salesforce platform shows poor performance for CPU intensive workloads. This is specially severe as in their tests the cloud was handling a tenth of the load of the other cloud providers due to Salesforce's cloud limitations. Amazon was also poor in terms of CPU performance, but the type of instance being used was the m1.small, the cheapest and least powerful type of instance that Amazon offers. Surely a more powerful instance type would have shown better processing performance and better I/O performance (however, the measured I/O performance was the worst of all of the clouds). Is again remarkable that it is difficult to accurately translate Amazon's performance specifications into real world work capacity, thus leading to a high degree of unpredictability in the planning stage. This is particulary visible for network I/O, where there are three levels of performance depending on the type of instance, literally: low, moderate, or high. Google App Engine outperformed every other competitor in terms of I/O. It is noteworthy though that this exceptional I/O performance is due to Google's particular database system, Bigtable, which features a perfomance scheme that can be summarized as "insert slow, query fast" [15]. This characteristic of the filesystem underlying the Google App Engine can have a direct impact on specific types of workloads, as will be discussed later. Beyond what is apparent in Figure 4.3, there are further limitations that implicitly focus the audience of some cloud providers

(i.e. the kinds of applications that fit a specific cloud). A clear example of this is the upper limit of 30 seconds Google App Engine places on compute time. What this means is that when an application is called to serve a web request it must issue a response within 30 seconds, clearly limiting the use of this platform for compute-intensive applications. In summary there is no single best cloud, but rather there are "best clouds" depending on the specific needs of the customer's application.

There has been a lot of talk about what kinds of applications can be moved to the cloud and about whether there are workloads that are simply not adaptable to a cloud computing environment. Gammage and Dawson [31] have studied the types of workloads that could suffer performance degradation or not be feasible at all if moved to virtualized environments. A large number of the applications that they point out as not suitable for virtualization occur because the application depends on or has specific requirements for perfomance of one or more hardware resources. How current hypervisors deal with hardware virtualization has improved since their report was published and it is doubtless that such improvements will continue thus continuing to lower the difference in performance between a virtualized resource and a physical one. However, the increase in unpredictability in shared-resource virtualized environments (such as public clouds) is an additional problem. Undertaking negotiations of performance and establishing performance SLAs can help to sort this out by increasing the determinism. It is obvious that this problem affects different types of workloads in different proportions, so it is worth analyzing how large these differences are.

Determining the precise classification parameters is not an easy task, as the range of existing classes of applications is very large and grows by the day due to frequent innovations. This particular classification was tackled by focusing on workload characteristics that can be linked to particular requirements or lead to problems when moving an application to a cloud. Some questions to ask in order to sort applications by its constraints or requirements on a cloud platform would be:

- Does it run across multiple compute nodes, or just on one?
- What resource does it stress the most - disk, network, CPU?
- How much data does it work on?
- Is there any task that requires near real-time interaction?

With these questions, the application or workload could be classified into one of the following types: data-intensive, latency-intensive, highly geo-distributed, or mission critical applica-

tions. The main impacts of moving each of these types of application to a cloud platform and specific recommendations or guidelines to look for when moving such an application to the cloud are given in the following sections.

4.2.3.1 Data-intensive

Current cloud offerings usually offer several storage options that are internally connected to the servers through a high speed link at no cost. This enables applications hosted in virtual machines (such as Amazon's EC2 instances) fast and free access to the data. However, uploading the data from local databases to the cloud does have a cost and, furthermore, the data travels to the cloud at varying speeds typical of the Internet. Applications which work on large datasets require this data to be uploaded to the cloud, as doing computations from a cloud but using local storage is not only expensive but requires an unpredictable amount of time to process this data (due to the varying data access times).

According to *The Economist* [26], the amount of data being collected and stored globally is doubling each year , and an increasing number of businesses are using and taking advantage of this data through data mining (i.e. analyzing large datasets to extract patterns and useful information). A data-intensive type of workload is also often found in scientific computation, especially in biomedical and life sciences. Applications such as data mining or biomedical computation usually have requirements for high processing for a limited period of time, making a cloud attractive as it can offer this resource at a reduced price (in comparison to buying and operating compute servers). But the large datasets such applications work on have to be moved to the cloud and, moreover, in many cases, this dataset needs to be refreshed frequently. This leads to a bottleneck for data-intensive applications that has not been solved succesfully to the date. Indeed, as Armbrust et al. suggested in their report [4], Amazon features an "Import/Export" service so that the customer can ship physical disks to Amazon to move data into and out of the Amazon cloud. Until there are connections with high enough and/or cheap enough bandwidth, shipping portable storage devices to and from the cloud datacenters will be more economical than transferring the bits.

As a final note, the importance in choosing cloud provider, and concretely examining its database system, is a major factor for these applications. For example when benchmarking the Google App Engine, Bitcurrent and Webmetrics [6] remark that it took them 37 hours to insert a relatively small amount of data in Google's Bigtable database, much more than in any other provider.

4.2.3.2 Latency-sensitive

Latency-sensitive applications are one of the most difficult workloads to move to a cloud given that, as it has been said throughout this thesis, all the problematic areas in cloud computing performance, from lack of isolation between virtual machines to the user and datacenter location effects, negatively impact over response time and latency of operations. At the same time, this category includes a large number of business opportunities such as:

- Online gaming. A group of users can rent a cloud server to host their game and run their game clients on their laptops.
- Audio or video streaming. With cloud computing it is easy for anyone to start an streaming service by harnessing the scalability the cloud provides, without too much planning upfront and the ability to scale the service to any level of demand.
- Media broadcasting. Similar to the previous use case, newspapers or online TV channels, who already own web servers, can outsource very popular events to a cloud instead of provisioning more capacity in their own infrastructure.
- VoIP telephony. The cloud could host VoIP servers, as the telephony demand curve is usually spiky, rather than uniform. Particularily, it would be advantageous to host the Session Initiation Protocol (SIP) servers in the cloud as they are the part of the VoIP system that handles call requests from users. Thus the cloud could help to deal with the wide variation in the demand the SIP servers usually face.

These are examples of latency-sensitive applications for which moving to the cloud makes sense for one of two reasons: no upfront investing or scalability (in the face of unknown or high temporary levels of demand). The essential problem is that these applications usually require QoS that, as it has been discussed before, current clouds do not offer. The QoS needs of these applications is very similar and focuses on the following service level guarantees, as deduced from the survey and analysis by Miras et al. [50]:

- Bandwidth: Audio streaming and VoIP applications require a modest yet sustained amount of bandwidth (e.g. VoIP can run with just around 64Kbps). For audio other than voice, the required bandwidth is usually higher as users expect at least FM or CD quality, but it is usually not high enough to impose a constraint or decision parameter on the choice of cloud provider. Video streaming can be very bandwidth consuming depending on the encoding format. Even with the most advanced H.264 encoder, a

medium resolution video will consume at least 512Kbps to achieve good visual quality, whereas a high definition 1080p-resolution video can require as much as 6 Mbps.

- Delay: Streaming applications usually have relaxed requirements about absolute values of delay. Delay itself only impacts applications that are based on human interaction such as VoIP or video conferencing. In the particular case of VoIP, delays over a certain amount (around 30ms) cause echo, but this is usually solved using echo cancellation techniques in the terminals. Miras et al. [50] cite ITU-TG.114 among other sources to quantify the maximum acceptable delay for VoIP between 150 and 250 ms. Above 250 ms talker overlap happens (i.e. multiple people speaking at the same time).

- Jitter: Jitter refers to variation in the delay with which packets arrive. In interactive services, be it audio such as VoIP or video, the requirements for jitter are very strict: the maximum jitter to ensure good perceived quality is 75 ms, but recommended values are around 40 ms. Jitter is generally hidden by a de-jitter buffer, but this is hardly possible to implement in interactive services like VoIP or video conference.

- Packet loss: Packet loss below 2% is considered necessary for VoIP. For video over IP, the maximum tolerance depends on the encoder used, with a maximum acceptable value of packet loss around 1.5% to 3% [8], [7].

For a latency-sensitive application to be hosted on the cloud the SLA requirements should be very specific about network performance, particulary with regard to fluctuations in bandwidth and delay. Otherwise, applications that involve a level of interactivity would see their end-user perceived quality being degraded in a much higher proportion than a normal web application. The extent to which these fluctuations can be controlled is limited as in cloud computing there are numerous sources of jitter starting with the server. Therefore, multi-tenant servers that are used to host this kind of workload should have fine-tuned performance isolation mechanisms to minimize the interference. One direct deduction from my experiments is that a latency-sensitive, specially jitter-sensitive, applications can suffer significant unexpected degradations when co-located with disk-hogging or network-hogging applications of other cloud users. Unfortunately, current mechanisms fail to enforce complete isolation, thus current clouds are not prepared to host applications that really need real-time interaction, such as high-frequency trading.

4.2.3.3 Highly geo-distributed

There is a group of applications that can benefit from cloud offerings of providers that own a collection of datacenters plus a large number of edge locations. The term "edge location" refers to smaller sized datacenters that receive data from "parent" datacenters to be used temporarily, in a form of data caching. This data will later be erased. This leads to geographically distributed deployment, enabling more users to experience lower latencies. Applications that can take advantage of this features have the following pattern in common:

- widely geographically distributed users
- shared data: many reads and writes made by different users to the same data. Agarwal et al. [2] consider the example of a user's Facebook wall.
- user mobility: users want to be able to access the information they put in the cloud from anywhere while experiencing similar performance.

Examples of highly geo-distributed applications include all of the social networks and other web 2.0 services, such as photo sharing, video sharing, and such. Widely geographically distributed sensor networks are an additional example. The capacity of the cloud to facilitate distribution of content through these edge locations or content delivery networks (CDN), be it propietary (such as Amazon's Cloud Front) or using a third party (e.g. Akamai is one of the largest CDN providers), fits very well with this type of applications. This fit occurs because it combines the scalability typical of clouds (perfect for workloads with bursty demand, such as that of a video sharing service - Youtube) with the delay-reducing effect of CDNs. However, this approach faces some challenges, the main one being how to optimally decide upond data placement. As Agarwal et al. [2] noted, placing each user's data as closest to the user as possible is not always the best strategy and, in fact, there are significant benefits to placing data closest to those who use it most heavily. So, for this type of application the data placement methodology followed by the cloud provider should be carefully examined.

4.2.3.4 Mission-critical applications

As with the first steps of adoption of any other new technology, customers and enterprises who moved to the cloud started by hosting mostly non-business critical processes. But as cloud computing evolves this will be changing, with customers partly or entirely hosting its

main revenue-generating products in the cloud . Leaving aside other performance requirements that might be addressed in some of the three previous categories, it is reasonable that mission-critical applications require strict definitions of availability. Here there are some distinctions: services which need to access to the actual data at literally any time versus services that might be satisfied with recent enough data. The latter applications will not worry about reaching near 100% availability, but instead tolerate some short periods of downtime that could be scheduled to fit between consecutive refreshes of data. For the time in between, the user access to data is sorted out through offline client caching methods, such as the capability provided by HTML5, as was discussed in section 2.5.10. On the other hand, services that really need the nearly constant uptime guarantees will probably not see their needs satisfied in current clouds. According to Durkee [24], every additional nine of availability required doubles the cost to deliver that service, and beyond four or five nines of availability it is necessary to ensure that only multiple points of failure could bring down the service. More than four nines of availability is considered unaffordable and unnecessary for most applications and therefore chances are that in current all-purpose clouds (i.e. that support every type of application, as all of the providers cited in section 2.4.2 do) it will not be feasible. Customers with these needs will not move to the cloud or will switch to *Clouds 2.0,* as Durkee puts it: next generation clouds designed to meet enterprise requirements. These new clouds will need to be more transparent and have more appropriate service level agreements. Customers willing to store mission-critical data in the cloud should look for specific level agreements on two parameters:

availability: the probability that an object stored in the service will be reachable.

durability: the probability that an object stored in the service will remain intact and accessible after a determined period of time. For example, calculating based on one year, 100% would mean there is no possible way for the object to be lost and 90% would mean that there is a 1-in-10 chance of an object being lost within the year.

Both parameters are based on introducing redundancy into infrastructure, the degree of redundancy will have to be also transparently specified.

Chapter 5

Conclusions and future work

5.1 Conclusions

The aim of this thesis was to explore whether cloud computing has any implicit limitations that could restrict its usefulness in some fields. As previous well-known research, such as [4], had pointed out there exist in fact technical, economic, and legal factors that put boundaries on the adoption and growth of cloud computing.

This thesis focused on examining the performance of the cloud, as this factor has been reported to be one of customer's top three concerns. The methodology was based on finding the factors which the performance of cloud-hosted applications depend on. Firstly, the impact on performance of a virtualized and shared physical server was tested throughout the implementation of a private cloud. This experiment, in contrast to similar benchmarks found in the literature about the topic, was executed in a controlled environment rather than in a public cloud, where setup options are limited and background load is generally difficult to know and uncontrollable, and therefore allowed drawing some clear conclusions. The analysis of these measurements yielded the following conclusions:

- There is always a performance decrease due to co-located virtual machines running resource-intensive tasks.
- The drop in perfomance is slight for CPU and memory intensive workload and very significant for disk and network I/O intensive workloads.
- Variability of the measurements increases dramatically in every case when there is high background load.

- Network peformance isolation is non-existent in the KVM virtualizer nor implemented in the Eucalyptus platform. In the experiments this was resolved by using traffic shaping, but there might be better ways to do this.
- New disk scheduling mechanisms for virtualized environments should be developed because traditional schedulers, which are currently in use, were not designed to suit virtualized operation.
- Alternatively, fairness in access to disk could be implemented by using network-attached storage and using the traffic shaping mechanism discussed before to share the access to disk.

The amount of networking inside the datacenter and, obviously, from the datacenter to the end-user's location are also factors in the performance. Partial conclusions are:

- Previous research demonstrated that the network performance inside current datacenters is not optimal, highly variable, and the theorical peak throughputs are hardly ever reached. This can greatly hinder the performance of application deployments based on large numbers of instances that need to intercommunicate frequently, such as in many parallel computing frameworks.
- Internet latencies make clearly determining the geographic distribution of users and knowing the application's latency requirements essential to ensuring a good user experience. The combination of both types of information can be used as a guideline to choose the number of hosting locations.

When offering applications as a service, software licenses are replaced by service level agreements (SLAs). The current state of SLAs in cloud computing is inadequate and there is a need for fine-grained specifications about more performance parameters than simply availability. A framework including throughput, response time, and availability would be a good starting point. This thesis proposed a SLA method based on client perceived performance as estimated from the server-side or provided by a third-party monitor.

Further insights gained were the fact that different cloud providers, although their marketing usually describes most of them as all-purpose, actually perform better in some senses than others. Therefore, choosing a cloud provider should be only done when the workload of the application to be hosted is fully characterized. This supports the hypothesis that the application type is one of the main parameters when choosing a cloud provider

or even when deciding whether to move to a cloud or not. Here, a classification of cloud applications with specific requirements when moved to a cloud was provided, alongside with type-specific recommendations for SLA negotation.

The overall conclusion of this thesis is that cloud computing is in general prepared to succesfully host most typical web applications and with great cost savings, but those applications with strict latency requirements or other network performance requirements, those that require working with large datasets, or those whose needs for availability are critical need to be studied carefully. In these cases, consideration of specific performance requirements, different for each type, and compensation models for violation of the SLA are crucial. Even with the evolution of SLAs, all-purpose clouds might not be willing to offer guarantees for the most demanding applications, thus there will be an opportunity for clouds targeted at and designed for particular workloads.

5.2 Future work

The experiments in chapter 3 highlighted the need to improve performance isolation mechanisms in virtualized environments and chapter 4 made some efforts in the direction of describing complete SLAs for cloud services, and in both areas there are a large number of possibilities for future research. Additionally, studying the performance of cloud computing offerings is not an straightforward task and more of it should be done, especially concerning how different application types behave on different cloud providers. A follow-up work, using the findings of this thesis would be to benchmark the performance of different applications across every cloud provider periodically over a longer time span, building a database of historical values. This could be the basis of a service similar to Cloudsleuth [19], but with enhanced capabilities: selecting an application type from among a few options and selecting a cloud provider, would provide an approximation of the performance of an application of that type in that particular cloud provider. The output would be very useful for application developers wondering where to host their application.

Bibliography

[1] John Adams. Chirp 2010: Scaling Twitter. In *http://www.slideshare.net/netik/billions-of-hits-scaling-twitter*, 2010.

[2] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan. Volley: automated data placement for geo-distributed cloud services. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, page 2. USENIX Association, 2010.

[3] W. Almesberger et al. Linux network traffic control: implementation overview. In *Sixth IEEE Symposium on*, pages 296–301. Citeseer, 2001.

[4] M. Armbrust et al. Above the clouds: A Berkeley view of cloud computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 2009.

[5] Microsoft Windows Azure. *http://www.microsoft.com/windowsazure/windowsazure/*.

[6] Bitcurrent and Webmetrics. Cloud computing performance. 2010.

[7] F. Boulos, B. Parrein, P. Le Callet, and D. Hands. Perceptual effects of packet loss on H. 264/AVC encoded videos. 2009.

[8] J.M. Boyce and R.D. Gaglianello. Packet loss effects on MPEG video sent over the public internet. In *Proceedings of the sixth ACM international conference on Multimedia*, pages 181–190. ACM, 1998.

[9] G. Brunette and R. Mogull. Security guidance for critical areas of focus in cloud computing v2. 1. *CSA (Cloud Security Alliance), USA. http://www. cloudsecurityalliance. org/guidance/csaguide. v2*, 1, 2009.

[10] E. Brynjolfsson, P. Hofmann, and J. Jordan. Cloud computing and electricity: beyond the utility model. *Communications of the ACM*, 53(5):32–34, 2010.

[11] R. Buyya, C.S. Yeo, and S. Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, pages 5–13. IEEE, 2008.

[12] RN Calheiros, R. Ranjan, A. Beloglazov, C. Rose, and R. Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience, Wiley Press, New York, USA*, 2010.

[13] Devin Carraway. Lookbusy, 2005.

[14] D. Catteddu and G. Hogben. Cloud computing: benefits, risks and recommendations for information security. Technical report, European Network and Information Security Agency, 2009.

[15] F. Chang et al. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008.

[16] Rackspace Cloud. *http://www.rackspacecloud.com/*.

[17] Oracle Cloud Computing. *http://www.oracle.com/us/technologies/cloud/index.htm*.

[18] CIO Magazine: Cloud computing survey. *http://www.cio.com/documents/whitepapers/CIOCloudComp pdf*, 2009.

[19] Compuware. Cloudsleuth. *https://www.cloudsleuth.net/web/guest/home*, 2010.

[20] R.J. Creasy. The origin of the VM/370 time-sharing system. *IBM Journal of Research and Development*, 25(5):483–490, 1981.

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

[22] M. Devera. Hierarchical token bucket. *http://luxik.cdi.cz/ devik/qos/htb/*.

[23] Nandita Dukkipati et al. An argument for increasing TCP's initial congestion window. *SIGCOMM Comput. Commun. Rev.*, 40(3):26–33, 2010.

[24] D. Durkee. Why cloud computing will never be free. *Queue*, 8(4):20–29, 2010.

[25] Amazon EC2. *http://aws.amazon.com/ec2/*, pages Last accessed at September 17, 2010.

[26] The Economist. Special report: The data deluge. 2010.

[27] Google App Engine. *http://code.google.com/appengine/*, pages Last accessed at September 25, 2010.

[28] EyeOS. Cloud computing operating system. *http://eyeos.org/*.

[29] Inc. Forrester Research. Cloud privacy heat map. *http://www.forrester.com/cloudprivacyheatmap*, 2010.

[30] I. Foster and C. Kesselman. *The grid: blueprint for a new computing infrastructure*. Morgan Kaufmann, 1999.

[31] Brian Gammage and Philip Dawson. Server workloads: What not to virtualize. *Gartner Research*, page 7, 2008.

[32] S.L. Garfinkel. An evaluation of amazon's grid computing services: Ec2, s3 and sqs. 2007.

[33] C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[34] GoGrid. Cloud hosting. *http://www.gogrid.com/cloud-hosting/*.

[35] Google. The Chromium projects: Chromium OS. *http://www.chromium.org/chromium-os*.

[36] Google. Data liberation front. *http://www.dataliberation.org/*.

[37] Google. SPDY: An experimental protocol for a faster web. *http://dev.chromium.org/spdy*, 2009.

[38] The Guardian. Cloud computing is a trap, warns GNU founder Richard Stallman. *http://www.guardian.co.uk/technology/2008/sep/29/cloud.computing.richard.stallman*, 2008.

[39] Z. Hill, J. Li, M. Mao, A. Ruiz-Alvarez, and M. Humphrey. Early observations on the performance of Windows Azure. *Science Cloud Workshop*, 2010.

[40] Ubuntu Enterprise Cloud Images. Release candidate server 10.04. 2010.

[41] S. Kamara and K. Lauter. Cryptographic cloud storage. *Financial Cryptography and Data Security*, pages 136–149, 2010.

[42] A. Kamra, V. Misra, and E.M. Nahum. Yaksha: A self-tuning controller for managing the performance of 3-tiered web sites. In *Twelfth IEEE International Workshop on Quality of Service, 2004. IWQOS 2004*, pages 47–56, 2004.

[43] J. Katcher. Postmark: A new file system benchmark, 1997.

[44] A. Kivity et al. KVM: the linux virtual machine monitor. In *Proceedings of the Linux Symposium*, volume 1, pages 225–230, 2007.

[45] B. Krishnamurthy, P. Gill, and M. Arlitt. A few chirps about twitter. In *Proceedings of the first workshop on Online social networks*, pages 19–24. ACM, 2008.

[46] Dave Mangot. EC2 Variability: The numbers revealed: Measuring EC2 system performance. *http://tech.mangot.com/roller/dave/entry/ec2*, Web page, Part of the series urandom Mangot ideas, 2009.

[47] J.D. McCalpin. Memory bandwidth and machine balance in current high performance computers. *IEEE Technical Committee on Computer Architecture Newsletter*, 12:19–25, 1995.

[48] P. Mell and T. Grance. The NIST definition of cloud computing. *National Institute of Standards and Technology*, 2009.

[49] D.A. Menascé and P. Ngo. Understanding cloud computing: Experimentation and capacity planning. In *Computer Measurement Group Conference*, 2009.

[50] D. Miras et al. A survey on network QoS needs of advanced internet applications. *Working Document of Internet 2 QoS Working Group*, 2002.

[51] W.D. Norcott and D. Capps. Iozone filesystem benchmark. *www.iozone.org*, 55.

[52] D.P. Olshefski, J. Nieh, and D. Agrawal. Inferring client response time at the web server. *ACM SIGMETRICS Performance Evaluation Review*, 30(1):171, 2002.

[53] Takuya Ooura. Pi CSS5. *http://myownlittleworld.com/miscellaneous/computers/piprogram.html*, 2006.

[54] Eucalyptus Cloud Platform. *http://open.eucalyptus.com/*.

[55] DC Plummer, TJ Bittman, T. Austin, D. Clearley, and DM Smith. Cloud computing: Defining and describing and emerging phenomenon, Gartner, Inc. *Retrieved September*, 25:2008, 2008.

[56] S. Ramachandran and A. Jain. Web page stats: size and number of resources. *http://code.google.com/speed/articles/web-metrics.html*, 2010.

[57] Kelton Research. Survey: Cloud Computing "No Hype", but fear of security and control slowing adoption. *http://tv.sys-con.com/node/852659*.

[58] Salesforce. *http://www.salesforce.com/*, pages Last accessed at September 19, 2010.

[59] N. Santos, K.P. Gummadi, and R. Rodrigues. Towards trusted cloud computing. In *Proceedings of the 2009 conference on Hot topics in cloud computing*, page 3. USENIX Association, 2009.

[60] IDC IT Cloud Services. *http://blogs.idc.com/ie/?p=730*, 2009.

[61] Amazon EC2 SLA. *http://aws.amazon.com/ec2-sla/*, pages Last accessed at September 20, 2010.

[62] Google App Engine SLA. *http://code.google.com/appengine/business/sla.html*, pages Last accessed at September 20, 2010.

[63] Rackspace Cloud SLA. *http://www.rackspacecloud.com/legal/sla*, pages Last accessed at September 20, 2010.

[64] J. Staten. Is cloud computing ready for the enterprise? *Forrester Research, March*, 7, 2008.

[65] Los Angeles Times. What's powering Web apps: Google waving goodbye to Gears, hello to HTML5. *http://latimesblogs.latimes.com/technology/2009/11/google-gears.html*, 2009.

[66] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs. Iperf: The TCP/UDP bandwidth measurement tool. *http://dast.nlanr.net/Projects/Iperf*, 2004.

[67] Twitter. *www.twitter.com*, pages Last accessed at September 16, 2010.

[68] Verizon. IP latency statistics. *http://www.verizonbusiness.com/about/network/latency/*, pages Last accessed at September 15, 2010.

[69] G. Wang and TSE Ng. The impact of virtualization on network performance of amazon EC2 data center. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[70] B. Wickremasinghe, R.N. Calheiros, and R. Buyya. CloudAnalyst: A CloudSim-based visual modeller for analysing cloud computing environments and applications. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 446–452. IEEE, 2010.

[71] D. Wischik, M. Handley, and M.B. Braun. The resource pooling principle. *ACM SIGCOMM Computer Communication Review*, 38(5):47–52, 2008.

[72] Dan Zarella. State of the Twittersphere, hubspot.com whitepaper. *http://blog.hubspot.com/Portals/249/sotwitter09.pdf*, 2009.

Acronyms and Abbreviations

API Application Programming Interface

AWS Amazon Web Services

CC Cluster Controller

CDN Content Delivery Network

CFS Completely Fair Scheduler

CLC Cloud Controller

CRM Customer Relationship Management

DHCP Dynamic Host Configuration Protocol

EC2 Amazon Elastic Compute Cloud

EMI Eucalyptus Machine Image

HTB Hierarchical Token Bucket

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

IaaS Infrastructure as a Service

I/O Input/Output

KSM Kernel Samepage Merging

KVM Kernel-based Virtual Machine

NC Node Controller

PaaS Platform as a Service

QDISC Queuing Discipline

QoS Quality of Service

REST Representational State Transfer

RSA Rivest, Shamir and Adleman

RTT Round-Trip Time

SaaS Software as a Service

SC Storage Controller

SIP Session Initiation Protocol

SLA Service Level Agreement

SLO Service Level Objective

SOAP Simple Object Access Protocol

SSD Solid-State Drive

SSH Secure Shell

TCP Transport Control Protocol

UDP User Datagram Protocol

VLAN Virtual Local Area Network

VM Virtual Machine

VMM Virtual Machine Monitor

VoIP Voice over Internet Protocol

WS3 Walrus Storage Controller

XML Extensible Markup Language

TRITA-ICT-EX-2010:277

www.kth.se