UNIVERSITÀ
CATTOLICA
del Sacro Cuore

CLOUD COMPUTING: CASE STUDY

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Carles Pérez Gassió

In partial fulfilment

of the requirements for the degree in

TELECOMMUNICATIONS SYSTEMS ENGINEERING

Coadvisor: Alfonso Rojas Espinosa

Coadvisor: Daniele Tessera

Brescia & Barcelona, July 2016

Abstract

The interest of applications in adopting cloud computing has grown steadily. Requirements for users to plan for provisioning turned more flexible, and allows to increase them when there is a rise in service demand. To understand the different services that the "cloud" can offer to companies and users, we wrote the State-of-the-Art.

Studies over already deployed cloud-based infrastructures consume too many resources. Simulation tools have become key to the evolution of cloud computing, and one of the most valued for this task it is *CloudSim*. The practical part of the Bachelor degree thesis is based on a web service model from this toolkit, and we have modelled its workload to process with different probability distributions. The default model provided by the program presented some limitations when the workload has large differences, so we have proposed a new model.

Resum

L'interès d'aplicacions per adoptar el cloud computing no ha parat de créixer. Es flexibilitzen els requisits dels usuaris a l'hora de planificar l'aprovisionament de recursos, i permet incrementar-los quan hi ha un augment de la demanda del servei. Per tal d'entendre el funcionament dels diferents serveis que pot oferir el "núvol" tant a empreses com a usuaris, hem escrit l'State-of-the-Art.

Fer estudis sobre estructures desplegades al "núvol" consumeix masses recursos. Les eines de simulació s'han tornat clau per a la evolució del cloud computing i una de les més valorades per aquesta tasca és el *CloudSim*. La part pràctica del treball final de grau es basa en un model de servei web d'aquesta eina, i hem modelat la seva càrrega de treball a processar amb diferents distribucions de probabilitat. El model que ofereix el programa per defecte presenta certes limitacions quan la carrega de treball varia massa, i hem proposat un nou model.

Resumen

El interés de aplicaciones para adoptar el cloud computing no ha parado de crecer. Se flexibilizan los requisitos de los usuarios a la hora de planificar el suministro de recursos, y permite incrementarlos cuando hay un aumento de la demanda del servicio. Para entender el funcionamiento de los diferentes servicios que puede ofrecer la "nube" tanto a empresas como a usuarios, hemos escrito el State-of-the-Art.

Hacer estudios sobre estructuras desplegadas en la "nube" consume demasiados recursos. Las herramientas de simulación se han vuelto clave para la evolución del cloud computing y una de las más valoradas para esta tarea es el *CloudSim*. La parte práctica del trabajo final de grado se basa en un modelo de servicio web de esta herramienta, y hemos modelado su carga de trabajo a procesar con diferentes distribuciones de probabilidad. El modelo que ofrece el programa por defecto presenta ciertas limitaciones cuando la carga de trabajo varía demasiado, y hemos propuesto un nuevo modelo.

Acknowledgements

I would like to thank Alfonso Rojas for his predisposition to support me during my mobility period and his corrections.

I would like to express my gratitude to Daniele Tessera for guiding and supporting me during all this experience. His advices and comments helped me a lot in the development and writing of the thesis.

Thanks to Diego Tiziani for the opportunity to access the *Cloud Academy*'s free trial.

Finally, I wish to thank Jordi Fornes his recommendations for illustrate *Java* relations between classes and methods.

Document Control Log

General Information

Title	Cloud computing: Case study	
Approval By		
Est. Approval Date		
Sign- Off Date	20-Jul-2016	
Distribution List	Alfonso Rojas Espinosa	alfonso@entel.upc.edu
	Daniele Tessera	daniele.tessera@unicatt.it
File Name	TFG_Degree Thesis_v04	

Version Information

Date	Version	Author	Comments
Mar 2016	01	Carles Pérez	Document Creation
Apr 2016	02	Carles Pérez	Added Business Model section
May 2016	02.1	Carles Pérez	Corrections after May 12th meeting
May 2016	03	Carles Pérez	First CloudSimEx's results: Ram values
Jun 2016	04	Carles Pérez	cpuGen simulations
Jul 2016	05	Carles Pérez	Confidence intervals

Table of Contents

Abstract 2
Resum 3
Resumen 4
Acknowledgements 5
Document Control Log 6
Table of Contents 7
List of Figures 8
List of Tables 8
I. Introduction 9
II. State-of-the-Art 13
2.1. Introduction to cloud computing 13
2.2. Deployment models 14
2.3. Cloud computing models (service models) 16
2.4. History 18
2.5. Benefits of cloud computing 19
2.6. Vendors 20
III. Business Model 21
IV. Argument 23
4.1. CloudSim 23
4.2. Project development 25
4.3. Simulations 29
4.4. Results 31
V. Budget 37
VI. Conclusions and future development 38
References 38
Appendices 41
Glossary 43

List of Figures

Figure 1. Cloud computing models.....16

Figure 2. Users and providers of cloud computing.....21

Figure 3. Layered *CloudSim* architecture.....24

Figure 4. Simulation environment.....26

Figure 5. Effect of time-sharing provisioning policy.....27

Figure 6. Gaussian model: Increase Average ActualCPUTime.....32

Figure 7. Detail figure 6 with CV between 0.001 and 0.5.....33

Figure 8. Uniform model: Average ActualCPUTime.....34

Figure 9. Log-normal model: Increase Average ActualCPUTime.....35

Figure 10. Cloudlet processing update process.....41

List of Tables

Table I. Number of Cloudlets generated function of RAM.....29

Table II. Number of Cloudlets generated (Gaussian case).....32

Table III. Number of Cloudlets generated (Log-normal case).....36

Table IV. Total costs of the project.....37

I. Introduction

The project was carried out at the department of Mathematics and Physics at the Università Cattolica del Sacro Cuore at Brescia (Italy). The purpose of this project abroad was to realize a Bachelor thesis (final project) as a student from the host university. This project consists on the discussion of a short written paper, which is prepared by the student under the guidance of a supervisor (tutor). According the study program called *Laurea Triennale*, the student must do a research on a topic out of his academic expedient. The main goal is collect and process the new material, and write a State-of-the-Art.

Actually the field of study is chosen by the student, but in this case, as the host university is focused on the Mathematics degree, the topic proposed came from the Data Processing Systems' teacher. This was the study field more appropriate for a student coming from a degree in a Telecommunications Systems as was discussed with academic coordinator Silvia Pianta. Daniele Tessera is the responsible of that course and currently his research interests are involved with Cloud Computing.

There was no option for foreign students to discuss their thesis in the host university, in order to adapt from an Italian study program to the UPC's study program, we added an argument section. Professor Tessera proposed a case study, out of Italian Bachelor thesis scope, focused on the evaluation of the variability of the workload to be processed by cloud services. To continue with this idea we worked with *CloudSim*, the simulation toolkit developed by *Cloud Computing and Distributed Systems Laboratory (CLOUDS Lab)* from the department of Computer Science and Software Engineering of the University of Melbourne.

The first setting up issues described on *Project Critical Review_v01.docx* led to download the *CloudSimEx* package. After some requests for new features, on 2014 Nikolay Grozev started *CloudSimEx*. It is a set of *CloudSim* extensions making simulation development easier and enabling the modelling of new types of applications, not supported by *CloudSim*. Using *CloudSimEx* allowed us to recover delayed time skipping some *CloudSim*'s set-up requirements.

After the first analysis of the different parts of the project, a breakdown of tasks was defined. With this document you will find attached another file called "*TFG_Planning.xlsx*". The first spreadsheet is a baseline of tasks with an extra row called "Documents" where is marked the

project deliverables. Columns C, D, and E are used as a summary. Also cells in column D turns red if worked time exceeds the estimation, but turns green once the task is finished. The lector has the option to contract or expand the tasks and subtasks.

Work packages breakdown diagram:

WP1. Documentation

WP2. State-of-the-Art

WP3. Software

WP4. Results assessment

WP5. Oral defence

All the WP's updates after the CDR, are written in blue:

Project: Documentation	WP ref: WP1	
Major constituent: Documentation	Sheet 1 of 5	
Short description: Different documents that describe the project and its development.	Start date: 3-3-2016 End date: 20-7-2016	
Internal task T1: Project planning Internal task T2: Work Plan redaction Internal task T3: Critical Design Review redaction Internal task T4: Final report redaction	Deliverables: Work Plan CDR Final Report	Dates: 24-Mar-2016 9-May-2016 20-Jul-2016

Project: State-of-the-Art	WP ref: WP2
Major constituent: Documentation	Sheet 2 of 5
Short description: Writing an own State-of-the-Art according the bibliography.	Start date: 7-3-2016 End date: 15-5-2016
Internal task T1: Searching for papers and publications about cloud computing Internal task T2: Write an extended index maximum detailed Internal task T3: Write a State-of-the-Art	

Project: Software	WP ref: WP3
Major constituent: Software analysis and programming	Sheet 3 of 5
Short description: Getting used to the *CloudSim* and analyse the source code of professor Buyya's examples.	Start date: 5-4-2016 End date: 26-5-2016
Internal task T1: Understanding about computer architecture multi-tier Internal task T2: Acquiring all the software needed to perform the project Internal task T3: Learning how to use *CloudSim* in a *Linux* environment Internal task T4: Developing the code needed	

Project: Results assessment	WP ref: WP4
Major constituent: Assessment	Sheet 4 of 5
Short description: Analyse the results from *CloudSim* work.	Start date: 28-6-2016 End date: 15-7-2016
Internal task T1: Analyse the results from simulations Internal task T2: Writing conclusions	

Project: Oral Defence	WP ref: WP5
Major constituent: Documentation	Sheet 5 of 5
Short description: Preparation of the thesis' oral defence.	Start date: 20-7-2016 End date: 1-9-2016
Internal task T1: Preparing the support slides Internal task T2: Presentation rehearsal Internal task T3: Oral defence	

Milestones:

WP#	Task#	Short title	Milestone / deliverable	Date
1	2	Work Plan approval	TFG Work Plan	**24-Mar-2016**
2	2	Extended index	State-of-the-Art	**18-Apr-2016**
1	3	CDR approval	TFG Critical Design Review	**9-May-2016**
2	3	State-of-the-Art	State-of-the-Art	**15-May-2016**
1	4	Final report approval	TFG Final Report	**20-Jul-2016**
6	3	Oral defence	TFG Oral Defence	**1-Sep-2016**

Find attached the Excel file called "*TFG_Planning.xlsx*". The Grantt diagram is on the spreadsheet "Test cycle 1 Execution" and all the plan details are calculated on "Details" spreadsheet.

II. State-of-the-Art

2.1. Introduction to cloud computing

Cloud Computing is more known every day by the general public and not only for big companies. There are a lot of possible implementations of this technology that is available for everyone nowadays. It goes from a supported email service to a file hosting service, or can be used for an entire server network.

National Institute of Standards and Technologies (*NIST*) definition of cloud computing: **Cloud computing is a model for enabling convenient, on-demand network access to 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.** [1]

This description was established on September 2011 to confirm a standard definition after several publications started a survey or state-of-the-art about clouds attempting to define it on various ways [2][3]. Also *NIST* describes the essentials characteristics of cloud computing that are going to be discussed on following sections.

In order to summarize, cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services [4]. This provider rents you out computational resources, storage, bandwidth and the power to run the servers according to your own needs. Some huge benefits are arisen such as decrease in total costs or virtually unlimited scalability of the infrastructure.

Cloud computing is based on virtualized resources. **Virtualization** had been studied upon decades, and is a technique to allow an operating system and the software to believe they are running on standard hardware while in reality they are running on top of another kind of software known as the hypervisor, which simulates a real computer. After a hypervisor is created on a

server, then a number of virtual machines can be created inside of it [5]. Each virtual machine acts as a different server and is an insulated environment without possibilities of being contaminated by any of the rest virtual machines. In fact, one of the benefits of virtualization is the possibility to provision new virtual machines. As virtual machines are independent of the underline hardware, you can migrate to one host to another without any modification.

Other advantages are to roll back a previous configuration in case it is needed, avoid cross-contamination between applications or the granular control of the infrastructure that allows to shut down broken software without affecting the rest [6]. As cloud computing is based on virtualization, all this technology allows to create, provision and de-provision computer power and storage without changing the hardware setup. Only creating more virtual machines or improving the existing ones. This can be done from a user interface offered by the provider or can be programmed depending on your needs.

One of the major reason to choose using clouds is for reducing costs. The model used to achieve such thing is to pay only for the resources you contract. A pay-as-you-go model, that with virtualization technology, can adapt the resources provision to your needs [3]. Some authors as M. Armbrust et Al. (2010), proposed to use for addressing this peculiarity of cloud computing the term elastic capability [4].

2.2. Deployment models

The natural evolution of companies is to create a cloud-like environment with its own data center. This infrastructure operated by a single organization is called private cloud and maintains all the cloud's advantages and a full control over it [1]. But what providers like *Amazon*, *Google* or *Microsoft* can facility from their data center to a client is known as public cloud. Also exists a third different model, a mix between the other two: a hybrid cloud.

Sometimes public cloud is considered as the only true cloud. It is the most common deployment model and the scenario that majority thinks first when they refer to clouds. A **public cloud** infrastructure provides services over a network that is open for public use [1]. Could be a free service or according a pay-per-use model.

This public deployment model is actually what allows a low maintenance and avoids a massive investment in hardware to build the data center [7]. Works as an operating expense (OpEx) where the ongoing cost is just for running the service.

Nowadays, exists a problem for some users that are worried to put their costumer's data in public clouds. They have a feeling of losing security. The regulations are still in progress, and for example, in European Union, personal data belonging to European citizens, have to be stored physically in a European country [8].

On the other side, **private clouds** are operated for a single organization. The difference with public clouds is not in the technology, just in the infrastructure. A different size that is explained because a public cloud consists on thousands hardware servers and a private may be ten.

Following the last example of the public cloud's disadvantage, one reason to use private clouds is: security. All the traffic among the services deployed in the cloud can be routed on a trusted network. Another advantage is the 100% control over the entire infrastructure, something impossible in a public cloud, even on the most general service that will be discussed later on the services description.

Detriment of private clouds appears with a not easy set up. It requires skills and a significant level and degree of engagement to virtualize the business environment and reaching a high security level. The reason for not being considered for everyone as a true cloud environment is because the lack of scalability. The network infrastructure must be able to scale to a large number of servers and allow for incremental expansion, and with the privates' infrastructures is limited.

Private clouds can get more of the benefits of public clouds through hybrid cloud [4]. A **hybrid cloud** is a composition of two or more clouds (usually a private and a public one) that remain unique entities, but are bound together offering the benefits of multiple deployment models [1].

Some use cases that fit best with this deployment model can be for example: an organization that may store sensitive client data in-house on a private cloud application, but interconnects that application to another app provided on a public cloud as a software service. This example of hybrid cloud extends the capabilities of the enterprise to deliver a specific business service for the addition of externally available public cloud services [5].

Cloud bursting is another interesting case for hybrid clouds. Employ public cloud computing resources to deal with temporary capacity demand that cannot be met by private cloud [3]. This is an application deployment model in which an application runs in a private cloud or a data center and bursts to a public cloud when the demand for computing capacity increases. With this approach, the organization only pays for extra compute resources when it needed [5].

2.3. Cloud computing models (service models)

Above the deployment models can be found the different delivery/service models that are utilized within a particular deployment model. Cloud applications can be run according to three different service models: SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service).

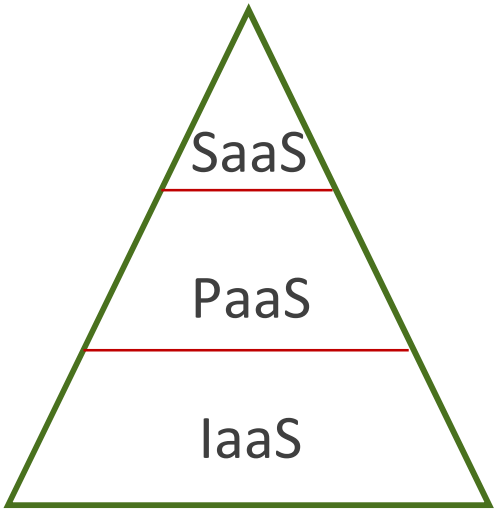

Figure 1. Cloud computing models.

The highest level cloud computing model is SaaS. **SaaS** is a software licensing and delivery model in which software is licensed on a subscription basis and it is centrally hosted on the cloud [1][4]. As a customer you get a complete software package ready to use, on-demand at a monthly or yearly fee. However, you can't customize it more than the provider allows. One of the most common examples is *Gmail* and this kind of applications are based on a Multi-Tenant architecture. With this model a single version of the application with a single configuration is used for all the clients. Also the application is typically scaled horizontally to support scalability [5].

SaaS do not need any user installation and usually allows cross-platform compatibility between different applications accessed from the same browser, for example, using your *Gmail* account with *Google Docs*. Also the total costs of the initial setup and ownership are lower than the traditional ones. Furthermore, the providers have an easy option for debugging the application or new actualizations, and have robust ways of keeping their intellectual property.

Some related issues were described by S. Subashini on Journal of Network and Computer Applications (2011). If a company only depend on one provider and its system fails, this can affect all their users. Meanwhile, that company may not have access to their confidential data or where are stored and secured [9].

The second level is **PaaS**, useful for developers who need to build their own application, to deploy it and take care of it while the provider will manage all the underlying infrastructural aspects as networks, servers, storage, etc [9]. It provides a computing platform and a solution stack as a service with tools and libraries for the costumer [1]. Examples of this solution are based on Software Development Kits (SDKs) and Application Programming Interfaces (APIs) that easy to develop environment and apps.

Andrea Colangelo, a Debian developer who collaborates with the *Cloud Academy*, talked about PaaS solutions. They benefit from automatic scaling and load balancing which are done by the provider controlling the underlying infrastructure, also developers can obtain integration with other services provided through their PaaS provider and can usually take advantage of authentication services, email services and user interfaces which are provisioned by the PaaS platform itself. Development times are reduced and using the PaaS frameworks and APIs slightly reduces the risk of bugs.

The biggest downside for this kind of solution is the vendor lock-in. Once you have built and developed your application on one provider's PaaS, it is usually not cheap to relocate to another platform from another provider. This is often considered a major deterrent for developers interested in deploying on the cloud. Among the most common PaaS solutions are *Google App Engine* and *Amazon Beanstalk*. Other very common PaaS platforms are *Cloud Foundry*, *AppScale* and *Oracle*.

IaaS is one layer below PaaS, the most basic service model and the highest level of control on the infrastructure resources. IaaS offers virtual machines where low-level services are provisioned as storage, perimeter firewall, load balancing or IP Addresses. Seems logic that companies who need to manage all their virtual infrastructure paying only for the resources they use (OpEx cost model) will choose an IaaS service [9].

Sometimes is not easy to define a service strictly as only one of the previously described, and could have a characteristic of another service. But always must follow the "as-a-service approach". To summarize is when certain resource can be provisioned and delivered on-demand in an automated way, with APIs to access it programmatically and with certain number of infrastructural and management task hidden away from the costumer by the provider [5].

2.4. History

Back in 1961, the first sparkle of a main idea behind cloud computing was envisioned by a computer pioneer named John McCarthy. He opined that "computation may someday be organized as a public utility" [10]. Also is recognized that in 1969, Leonard Kleinrock one of the chief scientists of the original *Advanced Research Projects Agency Network (ARPANET)* project which seeded the Internet, said: "As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of 'computer utilities' which, like present electric and telephone utilities, will service individual homes and offices across the country" [3].

The modern idea of cloud first appeared around 2003 when Chris Pinkham and Benjamin Black presented a paper describing a vision for *Amazon* infrastructure that was completely standardized, automated, relied extensively on web services for things like storage [11]. Near the end of their paper, they mentioned the possibility of selling virtual servers as a service proposing that the company could generate revenues from the new infrastructure investment [12][13].

Representative dates for cloud computing's history

1961: John McCarthy. "Computation may someday be organized as public utility".

2003: Chris Pinkham and Benjamin Black write about *Amazon*'s vision.

2004: *Amazon Web Service*.

2006: Eric Schmidt uses the word "cloud" to define the business model of providing services across the Internet.

2007: 180,000 developers use *AWS*.

2008: open source AWS API compatible platforms & *Google App Engine* was released as a preview.

2010: *Microsoft Azure*.

2017: end-user will be spending on cloud services around $250 billion.

2.5. Benefits of cloud computing

At the end of section '2.1' we introduced the term elasticity. It is cloud computing's ability to add or remove resources at a fine grain and with a lead time of hours rather than weeks allows matching resources to workload much more closely [4]. The importance resides on a dynamic resource provisioning, to easily expand the service to large scales in order to handle rapid increase in service demands and manage consumption according to own needs [7].

An advantage of public clouds respect private data centers is lower administration burden, because a huge part of it is managed by the provider. Also public cloud's physical locations are more used to manage power, cooling and networks to keep save data from different organizations [4]. Provider's reputation must ensure their costumers' data and show to them that even in a failure case, their data could be quickly recovered.

The economic appeal of cloud computing is often described as "converting capital expenses to operating expenses" (CapEx to OpEx) [4]. Capital expenditure (CapEx) are used by a company to acquire or upgrade physical assets such as property, industrial buildings or equipment. CapEx

costs are then amortized or depreciated over the life of the asset in question [14], but computing assets live is too short and their value depreciates quickly.

Treating IT as an OpEx helps in dramatically reducing the upfront costs in corporate computing. Some Internet start-ups were realized with investments in information technology that are orders of magnitude lesser than that required just a few years ago. The cloud becomes an adaptive infrastructure that can be shared by different end users, each of whom might use it in very different ways. The users are completely separated from each other, and the flexibility of the infrastructure allows for computing loads to be balanced on the fly as more users join the system (the process of setting up the infrastructure has become so standardized that adding computing capacity has become almost as simple as adding building blocks to an existing grid). The beauty of the arrangement is that as the number of users goes up, the demand load on the system gets more balanced in a stochastic sense, even as its economies of scale expand [8].

2.6. Vendors

In this section, we provide a survey of some of the dominant cloud computing commercial products. Following the “History” section, *Amazon* was the first offering cloud services with ***Amazon Web Services*** (*AWS*). *AWS* is a secure cloud services platform, offering compute power, database storage, content delivery and other functionality to help businesses scale and grow [15]. The most central and best-known of these services include *Amazon Elastic Compute Cloud*, known as *EC2*, and *Amazon Simple Storage Service*, also known as *S3*.

After *AWS* come out ***Google Cloud Platform***, that lets you build and host applications and websites, store data, and analyse data on *Google*'s scalable infrastructure. Its two most important products are *Google App Engine* as PaaS for sandboxed web applications, and *Google Compute Engine* as IaaS that enables users to launch virtual machines on demand [16].

And 6 years ago, *Microsoft* started in this business with ***Microsoft Azure***, a platform that provides a *Windows*-based environment for running applications and storing data on servers in data centers. *SQL Azure* provides data services in the cloud based on *SQL Server*, and *.NET Services* offer distributed infrastructure services to cloud-based and local applications [7].

III. Business Model

The business model of cloud computing follows a relationship provider-user:

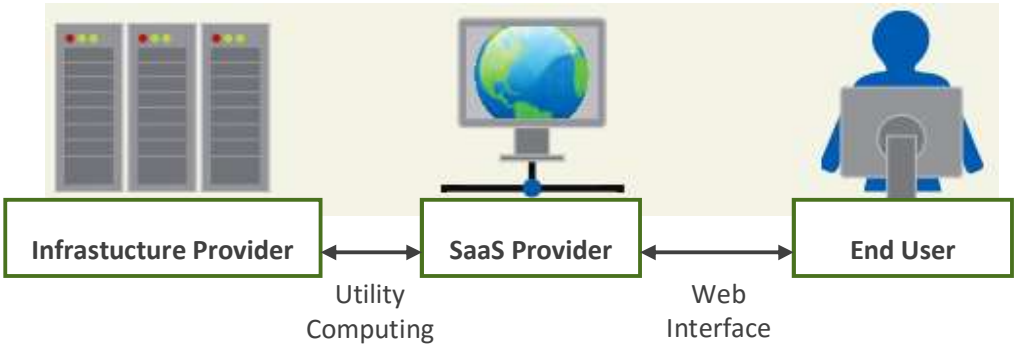

Figure 2. Users and providers of cloud computing.

According to the layered architecture of cloud computing, it is entirely possible that a PaaS provider runs its cloud on top of an IaaS provider's cloud. However, in the current practice, IaaS and PaaS providers are often parts of the same organization. This is why PaaS and IaaS providers are often called the infrastructure providers or cloud providers [7].

A data analysis known as SWOT analysis can be used to conclude research results, therefore define the business objectives: Company's SW (strengths and weaknesses) and market's OT (opportunities and threats).

Following the terminology, the first aspect to cover are strengths. In fact, in previous sections they have been discussed, but to resume the most important fact is to scale up services at a very short notice. Is possible to request more computing resources on the fly, and to use time-distributed computing resources without incurring the costs of costing a traditional infrastructure for the rest of the year. Cloud computing leads to reduced infrastructure costs and energy savings as well reduced upgrades and maintenance costs. Management of technology is simple by using a cloud computing service, makes a securer environment having better control of the resources on the network. Cloud computing services allow an organization to control when, where, and how

employees have access to the organization's computer systems, all managed over a simple web-based interface.

The main weakness is the reason that creates more lack of confidence to the companies: the loss of physical control of the data that is put on the cloud. Also some providers cannot guarantee the high quality of service and availability that some organization could demand.

One of the significant opportunities of cloud computing resides on the companies' objective to achieve a "green" credential. In order to reduce their carbon footprint, IT departments believe on energy efficiency and equipment recycling. Moving to the cloud will allow organizations to not only reduce their IT infrastructure, but, since it is much cheaper to transport computing services than energy, it will also represent a better use of energy.

Some questions are still waiting an answer and represent some threats. For example, if a cloud provider goes bankrupt, what happens with its client's data? Also nowadays, there is a lack of standards and *ISO* is still working on an international regulation.

In order to conclude the business section, we are going to present with some statistics researched by *IBM* (January 2014) to confirm why more business are adding the cloud to their technological departments [17].

Businesses are using cloud technology to support their analytics efforts: 54% use analytics extensively to derive insights from big data and 59% use cloud to share data seamlessly across applications.

Cloud allows work to be accessed from anywhere on multiple devices, making cross-functional collaboration much easier: 59% of organizations that use it, improve integration between development and operations.

Business functions companies that have been migrated to the cloud: 18% messaging, 15% storage.

The cloud allows for rapid development of new products and services: 52% use it to innovate products and services rapidly. 24% are able to offer additional products and services.

Results: 25% of business saw a reduction in IT costs. 55% saw an increase in efficiency. 49% saw an improvement in employee mobility.

IV. Argument

Previously in this document have been presented a few challenges with the cloud computing. In the laboratory, we tried to analyse one of the most interesting things that concerns some companies: resource prevision. How much resources do you need? A possible answer to this question could mean a huge decrement on the businesses' costs.

Simulation can be a viable way to size the capacity requirement. *CloudSim* simulates real time entities of cloud computing environment and helps to understand the behaviour of some application configuration instances without deploying it on a real cloud platform [18][19]. With *CloudSimEx* we recreate a web session model with workload and the experiment must show if the stablished provision could give the correct QoS to the user. In our *CloudSimEx* model, the workload describes the work to be processed by the infrastructure, and it is named as Cloudlet.

4.1. CloudSim

Before our experiment explanation, is needed a description of this toolkit *CloudSim*. Figure 3 shows the layered implementation of the *CloudSim* software framework. It manages the instantiation and execution of core entities (VMs, hosts, data centers, application) during the simulation period. This layer is capable of concurrently instantiating and managing a large scale cloud infrastructure consisting of thousands of system components. The fundamental issues such as provisioning of hosts to VMs based on user requests, managing application execution, and dynamic monitoring are handled by this layer [20].

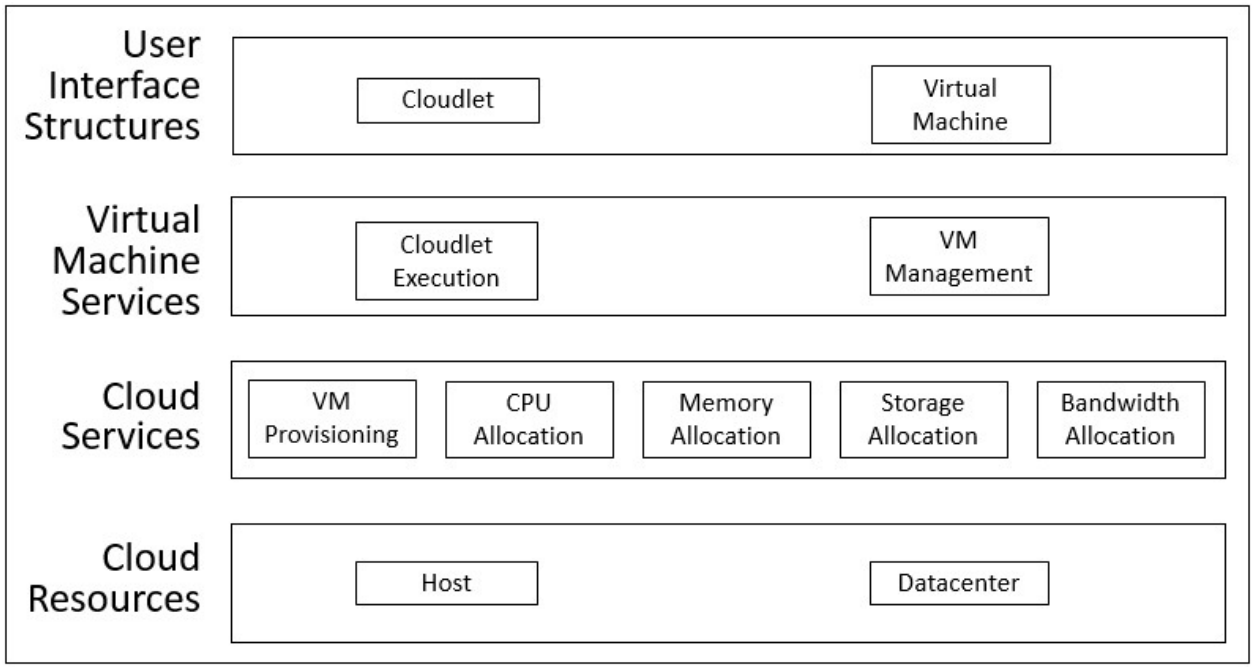

Figure 3. Layered *CloudSim* architecture.

On top this layer we have the highest layer in the *CloudSim* stack. It is the User Code that exposes basic entities for hosts (number of machines and their specification), applications (number of tasks and their requirements), VMs, number of users and their application types, and broker scheduling policies [21].

The infrastructure-level services related to the clouds are modelled in the simulator by a Datacenter component for handling service requests. These requests are application elements sandboxed within VMs, which need to be allocated a share of processing power on datacenter's host components. The data center entity manages a number of host entities. The hosts are assigned to one or more VMs based on a VM allocation policy that should be defined by the cloud service provider. An entity is an instance of a component, and a *CloudSim* component can be a class that represent one *CloudSim* model (data center, host) [21].

A data center is composed by a set of hosts, which are responsible for managing VMs during their life cycles. Host is a component that represents a physical computing server in a cloud: it is assigned a pre-configured processing capability (expressed in million of instructions per second

– MIPS), memory, storage and a provisioning policy for allocating processing cores to virtual machines [20].

VM allocation is the process of creating VM instances on hosts that match the critical characteristics (storage, memory), configurations (software environment), and requirements (availability zone) of the service provider.

For each Host component, the allocation of processing cores to VMs is done based on a host allocation policy. This policy takes into account several hardware characteristics, such as number of CPU cores, CPU share, and amount of memory, that are allocated to a given VM instance. Hence, *CloudSim* supports simulation scenarios that assign specific CPU cores to specific VMs (a space-shared policy), dynamically distributed capacity of a core among VMs (time-shared policy), or assign cores to VMs on demand [21].

4.2. Project development

Project requirements:

- Simulation of a couple environments multi tier.
- Generate different amount of requests, and with the workload study the execution time simulation.

Project specifications:

- *CloudSim* and *CloudSimEx* works better in a *Linux* environment.
- *JDK* 7 and *Maven* 3.2 or later.
- *Eclipse IDE Java EE Developers* and *SVN* and *GIT* clients.

On next page we can see our simulation environment and some of the variables. The first step with *CloudSim* (after initialize it with `CloudSim.init()`) is to create the datacenter. It will be the resource provider, it is created with `createDatacenter()` and will be defined with the number of process elements and all the host's properties (`int ram, long storage, int bw`).

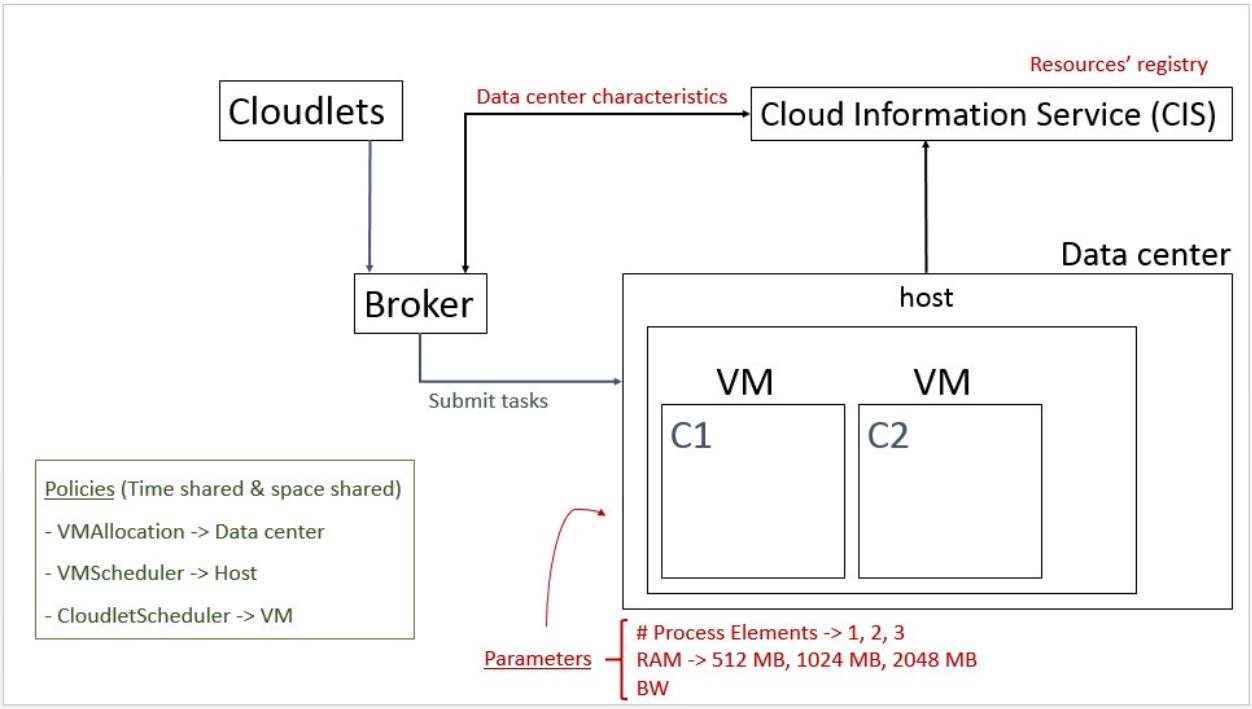

Figure 4. Simulation environment.

As a real cloud environment, *CloudSim* works on virtualization, and this created host will be virtualized on a number of virtual machines. Those virtual machines will again have the same parameters and you just need to register your datacenter to the Cloud Information Service (CIS) as to the *CloudSim* framework.

Once the datacenter is ready[1], we need the broker (who submits tasks to the datacenter) and to create virtual machines. If we create more than one VM, we need to submit the list of VMs to the broker.

The broker is an entity that initially obtains the resources' information from CIS, so the broker gets the details of the datacenter's characteristics.

Simulation is started with `CloudSim.startSimulation()` and the console will show the results after the method `broker.getCloudletReceivedList()` is executed. The different Cloudlets are submitted to the broker and once the broker have the datacenter's details, interacts

[1] The datacenter internal processing has not been studied in this thesis, and its performance description is added on the appendices section.

directly with the datacenter and assigns the Cloudlets to our created virtual machines. Exist some policies to determinate if these processes are going to be executed in parallel (Time shared) or one after the other (Space shared).

Our allocation policy on task unit execution is time-shared. In Figure 5 we show a VM provisioning scenario described by R. Buyya, R. Ranjan and RN. Calheiros: a host with two CPU cores receives request for hosting two VMs, such that each one requires two cores and plans to host four task's units. More specifically, tasks t1, t2, t3 and t4 to be hosted in VM1, whereas t5, t6, t7 and t8 to be hosted in VM2.

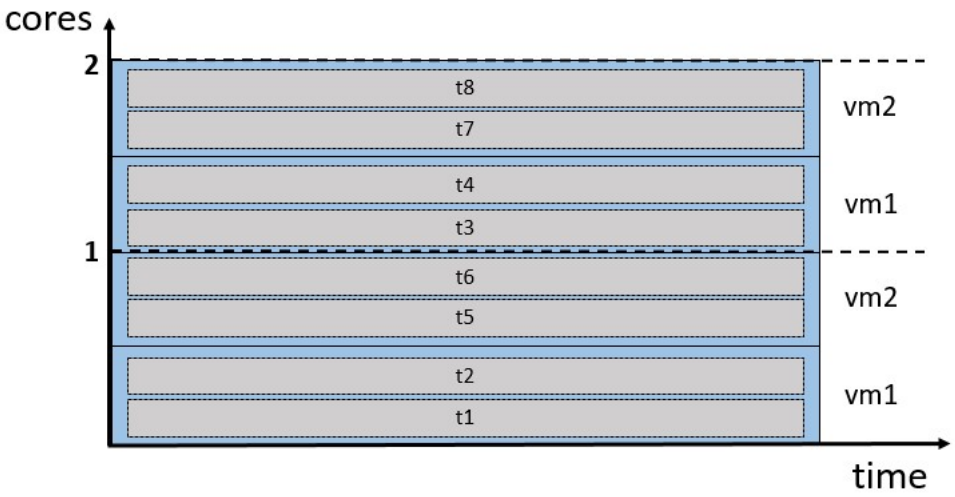

Figure 5. Effect of time-sharing provisioning policy.

CloudSim is structured as *Java* custom packages, where each package will contain the correlated classes in it. The *CloudSim*'s packages imported in our model:

1. `org.cloudbus.cloudsim`: contains classes that once instantiated will perform like some component in the system, or support a specific component of the system for producing its relevant behaviour during the simulation process. This package was categorized into two sections by Anupinder Singh.
   - Simulating components classes: These are the set of classes that imitates a particular part of cloud setup. Classes that come under this category are Cloudlet, Datacenter, DatacenterCharacteristics, Host, Pe, Storage, Vm.
   - Policy classes: These are the set of classes that imitate the policy behavior of a cloud component. Classes that comes under this category are VMAllocationPolicy and VMSchedulerTimeShared.

2. `org.cloudbus.cloudsim.core`: This package contains the main classes of this project and are directly responsible for initiating, starting, maintaining and end the simulation process.

3. `org.cloudbus.cloudsim.provisioners`: Classes in this package contains policies related to allocation of bandwidth, processing elements and RAM.

4. `org.cloudbus.cloudsim.ex.util`: Classes in this package provide the basic utility operations related to perform some math operations related to cloud computing services or some calculation of execution time during the simulation process.

After mentioned the fundamental classes of *CloudSim*, which are also the building blocks of the simulator, we provide a finer details related to them as Rodrigo N. Calheiros et Al did it in 2010 (alphabetical order):

- `BwProvisioner`: Abstract class that models the policy for provisioning of bandwidth to VMs. The main role of this component is to undertake the allocation of network bandwidths to a set of competing VMs that are deployed across the data center. The `BwProvisioningSimple` allows a VM to reserve as much bandwidth as required, but this is constrained by the total available host's bandwidth.
- `Cloudlet`: This class models the cloud-based application services. *CloudSim* orchestrates the complexity of an application in terms of computational requirements. Every application service has a pre-assigned instruction length and data transfer overhead that it needs to undertake during its life cycle.
- `CloudletScheduler`: Abstract class extended by the implementation of different policies that determine the share of processing power among Cloudlets in a VM. In our case is used the class `CloudletSchedulerTimeShared`.
- `Datacenter`: Class that models the core infrastructure-level services (hardware) that are offered by cloud providers. It encapsulates a set of compute hosts.
- `DatacenterBroker`: This class models a broker, which is responsible for mediating between users and service providers depending on users' QoS requirements and deploys service tasks across clouds. The broker acting on behalf of users identifies suitable cloud service

providers through the Cloud Information Service and negotiates with them for an allocation of resources that meet users' QoS needs.

- `DatacenterCharacteristics`: Contains configuration information of data center resources.
- `Host`: This class models a physical resource such as a compute or storage server. It encapsulates important information such as the amount of memory and storage, a list and type of processing cores, an allocation of policy for sharing the processing power among VMs, and policies for provisioning memory and bandwidth to the VMs.
- `RamProvisionerSimple`: Abstract class that represents the provisioning policy for allocating primary memory (RAM) to VMs. It does not enforce any limitation on the amount of memory that a VM may request. However, if the request is beyond the available memory capacity, then it is rejected.
- `Vm`: This class models a virtual machine, which is managed and hosted by a cloud host component. Every VM component has access to a component that stores the following characteristics related to a VM: accessible memory, processor, storage size and VM's internal provisioning policy that is extended from the abstract class `CloudletScheduler`.
- `VmScheduler`: Abstract class implemented by the host that models the policies required for allocating processor cores to VMs. In our case is used the class `VmSchedulerTimeShared`.

4.3. Simulations

We changed the virtual machines' RAM value between 512 MB and 2048 MB and keeping fixed the host's RAM value. Also we made the same simulation fixing the VM's RAM value and changing with the same values the host's RAM.

Table I. Number of Cloudlets generated function of RAM.

VM RAM (MB)	Host RAM (MB)	# Cloudlets
512	2048	**3930**
1024	2048	**3934**
2048	2048	**1**
512	1024	**3932**
512	512	**1**

The first obvious thing is that if is used the same RAM value for the host and the virtual machines, the host only could manage one VM. But comparing with a real case, it is the provider's task to preserve the host. So it has no sense to change the host's parameters, because in a real case you do not rent physical machines. Henceforth, we are going to keep the same value for host's RAM, and 32 GB (32768 MB) is a fair amount for a real server.

The amount of hardware resources available to each virtual machine is constrained by the total processing power and system bandwidth available within the host. This critical factor must be considered during the VM provisioning process, to avoid creation of a VM that demands more resources that are available in the host. The cases where just one cloudlet is generated is because the utilization of the class RamProvisionerSimple that rejects the requests.

Our first evidence to keep on the next simulations (in addition to the RAM's fixed value) is that if the number of cloudlets generated is singularly low, the results will be corrupted and the returned values will be worthless.

Once the initial contact with the simulator was done, we decide to focus our experiment on the usage of **statistical distributions to model the amount of computation associated to a web page**.

CloudSimEx's web session modelling has settled by default a Gaussian function (mean = 1000 and deviation = 20) as the distribution function that models our study parameter named cpuGen:

```
GaussianGenerator cpuGen = new GaussianGenerator(1000, 20, rng);
```

This constructor creates a generator of normally-distributed values from a distribution with the specified mean and standard deviation. The third parameter 'rng' is the source of randomness based on the Mersenne Twister algorithm.

Because of the central limit theorem when the number of random variable observations is enough, it eventually becomes a Normal distribution. Widely used to model unknown situations, it is a well-known distribution and with easy estimation parameters. This is the reason that makes so easy to work with it.

With this first approximation, we based our study maintaining fixed the mean to 1000, and changing the standard deviation between 1 to 2000. 'Results' section describes what is generated with the simulation. Especially, we focus on the impact of the standard deviation over the average value of 'ActualCPUTime' generated by the amount of CPU requirements of each cloudlet.

4.4. Results

This example of Web session modelling returns the following results for each Cloudlet:

- CloudletId is the assigned number to identify a bench of instructions to be executed, the workload unit (Cloudlet). It starts with CloudletId = 1.
- Ram is the amount of MB used by the VM to process the cloudlet.
- VmId specifies which virtual machine is used. In our simulated case could be VM1 or VM2.
- Delay: amount of the waiting for the server. More virtual machines, less delay. [seconds]
- IdealStartTime comes from the last FinishTime of the same VM that is going to be used. The first value for the first Cloudlet of each VM is 1 second.
- ExecStartTime = Delay + IdealStartTime [seconds]
- CloudletLength represents the amount of instruction to be executed when processing the cloudlet (it is expressed in millions of CloudletIOLength).
- CloudletIOLength is the amount of data to be written / read.
- **ActualCPUTime**: time that the VM needs to process the Cloudlet, without delays. [seconds]
- FinishTime = ActualCPUTime + ExecStartTime [seconds]
- CloudletStatusString notifies with 'Success' about the end of simulation.
- Finished is stablished with a Boolean parameter and if its value is 'true' the VM is available for the next execution.

FIRST MODEL (Gaussian):

This first case study, focused on the evaluation of the variability of the workload to be processed by cloud services, shows the results of the described case on the Simulations section. For each different value of the standard deviation introduced, thousands of Cloudlets could be generated. Next graphic on figure 6 represents the average of all these Cloudlets' values of ActualCPUTime.

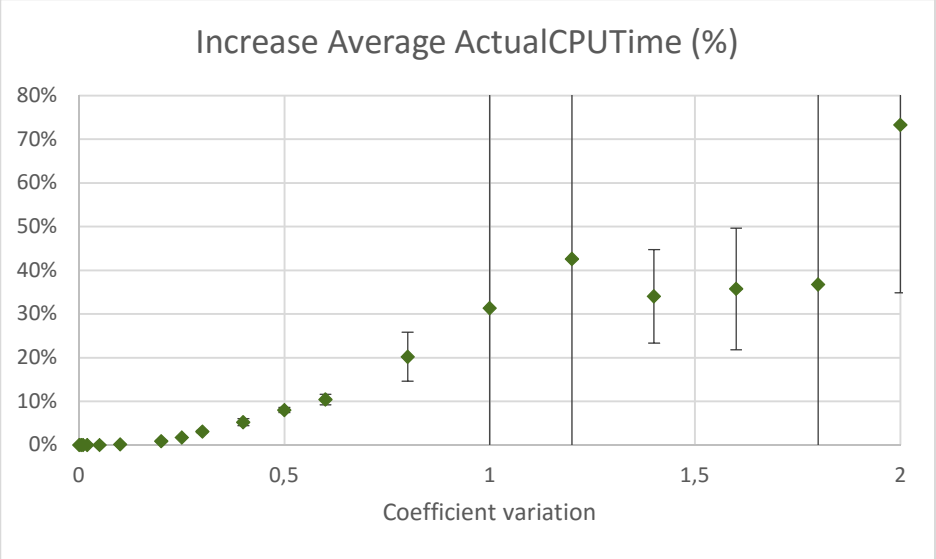

Figure 6. Gaussian model: Increase Average ActualCPUTime.

With more standard deviation (represented by $coefficient\ variation = \frac{standard\ deviation}{mean}$), the average time is higher, but what is not obvious, is how much higher will be.

Table II. Number of Cloudlets generated (Gaussian case).

CV	# Cloudlets
0.5	**3313**
0.6	**1051**
0.8	**127**
1	**5**
1.2	**7**
1.4	**97**
1.6	**71**
1.8	**11**
2	**17**

Due the possible negative values of a Gaussian distribution, *CloudSim* discarded too much instructions workload becoming meaningless the number of Cloudlets that the simulator resulted.

Our first case allows for a limited variations of web computational needs because of negatives values that are related to large value of standard deviation. So we establish a **Gaussian distribution as an acceptable model for a maximum coefficient variation of 0.5**.

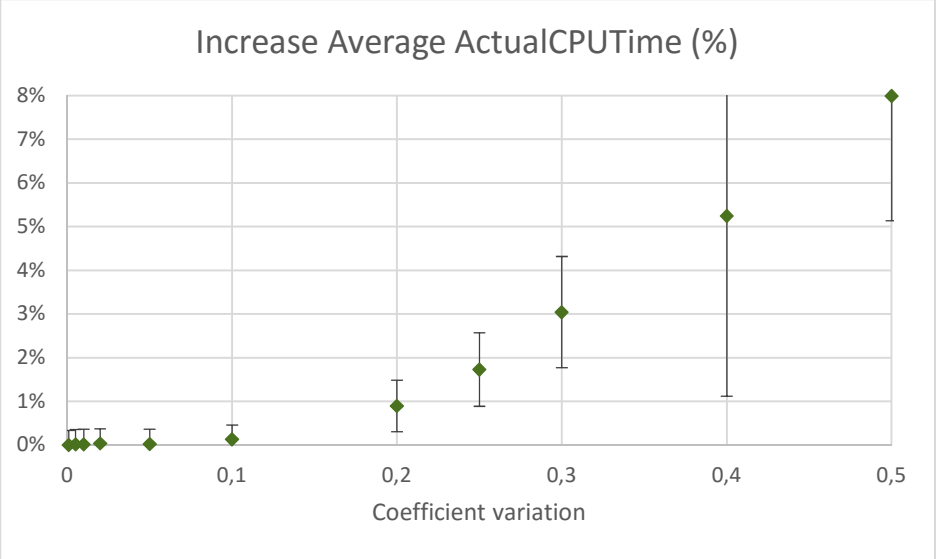

Figure 7. Detail figure 8 with CV between 0.001 and 0.5.

The initial reference for the percentage increase is established by the standard deviation = 1 (Coefficient variation = 0.001) with an average ActualCPUTime of 5.0091 seconds. At the maximum limited value is 5.4095 seconds, and this is a 7.99% more.

If you need to model web pages with large differences in their computational needs, you have to resort to different statistical distributions. Thinking on a distribution with absolute positive values to avoid the lack of Cloudlets, we proposed three options:

- Exponential distribution
- Continuous uniform distribution
- Log-normal distribution

With the exponential distribution we cannot control the coefficient variation, so to be consistent with the whole project we are not going to use it. Even though we prepared this option in the code delivered, but the user only could modify the rate.

We repeat the same experiment with the rest of these new distributions, the code's changes have been added at the beginning of the following subsections.

SECOND MODEL (Uniform):

```
//Variables for continuous uniform distribution
double mean = 1000;
double sd = 1; //standard deviation
//Operations
double a = mean-(1/2)*Math.sqrt(12)*sd;
double b = 2*mean + a;
ContinuousUniformGenerator cpuGen = new ContinuousUniformGenerator(a, b, rng);
```

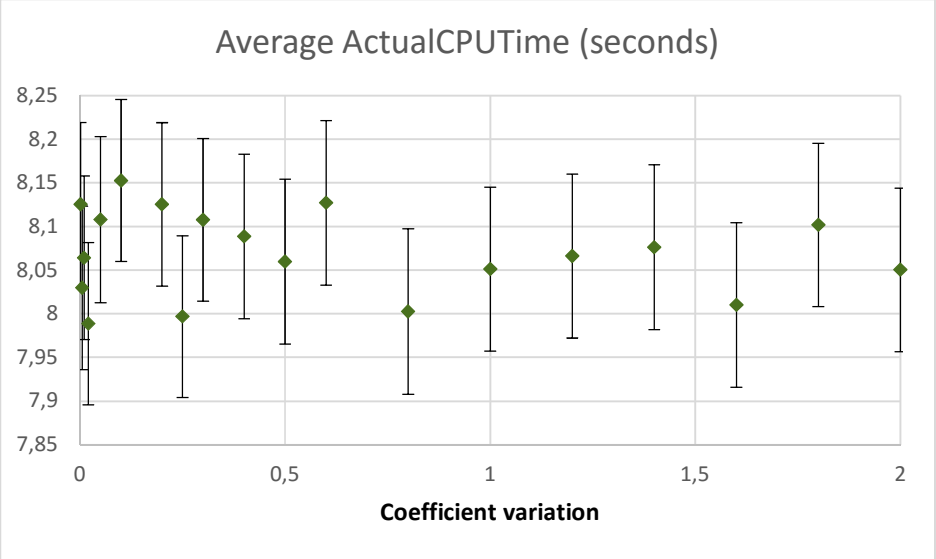

Figure 8. Uniform model: Average ActualCPUTime.

This is result of the distribution maintaining the same probability. All the cases of the average ActualCPUTime are between 7.9887 and 8.1525 seconds, and the maximum variation between the bounds is 2.02%.

Although all the standard deviation's value variation provided enough Cloudlets (always more than 2121), is **difficult to take this model as a reliable case**. Currently studies are working on consumption patterns that experience a periodic demand variation [4].

THIRD MODEL (Log-normal):

We created a new class called MyLognormalGenerator [2] implementing the interface NumberGenerator<>, what is mandatory to take advantage of the official code.

```
//Variables for Log-normal distribution:
double mean = 1000;
double sd = 1; //standard deviation
//Operations
double var = sd*sd; //variance
double location = Math.log((mean*mean)/Math.sqrt(var+mean*mean));
double scale = Math.sqrt(Math.log(var/(mean*mean)+1));
MyLognormalGenerator cpuGen = new MyLognormalGenerator(location, scale, rng);
```

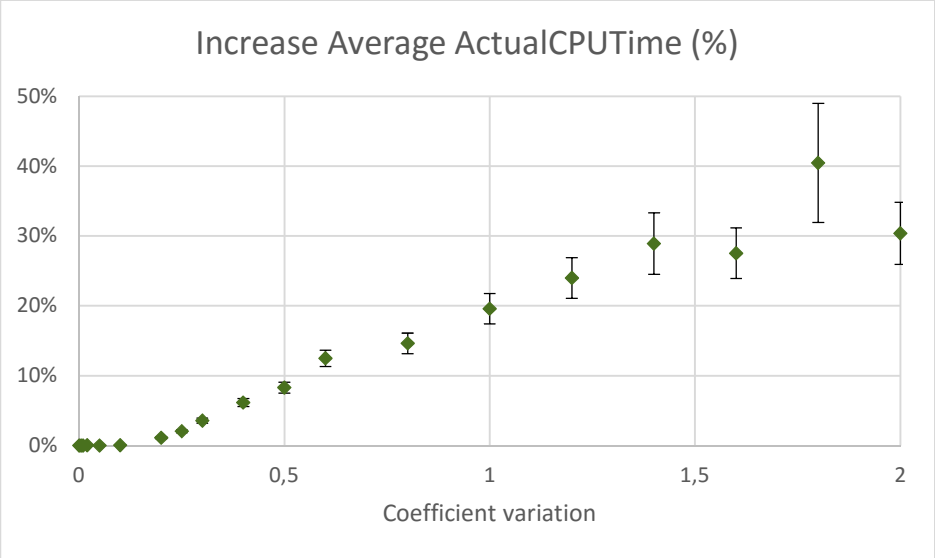

Figure 9. Log-normal model: Increase Average ActualCPUTime.

Comparing with the Gaussian case, below a coefficient variation of 0.5, both cases got average times between 5 and 5.5 seconds, and an increase under 10%. But over that range, with a Log-normal distribution, the average times are lower.

[2] Code added at Appendices section.

Table III. Number of Cloudlets generated (Log-normal case).

CV	# Cloudlets
0.6	**3209**
0.8	**3117**
1	**2895**
1.2	**2728**
1.4	**2560**
1.6	**2603**
1.8	**2255**
2	**2508**

With this amount of Cloudlets endorsing the average times, we propose a **new condition**. **If** the **coefficient variation** is **higher than 0.5**, in order to not compromise the simulations with large differences in the computational needs, **MyLognomalGenerator class will be used instead the GaussianGenerator class**.

All the charts include the confidence interval for every result stated at a 95% confidence level. This interval is related with the number of samples (Cloudlets), and with more samples, the result will tend more to the expected value and the error will be smaller. With fewer samples, will be more error and the standard deviation becomes larger and thus the interval will be larger.

4.3. Budget

This project started with the lecture of papers about cloud computing published in scientific magazines. These articles have been obtained without cost because the final result is for academically purposes.

The access to the *Cloud Academy*'s courses with the "*Student plan*" costs $9 / month. As most courses are out of the thesis scope, with one payment it could be enough to finish all the initial courses. In our case this expense is not considered because our agreement with the "Customer Success Representative", which allows us a non-limited free trial in exchange our feedback.

Since first day, the number of hours dedicated to the different tasks of every thesis' block have been weekly annotated on the document *TFG_Planning.xlsx*. Main tasks:

Table IV. Total costs of the project.

	Summary		
	Estimation (h)	Worked (h)	Costs
TOTAL	849	683	**5008 €**
MG	**312**	**270**	2160 €
LANGUAGE	**90**	**57**	0 €
PREVIOUS RESEARCH	**102**	**90**	720 €
CLOUD ACADEMY	**87**	**14**	112 €
DELIVERIES	**120**	**120**	960 €
SIMULATIONS	**138**	**132**	1056 €

*Also with this document, is added the Excel file where the "Details" spreadsheet defined on *TFG_Work Plan_v02.docx* is available with all the tasks of every block.

V. Conclusions and future development

Results derived from our simulation study has guided us to a new option due an original *CloudSim* limitation described on 'Results' section. The model of web application uses a Gaussian distribution to define the variability of web requests. The limitation appeared when we tried to increase that variability extending the difference of the web computational needs with higher values on the distribution's coefficient variation. With CV values higher than 0.5, Gaussian distribution is not valid because the amount of negative values generated are not compatible with a real case. *CloudSim* dismisses it, and the lack of Cloudlets processed (30% less for each CV's different values) are not enough to ratify the model.

Therefore, we propose to adapt the model with a new condition that changes to a Lognormal distribution when the coefficient variation is higher than 0.5. Our case study presented a stable number of Cloudlets and similar results of both distributions below CV = 0.5.

The extra cases studied have been chosen in order to obtain all positive values. Future work related with this model of web session could be followed by modelling the variability of web requests with other distributions.

Besides, there are more parameters to create different workload scenarios and the variability of the performance. The RAM consumption from users' activity can be also modelled with a probability distribution.

Another interesting case study could be a simulation changing the bandwidth and figure what is the impact overall execution time.

With more diversity of cases after these initial ones, results could answer to which tool sides better the properties of VM. In order to stablish the accurate provision to give the correct QoS to the users. Also correct size for capacity planning and for sizing the most appropriate number of virtual machines.

References

[1] Mell P, Grance T. The NIST Definition of Cloud Computing. NIST. 2011.

[2] Vaquero L, Rodero-Merino L, Caceres J, Lindner M. A break in the clouds: towards a cloud definition. ACM SIGCOMM computer communications review. 2009.

[3] Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems. 2008; 25 (6): 599-616.

[4] Ambrust M et al. A View of Cloud Computing. Communications of the ACM. 2010; 53 (4): 50-59.

[5] Cloud Academy. San Francisco: Cloud Academy; 2016. http://www.cloudacademy.com.

[6] Vouk MA. Cloud Computing – Issues, Research and Implementations. Journal of Computing and Information Technology. 2008; 16 (4): 235-246.

[7] Zhang Q, Cheng L, Boutaba R. Cloud computing: state-of-the-art and research challenges. J Internet Serv Appl. 2010; 1: 7-18.

[8] Marston S, Li Z, Bandyopadhyay S, Zhang J, Ghalsasi A. Cloud computing – The business perspective. Decision Support Systems. 2010; 51: 176-189.

[9] Subashini S, Kavitha V. A survey on security issues in service delivery models of cloud computing. Journal of Network and Computing Applications. 2010; 34: 1-11.

[10] Foster I, Zhao Y, Raicu I, Lu S. Cloud Computing and Grid Computing 360-Degree Compared. Grid Computing Environments Workshop. 2008.

[11] Benjamin Black causes trouble here. Benjamin Black; 2009. EC2 Origins. http://blog.b3k.us/2009/01/25/ec2-origins.html.

[12] Wikipedia, “Amazon Web Services”. https://en.wikipedia.org/wiki/Amazon_Web_Services. 2016.

[13] Dazeinfo. India: Bose P; 2015. End-User Spending On Cloud Services To Reach Almost $250 Billion By 2017. http://dazeinfo.com/2015/03/30/cloud-service-end-user-spend-250-billion-2017-csb-popular.

[14] Wikipedia, “Capital expenditure”. https://en.wikipedia.org/wiki/Capital_expenditure. 2016.

[15] Amazon Web Services (AWS) – Cloud Computing Services. Seattle: Amazon; 2016. What is AWS?. https://aws.amazon.com/what-is-aws.

[16] Wikipedia, “Google Cloud Platform”. https://en.wikipedia.org/wiki/Google_Cloud_Platform. 2016.

[17] Forbes. USA: McCue TJ; 2014. Cloud Computing: United States Business Will Spend $13 Billion On It. http://www.forbes.com/sites/tjmccue/2014/01/29/cloud-computing-united-states-businesses-will-spend-13-billion-on-it/#33a8fa2b341b.

[18] Nikolay Grozev – Staying on top of it. Nikolay Grozev; 2014. CloudSim and CloudSimEx [Part 1]. http://www.nikgrozev.org/2014/06/08/cloudsim-and-cloudsimex-part-1.

[19] Google Code Archive - CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services. The University of Melbourne; 2009. FAQ. https://code.google.com/archive/p/cloudsim/wikis/FAQ.wiki.

[20] Buyya R, Ranjan R, Calheiros RN. Modelling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities. Proceedings of the 7th High Performance Computing and Simulation Conference. 2009.

[21] Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R. CloudSim: a tool for modelling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software – Practice and Experience. 2011; 41: 23-50.

Appendices

1. DATA CENTER INTERNAL PROCESSING (described in [21])

Processing of task units is handled by the respective VMs. Their progress must be continuously updated and monitored at every simulation step. An internal event is generated to inform the Datacenter entity that a task unit completion is expected in the near future. Thus, at each simulation step, the Datacenter invokes a method called `updateVMsProcessing()` for the host. Following this, the contacted VMs update processing of currently active tasks with the host. The input parameter type for this method is current simulation time and the return parameter type is the next expected completion time of a task currently running in one of the VMs on the host. The next internal event time is the least among all the finish times, which are returned by the host.

At the host level, invocation of `updateVMsProcessing()` triggers an `updateCloudletsProcessing()` method that directs every virtual machine to update its task status with the Datacenter. This method implements similar logic as described previously for `updateVMsProcessing()` but at the VM level. Once this method is called, VMs return the next expected completion time of task units currently managed by them. The least completion time among all the computed values is sent to the Datacenter.

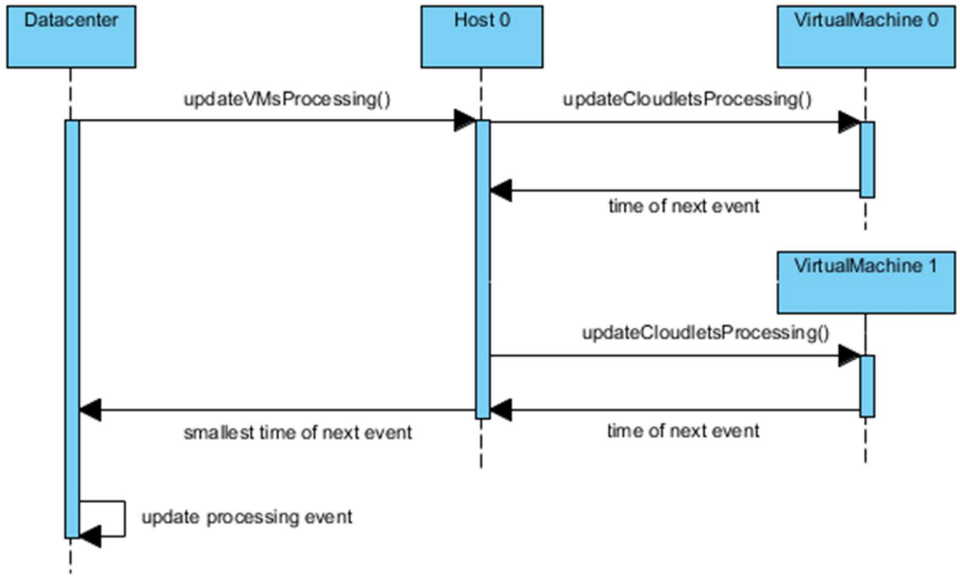

Figure 10. Cloudlet processing update process.

2. MYLOGNORMALGENERATOR.JAVA

New class called in *CloudSimWebExample.java* after the conversion between Gaussian and Lognormal parameters is done.

```
package org.cloudbus.cloudsim.ex.web.examples;

import java.util.Random;
import org.uncommons.maths.number.NumberGenerator;
import org.uncommons.maths.random.GaussianGenerator;

public class MyLognormalGenerator implements NumberGenerator<Double> {

  private NumberGenerator<Double> my_gen;

  public MyLognormalGenerator(double mean, double stddev, Random rng) {
    my_gen = new GaussianGenerator(mean, stddev, rng);
  }


  public Double nextValue() {
    Double d = my_gen.nextValue();

    double x = d.doubleValue();

    return new Double(Math.exp(x));
  }
}
```

Glossary

API: Application Programming Interface

AWS: *Amazon Web Service*

BW: Bandwidth

CIS: Cloud Information Service

CloudSim: *Cloud Simulation Toolkit for Modelling and Simulation of Clouds*

CloudSimEx: *CloudSim Extension*

CPU: Central process unit

CV: Coefficient variation

IaaS: Infrastructure as a Service

IDE: Integrated development environment

ISO: International Organization for Standardization

IT: Information Technology

JDK: *Java Development Kit*

NIST: *National Institute of Standards and Technologies*

OpEx: Operating expense

PaaS: Platform as a Service

PE: Process elements

QoS: Quality of Service

RAM: Random Access Memory

SaaS: Software as a Service

SDK: Software Development Kits

SQL: Structured Query Language

VM: Virtual machine