Management of Cloud systems applied to eHealth

Jordi Vilaplana Mayoral

Dipòsit Legal: L.1439-2015
http://hdl.handle.net/10803/311417

Management of Cloud systems applied to eHealth

by

Jordi Vilaplana Mayoral

Submitted to the Department of Computer Science and Industrial Engineering
in partial fulfillment of the requirements for the degree of

PhD Thesis in Engineering and Information Technology

at the

UNIVERSITY OF LLEIDA

September 2015

Thesis Supervisor .....
Francesc Solsona
Associate professor

Thesis Supervisor .....
Francesc Abella
Adjunct Professor

Management of Cloud systems applied to eHealth

by

Jordi Vilaplana Mayoral

Submitted to the Department of Computer Science and Industrial Engineering
on July 21, 2015, in partial fulfillment of the requirements for the degree of
PhD Thesis in Engineering and Information Technology

Abstract

Background: Cloud computing is a new paradigm that is changing how companies, institutions and people understand, perceive and use current software systems. With this paradigm, organizations have no need to maintain their own servers, nor host their own software. Instead, everything is moved to the Cloud and provided on demand, saving energy, physical space and technical staff. Cloud-based system architectures provide many advantages in terms of scalability, maintainability and massive data processing. The healthcare sector can greatly benefit from these advantages.

Methods: Queueing theory and nonlinear programming were used respectively as a mathematical approach to modelling Cloud-based architectures and to develop an algorithm aimed at reducing both energy consumption and response time. An event-driven simulator was used to implement and compare some of these techniques. Finally, two eHealth applications were designed, deployed and tested in a real Cloud environment.

Results: This thesis explores techniques, models and algorithms for an efficient management of Cloud systems and how to apply them to the healthcare sector in order to improve current treatments. It presents two Cloud-based eHealth applications to telemonitor and control smoke-quitting and hypertensive patients. Different Cloud-based models were obtained and used to develop a Cloud-based infrastructure where these applications are deployed. The results show that these applications improve current treatments and that can be scaled as computing requirements grow.

Conclusions: Multiple Cloud architectures and models were analyzed and then implemented using different techniques and scenarios. The Smoking Patient Control (S-PC) tool was deployed and tested in a real environment, showing a 28.4% increase in long-term abstinence. The Hypertension Patient Control (H-PC) tool, was successfully designed and implemented, and the computing boundaries were measured.

Resum

Introducció: La computació al Núvol és un nou paradigma que està canviant com les empreses, les institucions i la gent entenen, perceben i usen els sistemes de programari actuals. En aquest paradigma, les organitzacions no tenen la necessitat de mantenir els seus propis servidors ni d'allotjar el seu propi programari. En canvi, tot això es mou al Núvol i es proporciona sota demanada, estalviant energia, lloc físic i personal tècnic. Les arquitectures dels sistemes basats en el Núvol ofereixen múltiples avantatges en termes d'escalabilitat, manteniment i processament massiu de dades. El sector de la salut pot beneficiar-se àmpliament d'aquests avantatges.

Mètodes: La teoria de cues i la programació no línial s'han utilitzat com una aproximació matemàtica per modelar arquitectures basades en el Núvol i per desenvolupar un algorisme amb l'objectiu de reduir tant el consum energètic com el temps de resposta, respectivament. Un simulador basat en esdeveniments s'ha utilitzat per implementar i comparar algunes d'aquestes tècniques. Finalment, dues aplicacions de salut electrònica (eHealth) s'han dissenyat, desplegat i provat en un entorn real al Núvol.

Resultats: Aquesta tesi explora tèniques, models i algorismes per una gestió eficient en sistemes al Núvol i com aplicar-ho en el sector de la salut per tal de millorar els tractaments actuals. Presenta dues aplicacions de salut electrònica basades en el Núvol per telemonitoritzar i controlar pacients fumadors i hipertensos. S'ha obtingut diferents models basats en el Núvol i s'han utilitzat per a desenvolupar una infraestructura on desplegar aquestes aplicacions. Els resultats mostren que aquestes aplicacions milloren els tractaments actuals així com escalen a mesura que els requeriments computacionals augmenten.

Conclusions: Múltiples arquitectures i models han estat analitzats i implementats utilitzant diferents tècniques i escenaris. L'aplicació Smoking Patient Control (S-PC) ha estat desplegada i provada en un entorn real, aconseguint un augment del 28,4% en l'abstinència a llarg termini de pacients fumadors. L'aplicació Hypertension Patient Control (H-PC) ha estat dissenyada i implementada amb èxit, i els seus límits computacionals han estat mesurats.

Resumen

Introducción: La computación en la Nube es un nuevo paradigma que está cambiando como las empresas, instituciones y la gente entienden y usan los sistemas de software actuales. En este paradigma, las organizaciones no tienen la necesidad de mantener sus propios servidores ni de alojar su propio software. En cambio, todo esto se mueve a la Nube y se provee bajo demanda, ahorrando energía, espacio físico y personal técnico. Las arquitecturas de los sistemas basados en la Nube ofrecen múltiples ventajas en términos de escalabilidad, mantenimiento y procesamiento masivo de datos. El sector de la salud puede beneficiarse ampliamente de estas ventajas.

Métodos: La teoría de colas y la programación no lineal se han utilizado como una aproximación matemática para modelar arquitecturas basadas en la Nube y para desarrollar un algoritmo con el objetivo de reducir tanto el consumo energético como el tiempo de respuesta, respectivamente. Se ha utilizado un simulador basado en eventos para implementar y comparar algunas de estas técnicas. Finalmente, dos aplicaciones de salud electrónica (eHealth) se han diseñado, desplegado y probado en un entorno real en la Nube.

Resultados: Esta tesis explora ténicas, modelos y algoritmos para una gestión eficiente de sistemas en la Nube y como aplicarlos en el sector de la salud con el fin de mejorar los tratamientos actuales. Presenta dos aplicaciones de salud electrónica basadas en la Nube para telemonitorizar y controlar pacientes fumadores e hipertensos. Se han obtenido diferentes modelos basados en la Nube y se han utilizado para desarrollar una infraestructura donde desplegar estas aplicaciones. Los resultados muestran que estas aplicaciones mejoran los tratamientos actuales así como escalan a medida que los requerimientos computacionales aumentan.

Conclusiones: Múltiples arquitecturas y modelos han sido analizados e implementados utilizando diferentes técnicas y escenarios. La aplicación Smoking Patient Control (S-PC) se ha desplegado y probado en un entorno real, consiguiendo un aumento del 28,4% en la abstinencia a largo plazo de pacientes fumadores. La aplicación Hypertension Patient Control (H-PC) ha sido diseñada e implementada con éxito, y sus límites computacionales han sido medidos.

Our greatest weakness lies in giving up.

The most certain way to succeed is always to try just one more time.

— Thomas A. Edison

Acknowledgments

Doing this PhD has been an eye-opening experience and a great opportunity. It has been tough at times, and I had to step out of my comfort zone, but it has taught me a lot and I am truly happy that I have had a chance to complete it. It would not have been possible without all those people who helped me along this journey. First of all, I would like to thank my supervisors, Dr. Francesc Solsona and Dr. Francesc Abella, whose constant attention and care pushed me unhurriedly and steadily throughout this adventure.

I want to acknowledge Prof. Yaxin Bi for being an excellent host and supervisor during my doctoral stay at the Ulster University, and for providing valuable comments and suggestions on improving my work.

I would like to thank all the past and current members of the Department of Computer Science and Industrial Engineering, at the University of Lleida. In particular, I thank Fernando Cores, Francesc Giné, Fernando Guirado, Josep Lluís Lérida and Concepció Roig. I also want to thank my past and current co-workers, Eloi Gabaldón, Ivan Teixidó, Jordi Mateo, Jordi Lladós, Ismael Arroyo, Miquel Orobitg, Josep Rius and Anabel Usié, for being such a wonderful bunch of people, I have had a great time with them. I also want to mention Montse Espunyes for being the best clerk a department can hope for.

I would also like to thank my friends and family for the support they have provided me with throughout my life, especially my parents, Ignasi and Conxita, who exposed me to all sorts of opportunities as I grew up, and have always been incredibly encouraging of everything I wanted to do. Also Ignasi and Anna, for being a great brother and sister-in-law. And in particular, I must acknowledge my wife and best friend, Àgata, without whose love, encouragement and assistance, I would not have finished this thesis.

Jordi Vilaplana

Lleida, Catalonia

10 September 2015

Contents

List of Figures

Chapter 1

Introduction and scope of the research

Cloud computing has gained worldwide attention in the last few years, not only for its benefits to the Information and Communication Technology (ICT) industry, but also for its application in several different fields. This paradigm allows organizations to delegate their computing infrastructure to a Cloud provider, paying only for what they are really using. Consequently, many companies have gained access to computational resources that were limited to high performance computing centers.

Several industries have embraced Cloud computing as a means to improve their efficiency and productivity, and to offer better service to their customers. The health care sector can benefit greatly from the advantages of Cloud infrastructures in order to improve treatment, reduce costs and offer patients better service.

Applying Cloud computing to the health care sector opens new opportunities when dealing with several medical institutions containing thousands of patients. However, it is essential to guarantee a certain level of provisioning and performance. This task can be achieved by means of different techniques and approaches that are dealt with in this work.

Figure 1-1 shows an overall schema of the work involved in this thesis where Cloud systems are modelled using such formal mathematical techniques such as queueing theory and nonlinear programming. Then, performance evaluation is carried out

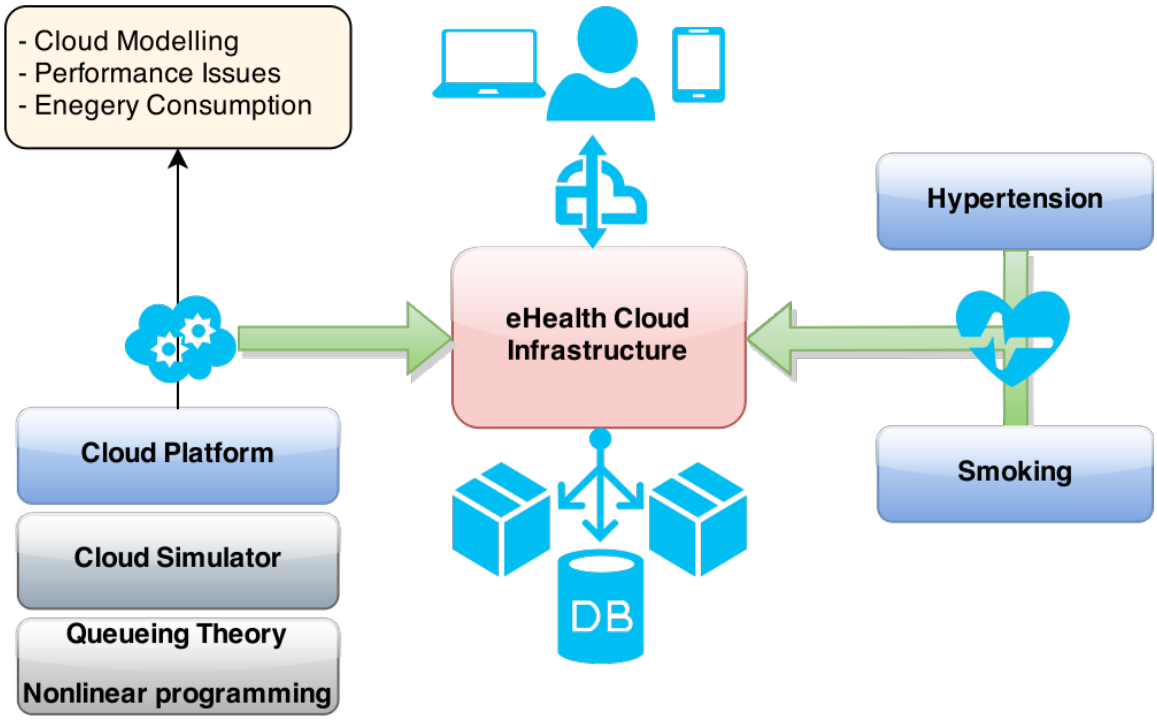


Figure 1-1: Overview of the eHealth Cloud Infrastructure.

through a simulation tool and a Cloud platform, taking performance and energy consumption issues into account. Finally, the Cloud infrastructure applied to eHealth is designed. We have dealt mainly with two specific scenarios: hypertension and smoking.

1.1 Modelling Cloud systems

Cloud systems can be served in three different ways (see Figure 1-2). The first layer, named *Infrastructure as a Service (IaaS)*, consists of offering hardware, storage and physical devices over the Internet. The *Software as a Service (SaaS)* layer offers software and hosted applications over the Internet. Finally, as a combination of both, the *Platform as a Service (PaaS)* layer offers the capability of deploying applications created using programming languages, libraries, services, and tools supported by the Cloud provider, where the consumer does not manage or control the underlying Cloud infrastructure, but has control over the deployed applications [23, 38].

Moreover, Clouds can be deployed as *public*, *private*, *hybrid* or *community*, depending on how they are managed. *Private* Clouds are operated by a single company or organization, whilst *public* Clouds are available for external and public usage. *Community* Clouds share their infrastructure between multiple companies or organizations. Finally, *hybrid* Clouds are a composition of the previous deployment models.

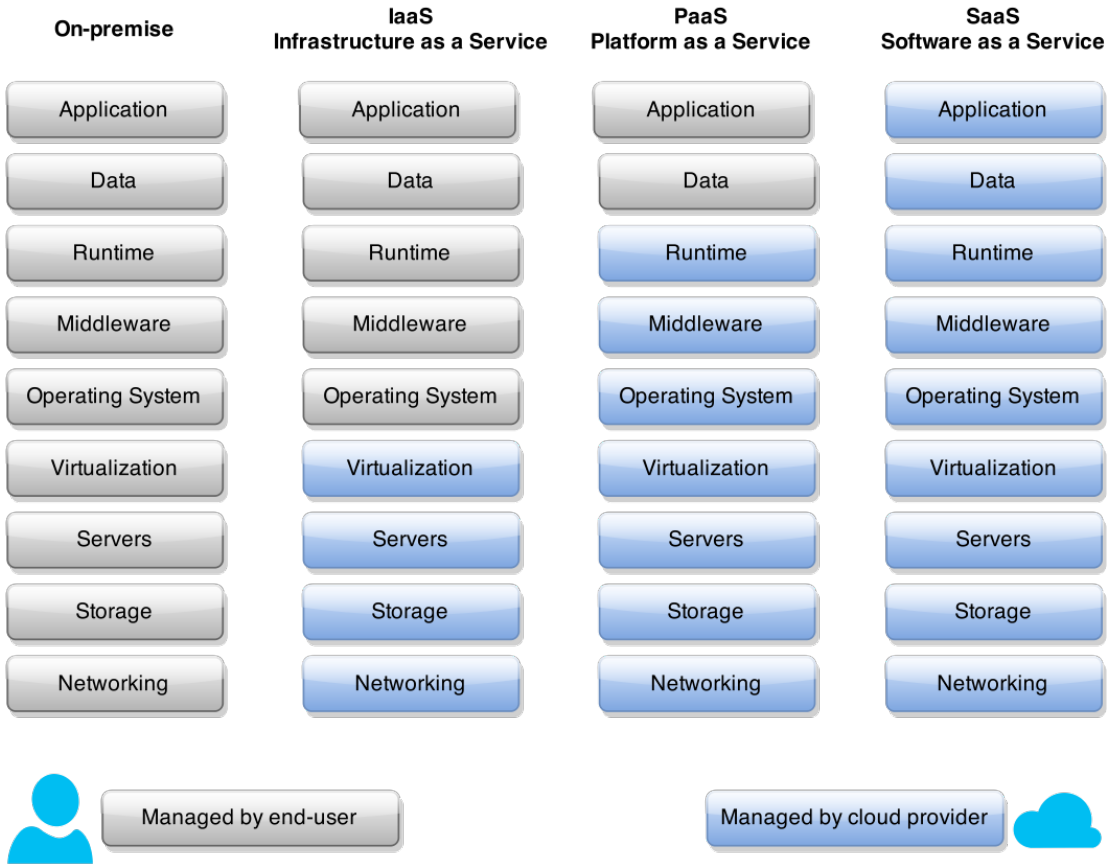


Figure 1-2: Cloud service models.

When handling large amounts of users and data, saturation problems can arise. In Cloud computing, hardware and software services are efficiently handled, as they can be added and released dynamically [3]. Problems arise when scaling the system, that is, when trying to deploy a platform to support the computing needs of many hospitals with different clinical departments and their corresponding clinicians and patients. The need for scalability further increases when the Cloud must also provide support for additional common and specific desktop or Internet applications (administration, specialized, general purpose, etc.).

Models can be designed in order to predict different performance metrics, such as *response time*, *task blocking probability*, *immediate service* and *mean amount of system tasks* [59], for specific scenarios. The models can usually be correctly sized, thus avoiding or minimizing performance issues. There are several ways to develop these models.

1.1.1 Queueing Theory

Queueing theory consists of studying queues from a mathematical point of view, and it provides tools to define several performance metrics [24].

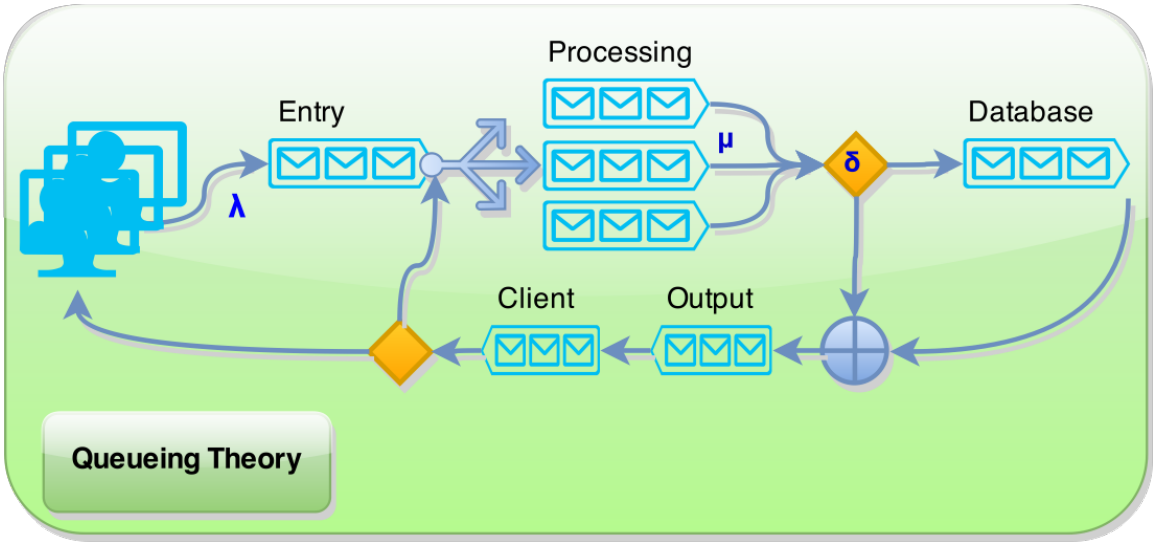


Figure 1-3: Queueing theory concept model.

Figure 1-3 shows a simple queueing theory model. Servers are seen as queueing nodes, each one with its own arrival (λ) and service (μ) rates.

Parameters such as the utilization factor (ρ), the average number of costumers in the system (N) and the average time in the system (T) can be calculated for different queues. One of the most widely-used queues is the $M/M/1$, which represents a one-server system where both the arrival rate and the service time follow an exponential distribution. Equations 1.1, 1.2 and 1.3 respectively show how the utilization factor, average number of customers and the average time in the system are obtained.

$$\rho = \frac{\lambda}{\mu}, \lambda < \mu \tag{1.1}$$

$$N = \frac{\rho}{1-\rho} \tag{1.2}$$

$$T = \frac{\frac{1}{\mu}}{1-\rho} \tag{1.3}$$

Moreover, several different queues can be modelled depending on multiple factors, such as the arrival and service distributions or the number of servers. The resulting analyses sharply grow in complexity with more elaborate models.

Although queueing theory results are not directly applicable to Cloud performance analysis when there is a large number of servers, the distribution of service times is unknown or the traffic intensity varies over a wide range [23], these models allow us to obtain reasonable approximations to the response time distribution of Cloud systems [61] and determine the level of service and relationship between the maximum number of tasks and the minimum number of resources.

1.1.2 Nonlinear programming

Like queueing theory, nonlinear programming is a mathematical approach that can be used to study and model performance issues. A nonlinear programming problem (NLP) deals with mathematical optimization problems where the objective function to be maximized or minimized, or some of its constraints, are nonlinear. It is a technique for solving an optimization problem that is subject to a set of constraints (equalities and inequalities), over a set of unknown real, integer and Boolean variables, along with an objective function to be maximized or minimized.

Generally, a NLP can be defined as:

$$Max(f(x_1, x_2, ..., x_n)), \tag{1.4}$$

s.t. (subject to):

$$\begin{aligned} h_i(x) &= b_i \quad (i = 1, 2, ..., m) \\ x_j &\geq 0 \quad (j = 1, 2, ..., n) \end{aligned} \tag{1.5}$$

Once the NLP has been defined, it can be processed by a solver (i.e. CPLEX, lp_solve, Excel Solver Function, etc.) to extract an optimal solution.

Linear and nonlinear programming are widely used in scheduling research [1]. It has been used to design a nonlinear programming scheduling algorithm to minimize energy consumption and maximize SLA guarantees at the same time.

1.1.3 Cloud simulation

Although queueing theory can provide very efficient models for relatively simple scenarios, the complex mathematical analyses hinder us from obtaining simple and fast solutions for more elaborate systems.

In addition, it is not possible to perform benchmarking tests in dependable, repeatable and scalable environments using real Cloud environments [9, 5].

Therefore, Cloud simulator tools are best suited for large Cloud system testing to decrease the complexity and enable performance analysts to assess system behavior by focusing on quality issues of specific components under different scenarios [36].

Several Cloud simulators have been specifically developed for the performance analysis of Cloud computing environments, with CloudSim[1], CloudAnalysit [60], GreenCloud[2], iCanCloud[3], MDCSim [34], NetworkCloudSim [17] and VirtualCloud[4] being the most prominent ones. Among all these tools, CloudSim is considered to be the most sophisticated [36] and is also the most widely-used and referenced in the literature.

CloudSim is an event-driven and extensible simulation toolkit that enables Cloud systems and scenarios to be modelled and simulated. It facilitates the development

[1]CloudSim webpage: http://www.cloudbus.org/cloudsim/
[2]GreenCloud webpage: http://greencloud.gforge.uni.lu/
[3]iCanCloud webpage: http://www.arcos.inf.uc3m.es/ icancloud/Home.html
[4]VirtualCloud webpage: http://sourceforge.net/projects/virtualcloud/

of personalized settings involving multiple data centers, physical hosts and virtual machines with their own characteristics. It also allows custom allocation policies to be developed for both user jobs or tasks and for virtual machines. The high level of customization and its default policies make CloudSim a remarkable tool for simulating both simple and complex scenarios.

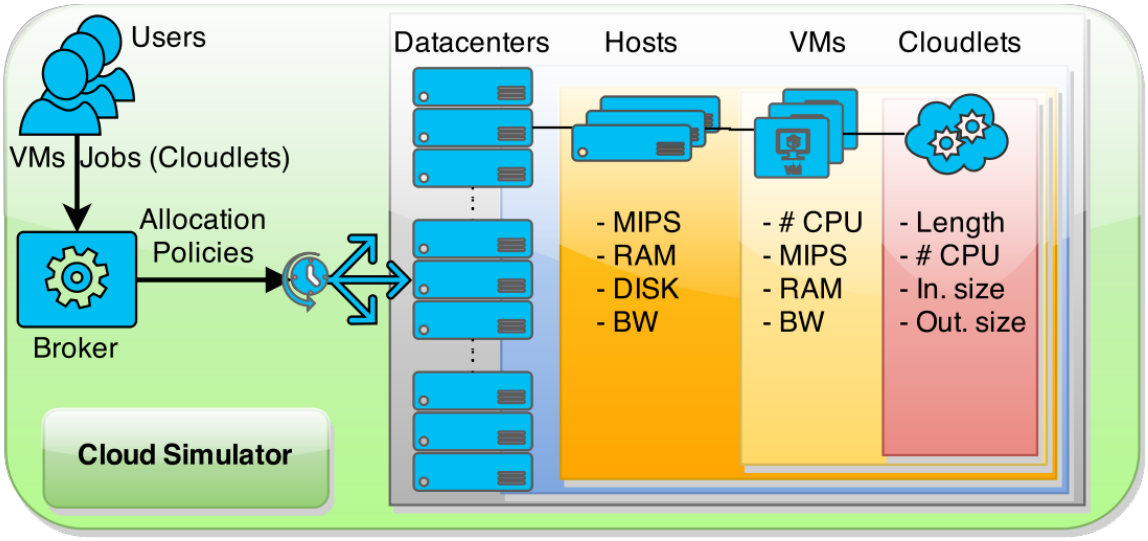


Figure 1-4: Basic CloudSim architecture.

A simplified architecture schema of CloudSim is described in Figure 1-4. There are multiple entities that must be taken into account in CloudSim. A Datacenter is an entity that represents an aggregation of Hosts. A Host represents a physical machine with its own computing capacity (as Millions of Instructions per Second or MIPS), RAM memory, storage space or disk, and bandwidth (BW). Virtual machines (VMs) are allocated to Hosts following a Virtual Machine Allocation Policy that can be fully customized. Jobs or tasks are represented by the Cloudlet entity. Each Cloudlet has its own length (in Millions of Instructions), the required number of processing elements or CPUs and its input and output size. Cloudlets are allocated to VMs following a Cloudlet Allocation Policy that can also be fully customized. Finally, the Broker entity coordinates things between the Datacenters and the Users, receiving Cloudlets and VMs from the users and scheduling them to the different Datacenters according to specific policies.

When programming a CloudSim simulation, one must provide the initial Cloud

environment, describing the Datacenters, Hosts and types of Virtual Machines. Simulations can be then performed where simulated users can submit their own VMs and Cloudlets. Multiple metrics can be obtained and a wide range of scenarios can be simulated.

Although the CloudSim experimentation performed in the course of this thesis does not appear in the main articles listed in Chapter 2, some of it can be seen in Appendix B.

1.1.4 Cloud platform

Cloud platforms allow us to implement our models and deploy our applications within a real Cloud infrastructure.

Many companies offer commercial Cloud platforms in terms of IaaS (Infrastructure as a Service), where users can purchase their services on a pay-as-you-go basis. Among the most prominent ones are the Amazon EC2[5], Windows Azure[6] and the Google Cloud Platform[7].

Moreover, there are also several open source Cloud platforms that can be freely downloaded and installed. OpenStack[8], XCP[9], Eucalyptus[10], OpenNebula[11], Nimbus[12] and Apache CloudStack[13] are probably the most mature and best-known ones.

Among all the possible solutions, OpenStack is the one used throughout this work. The main reasons are its open-source license, the availability of APIs so that user-deployed applications can interact with the platform and the fact that OpenStack is a solid and well-known solution adopted by many researchers.

The basic architecture of OpenStack can be seen in Figure 1-5. This platform can be installed on top of one or several physical machines or servers connected throughout

[5]Amazon EC2 webpage: http://aws.amazon.com/ec2/
[6]Windows Azure webpage: http://azure.microsoft.com/
[7]Google Cloud Platform webpage: https://cloud.google.com/
[8]https://www.openstack.org/
[9]XCP webpage: http://www.xenproject.org/
[10]Eucalyptus webpage: https://www.eucalyptus.com/
[11]OpenNebula webpage: http://opennebula.org/
[12]Nimbus webpage: http://www.nimbusproject.org/
[13]Apache CloudStack webpage: https://cloudstack.apache.org/

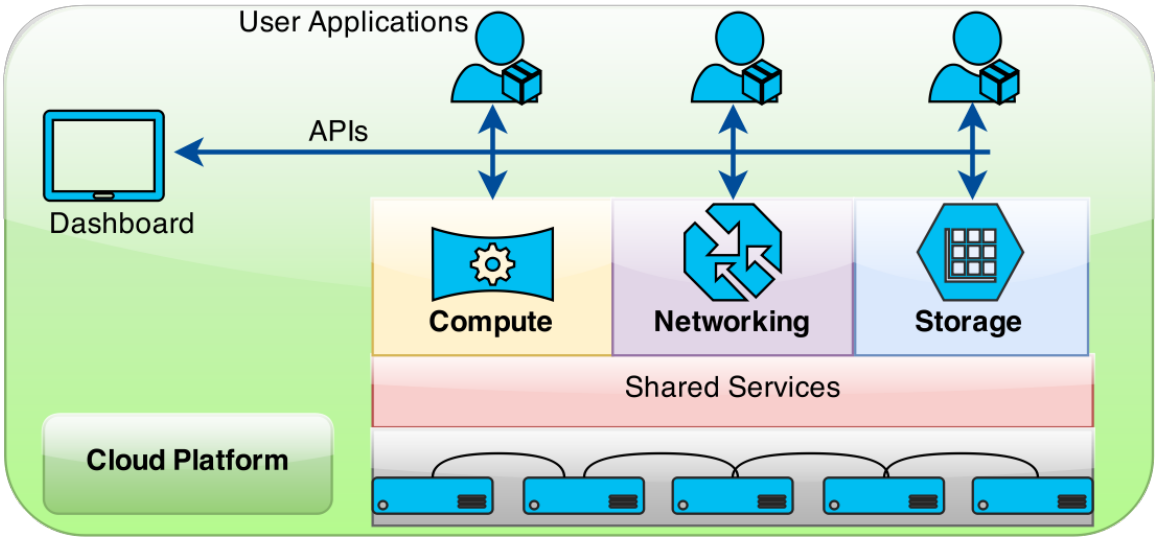


Figure 1-5: Architecture of OpenStack.

a local network. Once installed, OpenStack offers several services distributed in four different blocks or layers. The *compute* layer provisions and manages large networks of virtual machines. The *networking* layer offers a pluggable and scalable network and IP address management. The *storage* layer offers object and block storage capabilities for use with servers and applications. Finally, the *shared services* layer offers additional features that expand the three previous layers, such as the identity service that manages the OpenStack users or the image service that provides capabilities for managing disk and server images, among others.

1.2 Cloud computing challenges and issues

Cloud computing is a large paradigm and involves many challenges and issues that can be studied from different ICT perspectives. Cloud architectures must provide high *availability* [37], *serviceability* [55], *variability* [20], *reliability* [58] and system *security* [62] while maintaining a negotiated performance and being energy efficient.

1.2.1 Reliability, availability and serviceability

There are many aspects to take into account in Cloud computing. However, the *reliability*, *availability* and *serviceability* (RAS) of the Cloud infrastructure are major aspects of the virtualization technology and require special attention [51].

The *reliability* of the Cloud systems is critical, although it is hard to analyze due to its characteristics of massive-scale service sharing, wide-area network, heterogeneous software and hardware components and the complex interactions between these [10]. Therefore, typical software or hardware reliability models or conventional networks simply cannot be applied to studying the reliability of Cloud systems. Reliability models must take into account several types of failures that can have a significant impact on the overall functioning of Cloud services, such as overflow, timeout, data resource missing and software, hardware and network failures.

Availability refers to the ratio of time a Cloud system is functional to the total time it is required or expected to function. This can be expressed as a direct proportion, a percentage (i.e., 90%) or in terms of average downtime. Delivering a high level of availability has been a major challenge, as Cloud technology has been deployed on a large scale, the likelihood of incidents has progressively increased over the last few years [46].

In Cloud computing, high *serviceability* indicates the ability to maintain and repair the system easily and it is directly related to fault tolerance. Being able to detect potential problems is critical in this respect, and once a problem is found, the maintenance and repair operations should cause as little service disruption as possible.

This thesis does not focus on these issues.

1.2.2 System security, user privacy and trust issues

Security can be seen as the combination of confidentiality (prevention of the unauthorized disclosure of information), integrity (prevention of the unauthorized amendment or deletion of information), and availability (prevention of the unauthorized withhold-

ing of information). Security is the absence of unauthorized access to, or handling of, the system state [4], and is one of the most significant obstacles for opening up the new era of the long dreamed vision of computing as a utility [56]. For Cloud users, *privacy* means "that nobody else has access to my data" [19]. That is, the ability of an individual or group to seclude themselves, or information about themselves, and thereby reveal themselves selectively. Finally, *trust* is seen as a measurable belief that utilizes experience to make trustworthy decisions. Trust is one of the most important means of improving security and enabling the interoperability of current heterogeneous independent Cloud platforms. Moreover, it is a complex relationship between entities because it is extremely abstract, unstable and difficult to measure and manage [33].

This thesis does not focus on these issues.

1.2.3 Performance and energy consumption issues

In Cloud computing, a Service-Level Agreement (SLA) is a contract or agreement between a service provider and a consumer where the former agrees to deliver a service to the latter under specific terms, such as time or performance. In order to comply with the SLA, the service provider must monitor its quality of service (QoS) closely through such performance metrics as response or waiting time, throughput or makespan [21]. Studying and determining SLA-related issues is a big challenge [20].

In this thesis, a policy named *GreenC* was designed to lower energy consumption while maintaining a desired SLA. GreenC focuses on these two aspects and does not consider such other Cloud-related issues such as variability, system security and availability.

Job response time is one of the most important QoS metrics in a Cloud computing context. Good solutions dealing with QoS and energy consumption have been presented by some researchers in the literature [6, 25]. However, the model presented in GreenC aims to obtain the best scheduling consolidation of virtual machines to hosts taking both criteria into account.

The proposed policy has been theoretically tested by means of nonlinear program-

ming and it has been implemented and tested using the OpenStack Cloud platform.

1.3 Motivations for eHealth

Healthcare is a critical area that can benefit from the advantages of Cloud computing. Although significant work is being done in this field, there is still much to do in order to bring computer science into the health area. The union of the healthcare and the information and communications technology (ICT) areas, usually referred as eHealth, supplies unique and valuable benefits to the healthcare practice. Cloud computing can offer many opportunities for improving health care services from the viewpoint of management, technology, security and legality [27].

By moving the infrastructure to the cloud, valuable data extracted from the databases of treatment, patients, diseases, and so on will be accessible to doctors for analytical studies and to see statistical results. By hiding the personal details of patients, data could be shared between doctors and even hospitals, and could also be cross-referenced between different diseases and treatments. In [50], the authors examine how the biomedical informatics community, especially consortia that share data and applications, can take advantage of Cloud computing. Cloud computing systems offer the illusion of infinite computing resources available on demand, allowing an expansion of the resources when needed. Hardware and software services are more efficiently handled than in other High Performance Computing (HPC) infrastructures as they can be added and released dynamically [2]. However, problems arise when scaling the system, that is, when trying to deploy a platform to support the computing needs of many hospitals, with different clinical departments and their corresponding clinicians and patients. We can say that this health approach can be extrapolated to many other areas, such as administration, education, social care, etc.

A recent study [16] showed that personalized follow-up by telematic tracking applications via SMS messaging improved the results of patients trying to quit smoking. Related experiments also proved that the same method is useful for applications related with the treatment of hypertensive patients [7] and in patients with chronic

disease in general [31]. By using telematic applications, the time dedicated to personalized clinical attention to patients increased, and clinicians more effectively scheduled and managed that time. It also avoided unnecessary travelling by patients, while allowing them to feel closely followed by the clinician. This is just one example of the benefits that telematic applications can provide, and which are increasingly being implemented in health centres.

1.4 Description of eHealth

eHealth involves a wide range of aspects related to medicine and healthcare where information and communication technologies are involved. One of the main ones is telemedicine, where diagnosis and/or treatment is performed remotely. Other aspects are the management of electronic health records, electronic prescribing options and patient data management. Moreover, mHealth refers to the use of mobile devices to collect and receive data, communicate and perform real-time monitoring.

These systems can integrate multiple aspects of eHealth to provide an enhanced diagnosis or treatment and improve the existing ones. This work is focused on telemedicine applications, where patient-doctor communication is partially performed remotely.

1.5 Applied cases: Smoking and Hypertension

In this thesis, two applied cases were researched and, as a result, two eHealth telemedicine applications, based on a Cloud infrastructure and delivered in a SaaS model (see Figure 1-2), were developed. In particular, two eHealth applications for treating smoke-quitting and hypertensive patients.

Smoking and hypertension have been chosen as applied cases due to the fact that both conditions involve a strong psychological factor in their treatment, where a close follow-up and enhanced doctor-patient communication can be decisive for a successful outcome of the treatment.

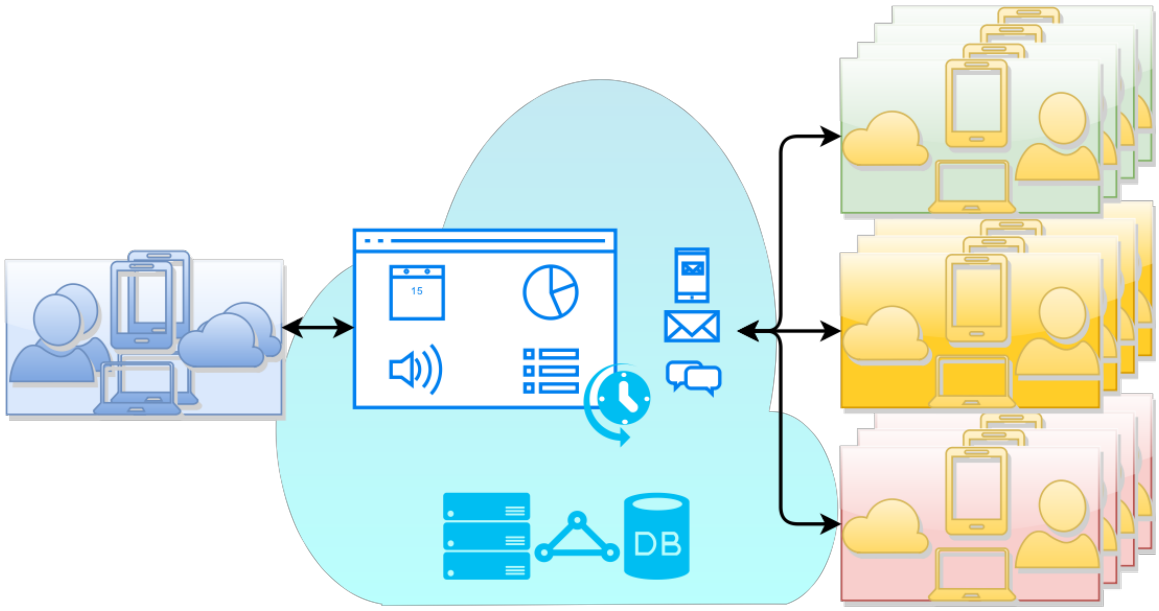


Figure 1-6: General diagram for Smoking and Hypertension platforms.

Figure 1-6 shows a concept diagram for the two platforms, where doctors and patients interact with each other by means of the eHealth applications. Doctors have a web-based dashboard where they can easily communicate with their patients and check their status in terms of statistics, tables and charts. Patients receive the scheduled messages from their doctors and can also send their current status back. The applications react differently depending on their status, allowing doctors to focus on the ones that need more attention.

In the following subsections, both eHealth applications are thoroughly described.

1.5.1 Smoking

Smoking is a major risk factor for active and passive smokers in certain respiratory [14] and circulatory diseases [43] as well as in some types of cancer and infections, among other diseases. Because of this, both public and private medical institutions in an increasing number of countries provide services for people who want to stop smoking.

There are several treatments and techniques to achieve abstinence in smoking patients like the Nicotine Replacement Therapy (NRT). These consist of replacing the

source of nicotine using transdermal patches, gums, nasal spray, inhaler or sublingual tablets. NRT has proved effective in treating short-term nicotine withdrawal, increasing the chances of successfully quitting smoking by between 50% and 70% [54].

However, NRT treatments are not so effective in keeping a patient off smoking over longer periods of time, after around eight weeks. Quitting smoking is just a first step and achieving a long-term abstinence in ex-smokers can be a difficult challenge. For this reason, reinforcing this part of the treatment can be critical in terms of achieving a successful outcome.

Several cessation programs combine pharmacological treatment with a simultaneous psychological treatment to control the progress and reinforce the motivation of the patient, which can be done on an individual basis [28], in the context of group therapy [52], or via long-distance support through phone calls [53]. A previous study proved that individual counselling, combined with telephone counselling were associated with higher 52-week abstinence rates than telephone counselling alone [48]. Moreover, previous studies have shown that social support was associated with cessation and short-term maintenance of abstinence [39]. Therefore, there is the need to develop efficient eHealth tools in order to optimize both the time spent by the clinicians who follow a patient and the efficacy of their service, minimizing the probability of relapse. In consequence, tools are required to allow professionals to follow patients without making this too time consuming.

There is previous research showing that the use of telephones, mobile phones and text messages to develop a tighter follow-up of the patients is more effective for successfully stopping smoking and reducing cigarette consumption [15, 57, 16]. Such contact and follow-up are very important psychological aspects of the process of quitting smoking, because they provide support and help maintain patient motivation [11].

In this thesis, an eHealth tool named S-PC (Smoker Patient Control) was designed, developed and deployed. This tool allows fluid communication between clinicians and patients while providing the former with an efficient interface to control, manage and personalize the follow-up and treatment of their patients. It allows clinicians and

patients to communicate and interact via mobile text messages (SMS) and e-mails.

Given that this requires a large time investment by health professionals, it is important for them to have tools that automate this part of the treatment as far as possible, while maintaining or increasing the efficiency of the professionals. Taking the facts described in the previous paragraphs into account, it was our objective to develop and benchmark the effectiveness of an eHealth tool that would: a) be generally applicable in smoking cessation treatment programs, b) automate much of the work that needs to be done by the clinicians, c) allow professionals to maintain a personalized support and follow-up of patients more effectively, d) give patients the psychological support that they require to succeed at stopping smoking, and e) decrease the time needed by clinicians to manage the patients and reduce the average length of waiting lists. This tool was named S-PC (Smoker Patient Control). An additional objective was to understand to what extent patients were satisfied with being treated using the tool.

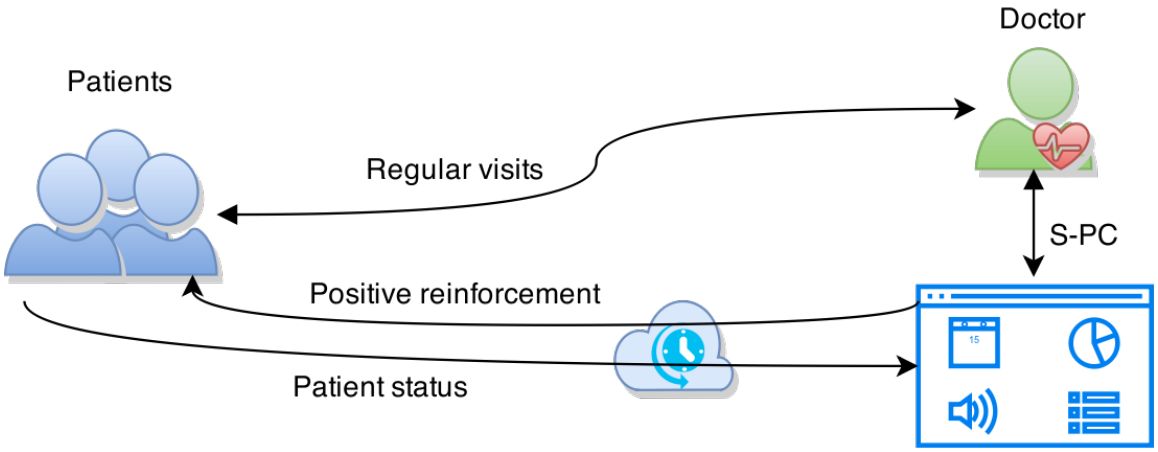


Figure 1-7: Smoking application (S-PC) diagram.

1.5.2 Hypertension

Hypertension is a factor of risk in cardiovascular disease, the leading cause of death worldwide. The difficulty of ensuring satisfactory control of blood pressure is still a great challenge, despite the major steps that have been taken to improve the lifestyle through new pharmaceutical treatments [29, 12].

Monitoring blood pressure at home consists of patients taking the readings at home and registering these using a digital device. Then, the patients send the readings to a health professional, who is responsible for taking appropriate action. One home blood pressure approach is patient self-management, defined as the ability and willingness of a patient to self-monitor. Some studies show that self-managed blood pressure control is at least as, or even better than, office-monitored blood pressure [8, 41].

Our motivation is based on the work published in [26], which demonstrates the effectiveness of telemonitoring.

High blood pressure monitoring and telemonitoring should become a routine component of blood-pressure measurement in the majority of patients with known or suspected hypertension [47], given that such readings may be better predictors of cardiovascular and renal outcomes than surgery readings [42, 7].

A telemedicine eHealth application called Hypertension Patient Control (H-PC) was developed to allow patients to monitor their blood pressure at home and send the measurements to their clinicians via SMS or e-mail. In addition, through H-PC, the professional can also send SMS or e-mail messages back to patients. In telemonitoring, readings taken at home are relayed by communication media to health-care professionals who can take appropriate action [44].

Figure 1-8 shows a flow diagram for the H-PC tool. First, the application server sends a scheduled message to the subscribed patients requesting their blood pressure readings. Once the patients have measured and sent their readings back to the server, the system checks whether those readings are within the established limits or not. In case of abnormally high readings, the system will automatically notify the doctor about this situation. The doctor, through the application web interface, can check all the data and decide the best course of action.

The purpose of this tool is not to implement a new protocol or treatment procedure for hypertensive patients, but rather to develop a telemonitoring application that can be used by both patients and clinicians in a simple and user-friendly fashion.

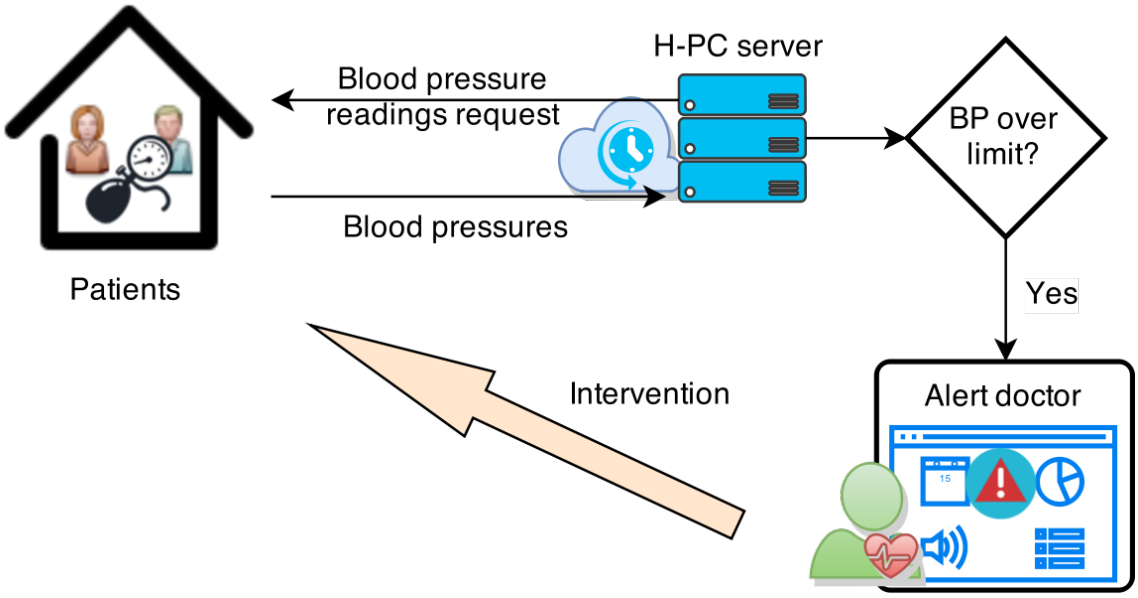


Figure 1-8: Hypertension application (H-PC) diagram.

1.6 Research objectives

The main purpose of this thesis is to achieve eHealth solutions through Cloud-based technologies and architectures, specifically to allow clinicians to control and communicate with patients remotely and establish a bidirectional exchange to improve treatments where such contact is essential for a successful outcome.

To deal with this challenge, the following objectives were defined:

- Develop models to determine performance metrics (i.e. response time, throughput) and issues (i.e. SLA guarantee, QoS) in Cloud-based architectures. Specifically, design a Cloud system by means of queueing theory and validate it using the OpenStack Cloud simulator.

- Design of a nonlinear programming scheduling algorithm to optimize both energy consumption and SLA guarantees.

- Explore and analyze current solutions and environments where these models can be applied, and design a Cloud architecture for an eHealth environment and analyze its behaviour with the CloudSim simulator.

- Design and implement a Cloud-based eHealth telemonitoring tool for smoke-quitting patients and test it in a real environment to assess its usefulness through statistical studies of prevalence.

- Design and implement a Cloud-based eHealth telemonitoring tool for hypertensive patients, tests its performance and scalability and finally, test it in a real environment.

- Develop a scalable and power-aware Cloud-based infrastructure to support these applications using the OpenStack platform.

1.7 Related Work and Contributions

In queueing theory, previous research showed that Cloud systems could be modelled through an M/M/m open network [61]. Other theoretical studies evaluated and analysed more complex queueing theory systems that did not follow exponential distributions [35]. However, in these complex models it is not possible to obtain a closed formula to represent the probability distributions of the response or waiting time of customers in the queue, thus requiring finding approximate models. The main contribution in this field is the analysis and design of a Cloud-based system, modelled using M/M/m and M/M/1 queues, with quality of service capabilities (see Section 2.1). The queueing theory models are later validated and implemented using the OpenStack Cloud platform and the CloudSim event-driven simulator. Moreover, they are also extended by adding a two-level scheduler to map tasks to processing nodes according to their processing power and availability (see Appendix B).

In the field of linear and nonlinear programming, prior work has been done in algorithms applied to scheduling [18, 32]. Other techniques, such as neural network predictors [13] and genetic algorithms [40], have also been used to develop power-aware systems. The main contribution in this regard is the design of a nonlinear programming scheduling algorithm to minimize energy consumption and maximize SLA guarantees at the same time. To date, there is no research work based on

these two criteria in Cloud computing. Another important contribution is the way that the power of virtual machines is modelled in function of their workload. This algorithm relies on the work done in [22], where the authors formulate the problem of assigning people from various groups to different jobs and who may complete them in the minimum time as a nonlinear programming problem. The job completion times have been assumed to follow a *Gamma* distribution. To model the influence of the workload, the computing capacity of the physical host was weighted by a load factor determined by an *Erlang* distribution (equivalent to a *Gamma*). Finally, a nonlinear programming problem was obtained and transformed into an equivalent deterministic problem with a nonlinear objective function. The algorithms were implemented and tested using the OpenStack Cloud platform (see Section 2.2).

From the eHealth point of view, this thesis presents two telemonitoring applications for the smoking and hypertension scenarios that offer a bidirectional communication between patients and clinicians via SMS and e-mail, allow efficient patient management and access to electronic medical records.

Most reported studies on stopping smoking with mobile phone technology follow patients for at most six months [57, 16]. In Section 2.4, the S-PC tool is presented, where the associated study followed the patients for twelve months. Additionally, it evaluates the effect of the tool on time management of both clinicians and patients, waiting list reduction and patient satisfaction with the mobile text intervention.

The most important applications that perform similar functions to that of S-PC are STOMP [49], PMC [30] and txt2stop [15]. Table 1.1 presents a comparison of the functionality of these tools compared to S-PC, the latter having the most complete set of features. The communication channels used by each of the applications vary. PMC uses e-mails to exchange messages and information with the patients, whereas STOMP and test2stop use mobile text messaging (SMS) for the same purpose. S-PC can communicate via mobile text messages and/or e-mails, although for the clinical study, it was set-up to use mobile text messages only. S-PC and PMC are the only tools that create customized lists of patients and represent clinical history and treatment progression graphically. In addition, S-PC sends warning messages to clinicians

when a patient is identified by the program as being at risk. It also allows messages to be customized at will. These two features are exclusive to S-PC.

Program	C. S.	C. C.	Charts	Lists	M. H.	C. M.	C. A.	T.
STOMP	Yes	SMS	No	No	No	Yes	No	No
PMC	-	E-mail	Yes	Yes	Yes	Yes	No	No
txt2stop	Yes	SMS	No	No	No	Yes	No	Yes
S-PC	Yes	SMS+E	Yes	Yes	Yes	Yes	Yes	Yes

Table 1.1: Comparing S-PC with other similar programs (STOMP, PMC and txt2stop and S-PC). C. S. stands for Clinician support, C. C. stands for Communication channel, M. H. stands for Medical history, C. M. stands for Custom messages, C. A. stands for Custom alerts, T. stands for Templates

For the hypertension scenario, a comparison between 20 web sites used to manage and present home blood-pressure readings was carried out between June and August 2009 [45]. The results showed that none of these 20 web sites were directly linked to common electronic medical records. Despite web sites having alert values, none of them provided any tools for sending alert messages in any format to patients, i.e. to telemonitor them.

The H-PC tool is presented in Section 2.3. This allows patients to monitor their blood pressure at home and send the measurements telematically, using both mobile text messages and e-mails.

1.8 Publications

The following publications have been derived from the work on this thesis.

1.8.1 Journal publications

The following publications in research journals are derived from the work in this thesis:

1. Vilaplana, J., Solsona, F., Abella, F., Filgueira, R., & Rius, J. (2013). The cloud paradigm applied to e-Health. BMC Medical Informatics and Decision Making, 13, 35. doi:10.1186/1472-6947-13-35

2. Vilaplana, J., Solsona, F., Abella, F., Cuadrado, J., Alves, R., & Mateo, J. (2014). S-PC: An e-treatment application for management of smoke-quitting patients. Computer Methods and Programs in Biomedicine, 115(1), 3345. doi:10.1016/j.cmpb.2014.03.005

3. Vilaplana, J., Solsona, F., Teixidó, I., Mateo, J., Abella, F., & Rius, J. (2014). A queuing theory model for cloud computing. The Journal of Supercomputing, 69(1), 492507. doi:10.1007/s11227-014-1177-y

4. Vilaplana, J., Solsona, F., Teixidó, I., Usié, A., Karathia, H., Alves, R., & Mateo, J. (2014). Database Constraints Applied to Metabolic Pathway Reconstruction Tools. The Scientific World Journal, 2014. doi:10.1155/2014/967294

5. Vilaplana, J., Solsona, F., Abella, F., Cuadrado, J., Teixidó, I., Mateo, J., & Rius, J. (2014). H-PC: a cloud computing tool for supervising hypertensive patients. The Journal of Supercomputing, 71(2), 591612. doi:10.1007/s11227-014-1312-9

6. Vilaplana, J., Mateo, J., Teixidó, I., Solsona, F., Giné, F., & Roig, C. (2014). An SLA and power-saving scheduling consolidation strategy for shared and heterogeneous clouds. The Journal of Supercomputing. doi:10.1007/s11227-014-1351-2

1.8.2 Conference publications and attendance

The following publications in international conferences are derived from the work in this thesis:

1. Abdelli, O., Usié, A., Karathia, H., Vilaplana, J., Solsona, F. & Alves, R. (2011). Parallelizing Biblio-MetReS, a data mining tool. XXII Jornadas de Paralelismo JP2012. La Laguna (Tenerife), Spain.

2. Vilaplana, J., Solsona, F., Abella, F. & Celma, J. (2012). Diseño de un Sistema Cloud Aplicado a e-Health. XXIII Jornadas de Paralelismo JP2012. Elche (Alicante), Spain.

3. Vilaplana, J., Solsona, F., Teixidó, I., Mateo, J., Rius, J. & Abella, F. (2014). An SLA&Power Aware Strategy for a Cloud. International Conference on Information Technology and Management Engineering (ITME2014), Hong Kong.

4. Vilaplana, J., Mateo, J., Teixidó, I. & Solsona, F. (2014). A Green Job Scheduling Policy for Heterogeneous Clouds. 14th International Conference on Computational and Mathematical Methods in Science and Engineering (CMMSE 2014), Rota, Cádiz - Spain, Volume: 4.

5. Vilaplana, J. & Solsona, F. (2014). SLA-Aware Load Balancing in a Web-Based Cloud System over OpenStack. LLNCS vol. 8377:281-293. ICSOC 2013 Workshops. CCSA. Berlin, Germany. doi: 10.1007/978-3-319-06859-6_26.

6. Vilaplana, J., Solsona, F., Teixidó, I., Mateo, J., Rius, J. & Abella, F. (2014). A Green Scheduling Policy for Cloud Computing. First International Workshop, ARMS-CC 2014, held in Conjunction with ACM Symposium on Principles of Distributed Computing, PODC 2014, Paris, France, Volume: 8907.

7. Vilaplana, J., Solsona, F. & Teixidó, I. A performance model for scalable cloud computing. 13th Australasian Symposium on Parallel and Distributed Computing (AusPDC 2015), Sydney, Australia, Volume: 163.

1.8.3 Other publications

1. Abella, F., Vilasarau, A., Cuadrado, J., Solsona, F., Alves, R., Vilaplana, J. & Serra, J. (2011). Seguimiento y control de pacientes fumadores en proceso de deshabituacin mediante SMS. Una experiencia en e-salud. Revista Espaola de Drogodependencia, 39(4) 7787.

2. Abella, F., Vilasarau, A., Vilaplana, J., Josa, I., Cuadrado, J. & Solsona, F. (2011). Programa de Control de Pacientes Fumadores (CPF), INFO TABAC 21:1.

3. Mateo, J., Vilaplana, J., Plà, Ll. M., Lérida, J. Ll. & Solsona, F. (2014). A Green Strategy for Federated and Heterogeneous Clouds with Communicating Workloads. The Scientific World Journal [14]. doi: 10.1155/2014/273537.

4. Vilaplana, J., Solsona, F., Teixid, I., Mateo, J., Usié, A., Torres, N., Comas, J. & Alves, R. (2014). MetReS: a Metabolic Reconstruction Database for Cloud Computing. Cloud Computing Project and Initiatives (CCPI'14). Rome, Italy.

5. Mateo, J., Plà, Ll. M., Vilaplana, J., Solsona, F. & Lérida, J. Ll. (2015). Parallel Lagrangian Decomposition for large-scale two-stage stochastic mixed 0-1 problems. 15th International Conference Computational and Mathematical Methods in Science and Engineering (CMMSE 2015), Rota (Cádiz), Spain.

1.9 Three-month doctoral stay

During the course of the present thesis, I spent a three-month doctoral stay at the University of Ulster in Belfast, where I was involved in a cyber-bullying detection project in the Twitter social network. My supervisor was Prof. Yaxin Bi from the School of Computing and Mathematics.

My work consisted of developing a user-friendly interface for an existing sentiment analysis engine that could detect bullying behaviour in text messages and connect it to the Twitter network with the aim of identifying possible bullies within that social network.

The main objective of my work was to develop a client application that would access Twitter datasets through its APIs based on predefined search queries and hashtags. Once the data is retrieved, the client sends it to the previously-developed server application through its APIs. Then, the results are retrieved from the server in JSON format and presented to the user through a web-based dashboard.

Appendix A describes this work in full detail.

[14]The Scientific World Journal. A Green Strategy for Federated and Heterogeneous Clouds with Communicating Workloads. http://www.hindawi.com/journals/tswj/2014/273537/

Chapter 2

Methodology

2.1 Paper 1: A queuing theory model for cloud computing

Springer and the original publisher The Journal of Supercomputing, Vol 69, 2014, pp 492-507, A queuing theory model for cloud computing, Jordi Vilaplana, Francesc Solsona, Ivan Teixidó, Jordi Mateo, Francesc Abella, Josep Rius, with kind permission from Springer Science and Business Media.

2.2 Paper 2: An SLA and power-saving scheduling consolidation strategy for shared and heterogeneous clouds

Springer and the original publisher The Journal of Supercomputing, Vol 71, 2014, pp 1817-1832, An SLA and power-saving scheduling consolidation strategy for shared and heterogeneous clouds, Jordi Vilaplana, Jordi Mateo, Ivan Teixidó, Francesc Solsona, Francesc Giné, Concepció Roig, with kind permission from Springer Science and Business Media.

2.3 Paper 3: H-PC: a cloud computing tool for supervising hypertensive patients

Springer and the original publisher The Journal of Supercomputing, Vol 71, 2014, pp 591-612, H-PC: a cloud computing tool for supervising hypertensive patients, Jordi Vilaplana, Francesc Solsona, Francesc Abella, Josep Cuadrado, Ivan Teixidó, Jordi Mateo, Josep Rius, with kind permission from Springer Science and Business Media.

2.4 Paper 4: S-PC: An e-treatment application for management of smoke-quitting patients

Reprinted from Computer Methods and Programs in Biomedicine, Vol 115, Jordi Vilaplana, Francesc Solsona, Francesc Abella, Josep Cuadrado, Rui Alves, Jordi Mateo, S-PC: An e-treatment application for management of smoke-quitting patients, Pages No. 33-45, Copyright (2014), with permission from Elsevier.

Chapter 3

Papers

J Supercomput (2014) 69:492–507
DOI 10.1007/s11227-014-1177-y

A queuing theory model for cloud computing

Jordi Vilaplana · Francesc Solsona · Ivan Teixidó · Jordi Mateo · Francesc Abella · Josep Rius

Published online: 9 April 2014

Abstract The ability to deliver guaranteed QoS (Quality of Service) is crucial for the commercial success of cloud platforms. This paper presents a model based on queuing theory to study computer service QoS in cloud computing. Cloud platforms are modeled with an open Jackson network that can be used to determine and measure the QoS guarantees the cloud can offer regarding the response time. The analysis can be performed according to different parameters, such as the arrival rate of customer services and the number and service rate of processing servers, among others. Detailed results for the model are presented. When scaling the system and depending on the types of bottleneck in the system, we show how our model can provide us with the best option to guarantee QoS. The results obtained confirm the usefulness of the model presented for designing real cloud computing systems.

J. Vilaplana · F. Solsona ( ) · I. Teixidó · J. Mateo
Department of Computer Science, University of Lleida, Jaume II 69, 25001 Lleida, Spain
e-mail: francesc@diei.udl.cat

J. Vilaplana
e-mail: jordi@diei.udl.cat

I. Teixidó
e-mail: iteixido@diei.udl.cat

J. Mateo
e-mail: jmateo@diei.udl.cat

F. Abella
IRB Lleida, Avda Alcalde Rovira Roure 80, 25198 Lleida, Spain
e-mail: abella@gss.scs.es

J. Rius
ICG Software, Pol. Industrial Torrefarrera. Mestral, s/n, Torrefarrera, 25123 Lleida, Spain
e-mail: jrius@icg.es

Keywords Cloud computing · Cloud architecture · Scalability · Queuing theory · Quality of Service · Simulation · Validation

1 Introduction

Cloud computing aims to shift the location of the computing infrastructure to Internet to reduce the costs of management and maintenance of hardware and software resources [1]. Cloud computing is a new cost-efficient computing paradigm in which information and computer power can be accessed by the customers through a web browser [2]. Cloud service providers offer high performance, scalability, security and high availability [3]. However, performance issues lead to the question of how to guarantee that the system can offer QoS (Quality of Service). Cloud computing has received worldwide attention from many researchers, but only a small portion of these have addressed the performance problem [4].

This article presents the design of a cloud platform with QoS guarantees based on response time for services. Response time is defined as the time for a request to be serviced, in other words, the sum of the waiting and servicing times in the cloud.

Cloud computing systems offer the idea of infinite computing resources available on demand, allowing the resources to be expanded as needed. Hardware and software services are more efficiently handled than in other high performance computing (HPC) infrastructure as they can be added and released dynamically [5]. However, problems arise when scaling the system, for example, when trying to deploy a platform to support the computing needs of many institutions, organisms or companies with different departments with their own staff and customers. The need for scalability increases even more when the cloud must also provide support for other common and specific desktop or Internet applications. Consequently, the model presented can be directly applied to almost all public and commercial areas.

As stated in [2], most current cloud computing infrastructures consist of services that are offered and delivered through a service center, such as a data center, that can be accessed from a web browser anywhere in the world. The two most significant components of a cloud computing architecture are the front-end and the back-end. The front-end is the gateway to the cloud and consists of the software components and interfaces needed to connect to the platform using remote client applications. These applications usually use standard web protocols to access the system and an authentication protocol, which allows access to authorized users. The back-end functions include management of the job queue, the servers and their virtual machines and the storage servers with their database system. Database inconsistencies are avoided by considering only one storage (i.e., database) server.

The main contribution of this paper consists of a computer service QoS model for the cloud architecture of the back-end (or simply called a cloud) made up of processing servers and a data service that, in line with the explanation above, consists of only one database server (see Fig. 1). In this model, the cloud is a single access point for the computing needs of the customers being served [2] through a web browser supported by a web server. The service center is a collection of service resources used by a

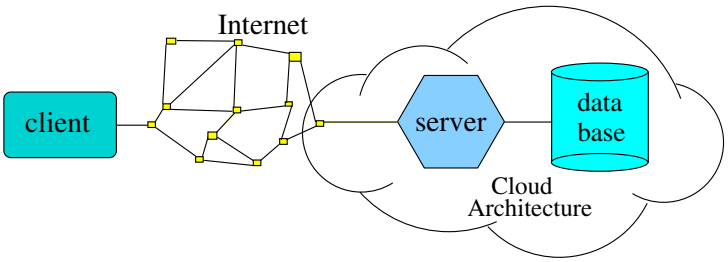


Fig. 1 Cloud computing paradigm

service provider to host service applications for customers. A service request from a user is transmitted to the web server running a service application [6], associated with an SLA (Service Layer Agreement). This process is performed at the front-end. The SLA is a contract negotiated and agreed between a customer and a service provider so a customer only pays for the resources and services used. Thus, the customer, who may represent multiple users, generates service requests at a given rate for processing at the service center hosted by the service provider through the cloud according to the negotiated QoS requirements at a given price.

Our proposal is modeled using queuing theory to identify and manage the users' response time for services. The model allows the cloud system to be scaled optimally to guarantee the QoS for the response time, and planning the proper deployment and removal of virtual machines (logical/virtual servers making up the cloud architecture) according to the system load. To guarantee the negotiated SLA, the model must be capable of determining the required number of virtual machines to reach the QoS (i.e., the response time of the applications entering the system). To comply with that, the cloud system should be able to add/remove virtual machines dynamically according to the outputs obtained by simulation. Those results can be obtained on- or off-line (in other words, respectively, on-time or in advance), depending on the speed of obtaining them. An interesting future work could be to try to solve this. Even though simulated predictions based on the model are not in real scale, they must be very reliable and, depending on the number of servers and the arrival speed, these could be very useful for generating an accurate approximation of the task response times to avoid exceeding the SLA.

In doing so, the cloud architecture is modeled with an open Jackson network [7,8] of *M/M/m* and *M/M/1* interconnected servers. There are two types of server, namely processing and data servers. We are interested in modeling QoS performance by scaling cloud platforms, leaving aside other issues such as cloud availability [9], energy consumption [10], variability [11] or reliability [12].

The remainder of the paper is organized as follows. Section 2 details the related work on queuing system theory and other alternative methods to address the problem of providing QoS and scalability in cloud computing. In Sect. 3, we present our modeling proposal to design cloud computing systems with open Jackson networks to guarantee QoS based on response times for customer services. Experimentation showing the good behavior of our proposal is presented in Sect. 4. Finally, Sect. 5 outlines the main conclusions and future work.

2 Related work

The problem of computer service performance modeling subjected to such QoS metrics as response time, throughput and network utilization, has been extensively studied in the literature [2,13–17]. For instance, in [16], Karlapudi proposed and validated a web application performance tool for the performance prediction of web applications between specified end-points. In [17], Mei addressed the problem of an end-to-end QoS guarantee for VoIP services.

In [2], the authors obtained the response time distribution of a cloud system modeled on a classic *M/M/m* open network, assuming an exponential density function for the inter-arrival and service times. Using the response time distribution, they determined the optimum level of service and the relationship between the maximum number of tasks and the minimum number of resources (virtual machines). The response time takes into account both the waiting time in the queue and the service time. For a given service resource, the authors obtained the level of QoS services that can be guaranteed in terms of response time.

In [14], the authors obtained the response time distribution for a cloud with an *M/M/m/m+r* system model. Both inter-arrival and service distribution times were assumed to be exponential and the system had a finite number of *m+r* size buffers. The complexity of other queues (*G/M/m*, *M/G/m*, *G/G/m*) comes from the impossibility of obtaining a closed formula to represent the probability distributions of the response or waiting times of customers in the queue, and therefore requires finding approximate models. However, the results mentioned above are not directly applicable to performance analysis of cloud computing server farms, where one or more of the following holds [4]: the number of servers is huge (in general, these models are reasonably accurate when there are few servers, typically below 10 or so); the distribution of service times is unknown and does not, in general, follow any of the "well-behaved" probability distributions, such as the exponential distribution. Finally, the traffic intensity is small or can vary over an extremely wide range (that means the covariance of the service time is large [18]). In a previous work [19], our group designed a first attempt to simulate an e-health cloud system, modeled simply by connecting two M/M/m queues. Although the main goal was to provide QoS capabilities based on the waiting time of the requests, the study lacked more experimentation and an accurate model validation. In the present work, we expand the e-health model developed in [19], presenting a more general and better-validated cloud model.

The nodes forming the cloud architecture can be analyzed independently when they make up an open Jackson network. The interconnection and behavior between the queues are ruled by Burke's [20] and Jackson's theorems [7,8]. Burke states that we may connect many multiple-server nodes together in a feedforward network and still preserve the node-by-node decomposition when arrival and servicing times are modeled by exponential density functions. In addition, Jackson pointed out that to calculate the total average arrival rate we must sum the arrivals from outside the system plus arrivals from all internal nodes. As a result, we can join different processing nodes to design the cloud architecture according to the response time as the QoS performance metrics. As the assumption that there are only exponential distributions for the arrival and servicing rates is a restriction, we only consider *M/M/1* and *M/M/m* nodes.

The authors in [5] showed how multiple virtual machines (VMs) can share CPUs and main memory surprisingly well in cloud computing, but that network and filesystem sharing is more problematic. They obtained a mean bandwidth of 75 instances of the same memory benchmark of 1,355 MB/s, with a standard deviation of just 52 MB/s (approx. 4 % of the mean). In an analogous experiment, they also obtained a mean disk-write bandwidth of nearly 55MB/s with a standard deviation across instances of a little over 9MB/s (approx. 16 % of the mean). That demonstrated the problem of I/O interference between virtual machines. Therefore, in the design of the architecture of our web-based application, we consider the distinction between processing and data servers.

In [21], Ming presented an integer programming mechanism to scale computing instances on a cloud based on workload information and performance desire automatically. It was based on activity scheduling by starting and shutting down VM instances. The mechanism enabled cloud applications to finish submitted jobs within the deadline by controlling underlying instance numbers and reducing user cost by choosing appropriate instance types.

In [13], Slothouber stated that a single queue is insufficient to model a complex system such as a web server. He defined the response time of the system treating the model as a Jackson network. He studied which parameters influenced the response time when the web server and network were the bottleneck. He was able to determine, for example, that in the first case, the best alternative was to increase the network bandwidth, and in the second, it was to double the server speed. Nah [22] stated that for users, the tolerable waiting time before abandoning the downloading of a Web page can vary under different circumstances and contexts. However, the findings from this study suggested that most users are only willing to wait for about two seconds for simple information retrieval tasks on the Web. So, if we apply this result to cloud computing, a reasonable design decision should be to design clouds with good and predictable response times as a valid QoS parameter.

We go further than the model presented in [13] by modeling a cloud architecture instead of a web server. We also model the system as an open Jackson network. As introduced above, the operation of a cloud system is very similar to a web server one, so we expand Slothouber's original idea [13] to model a cloud architecture. As in [2], we are interested in defining what level of QoS (i.e., response time) can be guaranteed for any given service. The model must also be useful as a guide for the creation/deletion of VMs, as in [21], or for studying the cause of the bottlenecks and providing solutions, as in [13,23].

3 Model

We propose a multi-server system with the queuing model (see Fig. 1, representing an Open Jackson network.

This network has a single entry point ES (Entering Server). The server acts as a load balancer, which forwards the user requests to one of the PS_i, where $i = 1 \ldots m$, namely the Processing Server nodes. The load balancer is represented by a *M/M/1* queue, with an arrival and service rate modeled on an exponential pdf with parameters

λ and L, respectively, where $\lambda < L$. Its purpose is to determine the PS_i node to deliver the service. In doing so, it uses an algorithm that distributes the requests depending on the averaged workload in each PS_i node.

A processing server PS_i is a node, core or processor representing the physical computational resources of our cloud architecture where the services are computed. The selected PS_i node performs all the services required by the user request. The PS_i is identical and is modeled as an *M/M/m* queueing system. Each PS_i node has the same service rate and is equal to μ, this is $\mu = \mu_i, i = 1 \ldots m$.

Each PS_i node accesses DS with a probability δ. *DS* represents a database server and serves to model the access to files, directories and databases or any kind of I/O access to secondary memory during the service in the cloud architecture. As stated in the introduction, this is an important consideration to be taken into account, because I/O interference between virtual machines is an important issue. In addition, data inconsistencies are avoided by choosing only one database server in our design. DS is modeled by an *M/M/1* queue with respective exponential arrival and service rates of $\delta\gamma$ (according to the rules of the open Jackson network) and D, respectively.

OS represents the Output Server of the cloud architecture. OS is the node forming the cloud architecture that transmits the response data over the Internet, back to the client that made the original request (CS).

CS is the Client Server. It sends requests in an exponential distribution with parameter λ to the entering server ES. It also receives the responses from the cloud architecture. CS receives files or pieces of files, until the request is fully satisfied. Both OS and CS are also modeled by an *M/M/1* queue. γ is the arrival rate of the exponential pdf to both nodes. However, like the other network nodes above, they can have different service rates.

Connecting servers with exponential arrival and service distributions in a feedforward (without feedback paths) are independent from each other and preserve the pdf distributions [20]. So λ is the feeding distribution for the customers leaving the entering server (ES). In addition, Jackson [7,8] stated that, to calculate the total average arrival rate, we must sum the arrivals from outside the system and the arrivals from all the internal nodes. By applying Jackson, we can obtain γ. Let τ be the output probability of leaving the open Jackson network, according to λ must be conserved (as Jackson stated in [7,8]), $\gamma = \lambda/(1 - \tau)$.

The response time (T) of the global cloud architecture, as a result of being considered as an open Jackson network, is the following:

$$T = T_{ES} + T_{PS} + T_{DS} + T_{OS} + T_{CS} \tag{1}$$

Then, each term of Eq. 1 is explained separately.

3.1 Obtaining T_{ES}

T_{ES} represents the response time of the Entering Server (ES) which acts as a load balancer. The ES node is modeled as an $M/M/1$ queue. Thus, the formula for obtaining

the response time for an $M/M/1$ queue is [24]:

$$T_{\mathrm{ES}} = \frac{1/L}{1-\lambda/L}, \tag{2}$$

where λ is the arrival rate (see Fig. 2), and L the service rate of the node ES.

3.2 Obtaining T_{PS}

T_{PS} represents the response time of the Process Servicing nodes that actually process the user requests. There is a maximum of m PS nodes, where m is determined by the physical computing resources of the cloud architecture. These nodes are modeled as an $M/M/m$ queue. According to [25], the response time of such a queue is defined as:

$$T_{\mathrm{PS}} = \frac{1}{\mu} + \frac{C(m,\rho)}{m\mu-\gamma}, \tag{3}$$

where m is the number of processing elements, and γ and $\mu = \mu_i, i = 1 \ldots m$, are, respectively, the arrival and service rates of each processing element, respectively (see Fig. 2). The term $C(m,\rho)$ represents Erlang's C formula, which gives the probability of a new client joining the *M/M/m* queue. Erlang's C formula is defined as [24]:

$$C\,(m,\rho) = \frac{\left(\frac{(mp)^m}{m!}\right)\left(\frac{1}{1-\rho}\right)}{\sum_{k=0}^{m-1}\frac{(mp)^k}{k!} + \left(\frac{(m\rho)^m}{m!}\right)\left(\frac{1}{1-\rho}\right)}, \tag{4}$$

where $\rho = \gamma/\mu$.

3.3 Obtaining T_{DS}

T_{DS} represents the response time of the Database Server. Requests are sent to the DS node with a probability δ. The DS node is modeled as a $M/M/1$ queue, and so (see Fig. 2):

$$T_{\mathrm{DS}} = \frac{1/D}{1-\delta\gamma/D}, \tag{5}$$

where $\delta\gamma$ is the arrival rate to DS (see Fig. 2), and D is the service rate. Eq. 5 is obtained from the mean response time formula of a $M/M/1$ queue in a similar way as T_{ES} was obtained in Eq. 2.

Note that the arrival rate at the Output Server (OS) is the sum of the arrival rates of the two crosspoint branches entering the summatory represented in Fig. 2. The one crossing the data server is the same than at its input ($\delta\gamma$). On the other branch, the γ term must be changed by its complement to one.

$$(1-\delta)\gamma + \delta\gamma = \gamma \tag{6}$$

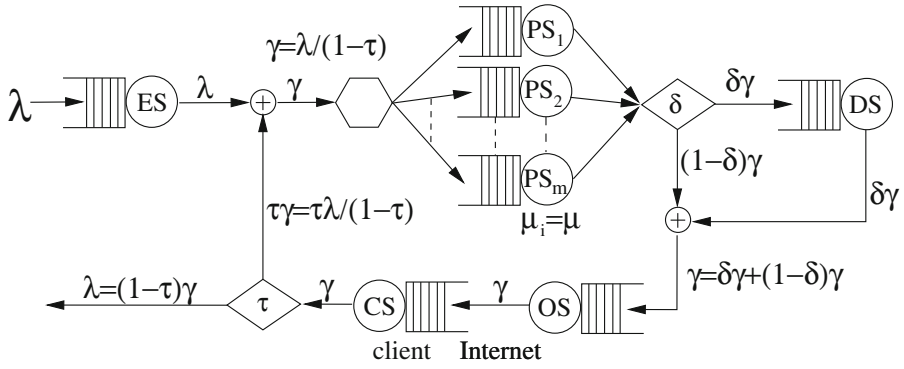


Fig. 2 Model of the cloud architecture

3.4 Obtaining T_{OS}

T_{OS} represents the response time of the Output Server (OS), the one that transmits data (i.e., files) back to the client. We also supposed that its operation is also modeled as a $M/M/1$ queue (as in Eq. 2). The service rate of this node is defined as O/F, where O is the average bandwidth speed (in bytes per second) of the OS, and F is the averaged size of the data responses of the system. Its response time (Eq. 7) is defined by the following formula:

$$\begin{aligned} T_{OS} &= \frac{F/O}{1-\gamma/(O/F)} \\ T_{OS} &= \frac{F}{O-\gamma F}, \end{aligned} \tag{7}$$

In the present work, the number of responses (i.e., response pdf) is not taken into account because we have insufficient information in this sense. More in-depth research into this is required. Furthermore, the performance of cloud systems should be adapted to the needs of the field (area) they are applied to.

3.5 Obtaining T_{CS}

Finally, T_{CS} is the response time of the Client Server (CS), which receives the data sent by the OS node through Internet, and then operates as an $M/M/1$ queue. The service rate in this case is defined as C/F, where C is the average bandwidth speed of the client server in bytes per second. As when obtaining T_{OS}, F is the average size in bytes of the received reply files. Accordingly, the response time is defined in this case as (see Fig. 2):

$$\begin{aligned} T_{CS} &= \frac{F/C}{1-\gamma/(C/F)} \\ T_{CS} &= \frac{F}{C-\gamma F}, \end{aligned} \tag{8}$$

4 Results

The following section presents an analysis of how the response time is affected by modifying some of the metrics presented in the model. Our purpose is to verify whether our model behaves as expected when a range of parameters and system configurations are tested.

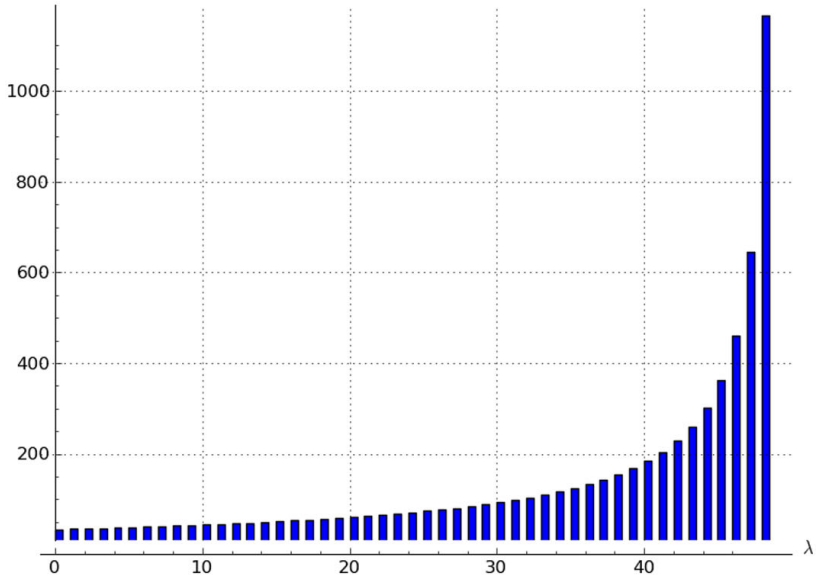
First, we give the results obtained in the simulation. Then, we validate the model by comparing the simulated results with those obtained in a real cloud system.

4.1 Simulation

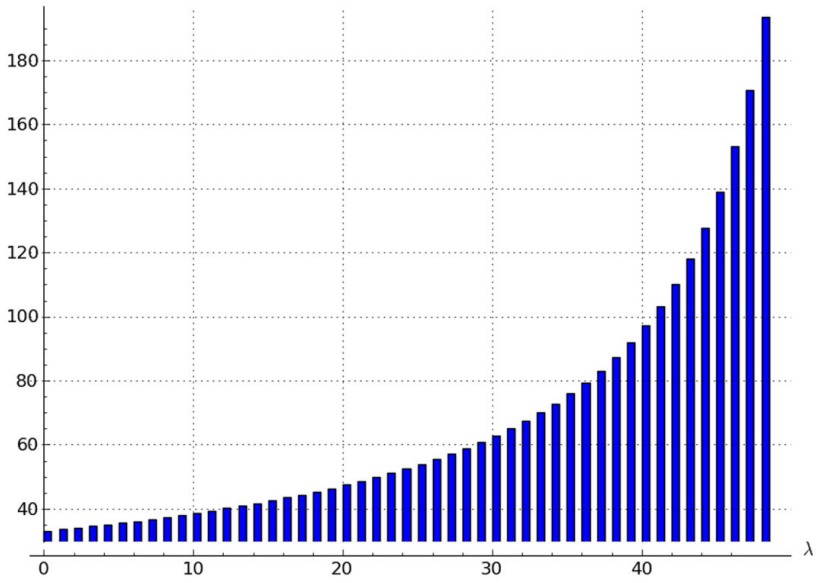
The model was implemented using Sage 5.3 mathematical software[1]. All the results (y-axis) are in simulation units of time, so no Y-labels were placed in the figures. The parameters used in the implementation are described below:

λ Arrival rate. This is the average number of requests reaching the system per unit of time. $1/\lambda$ is the mean inter-arrival time. We aimed to show how the total response time (T) is affected by varying the parameter λ. The number of processing servers (m) represents the total number of processors, cores or nodes dedicated to servicing requests. Changing this parameter shows the impact of adding or removing servers from the system.
F Average file size. This is the mean size of the files that are sent to clients via Internet as the service response of the overall system. This value depends on the web application that runs on the system, although it should be no greater than 1MB in most cases.
O Server bandwidth. This is the network speed at which files are sent to clients over the Internet.
C Client bandwidth. This is the average connection speed of the clients receiving the data sent from the system. This parameter will usually be outwith our control.
δ Database access probability. This is the probability that an incoming request needs to access the database node DS. As we are modeling a web-based system, not all requests will require access to the database server. Note, however, that this probability will usually be relatively high.
μ Service rate. This describes the speed at which the web servers handle the requests. $1/\mu$ is the mean service time. To simplify the results we have chosen the same value for all PS_i servers. The total service rate of the system must always be greater than λ for the system to be stable. This value was modified in the tests to check the performance impact. Although the service rates of ES, PS_i, $i = 1 \ldots m$, DS, OS and CS could be different, we assumed the same value to simplify the experiments. So, in this case, $\mu = L = D = O/F = C/F$ in all the tests.

[1] Sage. http://www.sagemath.org.



(a) $m = 1$.



(b) $m = 2$.

Fig. 3 Total response time (T) of the cloud model in function of λ

4.1.1 Response time

Figure 3 shows how the total response time (T) is affected by increasing the arrival rate (λ) for one (Fig. 3a) and two (Fig. 3b) servers.

Increasing low values of the number of servers has a great impact on lowering the response time. However, the response time stabilizes quickly when the number of servers increases. We found that no significant differences were obtained when the number of servers was higher than 5. We may find the explanation for this phenomenon in the utilization of the PS_i nodes, defined as $\rho = \gamma/\mu$.

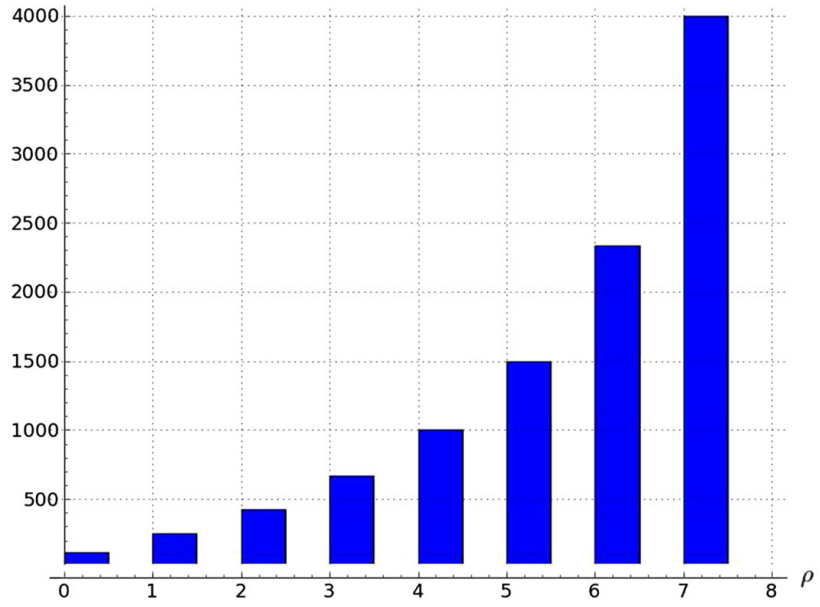


Fig. 4 Total response time (T) of the cloud model in function of ρ

From Fig. 3, we can derive that the reason the response time stabilizes so fast is because in Fig. 3a, when the entering arrival rate for requests to the system (λ) is high, also is $\gamma = \lambda/(1-\tau)$), the arrival rate at PS. By adding servers to the PS_i nodes, the utilization rate of the $M/M/m$ queue decreases and the response time stabilizes to the same value as service time. For this case, by adding more PS_i nodes, almost no improvement was observed in the response time. Thus, no further results are shown. There is an upper soft bound above which adding more PS_i nodes, hardly decrease the system's total response time (T). In this case, this soft bound was 2.

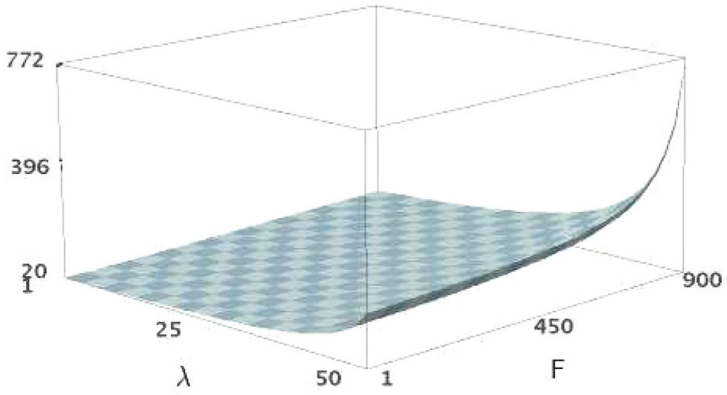
Figure 4 illustrates how the response time of the PS node (T_{PS}) lengthens exponentially as the utilization or traffic intensity (ρ) of the server increases. When the utilization is close to zero, the response time is almost equal to the service time. The response time increases with the utilization until it approaches one. At that moment, the response time grows rapidly towards infinity. We observe the same behavior as in Fig. 3.

Figure 5 shows the Total Response Time of the system when the arrival rate (λ), mean file size (F) and the output server bandwidth (O) are modified.

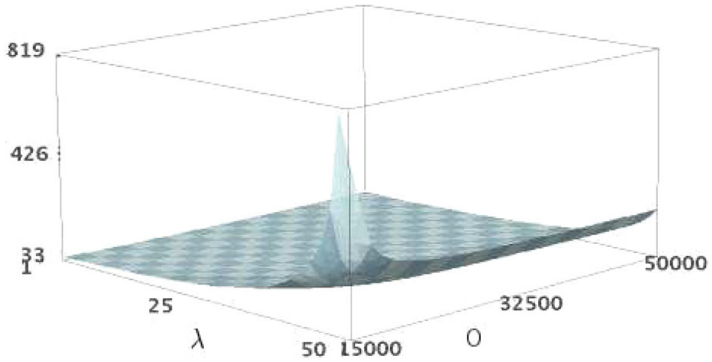
Total response time (T) grows exponentially when the mean file size (F) is increased or when the server bandwidth (O) is decreased. This behavior was expected. When files are too large, the output server OS needs more bandwidth to transfer data to clients across the network (i.e., Internet) efficiently. Similarly, when server bandwidth decreases, files are transferred at a much lower speed and consequently the total response time (T) of the system also increases.

4.1.2 Bottlenecks

Figures 6 and 7 present two different bottleneck studies, also by measuring the total response time (T).



(a) T given λ and F.



(b) T given λ and O.

Fig. 5 Total response time (T) of the cloud model in function of λ, F and O

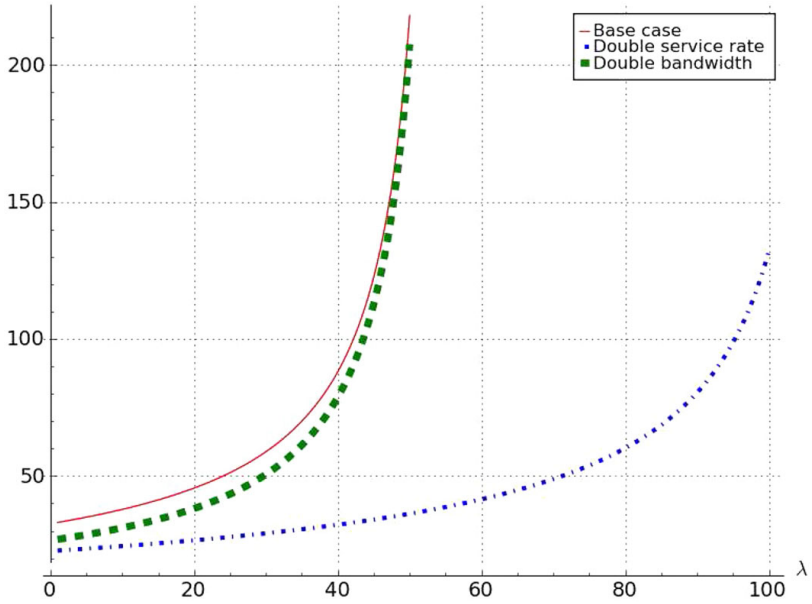


Fig. 6 Total response time (T). Servers are the bottleneck

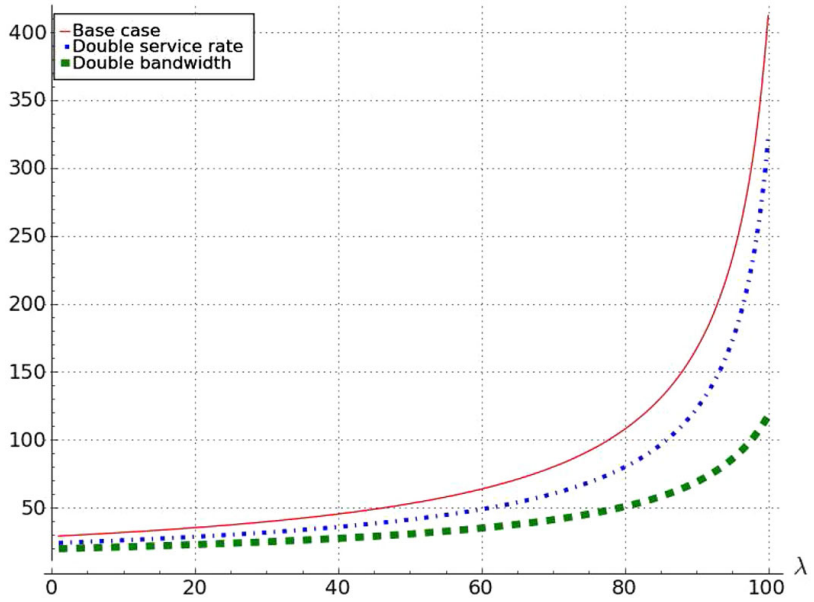


Fig. 7 Total response time (T). Bandwidth is the bottleneck

The figures show three different situations. The first is the base case, which presents a system where the servers are the bottleneck in the cloud system. The other two cases represent the system behavior (T) when the service rate and server bandwidth are doubled.

In Figure 6, we see that the best alternative is to double the service rate of the server. This increased the performance enormously in all the λ axis. From this result, we can conclude that the servers are really the current bottleneck, which means that the performance of our system is limited by the service capacity of our servers.

In Fig. 7, we adjusted our system so that our base case represented a situation where the bandwidth was the bottleneck. As expected, the best alternative was to double the server bandwidth, which significantly increased the performance from 25 to 360 %.

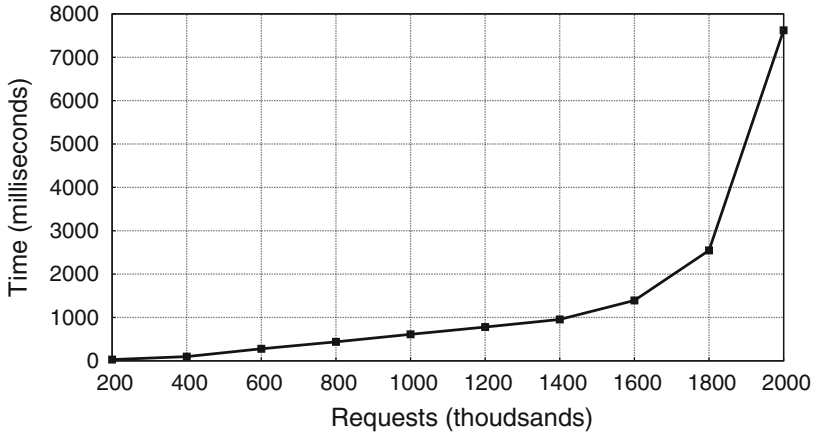
From the given tests, we can confirm that our model represents a cloud accurately. The response time of the system is consistent with extreme situations. We showed that, depending on the system configuration, some parameters become more important or can be rather insignificant for the overall system performance.

4.2 Model validation

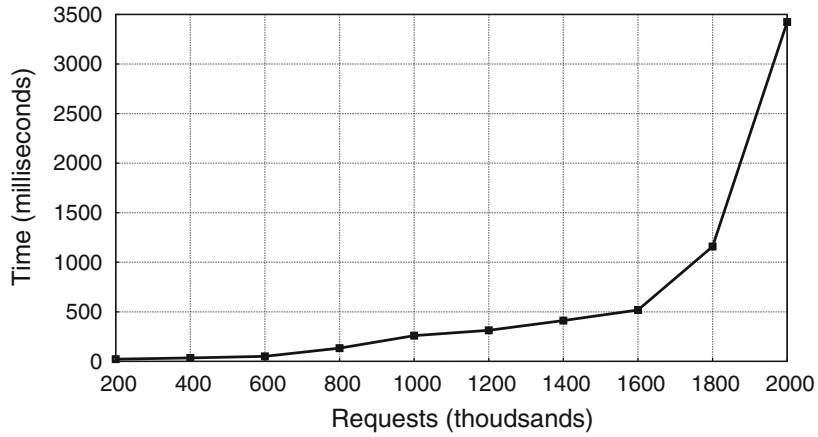
To validate the usefulness of this model in predicting the cloud behavior, we compared the above simulation results with the results from the experimentation in a real environment system.

The proposed queuing theory model was implemented using an open-source cloud platform called *OpenStack*[2]. *OpenStack* goes beyond a classic hypervisor (i.e., *Virtu-*

[2] *OpenStack*. http://www.openstack.org.



(a) Response time (T) with 1 server.



(b) Response time (T) with 2 servers.

Fig. 8 Response time obtained in a real OpenStack cloud system

alBox[3], *Xen*[4] and *VMware*[5]), which allows the creation and setup of virtual machines dynamically, as computational resources are needed. This enables the QoS restrictions in traffic peaks to be handled, in other words, when the arrival rate of the requests to be served increases. OpenStack can be set up to create new instances automatically when current servers are overwhelmed and to shut them down when traffic decreases. The architecture designed for this test was made up of different servers created as virtual machines (VMs) containing the entering server (ES), four servicing nodes, PS_1 ... PS_4, and a *MySQL*[6] database, DS, and the output and client servers. Each virtual machine had 4 GB RAM and two cores of an AMD Opteron 6,100 processor running at 2.1 GHz.

Figure 8 shows the average response time (in milliseconds) of the system when benchmark applications were executed using one and two servicing nodes. To simulate

[3] *VirtualBox*. https://www.virtualbox.org.

[4] *Xen*. http://www.xenproject.org.

[5] *VMware*. http://www.vmware.com.

[6] *MySQL*. http://www.mysql.com.

the incoming requests, the *Apache JMeter* [26] tool was used. This tool allows the creation of accurate and controllable testing workloads of web requests to be performed by sending them to an HTTP queuing server. VMs always perform the same task: when idle, they pick up a new task from the HTTP queuing server, if there is any, and process it. If not, they wait until a task becomes available. This way, specific parametrized performance tests can be automatically done, thus simulating a huge incoming load.

As in the simulation, the most significant gain was obtained by passing from one to two VMs, namely servicing nodes in the Model and Simulating sections (Sects. 3 and 4.1, respectively). Hardly appreciated gains were obtained when adding one additional server, and no more significant gains were achieved from three onwards. Then, no more figures with the results for more than two servicing nodes are shown. It can be seen that the shapes of the simulation and real results are very similar. It should be very difficult to give absolute times in the simulation case. However, the similarities in shapes between the simulation and real results prove the good behavior of the model. The real validation of the model adds additional value to this paper because it proves its usefulness.

5 Conclusions and future work

In this paper, a model for designing cloud computing architectures with QoS is presented. Queuing theory and the open Jackson's networks were selected as the basic means to guarantee a certain level of performance in line with the waiting and response times of such networks.

Through a preliminary analysis, the design of a cloud architecture with QoS requirements was proposed. The combination of $M/M/1$ and $M/M/m$ in sequence was proposed to model the cloud platform. Our work shows that to provide good QoS in terms of response time, we have to determine where the system has a bottleneck and then improve the corresponding parameter. We can then conclude that our model can be very useful for tuning service performance, i.e., QoS [response time (T)], thus guaranteeing the SLA contract between the client and the service provider.

As future work, we plan to investigate other issues related to cloud computing, such as user variability and the reliability of the cloud platform. The final purpose is to enable our design to satisfy the servicing needs of many areas (education, administration, e-health, etc.) easily, allowing the construction of web-based applications that implement all the needed workflows.

Acknowledgments This work was supported by the MEyC under contract TIN2011-28689-C02-02. Some of the authors are members of the research group 2009 SGR145, funded by the Generalitat de Catalunya.

References

1. Vaquero LM, Rodero-Merino L, Caceres J, Lindner M (2008) A break in the clouds: towards a cloud definition. ACM SIGCOMM Comput Commun Rev 39:50–55
2. Xiong K, Perros H (2009) Service performance and analysis in cloud computing. In: Proceedings of IEEE World Conference Services, pp 693–700

3. Varia J (2010) Architection for the cloud: best practices. Amazon Web Services
4. Khazaei H, Misic J, Misic V (2012) Performance analysis of cloud computing centers using M/G/m/m+r.Queuing Systems. IEEE transactions on parallel and distributed systems, vol 23, no 5
5. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M (2010) A view of cloud computing. Commun ACM 53(4):50–58
6. Martin J, Nilsson A (2002) On service level agreements for IP networks. In: Proceedings of the IEEE INFOCOM
7. Jackson JR (1957) Networks of waiting lines. Oper Res 5:518–521
8. Jackson JR (1963) Jobshop-like queueing systems. Manage Sci 10:131–142
9. Martinello M, Kaâniche M, Kanoun K (2005) Web service availability: impact of error recovery and traffic model. J Reliab Eng Syst Saf 89(1):6–16
10. Beloglazov A, Buyya R (2012) Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. Concurr Comput Pract Exp 24(13):1397–1420
11. Iosup A, Yigitbasi N, Epema D (2011) On the performance variability of production cloud services. 11th IEEE/ACM international symposium on cluster, cloud and grid, computing (CCGrid'2011), pp 104–113
12. Vishwanath KV, Nagappan N (2010) Characterizing cloud computing hardware reliability. In: Proceedings of the 1st ACM symposium on Cloud computing (SoCC '10), pp 193–204
13. Slothouber L (1996) A model of web server performance. In: Proceedings of the fifth international world wide web conference
14. Yang B, Tan F, Dai Y, Guo S (2009) Performance Evaluation of cloud service considering fault recovery. In: Proceedings of the first international conference on cloud, computing (CloudCom'09), pp 571–576
15. Ma N, Mark J (1998) Approximation of the mean queue length of an M/G/c queueing system. Oper Res 43:158–165
16. Karlapudi H, Martin J (2004) Web application performance prediction. In: Proceedings of the IASTED international conference on communication and computer networks, pp 281–286
17. Mei RD, Meeuwissen HB (2005) Modelling end-to-end Quality-of-Service for transaction-based services in multidomain environement. In: Proceedings of the 19th international teletraffic congress (ITC19), pp 1109–1121
18. Boxma OJ, Cohen JW, Huffel N (1979) Approximations of the Mean waiting time in an M=G=s queueing system. Oper Res 27:1115–1127
19. Vilaplana J, Solsona F, Abella F, Filgueira R, Rius J (2013) The cloud paradigm applied to e-health. BMC Med Inf Decis Making 13:35
20. Burke PJ (1956) The output of a queuing system. Oper Res 4:699–704
21. Mao M, Li J, Humphrey M (2010) Cloud auto-scaling with deadline and budget constraints. In: Proceedings of the 11th IEEE/ACM international conference on GRID, pp 41–48
22. Nah F (2004) A study on tolerable waiting time: how long are Web users willing to wait? Behav Inf Technol 23(3):153–163
23. Sai Sowjanya T, Praveen D, Satish K, Rahiman A (2011) The queueing theory in cloud computing to reduce the waiting time. IJCSET, vol 1, no 3, pp 110–112
24. Kleinrock L (1975) Queueing systems: theory, vol 1. Wiley-Interscience, New York
25. Barbeau M, Kranakis E (2007) Principles of ad-hoc networking. Wiley, New York
26. Apache JMeter website. http://jmeter.apache.org/. Accessed 10 March 2014

The Journal of Supercomputing
DOI 10.1007/s11227-014-1351-2

An SLA and power-saving scheduling consolidation strategy for shared and heterogeneous clouds

Jordi Vilaplana · Jordi Mateo · Ivan Teixidó · Francesc Solsona · Francesc Giné · Concepció Roig

Published online: 29 November 2014

Abstract This paper presents a power-aware scheduling policy algorithm of Virtual Machines into nodes called Green Cloud (GreenC) for Heterogeneous cloud systems.

GreenC takes into account optimal assignments according to physical and virtual machine heterogeneity, the current host workload and communication between the different virtual machines.

An initial test case has been performed by modelling the policies to be executed by a solver that demonstrates the applicability of our proposal for saving energy and also guaranteeing the QoS.

The proposed policy has been implemented using the *OpenStack* software and the obtained results showed that energy consumption can be significantly lowered by applying GreenC to allocate virtual machines to physical hosts.

J. Vilaplana
Department of Computer Science and INSPIRES, University of Lleida, Av. Jaume II 69, 25001 Lleida, Spain. Tel.: +34-973-702718. E-mail: jordi@diei.udl.cat

J. Mateo
Department of Computer Science and INSPIRES, University of Lleida, Av. Jaume II 69, 25001 Lleida, Spain. Tel.: +34-973-702730. E-mail: jmateo@diei.udl.cat

I. Teixidó
Department of Computer Science and INSPIRES, University of Lleida, Jaume II 69, Lleida, Spain 25001. Tel.: +34-973-702711. E-mail: iteixido@diei.udl.cat

F. Solsona
Department of Computer Science and INSPIRES, University of Lleida, Av. Jaume II 69, 25001 Lleida, Spain. Tel.: +34-973-702735. E-mail: francesc@diei.udl.cat

F. Giné
Department of Computer Science and INSPIRES, University of Lleida, Av. Jaume II 69, 25001 Lleida, Spain. Tel.: +34-973-702748. E-mail: sisco@diei.udl.cat

C. Roig
Department of Computer Science and INSPIRES, University of Lleida, Av. Jaume II 69, 25001 Lleida, Spain. Tel.: +34-973-702733. E-mail: roig@diei.udl.cat

Keywords Green cloud computing · SLA · power-aware scheduling · non-linear programming

1 Introduction

In cloud computing, an SLA (Service-Level Agreement) is an agreement between a service provider and a consumer where the former agrees to deliver a service to the latter under specific terms, such as time or performance. In order to comply with the SLA, the service provider must monitor the QoS (Quality of Service) closely through such performance metrics as response or waiting time, throughput or makespan [6]. Studying and determining SLA-related issues is a big challenge [2,5].

GreenC is designed to lower power consumption [16] as much as possible. At the same time, GreenC is aimed at guaranteeing a negotiated SLA and power-aware [12] solutions, leaving aside such other cloud-computing issues as variability [5], system security [16] and availability [10]. Job response time is perhaps the most important QoS metric in a cloud-computing context [2]. For this reason, it is also the QoS parameter chosen in this work. Good solutions have been presented by some researchers in the literature dealing with QoS [14,13] and power consumption [3,8]. However, the model presented (*GreenC*) aims to obtain the best scheduling consolidation of Virtual Machines (VMs) to hosts, taking both criteria into account.

There is a great deal of work in the literature on linear programming (LP) solutions and algorithms applied to scheduling, like the one presented in [9,4]. Other notable work was performed in [15], where authors designed a Green Scheduling Algorithm that integrated a neural network predictor in order to optimize server energy consumption in Cloud Computing. Also, in [11] authors proposed a genetic algorithm that takes into account both makespan and energy consumption. Our main objective is to design a NLP (Non-Linear Programming) scheduling algorithm to minimize energy consumption and maximize SLA guarantees at the same time. To date, there is no research work based on these two criteria in cloud computing.

An important contribution of this paper is the way we model the power of the virtual machines in function of their workload. We rely on the work done in [7], where the authors formulate the problem of assigning people from various groups to different jobs and who may complete them in the minimum time as a stochastic programming problem. The job completion times were assumed to follow a Gamma distribution. To model the influence of the workload, we weighted the computing capacity of the physical host by a load factor determined by an Erlang distribution (equivalent to a Gamma). Finally, we obtained a stochastic programming problem and transformed it into an equivalent deterministic problem with a non-linear objective function.

The proposed policy has been implemented and tested using *OpenStack*[1], which is an open source software platform for managing cloud infrastructures.

[1] http://www.openstack.org*OpenStak*. http://www.openstack.org

The remainder of the paper is organized as follows. Section 2 presents our main contributions, a sort of scheduling policy. These proposals are arranged in order of increasing complexity. In Section 3, our proposal was first tested with the Excel optimizer, which provides tools to implement the models presented. Then it was implemented in a real cloud platform. The implementation is presented in Section 4. The experimentation showing the good behavior of our cloud model is presented in Section 5. Finally, Section 6 outlines the main conclusions and possible research lines to explore in the near future.

2 Materials and Methods

Our scheduling proposals, based on non-linear programming (NLP), model a problem with an Objective Function (OF). This objective function represents several performance criteria. Multiple performance criteria can be taken into account in order to choose the optimal scheduler. The most widely used are the minimization of the power consumption of the cloud system and the mean response time of the tasks. These are also the ones chosen in this article.

The relationships between variables and parameters and their ranges appearing in the OF are defined by additional constraints. The OF is then executed in a solver jointly with the constraints. In our case, the solver gives the assignment of Virtual Machines to hosts (physical machines) that minimize power consumption while preserving the QoS (Quality of Service).

GreenC tries to assign as many tasks as possible to the most powerful hosts, leaving aside the remaining ones. This way, unused VMs can then be turned off. The method is based on the computing capacity of the hosts, making decisions about which ones will hold more Virtual Machines to give the opportunity to power off the idle ones to maximize the QoS and save as much energy as possible.

The GreenC policy assumes a cloud made up of V heterogeneous hosts and a set of T heterogeneous Virtual Machines. Each host (H_v) has a specific amount of Memory (M_v), which restricts the maximum workload it can host. The notation and different features the model has to support are introduced prior to presenting the GreenC OF and its constraints.

2.1 Host Heterogeneity

The *relative computing power* (Δ_v) of a Host H_v ($v = 1, \ldots, V$), is defined as the *normalized score* of such a host. Formally, given V hosts, $\Delta_v = \frac{\delta_v}{\sum_{k=1}^{V} \delta_k}$, where $\sum_{k=1}^{V} \Delta_v = 1$. δ_v is a valid score (i.e. the computing power) of H_v. Although δ_v is a theoretical concept, there are many valid benchmarks to obtain node scores to get the relative computing power (i.e. Linpack[2] or SPEC[3]).

[2] Linpack. http://www.netlib.org/linpack/

[3] SPEC. http://www.spec.org

Linpack (available in C, Fortran and Java) for example, is used to obtain the number of floating-point operations per second. In our case, the closer the relative computing power is to one (in other words, the more powerful it is), the more likely it is that the VMs will be mapped into such a host.

2.2 Virtual Machine Heterogeneity

In order to model Virtual Machine heterogeneity, each VM, $\mho^i$ ($i = 1, \ldots, T$), has its *Processing cost* P_v^i, representing the execution time of VM, $\mho^i$, in H_v with respect to the execution time of VM $\mho_i$ in the less powerful H_v (in other words, with the lowest Δ_v). M_v^i is defined as the amount of Memory allocated to VM $\mho_i$ in H_v. It is assumed that Memory requirements do not change between hosts, so $M_v^i = M_{v'}^i \quad \forall v, v' \leq V$. The Boolean variable $\mho_v^i$ represents the assignment of VM $\mho^i$ to host H_v. Once the solver is executed, $\mho_v^i$ variables will inform about the assignment of VM to hosts. This is $\mho_v^i = 1$ if $\mho^i$ is assigned to H_v, and $\mho_v^i = 0$ otherwise.

2.3 Host Workload

The performance drop experienced by hosts due to workload saturation is also taken into account. If a host is underloaded, its throughput will increase as more VM are assigned to it. When the host reaches its maximum workload capacity, its throughput starts falling asymptotically towards zero. We can model this behaviour with an *Erlang* distribution density function. *Erlang* is a continuous probability distribution with two parameters, α and λ. The α parameter is called the shape parameter, and the λ parameter is called the rate parameter. These parameters depend on the VM characteristics. When α equals 1, the distribution simplifies to the exponential distribution. The *Erlang* probability density function is:

$$E(x; \alpha, \lambda) = \lambda e^{-\lambda x} \frac{(\lambda x)^{\alpha-1}}{(\alpha - 1)!} \forall x, \lambda \geq 0 \tag{1}$$

We consider that the Erlang modelling parameters of each host can easily be empirically obtained (i.e. by increasing the workload and measuring the mean response times). Fig. 1 shows various Erlang plots for some α and λ values by varying the x, designed in this case to represent the workload.

The *Erlang* distribution was developed to examine the number of telephone calls that might be made to the operators on a switchboard at the same time. We use it here to weigh the *relative computing power* Δ_v of each H_v with its associated workload factor determined by an *Erlang* distribution. This way, we model the loss or gain of executing power in each H_v according to its workload. In our case, we changed the abscissas (x) by the summatory of the *Processing cost* P_v^i of each $\mho_i$ assigned to such a H_v. Provided that Boolean variable $\mho_v^i = 1$ is a Boolean variable informing that $\mho^i$ is assigned to H_v, and

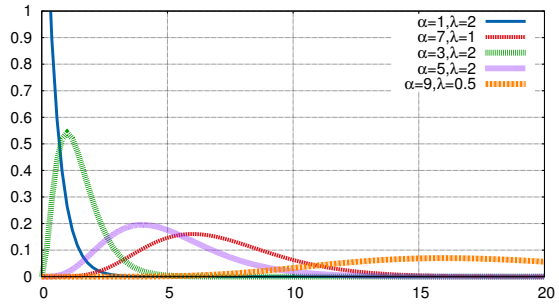


Fig. 1 Erlang plots for different α and λ values.

$\mho_v^i = 0$ if not, the *Erlang*-weighted *relative computing power* (Δ_v) of a Host H_v) would be:

$$\Delta_v E(\sum_{i=1} P_v^i \cdot VM_v^i; \alpha, \lambda) \tag{2}$$

2.4 Virtual Machine Communication

We also want to consider here the communication between VMs. The *Communication Cost* (representing the communication time) between VMs, $\mho^i$ and $\mho^j$, is denoted by C^{ij} and should be passed to the solver as an argument. For simplicity reasons, we consider that all the host communication links have the same bandwidth capacity. Notation C_v^{ij} represents the *Communication Cost* between $\mho^i$ residing in H_v with other $\mho^j$ (located in the same host or elsewhere). Provided that equivalent bandwidth between any two hosts, $C_v^{ij} = C_{v'}^{ij}$ $\forall i, j \leq T$ and $v, v' \leq V$. In other words, the *communication time* does not depend on the host or the links used between the hosts.

According to VM communication, the idea is to add a component in the OF that penalize (enhance) the communications performed between different hosts. Grouping VMs inside the same host not only will depend on their respective *Processing cost* (P_v^i), but also in the *relative communicating time* C_v^i.

Finally, the best VM scheduling assignment to hosts which take into account all the features (*GreenC* policy) is formally defined by the following non-linear programming model:

$$max(\sum_{v=1}^{V}(\sum_{i=1}^{T}\mho_v^i(P_v^i + \sum_{j\ st\ i>j}^{T}(C_v^{ij}\mho_v^j C_s + C_v^{ij}\mho_{\overline{v}}^j)))\Delta_v E(\sum_{j=1}^{T} P_v^j \cdot \mho_v^j;\alpha,\lambda)) \quad (3)$$

$$s.t.\ \sum_{v=1}^{V}\mho_v^i = 1 \quad \forall i \leq T \quad (4)$$

$$\sum_{i=1}^{T} M_v^i \leq M_v \quad \forall v \leq V \quad (5)$$

Equation 3 is the Objective Function (OF) to be maximized. Equality in Equation 4 and inequality in Equation 5 are the constraints of the objective function variables. Given the constants T (the total number of VMs), and Δ_v and M_v for each H_v, the solution that maximizes OF will obtain the values of the variables $\mho_v^i$, representing the VMs assigned to H_v. Thus, the $\mho_v^i$ obtained will be the assignment found by this model.

OF takes into account the *Processing costs* (P_v^i) and the *communication times* (C_v^{ij}) of the VMs assigned to each H_v. In the communication part of the OF ($C_v^{ij}\mho_v^j C_s + C_v^{ij}\mho_{\overline{v}}^j$), we select the j so $i > j$, because communication costs between VMs should only be counted once. In addition, we multiply the communicating cost of the two hosts by a given *Communication Speedup* (C_s) when $\mho^j$ is located in the same H_v. This happens when $\mho_v^j = 1$. Otherwise $\mho_{\overline{v}}^j = 1$, meaning $\mho^j$ is assigned to other hosts than H_v. C_s can be obtained by using a benchmark that measures communication speed. On this occasion, a communicating benchmark can be implemented by using such ICMP diagnostic tools as ping or traceroute unix commands and perhaps with some basic shell scripting.

3 Theoretical Experimentation

In this section, we present the theoretical results obtained from solving the scheduling problems to achieve the best VM to host assignation. Two representative experiments were performed in order to test the performance of GreenC.

The theoretical experiment was performed using the non-linear solver of the Excel program. As the objective of this section is to prove the correctness of the policy, we chose a small set of VMs and also simulated a small virtual architecture. The experimental framework chosen here was a cloud infrastructure made up of 3 hosts and 3 VMs. The parameters of each host are shown in Table 1. The configuration of each VM is shown in Table 2. Table 3 shows the *Communication Cost* between VMs and the speedup if communication is performed inside the host. This is the *Communication Speedup* (C_s).

Host	Memory	Δv	Erlang Distribution
1	**10**	**0.55**	$\alpha = 3, \lambda = 8$
2	**10**	**0.35**	$\alpha = 3, \lambda = 8$
3	**10**	**0.1**	$\alpha = 3, \lambda = 8$

Table 1 Host configurations.

	VM	Weight
	1	**1**
	2	**5**
	3	**1**
Totals	**3**	**7**

Table 2 VM Configurations

	VM i	VM j	Cost
	1	**2**	**0.2**
	1	**3**	**0.6**
	2	**3**	**0.3**
C_s	**0.6**		

Table 3 Communication configurations

3.1 Dominant host

In this example, we used the configuration shown in Table 1, where all the hosts followed the same Erlang distribution function with the same number of assigned VMs. Logically, in this case, the best option would be to assign all VMs to the most powerful host, as the communication costs between VMs will delay the execution.

The resulting assignment is $H_1 = \{\mho^1, \mho^2, \mho^3\}$, $H_2 = \varnothing$ and $H_2 = \varnothing$.

This example consolidated all the VMs in the most powerful host in order to minimize the communication costs between them. The assignment given by the model is consistent with the results expected by the model.

3.2 Erlang-shaped hosts

In this case, we supposed the configuration shown in Table 4, where all the hosts followed different Erlang distribution functions. In this case, we forced the Erlang distribution for Host 1 to obtain its maximum when assigning two VMs with a total weight of 6. The Erlang distribution for Host 3 reached its maximum when only one VM was assigned to it. Furthermore, in both cases the Erlang decreases sharply when more VMs are assigned to them. In this case, as could be expected, the best assignation is to assign two VMs to Host 1 and 1 VM to Host 3. That happens because the *communication costs* are compensated by the high *computing power* of the assigned nodes.

Host	Memory	Δ_v	Erlang Distribution
1	**10**	**0.55**	$\alpha = 7, \lambda = 1$
2	**10**	**0.35**	$\alpha = 8, \lambda = 3$
3	**10**	**0.1**	$\alpha = 2, \lambda = 1$

Table 4 Host configurations.

The resulting assignment is $H_1 = \{\mho^1, \mho^2\}$, $H_2 = \varnothing$ and $H_3 = \{\mho^3\}$. The assignment is different to the Powerful Host case due to the compensation of the communication costs with a smaller execution time required by the hosts. The results obtained were also consistent with the model objectives.

3.3 Erlang-shaped hosts with heterogeneous memory

In this scenario, the configuration shown in Table 5 is applied. Now all hosts also follow different Erlang distribution functions but, in addition, each host has different memory capacities. This way, although Host 1 is the most suitable one to host VM1 and VM2, due to its memory constraints only VM2 can fit. VM2 will be assigned to Host 1 instead of VM1 due to its Erlang distribution function.

Host	Memory	Δ_v	Erlang Distribution
1	**5**	**0.55**	$\alpha = 7, \lambda = 1$
2	**10**	**0.35**	$\alpha = 8, \lambda = 3$
3	**7**	**0.1**	$\alpha = 2, \lambda = 1$

Table 5 Host configurations.

The resulting assignment is $H_1 = \{\mho^2\}$, $H_2 = \{\mho^1, \mho^3\}$ and $H_3 = \varnothing$.

4 Implementation

The implementation was carried out by using four identical physical hosts making up the cloud infrastructure. Each host is a HP Proliant DL165 G7 with two AMD Opteron 6274 processors with 16 cores each at 2.2 GHz, 112 GB of RAM, 600 GB of SAS disk and 1 Gb Ethernet network. CentOS release 6.5 has been used as operating system.

Fig. 2 shows the implementation diagram. In each host, the *OpenStack* 2014.1 (Icehouse) software has been installed as the cloud platform. The *Wake-on-LAN* (*WoL*) program has been used in order to remotely boot the physical machines. *WoL* is a standard protocol for remotely waking computers up. Both, the remote host motherboard and network card must support this functionality. In our case, all the nodes supported the *WoL* protocol.

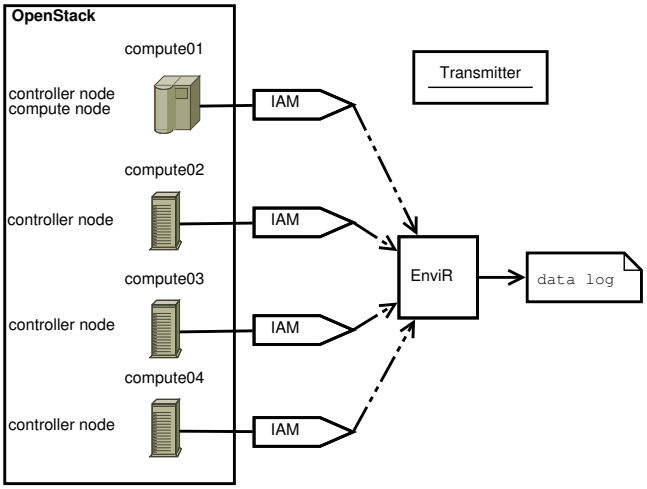


Fig. 2 Implementation diagram.

The hosts making up the cloud infrastructure are named *compute01*, *compute02*, *compute03* and *compute04* (see Fig. 2). The *compute01* host acts as the controller (i.e. front-end) node, that provides some additional services like the database (*MySQL*) and the web interface (*Dashboard*). The other three hosts act as compute nodes, that provide the processing, memory, network and storage resources to run the virtual machines.

Energy consumption has been empirically monitored. To do so, the following *Current Cost* devices have been used: four *Individual Appliance Monitors* (IAMs) were used alongside the nodes, and an *EnviR*, that transmits all the data collected from the IAMs (see Fig. 2). *EnviR* has a 433MHz SRD band receiver and offers an overall accuracy of >97%. *IAMs* operate at 230V AC voltage, 13A max Amps and C^2 433.92MHz communication. These devices allow us to individually monitor the energy consumption (in Watts) of each physical host every second and transfer the data to a text file for further analysis. The room temperature has also been monitored with a *Thermometer* and has remained constant at 27 degrees Celsius throughout all the experimentation.

The *GreenC* policy has been implemented using *Python* scripts. Algorithm 1 and Algorithm 2 show how physical nodes are powered off and on.

Algorithm 1 shutdown_host()

Require: *username*: remote host root username
Require: *ip*: remote host IP address
$host = username + '@' + ip$
$subprocess.call(["ssh", host, "poweroff"])$;

Algorithm 2 startup_host()

Require: cnt_{user}: controller host root username
Require: cnt_{ip}: controller host IP address
Require: $hwaddr$: remote host MAC address
$subprocess.call([\text{"}ssh\text{"}, cnt_{user} + \text{'@'} + cnt_ip, \text{"}WoL\text{"} + hwaddr])$;

To shut a host down, the *poweroff* command is executed in the selected host. To power a host on, the *WoL* command is executed from the controller host (*compute01*) targeting the MAC address of the remote host that needs to be awaken. Note that all nodes are connected in the same network.

Algorithm 3 allocate_vm()

Require: $name$: VM name
Require: $vcpu$: amount of VCPUs required
Require: ram: amount of RAM required
Require: $flavor$: VM flavor
Ensure: $host\_list$: sorted by descending *relative computing power* (Δ_v)
for $host$ in $host\_list$ **do**
if $host.vcpu\_used + vcpu < host.vcpuandhost.ram\_used + ram < host.ram$ **then**
$nova.servers.create(name, flavor = flavor, availability\_zone = host.av)$
end if
end for

Algorithm 3 shows, in a simplified way, how VMs are allocated to hosts based on the requested and available resources. Provided a list of sorted hosts, it chooses the host with more *relative computing power* that has enough resources to host the VM. A descriptive VM name needs to be provided, alongside the required resources and the desired *flavor*.

OpenStack defines different virtual hardware templates called *flavors*. These *flavors* have been used to create the VMs of the experimentation performed (Section 5). The principal VM parameters of the *OpenStack flavors* are listed in Table 6.

Flavor	VCPUs	RAM (MB)
tiny	**1**	**512**
small	**1**	**2048**
medium	**2**	**4096**
large	**4**	**8192**
xlarge	**8**	**16384**

Table 6 *OpenStack* flavors.

We first try to allocate the VM to the most powerful host that meets the desired requirements (in order to guarantee the SLA). Note that *OpenStack* supports *overcommitting* CPU and RAM on compute nodes, which allows to consume more resources than available on the hardware at the cost of reducing

the performance of the instances. This behavior may have sense since, in some scenarios, VMs are likely not to use all its VCPUs (Virtual CPUs assigned to a VM) or RAM. In such a situation, these VMs could lend underutilized computational resources to other(s) VMs neediest of them. However, in order to guarantee the SLA (this is, to provide to the client the amount of promised resources), the *GreenC* policy ensures that *overcommitting* is not allowed.

In order to allocate VMs to the desired physical host, *OpenStack* availability zones have been used. Availability zones allow users to define separated groups of nodes and assign new VMs to them.

In our case, an availability zone has been created for each host (*compute01*, ..., *compute04*). This way, it can be controlled to which specific host each VM is allocated.

Finally, Algorithm 4 shows the *controller process*, located in the *compute01* (the controller node). It is responsible for starting and shutting hosts down depending on the number of VMs assigned to them. If there are not enough available cloud resources, additional hosts will be turned on. If at the end of the VM assignment, a host remains idle, it will be then powered off.

Algorithm 4 Controller()

Require: $xlarge$
Require: $large$
Require: $medium$
Require: $small$
Require: $tiny$
Ensure: vm_list: sorted by descending *processing cost* (P_v)
Ensure: $compute\_list$: all compute nodes that are not controller nodes
  for vm in vm_list **do**
    $name = vm.name + str(i)$
    $allocated = allocate\_vm(name, vm.vcpus, vm.ram, vm.flavor)$
    if $allocated == error$ **then**
      $new\_host = select\_poweredoff\_host(compute\_list)$
      $startup\_host(new\_host)$
      $allocate_{vm}(name, vm.vcpus, vm.ram, vm.flavor)$
    end if
  end for
  for $compute$ in $compute\_list$ **do**
    if $compute.vcpu\_used == 0$ **then**
      $shutdown\_host(compute.user, compute.ip)$
    end if
  end for

Algorithm 4 takes as input the number of VMs to be deployed for each *flavor*, a sorted vm_list and $compute\_list$. Then, for each VM, tries to allocate it using the $allocate\_vm()$ function defined in Algorithm 3. If the VM could not be allocated, an additional host is started. Finally, when the VM assignment is complete, all the idle hosts are powered off.

5 Results

This section presents the experimental results obtained using the *OpenStack* software. The *GreenC* policy has been implemented using *python-novaclient*, which is a *Python* client for the *OpenStack Nova API* that allows us to interact with the *OpenStack* platform. Each test has been performed three different times, and in this section the mean values are presented.

First, a test to determine the energy consumption of the physical hosts when idle was performed. A host is considered to be idle when no virtual machines are deployed in it.

Fig. 3 shows the energy consumption (in Watts) of the four physical hosts making up the cloud for one hour. It can be seen how the controller node, *compute01* (Fig. 3, top left), consumes more energy than the remaining ones. Moreover, *compute04* consumes slightly more energy than *compute02* and *compute03*. This could be due to its physical location inside the data center that could affect the node temperature, or due to intrinsic hardware issues. During the time period shown in Fig. 3, no additional tasks were executed in the cloud.

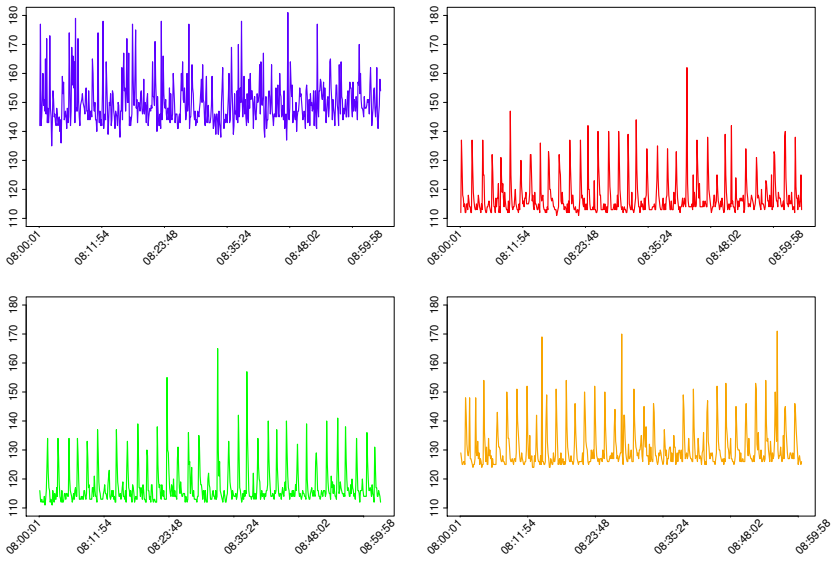


Fig. 3 Idle cloud energy consumption (in Watts) for 1 hour (between 8:00am and 9:00am). (top left) compute01, (top right) compute02, (bottom left) compute03, (bottom right) compute04.

Then, the energy consumption evolution when turning a node off and on has been tested. Fig. 4 shows the consumption process of the *compute04* node during fifteen minutes. It can be seen as energy consumption falls down to 7

Watts when a node is turned off and takes 137 seconds to be fully operational again. Moreover, when a node is turned back on, its energy consumption significantly increases during the start-up time and then gets back to its steady normal consumption.

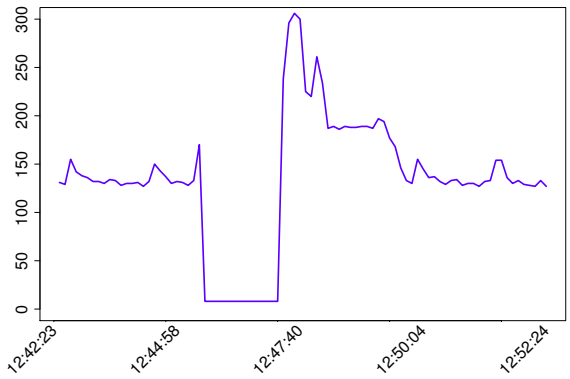


Fig. 4 Energy consumption evolution (in Watts) when stopping and starting a compute node (compute04).

Next, it has been tested how energy consumption increases when adding virtual machines to a node. In this case, a total of 100 VMs were deployed on *compute02*. Each second, one VM was added to the node, until a total of 37 VMs. Then, the deployment was paused. At the vertical dashed line (around 13:10), the VM deployment was resumed and the remaining 63 VMs were deployed. At the second vertical line (dotted-dashed), all the VMs were successfully deployed. From then on, energy consumption remained steady at around 250 Watts.

From Fig. 5, it can be seen how energy consumption actually increases when adding more VMs, although the major impact corresponds to the deployment of the first 37 VMs.

In the next test, a group of VMs has been deployed to the *OpenStack* infrastructure using its *default* scheduling policy. This policy first filters the hosts to only consider the ones that meet all the VM requirements. Then weights the hosts according to the request specifications, and selects the host with the largest weight. For this test, a total of 37 VMs were deployed with the *flavors* described in Table 7.

Table 8 shows the resulting allocation of VMs to hosts when applying the *default* policy.

Fig. 6 shows the energy consumption evolution (in Watts) when using the default *OpenStack* scheduling policy. In this case, the *compute04* node was powered off at the beginning of the test and it gets powered on immediately after. The VMs are deployed among all the hosts and it can be seen how the

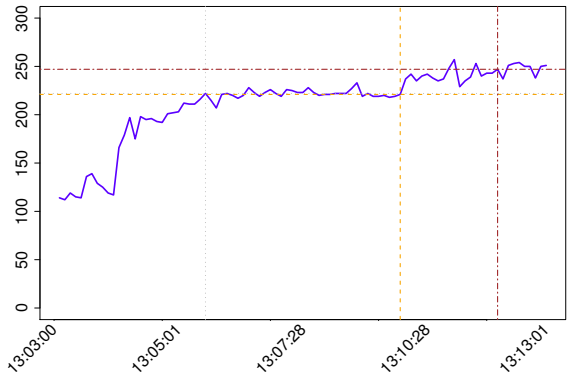


Fig. 5 Energy consumption evolution (in Watts) when increasing the load on a compute node (*compute02*).

Flavor	Deployed
xlarge	3
large	6
medium	16
small	8
tiny	4

Table 7 Deployed VMs grouped by *flavor*.

VM flavor	compute01	compute02	compute03	compute04
xlarge	1	0	1	1
large	1	2	1	2
medium	4	5	5	2
small	4	0	2	2
tiny	1	1	1	1

Table 8 Resulting allocation when using the default policy.

energy consumption increases homogeneously among all the hosts. At the end of the test, all the deployed VMs were terminated and it can be seen how the energy consumption levels return back to its previous values. The mean energy consumption for each node during the execution of the VMs can be seen in Table 9.

compute01	compute02	compute03	compute04	TOTAL
208.6542	187.94	188.29	202.3373	787.2215

Table 9 Mean energy consumption when using the default policy.

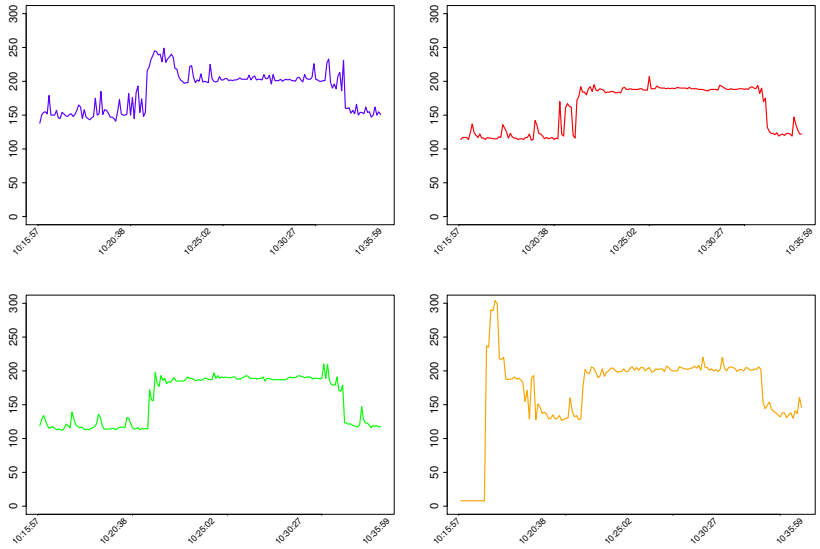


Fig. 6 Energy consumption evolution (in Watts) when applying the default policy. (top left) compute01, (top right) compute02, (bottom left) compute03, (bottom right) compute04.

Next, the same test has been performed using the *GreenC* policy. Table 10 shows the resulting allocation of VMs to hosts when applying the *GreenC* policy.

VM flavor	compute01	compute02	compute03	compute04
xlarge	**3**	**0**	**0**	**0**
large	**1**	**5**	**0**	**0**
medium	**1**	**5**	**10**	**0**
small	**1**	**1**	**6**	**0**
tiny	**0**	**0**	**4**	**0**

Table 10 Resulting allocation when using the *GreenC* policy.

It can be seen how the *GreenC* policy consolidates as much VMs as possible in a single host without overpassing its available resources. As no VMs have been assigned to the *compute04* node, it has been automatically powered off.

Fig. 7 shows the energy consumption evolution (in Watts) throughout the test. When VMs are deployed, energy consumption for nodes *compute01*, *compute02* and *compute03* greatly increases. However, *compute04* is powered off.

The mean energy consumption for each node during the execution of the VMs can be seen in Table 11.

The mean energy consumption for all nodes is 604.9 Watts when using the *GreenC* policy and 787.2 Watts when using the default *OpenStack* policy.

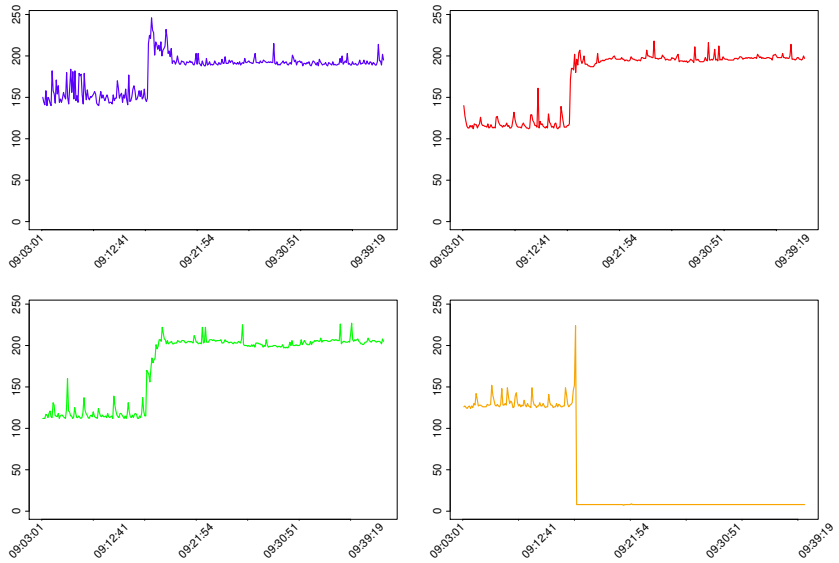


Fig. 7 Energy consumption evolution (in Watts) when applying the *GreenC* policy. (top left) compute01, (top right) compute02, (bottom left) compute03, (bottom right) compute04.

compute01	compute02	compute03	compute04	TOTAL
194.4395	196.2511	202.9821	11.2343	604,9070

Table 11 Mean energy consumption when using the *GreenC* policy.

Therefore, the *GreenC* policy reduced energy consumption by 182.3 Watts (23%).

6 Conclusions and Future Work

This paper presented a cloud-based system scheduling mechanism called *GreenC* that is able to achieve reduced power consumption and still maintain the SLA agreements. The complexity of the model developed was increased, thus adding more factors to be taken into account. The model was first tested using the Excel optimizer, and the results obtained proved consistent over a range of scenarios. Then, further experimentation in a real cloud platform has been performed and the energy consumption has been empirically monitored using a range of devices.

Initial results proved encouraging, achieving up to a 23% energy consumption reduction by using the *GreenC* policy. Nonetheless, much more experimentation is needed to demonstrate the usefulness of our proposals and should be contrasted with other power-aware policies.

In the near future, we plan to compare our policy with other power-aware policies and frameworks, such as *OpenStack-Neat*. We also plan to further expand our algorithms to consider further resources as disk usage and network communications, and to add live VM migration. We also plan to expand our architecture and to perform tests with additional physical hosts allocated in several data centers.

In the longer term, we consider to add machine learning techniques in order to predict future workloads so powered off hosts can be turned on in advance, as there is a significant start-up time and this can lead to unacceptable long response times when sudden workloads arrive to the system.

Acknowledgements

This work was supported by the MEyC under contracts TIN2011-28689-C02-02. The authors are members of the research groups 2009-SGR145 and 2014-SGR163, funded by the Generalitat de Catalunya.

References

1. M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, *A view of cloud computing*, Commun. ACM. Vol. 53, Issue 4, pp. 50–58. 2010.
2. R. Aversa, B. Di Martino, M. Rak, S. Venticinque, U. Villano, *Performance Prediction for HPC on Clouds. Cloud Computing: Principles and Paradigms*, John Wiley & Sons. 2011.
3. A. Beloglazov, R. Buyya, *Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers. Concurrency and Computation: Practice and Experience, vol. 24 pp. 1397-1420. 2012.*
4. A. Goldman, Y. Ngoko, *A MILP Approach to Schedule Parallel Independent Tasks. International Symposium on Parallel and Distributed Computing, ISPDC '08, pp. 115–122. 2008.*
5. A. Iosup, N. Yigitbasi, D. Epema, *On the Performance Variability of Production Cloud Services. 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'2011), pp. 104-113. 2011.*
6. A. Keller and H. Ludwig, *The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services*, J. Netw. Syst. Manage. Vol. 11, Issue 1, pp. 57–81. 2003.
7. M.F. Khan, Z. Anwar, Q.S. Ahmad, *Assignment of Personnels when Job Completion time follows Gamma distribution using Stochastic Programming Technique. International Journal of Scientific & Engineering Research, Volume 3, Issue 3. 2012.*
8. D. Kliazovich, P. Bouvry, S. Khan, *GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. The Journal of Supercomputing. 2010.*
9. J.Ll. Lérida, F. Solsona, P. Hernández, F. Giné, M. Hanzich, Josep Conde, *State-based predictions with self-correction on Enterprise Desktop Grid environments. Journal of Parallel and Distributed Processing, vol 71, n. 11, pp. 777–789. 2012.*
10. M. Martinello, M. Kaâniche, K. Kanoun, *Web service availability: Impact of error recovery and traffic model. Journal of Reliability Engineering and System Safety, Vol. 89, n. 1, pp. 6-16. 2005.*

11. M. Mezmaz, N. Melab, Y. Kessaci, Y.C. Lee, E.-G. Talbi, A.Y. Zomaya, D. Tuyttens, *A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. Journal of Parallel and Distributed Computing, Volume 71, Issue 11, November 2011, pp. 1497–1508. 2011.*
12. J. Varia. *Architection for the Cloud: Best Practices. Amazon Web Services. 2014.*
13. J. Vilaplana, F. Solsona, I Teixidó, F. Abella, J. Rius, *A queuing theory model for cloud computing. The Journal of Supercomputing. 2014.*
14. J. Vilaplana, F. Solsona, F Abella, R. Filgueira, J. Rius, *The cloud paradigm applied to e-Health. Bmc Medical Informatics and Decision Making. 2013.*
15. T. Vinh, T. Duy, Y. Sato, Y. Inoguchi, *Performance evaluation of a Green Scheduling Algorithm for energy savings in Cloud computing. Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW) , pp. 1–8. 2010.*
16. K.V. Vishwanath, N. Nagappan, *Characterizing cloud computing hardware reliability. In Proceedings of the 1st ACM symposium on Cloud computing (SoCC '10), pp. 193-204. 2010.*

J Supercomput
DOI 10.1007/s11227-014-1312-9

H-PC: a cloud computing tool for supervising hypertensive patients

Jordi Vilaplana · Francesc Solsona · Francesc Abella · Josep Cuadrado · Ivan Teixidó · Jordi Mateo · Josep Rius

Abstract Hypertension or high blood pressure is a condition on the rise. Not only does it affect the elderly but it is also increasingly spreading to younger sectors of the population. Treating it involves exhaustive monitoring of patients. Current health services can be improved to perform this task more effectively. A tool adapted to the particular requirements of hypertension can greatly facilitate monitoring and diagnosis. This paper presents the computer application Hypertension Patient Control (H-PC), which allows patients with hypertension to send their readings through mobile phone

This work was supported by the MEyC under contracts TIN2011-28689-C02-02. The authors are members of the research group 2009-SGR145, funded by the Generalitat de Catalunya.

J. Vilaplana · F. Solsona ( ) · J. Rius · I. Teixidó · J. Mateo
Department of Computer Science and INSPIRES, University of Lleida,
Jaume II 69, 25001 Lleida, Spain
e-mail: francesc@diei.udl.cat

J. Vilaplana
e-mail: jordi@diei.udl.cat

J. Rius
e-mail: jrius@diei.udl.cat

I. Teixidó
e-mail: iteixido@diei.udl.cat

J. Mateo
e-mail: jmateo@diei.udl.cat

F. Abella
Santa Maria Hospital, Avda Alcalde Rovira Roure 44, Lleida 25198, Spain
e-mail: abella@gss.scs.es

J. Cuadrado
Hesoft Group, Partida Bovà, 15, LLeida 25196, Spain
e-mail: josep@hesoftgroup.com

Published online: 18 October 2014

Short Message Service (SMS) or e-mail to a cloud computing datacenter. Through a graphic interface, clinicians can keep track of their patients, thus facilitating monitoring. Cloud-based datacenters provide a series of advantages in terms of scalability, maintainability, and massive data processing. However, the ability to guarantee Quality of Service (QoS) is crucial for the commercial success of cloud platforms. A novel and efficient cloud-based platform managing H-PC with QoS is also proposed in this paper.

Keywords Hypertension · Health monitoring systems · Patient tracking · Cloud systems · Quality of Service

1 Introduction

Cloud computing has gained worldwide attention from many researchers, but only a small portion of these have addressed the performance problem [1]. Performance issues affect the question of how to guarantee that the system can offer an acceptable level of Quality of Service (QoS).

In cloud computing, both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services.

In cloud computing, hardware and software services are efficiently handled, as they can be added and released dynamically [2]. Problems arise when scaling the system, that is, when trying to deploy a platform to support the computing needs of many hospitals with different clinical departments and their corresponding clinicians and patients. The need for scalability even increases when the cloud must also provide support for additional common and specific desktop or Internet applications (administration, specialized, general purpose, etc.). However, we focused our attention to scaling H-PC users and hospitals for only one speciality, namely hypertension.

Hypertension is a factor of risk in cardiovascular disease, the leading cause of death worldwide. Only about half the people under treatment for hypertension successfully maintain their blood pressure at the recommended levels nowadays [3,4]. Furthermore, the difficulty of ensuring satisfactory control of blood pressure is still a great challenge, despite the major steps that have been taken to improve the lifestyle through new pharmaceutical treatments [5,6]. Therefore, techniques to control hypertension such as home blood pressure (HBP) may have an important role [7].

Monitoring blood pressure at home consists of patients taking the readings at home and registering these using a digital device. Then, the patients send the readings to a health professional, who is responsible for taking appropriate action.

In this article, we present the H-PC tool, which allows patients to monitor their blood pressure at home and send the measurements via SMS or e-mail. In addition, through H-PC, the professional can also send short message service (SMS) or e-mail messages back to patients. In *telemonitoring*, readings taken at home are relayed by communication media to health care professionals who can take appropriate action [8]. Thus, H-PC implements a sort of *telemonitoring* based on SMS and e-mail messaging.

The H-PC tool has several advantages over traditional HBP. It allows the blood pressure measurements to be sent back to the health center immediately. This way, clinicians can monitor their patients in real time and immediately take appropriate actions when needed. Furthermore, controlled patients can delay or bring forward

visits depending on their measurements, allowing clinicians to focus on the patients who need immediate attention. Moreover, deploying this tool in a cloud environment enables several health care centers to be integrated easily, thousands of patients to be managed and the system scaled when more resources are needed.

We are working on H-PC for following hypertensive patients in the cardiology service at Santa Maria Hospital, in Lleida (Spain). The rationale for doing so is twofold. First, HBP monitoring and telemonitoring should become a routine component of blood pressure measurement in the majority of patients with known or suspected hypertension [9], given that such readings may be better predictors of cardiovascular and renal outcomes than surgery readings [10,11]. The scope of this work is not to develop an efficient protocol or treatment procedure for hypertensive patients. However, we are interested in developing a user-friendly tool able to implement any kind of *telemonitoring* treatment by the clinicians in a simple way.

Second, there is a possibility that the telemonitoring service will become saturated and unable to guarantee appropriate care for all its patients. Taking this into account, we developed a Software as a Service (SaaS) cloud architecture with QoS guarantees. Service requests are transmitted to a cloud server running the H-PC application, which is associated with a Service Level Agreement (SLA). An SLA is a contract negotiated and agreed between a customer and a service provider for which a customer pays only for the resources and services used according to negotiated QoS requirements at a given price [12]. In this work, the SLA is maintained between H-PC and the health care centers using it, which act as customers. Job response time is perhaps the most important QoS metric in a cloud computing context [12]. For this reason, it was also the QoS parameter chosen in this work.

We also present a cloud computing framework providing QoS for a given SLA and number of H-PC users (patients and clinicians). Basically, this is based on the response time as the QoS metric. The system also provides high reliability [13] and variability [14], leaving aside such other cloud computing issues as cloud availability [15] or energy consumption [16].

The remainder of the paper is organized as follows. Section 1.1 details the related work to address the problem of providing QoS and scalability in telemonitoring cloud computing, in our case, applied to hypertension. In Sect. 2.1, we present both the H-PC application and the cloud computing framework, which was specially designed to embed the H-PC application with QoS guaranteeing. Experimentation showing the good behavior of our proposal is presented in Sect. 3. Finally, Sect. 4 outlines the main conclusions and future work.

1.1 Related work

Quantifying the computing resources for servicing the required QoS in a real cloud computing environment (Amazon EC2 [17], Microsoft Azure [18], Google App Engine [19]) is a research challenge because clouds exhibit a high variability of demands for computing resources; and users also have dynamic and different QoS requirements. The use of real infrastructures for benchmarking the application performance under variable conditions (availability, task arrival rate, falling, scalability, and

occupancy) is often constrained by the rigidity of the infrastructure. Therefore, this makes the reproduction of the results that it can generate extremely difficult [20]. Some authors rely on queuing theory to simulate real cloud behavior [1,21]. Others instead rely more on the possibilities of event-driven simulators [20,22,23]. Thus, it is a challenge to perform benchmarking experiments using real-world cloud environments, the one chosen for this article.

Also, previous work has been done to analyze cloud computing applied to home health care systems [24]. In [25], the authors discuss several mitigation techniques focussing on patient-centric control and policy enforcement via cryptographic technologies.

There is a potentially important role for novel techniques to lower blood pressure, especially in primary care, where management of hypertension mainly takes place [26]. In a recent scientific article, the American Heart Association concluded that HBP monitoring "should become a routine component of blood pressure measurement in the majority of patients with known or suspected hypertension" [9]. HBP readings may also be better predictors of cardiovascular and renal outcomes than surgery readings [10,11]. Furthermore, the potential benefits of HBP also include more accurate assessment than surgery blood pressure, rapid titration of antihypertensive therapy, identification of white-coat and masked hypertension, and greater patient involvement in managing hypertension, a condition that is typically asymptomatic [26]. One HBP approach is patient self-management, defined as the ability and willingness of a patient to self-monitor. Some studies shown that self-management blood pressure control is at least as, or even better than, office-monitored blood pressure [27,28].

In another study [30], a comparison of web sites used to manage and present HBP readings was performed between June and August of 2009. A list of 33 desirable web site features was drawn up and a total of 60 web sites identified, of which 20 were free or free to try. The results showed that these 20 web sites offered between 41 and 77 % of the 33 features considered desirable, such as displaying HBP readings in tabular and graphic modes. In contrast, none of them were directly linked to common electronic medical records. Despite web sites having alert values, none of them provided any tools for sending alert messages in any format (e-mail or SMS) to patients, i.e., to telemonitor them. Our motivation is based on the work published in [29], which demonstrates the effectiveness of it it telemonitoring.

The main aim of H-PC is to provide access to standard medical records and allow physicians to control and communicate with patients by sending SMS messages and e-mails. The ability to perform this bidirectional communication makes H-PC an innovative and very useful tool for hypertension physicians. Furthermore, moving H-PC into a real-world cloud environment with QoS guarantees is a novel and interesting challenge that is dealt with in this paper.

2 Material and methods

2.1 H-PC

In this section, we explain the H-PC application in detail. In doing so, the main features and its operation are first presented. Next, the main principles of the H-PC design are

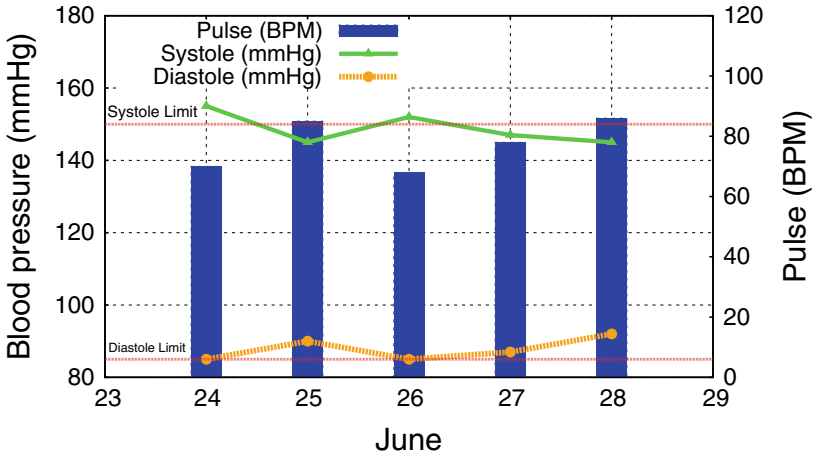


Fig. 1 H-PC readings

introduced. Finally, taking efficiency and performance into account, the architecture of the cloud-based system is explained.

2.1.1 Features and operation

Hypertension Patient Control was designed for collecting and managing data from hypertensive patients. Its functions are to record and print/display measurement statistics, to graphically show patients' evolution using charts, automatic control of risk communication by SMS messages or e-mail to aid clinicians to diagnose and generate alerts or suggestions for treatments, patient monitoring, medication, nutrition, etc.

Hypertension Patient Control allows target limits to be established individually from both systole and diastole blood pressures, depending on patient characteristics. If these limits are exceeded, an alert is shown in the main page of the H-PC tool so clinicians can act quickly and, if needed, perform an intervention or send an alert to the patient.

Figure 1 shows an example of the graphic data plot of a patient's readings. These readings can be registered automatically from SMS or e-mails sent by the patients or can be introduced manually by the clinicians. H-PC automatically calculates the mean values for each day, showing only one value per day only in the plot. Nevertheless, individual readings can also be viewed in table format. Note that pulse can also be registered when manually entering the data or when sending it via e-mail. When the data is sent by a patient, H-PC performs a data verification check to avoid incorrect or invalid measurements, like negative or impossibly high values.

Figure 2 shows the operation of H-PC. First of all, via bluetooth, the blood pressure device (i.e., Fully Automatic Wrist Bluetooth Blood Pressure Monitor HPL-108) sends the patient's readings to a mobile phone (1). Those readings can be sent via SMS or e-mail (2). There are two different pathways to the destination, one for the SMS messages and another for e-mails. The SMS messages are delivered via a GSM. Global System for Mobile communications GPRS. General Packet Radio Service. GSM extension on the 2/3G communication network to the *cloud computing framework*, made up of two sites with different domain names (*site 1* and *2*). *Site 1*, the one containing a GSM

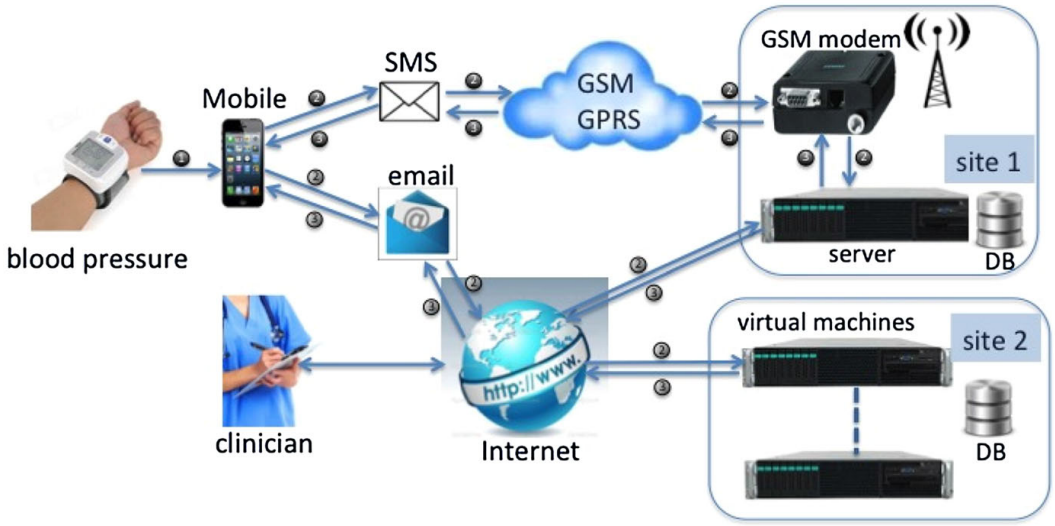


Fig. 2 H-PC operation

modem device with a Subscriber Identity Module (SIM) card associated with a mobile telephone number, will be responsible for receiving/sending SMSs from/to H-PC. Then, when receiving an SMS message through the GSM modem, the *cloud framework* redirects it to H-PC itself, which is responsible for checking and saving the data in the *database*. Next, depending on the data, H-PC responds to the patient's mobile with another SMS (3). Sending/receiving readings by means of e-mails through Internet follows a similar pathway as the SMS. The default destination server will also be *site 1*.

The H-PC tool can be freely downloaded from Hesoft Group website. To operate properly, it requires a GSM modem device with a SIM card.

Hypertension Patient Control is currently designed to operate with only one SIM (i.e., phone number), hence a single GSM Modem device is used. This device is located in *site 1*. So only *site 1* can send/receive SMS messages (see Fig. 2). This rather determines the architecture design. However, future trends will be on the addition of a SIM duplication on *site 2*. This will allow *site 2* to deal also with SMSs, potentially increasing the reliability and availability of the overall system when *site 1* fails.

In addition, as we will see in Sect. 2.1.3, *site 2* is mainly responsible for providing reliability and variability features and increasing the scalability of the cloud system, which is important to guarantee the QoS of the H-PC tool that is offered as SaaS on top of it.

2.1.2 Design principles

Hypertension Patient Control is a multi-platform and multi-language web-based application with a user-friendly graphic user interface (GUI) that provides the clinician with easy access and utilization of all its functions, implemented using Java for the back-end and Javascript, CSS, XHTML, and AJAX for the front-end. H-PC can run on any computer, operating system (Windows, Symbian, Leopard, Linux, etc.), and any of the major web-browsers (Firefox, Explorer, Chrome, Opera, Safari, etc.). The interface

Table 1 Usability criteria

Criterion	What is measured
Learnability	How much time is required for people to complete basic tasks?
Efficiency	How many steps are required to complete basic tasks?
Memorability	How much does the person remember afterwards or after periods of non-use?
Errors	How many mistakes did people make using the program?
Emotional response	How does the person feel about the tasks completed (confident, stressed)?

was developed using the responsive web design approach, thus providing an optimal viewing experience across a wide range of devices.

Nowadays, *HBP* is mainly used for measuring blood pressure in patients with recommended disparity between the query and the pressures obtained outside the hospital environment, which consists of placing a device for measuring blood pressure daily, including during sleep. However, it can be applied to any kind of hypertensive patient. Usually, the readings period is 24 or 48 hours, with a measurement taken every hour. However, as stated above, using H-PC, we provide a means for implementing a more efficient *telemonitoring* approach, able to avoid duration and frequency restrictions.

Although the efficiency of the *telemonitoring* technique has been demonstrated in several studies, the main problem is its high cost. In this work, the cost is significantly reduced due to the use of conventional devices.

Design decisions are focused on eliminating waiting lists, saving the patients and the health center time and money, and aiding diagnosis. However, to reach these goals, it is very important to provide this tool with a high degree of usability. This will facilitate its successful deployment among health professionals.

The design of H-PC also followed stringent usability and user-friendliness criteria (see Table 1) after an exhaustive analysis of the clinical requirements of medical staff in the Santa Maria Hospital and the University Hospital Arnau de Vilanova.

There are five criteria that determine usability and user-friendliness [31]. Firstly, a usable program must allow users to accomplish basic tasks the first time they use it (Learnability). Secondly, users who are familiar with the program should be able to perform the tasks for which the program is required quickly (Efficiency). Thirdly, users should quickly relearn how to use the program after some time without using it (Memorability). Fourthly, users should not make serious errors when using the program, and recovering from any error should be easy (Error rate). Fifthly and finally, the user should be satisfied with using the program (Emotional Response).

The principles of cloud computing were also incorporated into the design. As a consequence, H-PC can be accessed from standard computers, smartphones, and tablets. These criteria and the comments of six users were used to design and iteratively improve the GUI of the program. Development finished after eleven iterations. The number of users was determined using Nielsen's formula [32]: the number of usability problems found at least once by i users is:

$$Found(i) = N(1 - (1 - p)^i), \tag{1}$$

where N is the total number of problems in the interface and p is the probability of finding the average usability problem when running a single user. Provided that p is 0.33 (as was suggested in [32]), through simple calculations, it can be seen that finding 90 % (i.e., Found(i)=9 and N=10) requires the problem to be tested by 6 users (in our case, medical staff).

2.1.3 Architecture

Hypertension Patient Control can be executed in different Internet sites (i.e., domains). In this case, we only have two, *site 1* and *site 2*.

First, there is a *firewall* that will pass incoming requests from the H-PC users to the system. A *firewall* is a security system that controls the incoming and outgoing network traffic and determines, based on predefined rules, whether this traffic should be allowed or not. Next, the scheduler assigns the requests to the virtual machines, where a copy of H-PC is running. The data is managed by a clustered database.

The two sites are implemented using OpenStack (on *site 1*) and VMware (on *site 2*). Both environments are able to create virtual machines dynamically as needed. The difference, subtle but important, is that in *VMware* (*site 2*), this operation must be performed manually by one operator, while in *OpenStack*, the creation/deletion of virtual machines can be performed automatically. *OpenStack* (*site 1*) is more dynamic and flexible because no human interaction is needed unless the entire system has to be restarted.

This way, QoS can be guaranteed by adding more virtual machines when response time reaches a certain threshold. Moreover, saturation scenarios can be avoided, thus maintaining the maximum system throughput. These QoS metrics are studied in Sect. 3 below.

Site 1 hosts the Virtual Machine (VM) which acts as entry point (the Master), the one executing the scheduler (or load balancer, see Sect. 2.2.3) between the remaining Virtual Machines forming *site 1* and *2* Virtual Machines 1 and 2 (VM1 and VM2) to *site 1* and Virtual Machine 3 (VM3) to *site 2*. This will influence performance and scalability as well as tolerance to load variability of the cloud framework. The scheduler is exclusively dedicated to routing entering service requests to the slaves (virtual machines). The architecture proposal could be extended to any number of Virtual Machines, but for simplicity, we only deal with this composition. Experimentation with larger numbers of VMs is left for future work. To provide the system with the ability to save power whenever possible, idle VMs should be turned off. *OpenStack*, on *site 1*, is able to turn virtual machines on and off on the fly, thus saving energy whenever possible. However, power-aware consumption is outside the scope of this work, but it is planned to assess this in future trend.

To guarantee robustness against failures, there are two copies of the same relational database (and implemented with *MySql Cluster*, acting as a mirroring Raid (i.e., Raid 1). One copy is located at *site 1* and the other one at *site 2*. The database is synchronized by configuring *MySql Cluster* accordingly, so that registers in both database sites are properly synchronized. This guarantees a high degree of system reliability.

2.2 H-PC features

Cloud frameworks should provide a certain number of desirable performance features. Our contributions in the design of the cloud framework are mainly the following: scalability, reliability, and load balancing.

2.2.1 Scalability

Providing a high degree of scalability is a key to successfully implementing a cloud-based system. In addition to horizontal, vertical, and database scalability, the architecture has been designed to provide scalability at four different levels:

Administrative. Scalability in terms of the number of organizations or users sharing the system. The system must be able to handle an increasing number of registered users and organizations and, therefore, an increasing number of concurrent users and user requests. We go into this aspect in depth, because the cloud framework is designed to deal with large numbers of users, clinicians, and hospitals.

Functional. Scalability in terms of improving current functions and adding new ones. The system must be easily updated and able to add new functionality without excessive effort. Having multiple VMs allows new H-PC versions to be installed on the fly. For example, while a new version is being installed on VM1 and VM2, VM3 can be active, and vice-versa. The same is true for the cloud infrastructure, while updating one site the other one can be active, and vice versa.

Geographic. Scalability in terms of expanding the system from a local area to a wider geographic territory without loosing performance. It must be taken into account that the system may be eventually used globally and ensure that the system is not bound to any geographical pattern. Both sites are located in the same geographical area. Future work will deal with a) increasing the number of sites and b) deploying them in different geographical areas. For simplicity, this has not been performed yet.

Load. Scalability in terms of dynamic workload. The system must be able to distribute load efficiently between its computing resources. Note that it is also important to add/remove resources dynamically depending on the load in terms of optimal resource utilization and power-aware management. The scheduler has been designed to distribute service requests in a balanced way. However, future trends will be aimed at reducing power consumption by stopping idle VMs.

2.2.2 Reliability

Designing a reliable system is also crucial when developing a cloud architecture [13]. A fault tolerant system is able to continue operating when unexpected negative events occur. The main idea is to avoid a complete failure when one or multiple parts of the system fail, and keep providing service at a reduced performance rate.

In our architecture, each virtual machine is a component that can potentially fail. To avoid a complete failure when a single component fails, redundancy is applied.

Although this is an excellent way of improving reliability, it also has some drawbacks. Redundancy implies an increased cost in terms of the required resources, com-

plexity and power consumption, among others [33]. Therefore, we have followed a criterion to determine which components redundancy should be applied to.

Our criteria were driven by three factors. The probability of failure of the component, the critical nature of the component, and the associated cost of applying redundancy to the component. Taking this into account, redundancy was applied to the database. The failure of a single database in one site is possible and is not unreasonably expensive. This is examined in greater detail in the Implementation Section (Sect. 2.3).

The Master was not replicated as its probability of faulting is extremely low due to its scarce functionality. Only restarting the VM is enough. No serious damage will occur while the Master is restarted, except the time lost in doing so. Furthermore, processing VMs (VM1, VM2, and VM3) do not need to be duplicated. Simply, when one fails, its service requests it is executing are restarted when the VM is reestablished.

The GSM modem was not replicated due to the cost of the physical device. Moreover, it can be considered a non-critical component due to the fact that incoming and outgoing SMS messages will remain stored until the device is functional again, and therefore, no messages would be lost if this component stopped working for a short, or even a long, period of time. The same holds for e-mailing.

2.2.3 Scheduler

The scheduler is responsible for balancing the requests among the different computing virtual machines, and more specifically between VM1, VM2, and VM3, as can be seen in Fig. 3. It also acts as the entry point to the system. All incoming requests pass through the scheduler.

Several scheduling policies can be applied. However, we have focused on three different ones (*Request Counting, Weighted Traffic, and Pending Request*).

Request Counting Policy. In this policy, each virtual machine is given a normalized score that determines the number of requests that the scheduler will send to it. The normalized score is obtained as follows. Assuming that the scores of VM1, VM2, and

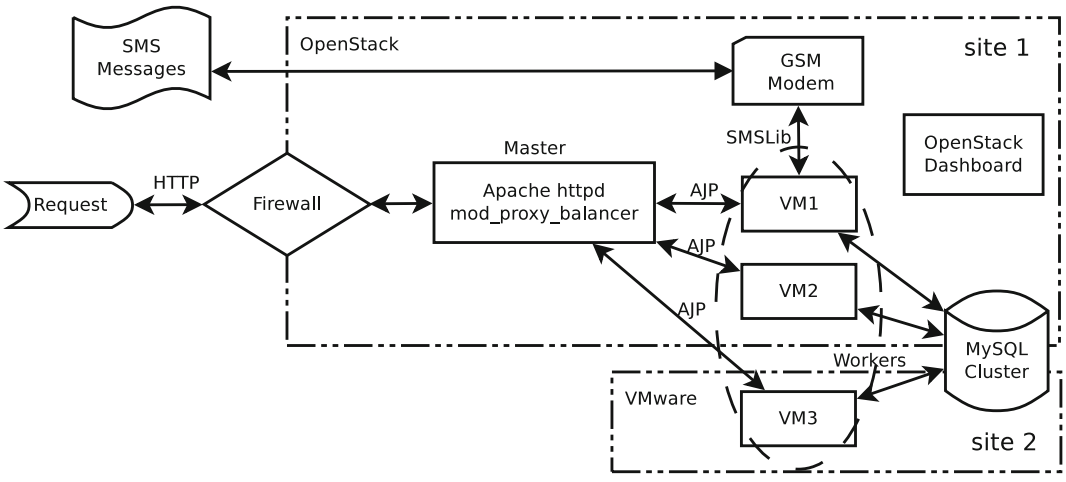


Fig. 3 Implementation of the system architecture

VM3 are S_{VM1}, S_{VM2}, and S_{VM3}, respectively, the normalized score of VMi, namely N_{VMi} (being $i = \{1, 2, 3\}$), is obtained as follows:

$$N_{\mathrm{VMi}} = \frac{S_{\mathrm{VMi}}}{S_{\mathrm{VM1}} + S_{\mathrm{VM2}} + S_{\mathrm{VM3}}} \tag{2}$$

Therefore, the closer the normalized score is to one, the more requests will be mapped to such a virtual machine. This policy is based on the computing capacity of the virtual machines, and allows us to decide which ones will process more requests.

Weighted Traffic Policy. This policy has the same mechanics as the previous one with the exception that the score will determine how much traffic, in bytes, a virtual machine will handle. So, in this case, instead of taking the number of requests sent to the virtual machine into account, it will take into consideration the amount of traffic. This policy allows a balancing to be performed by taking network traffic into account, instead of the number of requests.

Pending Request Policy. This policy assigns an incoming request to the virtual machine with the lowest number of active requests at that moment. Therefore, when a new request arrives, it is assigned to the virtual machine which is currently processing fewer requests. This policy performs a very accurate balancing among the virtual machines when all requests have a similar difficulty to be served. When some requests may need a lot of computing resources and some others may not, this policy would not be a good option.

2.3 Implementation

Next, the main details about the implementation of the cloud framework and the most complex part of the framework, responsible for scheduling requests, namely the Master, is presented.

2.4 Cloud framework

The framework is made up of six VMs, distributed in two different sites (*site 1* and 2). All the VMs run Ubuntu GNU/Linux 3.2.0-41-virtual x86_64. The first site allocates four VMs deployed on top of the OpenStack (Master, VM1, VM2, and a database VM). The second site is set up in a different physical location and deployed using VMware. The second one allocates two VMs deployed also on top of the VMware (VM3 and a database VM). As mentioned above, the degree of administrative and geographic scalability increases with the number of sites. Figure 3 shows the implementation schema.

The scheduler, located at the Master VM, is deployed in a VM with 512MB RAM and 1 Virtual CPU (VCPU). A VCPU corresponds to one core of an AMD Opteron 6100 processor running at 2,1 GHz. It is implemented using the *Load Balancer* of *Apache Tomcat 7*, through the *mod_proxy_balancer* module.

VM1 and VM2 have 4GB RAM and 2 VCPU. VM3 has 2GB RAM and 2 VCPU. These three VMs are the computing VM nodes, where the H-PC application copies (each performing the same operation) are deployed on top of the *Apache Tomcat* web

server. *Site 1* hosts two computing VMs (VM1 and VM2) and *site 2* allocates the remaining computing VM (VM3). The communication/synchronization between the Master and the computing VM nodes follows a master–worker paradigm. So, VM1, VM2, and VM3 are also named workers from now on.

All VMs are configured with the AJP(*Apache JServ Protocol - Apache Tomcat Connector*) protocol enabled, which is used by the scheduler to communicate with the nodes. AJP is a protocol that can proxy inbound requests from a web server (Apache HTTP server) to an application server (Tomcat), and it also allows the web server to monitor the application server through ping. Furthermore, all these instances are created from a unique snapshot instance, that is, an image created from a running VM.

The use of snapshots allows the VMs to be updated to newer versions of the H-PC application very easily. Therefore, we achieve a high degree of functional scalability.

The database is implemented using a *MySQL Cluster*, distributed between the two sites. The *MySQL Cluster* is implemented with two VM with 4GB RAM and 2 VCPU (each on different sites). Having multiple computing and data sites ensures a high degree of load and administrative scalability and reliability. The table engine used is MyISAM.

The GSM Modem device is connected to VM1 through a USB port. The communication between the VM and the device is performed using the *SMSLib* messaging library. This way, incoming messages are retrieved from the GSM Modem and stored in the database. When an outgoing message is generated, it is stored in a specific database table, which is constantly monitored by VM1. When a new message is detected, it is immediately forwarded to the GSM Modem.

Outgoing e-mails are managed using *MailChimp*, which allows a maximum of 12,000 e-mails to be delivered per month (a limitation of the ISP used). Note that if needed, it would be possible to have an unlimited number of e-mails per month, but at a cost fixed by the ISP.

2.4.1 Master

The scheduler works as a function in the Master VM, and is activated asynchronously on arrival of a request for service. The scheduler (in the Master VM) is implemented using the Apache 2.2 HTTP Server module *mod_proxy_balancer*. It distributes the load among the different available computing VMs or workers (see Algorithm 1).

Algorithm 1 Master. Scheduling function.

Input: *Scheduling_Policy* {*Request Counting, Weighted Traffic, Pending Request*};
1. **while**($requests$)
2. vm=Select_VM(*Scheduling_Policy*);
3. **If** (vm.off) vm.on;
4. w_{new}=Create_worker(vm);
5. start($request$,w_{new},vm);
6. **end while**

The scheduler operates until there are no more pending requests. These requests can be to process web services from the users, SMSs or e-mail processing, and/or reception/delivering (line 1). The VM selected to process a request (line 2) is obtained based on the scheduling policies presented in Sect. 2.2.3: *Request Counting*, *Weighted Traffic*, and *Pending Request*. The different policies can be selected by setting the *lbmethod* parameter inside the *Apache* configuration file. If the selected VM is off, it is switched on (line 3). Then, a new worker process (w_new) is created on the selected VM *vm* (line 4) and started on it (line 5).

The *Request Counting* policy is the default one. This is established by setting the *lbmethod* variable to *byrequests*. The Master assigns an *lbfactor* to each VMi, which, in our case, is the normalized score of VMi, N_{VMi}, as defined in the *NVM* formula. The *lbfactor* of each VM is computed as a function of the computing capacity of such a VM. The scheduler communicates with different VMs via the *AJP* protocol to monitor their condition. If a VM fails, it stops sending requests and the load is redistributed among the remaining VMs.

The *mod_proxy_balancer* also offers two different scheduling algorithms. The *Weighted Traffic* policy can be applied by changing the *lbmethod* variable to *bytraffic*. It has the same mechanics as the previous one with the difference that *lbfactor* is a normalized value representing how much traffic (bytes) the VM will handle. Therefore, in this case, instead of counting the number of requests, the amount of traffic in the VM is taken into account.

Finally, the *Pending Request* policy can be applied by setting *lbmethod* variable to *bybusyness*. It keeps track of how many requests are currently assigned to each VM and a new request is assigned to the worker with the lowest number of active requests.

These policies are thoroughly studied later on in Sect. 3.4.

3 Results

The following section presents an analysis of the H-PC behavior and the performance of the cloud framework. In doing so, we measure the scalability using the response time metric. Next, we give the results obtained to guarantee SLA agreement. System performance was measured using the throughput metric. With the same metric, we provide a means for controlling the saturation of the cloud framework. We also measure the scheduling policies and system reliability.

Application stress tests via HTTP requests were performed using the Apache JMeter [34] tool, which was used to measure performance and functional behavior. These requests represent both patients consulting or introducing their data and the clinicians using the H-PC application.

All tests scenarios were performed in a real system using the traffic generated with the Apache JMeter tool.

The scheduler (Load Balancer) is configured to distribute the requests across the three VMs depending on a predefined factor. OpenStack VMs (VM1 and VM2) is assigned an initial *lbfactor* of 40, and 20 for the external VM (VM3). This value represents the strengths of the VMs, and consequently, a higher *lbfactor* leads to more requests being assigned to the VM. It is set this way because VM1 and VM2 are in

Table 2 Test plan requests

# Users	Total Requests		
	T1 Req/user	10 Req/user	100 Req/user
50	50	500	5000
100	100	1000	10000
200	200	2000	20000
400	400	4000	40000
800	800	8000	80000
1600	1600	16000	160000

the same site as the scheduler, and have greater Memory capacity. This way we force the local VMs to be scheduled more often than VM3.

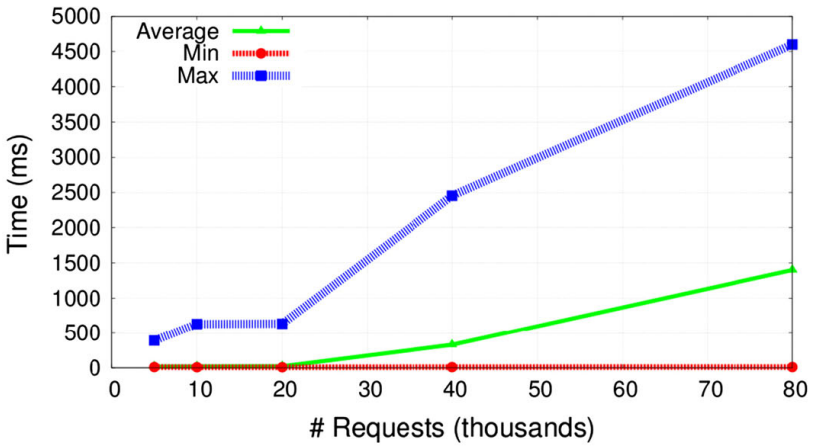
Requests were generated according to the test plan in Table 2. The test plan was performed with different sets of VMs enabled: VM1, VM2, and VM3 enabled, VM1 and VM3 enabled, VM1 enabled, and VM3 enabled. The test plan consisted of six tests with three different variations, with 1, 10, or 100 requests per user. The ramp-up period was set at 50 s, which means that all users would be running within that period of time. Also, the time between users is constant, and therefore, they are evenly distributed.

Similar results were obtained across all the low load tests. The average response time ranged between 16 and 27 ms when using only the OpenStack VMs (VM1 and VM2), and between 68 and 72 ms when using only the external VM (VM3). We also observed that the maximum response time varied between different repetitions of the same test when working with low loads. This is due to the unpredictable and heterogeneous usage of the networks by external agents. Furthermore, we must increase the stress of the overall resources to appreciate its usage effectively when they are high and evenly loaded. For these reasons, from here on, the experiments were focused on the high load tests.

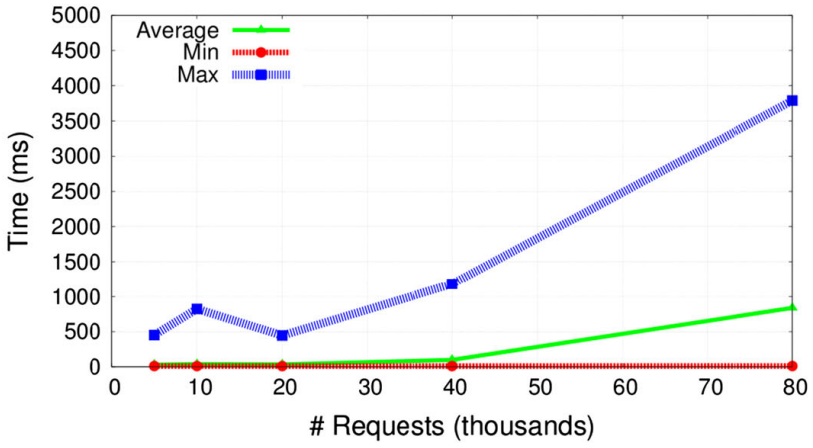
3.1 Response time

The first tests were performed using a single VM (VM1 from site 1) and performing 100 requests per user. Figure 4a shows the evolution of the response time (in ms) when increasing the number of users and, therefore, the number of requests, was increased. It can be seen that saturation of the system begins when requests are scaled over 20,000.

The next test was performed using all three available VMs and performing 100 requests per user. In Fig. 4b, the results are shown in terms of the minimum, maximum, and average system response time when the number of users increases from 50 to 800. It can be seen how increasing the number of VMs increases the performance of the whole system by reducing the mean response time to 500 ms. Even though at 40,000 requests, the system starts to overload, it still maintains an averaged response time below 1 s.



(a) 1 VM.



(b) 3 VM.

Fig. 4 Evolution of response time (min, max, and average)

3.2 SLA agreement

Similar tests to the ones performed in Fig. 4 are shown in Fig. 5. It shows the evolution of the average, the median and the 90 % line evolve. The 90 % line (or 90th percentile) is the value below which 90 % of the samples fall. That means that 90 % of the overall requests were processed in less than the given value. This metric is more meaningful than the maximum, minimum, median, or the average value in terms of SLA. As we do not have absolute control over the Internet network, there can be momentary fluctuations that can greatly affect the results in terms of the maximum time required to process a request through a test. For this reason, having a metric that takes most of the samples into account can offer a more reliable estimation of the performance of the system.

From Fig. 5, it can be seen that the system reaches its maximum performance at some point between 40,000 and 80,000 requests (i.e., between 400 and 800 users), as the 90 % line sharply increases until a certain point that endangers the compliance of

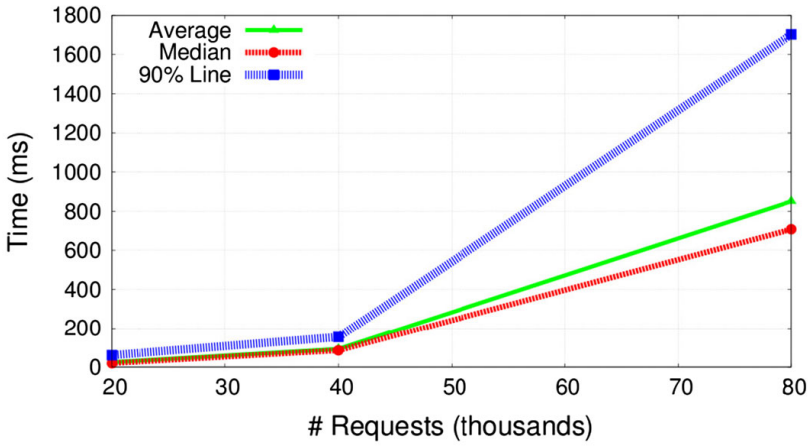


Fig. 5 Evolution of response time (average, median, and the 90 % line) when using three VMs

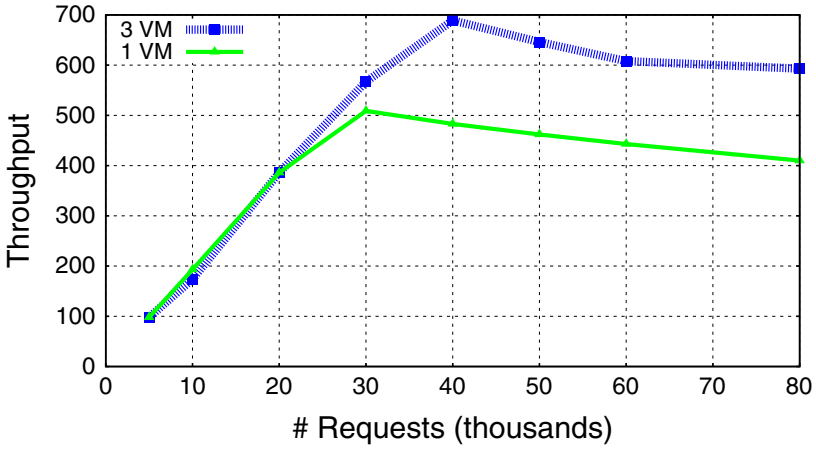


Fig. 6 Evolution of system throughput when using three VM

the SLA. These tests (i.e., metrics) are not useful in determining this point. Further metrics, like throughput, will help to find this.

3.3 System throughput and saturation

Another performance metric is the throughput. It is not as widely used for measuring QoS as response time. However, throughput is widely used for measuring system performance in terms of the number of requests served per unit of time. On this occasion, throughput can be defined as:

$$\text{Throughput} = \frac{number\ of\ requests}{time} \tag{3}$$

Figure 6 shows the evolution of the system throughput when using one single VM and when using all three VMs. When using three VMs, we can appreciate a constant throughput increment when the number of requests increases until a maximum of 40,000 requests, where throughput starts falling. The same behavior occurs when using a single VM, but the maximum performance is reached around the 30,000 requests.

Although no formal or heuristic method to manage variability is given in the paper, this figure show us that it can also be possible to determine the degree of variability offered by the system empirically.

From Fig. 6, we can obtain the turning point where the system starts saturating, and therefore, its performance starts to drop (beyond this number of requests, SLA is not guaranteed). We can also appreciate how the overall performance of the system increases when more VMs are active. In this case, we obtain a performance increment of 35.44 % when two more VMs are added to the system. The performance increment is not proportional to the added computing resources. This is due to the delay introduced by the remote communication between sites, which is often the bottleneck. In addition, VM3 is less powerful than VM1 and VM3. In any case, this result suggests the deployment of work first to local, and then to remote, VMs.

This behavior is consistent with the one presented in [21], where a similar system was designed through a mathematical approach using queuing theory-based model. In that model, the system became overloaded at some point, which led to a steep increase in the waiting time. This problem was solved by adding more servers, and hence more computing capacity, to the system. In this case, we face a similar situation where our system becomes overloaded, leading to a significant increase in the response time and a decrease in the system throughput. Also, it can be seen that adding more VMs to the system moves this turning point and extra load is required to overload the system.

3.4 Scheduler

The following tests were focused on testing the three different scheduling policies. The Request Counting Policy is referred to as *byrequests*, Weighted Traffic as *bytraffic*, and Pending Request as *bybusyness*. Each policy was tested through four tests: with 200, 400, 600, and 800 concurrent users performing 100 requests each, and a ramp-up period of 50 s.

Figure 7 shows the evolution of the 90 % line with respect to each policy. It can be seen that *bybusyness* was the best policy. For 80,000 requests, *bybusyness* obtained a 14.94 % time reduction compared with *bytraffic* and 29.06 % over *byrequests*. Past that point, all three policies were overloaded. However, *bybusyness* still continued to give the best performance. In all the cases, the system reached its maximum capacity at some point beyond 40,000 requests, as the 90 % line started increasing. This means that the system was overloaded and, therefore, requests took longer to be processed.

To confirm this, further measurements were performed for the *bybusyness* policy. Figure 8 shows the average, minimum, and median time (in ms) and the system throughput for the *bybusyness* policy. Looking at the system throughput, it can be clearly seen how performance starts decreasing over 40,000 requests.

3.5 Reliability

The architecture was based on a reliable design to minimize the fail probability of the system. Hence, redundancy was applied to the critical components of the system architecture.

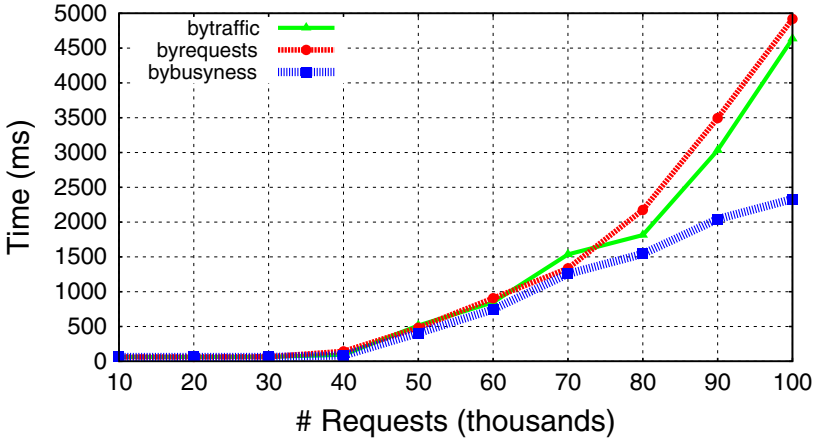


Fig. 7 Evolution of 90 % line with different scheduling policies

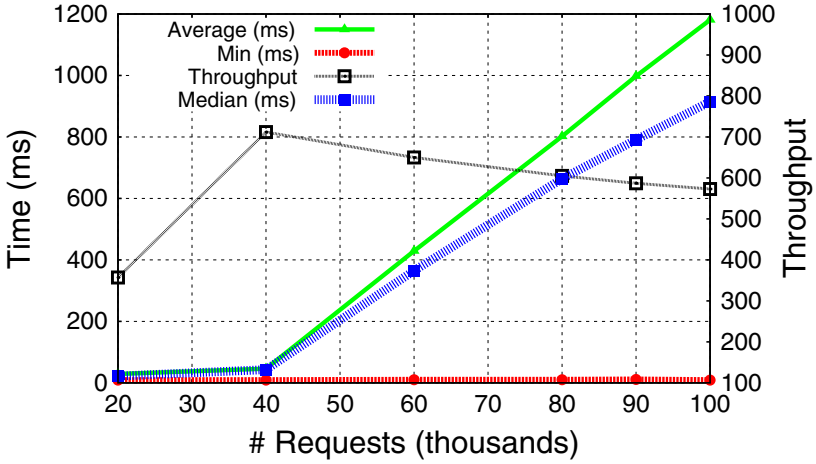


Fig. 8 Evolution of different metrics with the *bybusyness* scheduling policy

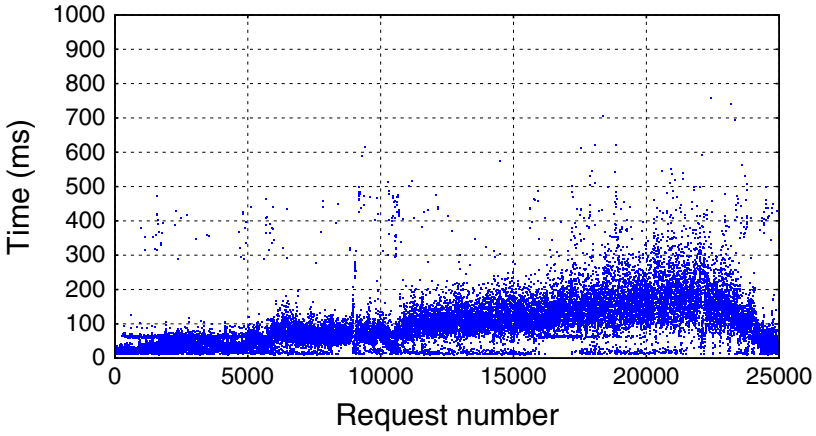


Fig. 9 Evolution of system response time when using three VMs

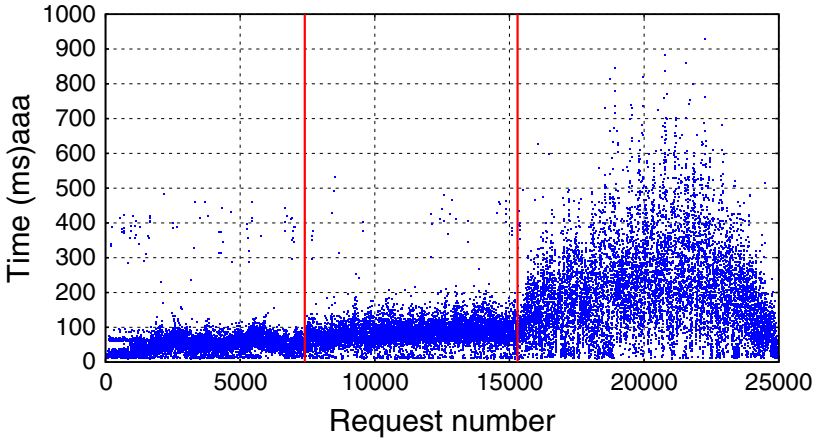


Fig. 10 Evolution of system response time when VMs fail

That said, the next test was focused on testing the fault-tolerance capabilities of the system. In this scenario, the evolution of the response time was considered in two different situations. In the first (Fig. 9), the system was running normally using the three available VMs. In the second one (Fig. 10), the first OpenStack VM failed at the 10th second and the external VM ceased to operate at the 20th second. The vertical lines indicate the moment when each VM failed. The test was configured with a total of 500 users with a ramp-up period of 30 s. Each user sent 50 requests to the system, thus a total of 25,000 requests were made in each test.

Figure 10 shows how the system performance significantly decreased when 2 VMs were disabled at different time slices. Even though the response time increased significantly, the percent error stayed at 0.00 %, which implies that no user request was lost or incorrectly responded to. These results verify the good reliability of our design of the system, and prove the effectiveness of the load balancer managing unstable situations.

4 Conclusions

The results indicate that H-PC was successfully designed, implemented, and used in the context of telemonitoring. It also supported the usage of text messaging and e-mailing in improving a) the control of patients and b) the management of time and patients by clinicians, and thus the optimization of health care resources and the reduction of patient waiting lists. We measured the H-PC computing boundaries and proved its good behavior and novel cloud design for deploying in clinical centers. We also proved the good results in scalability, performance, reliability, and variability.

We believe that H-PC can also be adapted to other detoxification programs, such as for alcoholism or drug addiction. Furthermore, treatment of chronic diseases, such as diabetes, obesity, cancer [29], or HIV [35], will benefit from new and effective e-care solutions.

Our design still has a single point of failure (SPOF), this being the Master (Scheduler) VM. Although the Master VM is quite robust, it is imperative to avoid any SPOF

when implementing a truly fault-tolerant design. For this reason, as part of our future work, we plan to implement a load-balancing replication through the use of DNS, providing sites with multiple IPs. This will influence the question of availability. Power awareness is another important issue we plan to add to the system.

It is also planned to address other key issues in our future work. Security and privacy are critical topics to be taken into account in cloud-based systems and health-related platforms. We intend to study and address these topics thoroughly in the near future. In this regard, we also plan to design mechanisms to determine the integrity of the transmitted data, as any transmission error may change the values, causing undesired adverse effects.

Section 2.1 shows that the H-PC design currently works using only one GSM modem. As part of our future work, we plan to operate using multiple modems with SIM replication. We also want to perform stress tests to determine how many SMS messages and e-mails can be sent/received per second. In this respect, we plan to study the implementation of priority assignments in our scheduling policies. This way, values above a certain threshold (for example, that exceed the target blood pressure limits), could be assigned a high priority and be processed immediately. Scaling the number of sites and providing the system with a power-aware policy are also among our future plans.

We also intend to scale the database system and incorporate big data techniques to perform statistical analysis of the data. This way, trends could be established by analyzing past data to predict near future values. Machine learning and data mining are very interesting methods that could give the proposed system great value and could also allow us to perform more realistic testing based on human behavior.

Also, further usability testing by medical staff should be done to ensure independent users. Therefore, we plan to invite medical staff from other hospitals and regions to test the H-PC tool.

Moreover, we plan to test this system in a public cloud like Amazon, to perform a study of the costs of running the system and provide feasible solutions on how to support these costs.

Acknowledgments This work was supported by the MEyC under contract TIN2011-28689-C02-02. Some of the authors are members of the research group 2009 SGR145, funded by the Generalitat de Catalunya.

References

1. Khazaei H, Misic J, Misic V (2012) Performance analysis of cloud computing centers using m/g/m/m+r queuing systems. IEEE Trans Parallel Distrib Syst 23(5):936–943
2. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M (2010) A view of cloud computing. Commun ACM 53(4):50–58
3. Craig R, Mindell J (eds) (2006) Health survey for England 2006. Her Majesty's Stationery Office, London
4. NHS Information Centre. Quality and outcomes framework 2008/09. Online GP practice results database. http://www.qof.ic.nhs.uk/
5. Law MR, Morris JK, Wald NJ (2009) Use of blood pressure lowering drugs in the prevention of cardiovascular disease: meta-analysis of 147 randomised trials in the context of expectations from prospective epidemiological studies. BMJ 338:b1665

6. Dickinson HO, Mason JM, Nicolson DJ et al (2006) Lifestyle interventions to reduce raised blood pressure: a systematic review of randomized controlled trials. J Hypertens 24:215–33
7. Green BB, Cook AJ, Ralston JD et al (2008) Effectiveness of home blood pressure monitoring, web communication, and pharmacist care on hypertension control: a randomized controlled trial. JAMA. 299(24):2857–2867. doi:10.1001/jama.299.24.2857
8. Pare G, Jaana M, Sicotte C (2007) Systematic review of home telemonitoring for chronic diseases: the evidence base. J Am Med Inform Assoc 14:269–77
9. Pickering TG, Miller NH, Ogedegbe G et al (2008) Call to action on use and reimbursement for home blood pressure monitoring: a joint scientific statement from the American Heart Association, American Society of Hypertension, and Preventive Cardiovascular Nurses Association. Hypertens 52:10–29
10. Ohkubo T, Imai Y, Tsuji I et al (1998) Home blood pressure measurement has a stronger predictive power for mortality than does screening blood pressure measurement: a population-based observation in Ohasama. Jpn J Hypertens 16:971–975
11. Bobrie G, Chatellier G, Genes N et al (2004) Cardiovascular prognosis of masked hypertension detected by blood pressure self-measurement in elderly treated hypertensive patients. JAMA 291:1342–1349
12. Aversa R, Di Martino B, Rak M, Venticinque S, Villano U (2011) Performance prediction for HPC on clouds. Principles and paradigms, Cloud Computing
13. Vishwanath KV, Nagappan N (2010) Characterizing cloud computing hardware reliability. In: Proceedings of the 1st ACM symposium on cloud computing (SoCC '10), 193–204 2010
14. Iosup A,Yigitbasi N, Epema D (2011) On the performance Variability of Production cloud services. 11th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGrid'2011), 104–113, 2011
15. Martinello M, Kaniche M, Kanoun K (2005) Web service availability: impact of error recovery and traffic model. J Reliab Eng Syst Saf 89(1):6–16
16. Beloglazov A, Buyya R (2012) Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. Concurr Comput Pract Exp 24(13):1397–1420
17. Amazon Elastic Compute Cloud (EC2). Available at: http://www.amazon.com/ec2/. 2013
18. Chappell D Introducing the Azure services platform. White Paper, October 2008.
19. Google App Engine. Available at: http://appengine.google.com. 2013
20. Calheiros R, Ranjan R, Beloglazov A, De Rose C, Buyya R (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Soft Pract Exp 41(1):23–50
21. Vilaplana F, Abella F, Filgueira R, Rius J (2013) The cloud paradigm applied to e-health. BMC Med Inf Decis Mak 13(1):35
22. Kliazovich D, Bouvry P, Khan S (2010) GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. J Supercomput
23. Lim S, Sharma B, Nam G, Kim E, Das c (2009) MDCSIM: a multi-tier data center simulation platform. In: Proceedings of IEEE international conference on cluster computing, 2009
24. Abbadi IM, Namiluko C, Martin A (2011) Insiders analysis in Cloud computing focusing on home healthcare system. In: 2011 international conference for internet technology and secured transactions, 350,357, 11–14 Dec. 2011.
25. Deng M, Petkovic M, Nalin M, Baroni I (2011) A home healthcare system in the cloud-addressing security and privacy challenges. In: 2011 IEEE International Conference on Cloud Computing (CLOUD), vol., no., pp. 549,556, 4–9 July 2011 doi:10.1109/CLOUD.2011.108.
26. McManus RJ et al (2010) Telemonitoring and self-management in the control of hypertension (TASMINH2): a randomised controlled trial. Lancet 376(9736):163–172
27. Bray EP, Holder R , Mant J , McManus RJ (2010) Does self monitoring reduce blood pressure? analysis with metaregression of randomized controlled trials. Ann Med 42(5):371–386
28. Ogedegbe G, Schoenthaler A (2006) A systematic review of the eff ects of home blood pressure monitoring on medication adherence. J Clin Hypertens (Greenwich) 8:174–80
29. Kroenke K et al (2010) Effect of telecare management on pain and depression in patients with cancer: A randomized trial. JAMA 304(2):163–171
30. Patel B, Turban S, Anderson C, Charleston J, Miller E, Appel L (2010) A comparison of web sites used to manage and present home blood pressure readings. J Clin Hypertens 12(6):389–395
31. Karwowski W, Soares MM, Stanton NA (2011) Human factors and ergonomics in consumer product design: methods and techniques (Handbook of human factors in consumer product design): needs

analysis: or, how do you capture, represent, and validate user requirements in a formal manner/notation before design?. CRC Press, Florida Chapter 26 by K Tara Smith
32. Nielsen J, Landauer T. A mathematical model of the finding of usability problems. In: Proceedings of ACM INTERCHI'93 Conference Amsterdam, ACM Press, Amsterdam, Netherlands, pp. 206–213 1993.
33. Dubrova E (2013) Fault-tolerant design. Springer, ISBN 978-1-4614-2112-2 2013
34. Apache JMeter. webpage http://jmeter.apache.org/
35. Len A et al (2011) A new multidisciplinary home care telemedicine system to monitor stable chronic human immunodeficiency virus-infected patients: a randomized study. PLoS ONE 6(1):e14515

COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE 115 (2014) 33–45

journal homepage: www.intl.elsevierhealth.com/journals/cmpb

S-PC: An e-treatment application for management of smoke-quitting patients

Jordi Vilaplana[a], *Francesc Solsona*[a,*], *Francesc Abella*[c], *Josep Cuadrado*[b], *Rui Alves*[d], *Jordi Mateo*[a]

[a] *Departament d'Informàtica i Enginyeria Industrial & INSPIRES, Universitat de Lleida, Av. Jaume II no 69, 25001 LLeida, Spain*

[b] *Hesoft Group, Partida Bovà, 15, 25196 Lleida, Spain*

[c] *Unitat de Tabaquisme de l'Hospital Santa Maria, Alcalde Rovira Roure, 44, 25198 LLeida, Spain*

[d] *Departament de Ciències Mèdiques Bàsiques & IRBLleida, Universitat de Lleida, Montserrat Roig no 2, 25008 LLeida, Spain*

ARTICLE INFO

Article history:
Received 24 August 2013
Received in revised form
6 February 2014
Accepted 11 March 2014

Keywords:
E-health
SMS
Smoking cessation
Treatment support

ABSTRACT

The main objective of this paper is to present a new program that facilitates the management of people who want to quit smoking, implemented through an e-treatment software called S-PC (Smoker Patient Control). S-PC is a web-based application that manages groups of patients, provides a bidirectional communication through mobile text messages and e-mails between patients and clinicians and offers advice and control to keep track of the patients and their status.

A total of 229 patients were enrolled in the study, randomly divided into two groups, although some variables were tested to ensure that there were no significant differences between the groups that could have an impact on the outcome of the treatment. There were no significant differences between the two groups regarding the ratio/number of males/females, tobacco dependence, co-oximetry, average cigarette consumption, current age and age when smoking started. The first group was made up of 104 patients (45.4% of the total) and followed a treatment that incorporated the S-PC tool, while the second one had 125 patients without the S-PC tool. S-PC was evaluated for its effectiveness at assisting the patients to give up smoking, and its effect on clinician time management.

74% of the S-PC group completed the treatment without relapses and remained abstinent three months after the completion of the treatment, understanding abstinence as being continuous (with no relapses allowed and co-oximetry below 1 ppm) from the day of stopping. In contrast only 45.6% of the No S-PC group completed the treatment without relapses and remained abstinent three months after completion of the treatment. The rate of admittance to the program has doubled in one year and patients went from having to wait for 3 months to be immediately admitted into the program.

* *Corresponding author at*: Departament d'Informàtica i Enginyeria Industrial, Universitat de Lleida, Av. Jaume II no 69, 25001 Lleida, Spain. Tel.: +34 973 70 27 35; fax: +34 973 70 27 02.
E-mail address: francesc@diei.udl.cat (F. Solsona).

http://dx.doi.org/10.1016/j.cmpb.2014.03.005

This therapeutic e-health program aims at maximizing the number of patients that a professional can effectively help to quit smoking. In addition, the system also detects patients who are not progressing appropriately, allowing the professional to improve their treatment parameters dynamically.

1. Introduction

Tobacco smoking is a major risk factor for active and passive smokers in certain respiratory [1] and circulatory diseases [2] as well as in some types of cancer [3,4] and infections [5], among other diseases [6–10]. Because of this, both public and private medical institutions in an increasing number of countries provide services for people that want to quit the habit of smoking.

NRT (Nicotine Replacement Therapy), in the form of nicotine patches and/or nicotine gum, is effective to treat the short-term nicotine withdrawal. Depending on the treatment and replacement, the chances that patients succeed in quitting smoking are increased between 50% and 70% upon NRT [11]. However, NRT alone becomes ineffective after about 8 weeks of starting the treatment and its effect in maintaining a smoke-free patient over a longer period of time (years) appears to be quite modest, as demonstrated by meta-analysis of different studies [12,13]. In light of this, some countries, such as the USA [14], the UK [15] and Australia [16], have published evidence-based guidelines to recommend effective tobacco cessation interventions ranging from brief instructions for quitting to extensive counseling combined with pharmaceutical adjuncts [17].

Because of the social context of tobacco smoking and extension of tobacco addiction in the population, it is not feasible to provide general cessation programs where patients are interned and only return to the streets upon completion of the program. A major issue in the treatment of addictions, and in smoking cessation, is the high relapse rates. There can be several reasons: decrease of the initial motivation, carelessness, yielding to peer pressure in certain situations (parties, dinners, situations of anxiety or relaxation), a conscious decision by the patient to start smoking again, craving, among others. A factor that can help in these situations is to develop a strong therapeutic link between the patient and the medical service. Constantly reminding them about the decision taken (to stop smoking), and making them feel closely connected to the professional team can minimize the chances of relapse. Strategies that favor this situation may have a positive impact on the obtained results regarding the long-term tobacco abstinence.

Therefore, many cessation programs combine pharmacological treatment therapy with a simultaneous psychological treatment to control the progress and reinforce the motivation of the patient. This following can be done on an individual basis [18], in the context of group therapy [19], or via long distance support through phone calls [20]. The increasing number of people taking advantage of public and private cessation programs overloads these programs and decreases their efficacy. This is so because the psychological part of the treatment is often as important as the pharmacological treatment in order for a patient to quit smoking. A previous study proved that individual counseling, combined with telephone counseling were associated with higher 52-week abstinence rates than telephone counseling alone [21]. Previous studies have shown that social support was associated with cessation and with short-term maintenance of abstinence [22]. Therefore, there is the need to develop efficient e-health tools in order to optimize both the time spent by the clinicians that follow a patient and the efficacy of their service, minimizing the probability of relapse by the patient. In consequence, tools are needed to allow professionals follow patients without making such following too long time consuming.

Preliminary work using mobile phones showed that this type of patient was twice as likely to successfully quit smoking as patients that did not have such support [23]. A more recent study [24] confirmed that proactive telephone counseling is effective in short term reduction of cigarette consumption and in increasing the percentage of smokers that attempts to quit by more than 5%, when compared to that of people without phone counseling. Another study [25] of the same group suggests that text messaging can double the likelihood of smoking cessation when compared to patients that have neither continuous contact with their caregivers nor personalized follow-up. Such contact and follow-up are very important psychological aspects of the process of quitting smoking, because they provide support and help maintain patient motivation [26,27]. Another study [28], that aimed at determining whether mobile phone-based interventions are effective in stopping smoking, concluded that mobile phone-based text messaging smoking cessation interventions have a positive effect on long-term outcomes. Another study [29] evaluated the effectiveness of the telemedicine interventions, comparing their applications at home and in the consulting room or hospital. It showed evidence for the efficacy of household applications in clinical outcomes for chronic disease management, as for example hypertension and AIDS. In hospital applications, it was found that telemedicine was comparable to face-to-face care in emergency medicine. Therefore, psychological following appears to be an essential part in the quitting of the smoking process, as it is the case for other cessation programs.

Given that such following requires a large time investment by health professionals, it is important to have tools that automate this part of the treatment as much as possible, while maintaining or increasing the efficiency of the professionals.

Taking the facts described in the previous paragraphs into account, it was our objective to develop and benchmark the effectiveness of an e-health tool that would: (a) be generally applicable in smoking cessation treatment programs, (b) automate much of the work that needs to be done by the clinicians, (c) allow professionals to more effectively maintain a personalized support and follow-up of patients, (d) give patients the

psychological support that they require for successfully quitting smoking, and (e) decrease the time needed by clinicians to manage the patients and reduce average length of waiting lists. This tool was named S-PC (Smoker-Patient Control). An additional objective was to understand to what extent patients were satisfied with being treated using the tool.

S-PC is an e-medicine service based on a computer program that manages a central database of information on patient progression. It was benchmarked in the smoking cessation program being run at the public hospital Santa Maria in Lleida, Spain. In this paper we present the tool and its functionality, as well as the results of the benchmark and studies on patients' satisfaction. Those studies suggest that S-PC meets the objectives of its development. This tool can be freely downloaded from Hesoft Group web page.[1]

2. Background and significance

Most reported studies of mobile phone technology used for smoking cessation follow patients by at most 6 months [23–26]. This is at odds with the current study, which has followed patients for a year.

Nevertheless, the effect of S-PC on the likelihood of smoking cessation by patients appears to be comparable to that found in other studies that measure the effect of SMS on smoking cessation in the short term [23–26,34]. All such studies present likelihoods of smoking cessation that are approximately twice as high using mobile phone technology as in control groups. In some of these studies that likelihood decreases at 6 months while in others it remains at about 2 months, as is the case with ours. Studies that followed the patients for longer periods usually also considered the effect of Internet messages on improving the outcome of smoking cessation interventions.

In the "Free C" study [23], the response rate at 6 months was 92%. In "Tzelepis F" [24], the results showed that telephone-counseling participants were more likely than the controls to have attempted to stop (48.6% vs. 42.9%, $p = 0.01$) and they reduced their cigarette consumption (16.9% vs. 9.0%, $p = 0.0002$). In the "*Free C*" [25], continuous abstinence at 6 months increased significantly in the intervention group compared to the control group (10.7% intervention vs. 4.9% control, RR 2.20, 95% CI, 1.80–2.68; $p < 0.0001$). The "Whittaker R" [28] six-months studies concluded that mobile phone interventions increased rate long-term of stopping (RR 1.71, 95% CI, 1.47–1.99). However, none of these studies lasted more than six months, while the present study was conducted over a one-year period.

What is new about the current study is the evaluation of the effect of the tool on clinician and patient's time management, waiting list reduction, and the patient satisfaction with the mobile texting intervention. As far as we know none of the other studies have performed such evaluation.

The most important applications that perform a function similar to that of S-PC are *STOMP* [32], *PMC* [33] and *txt2stop* [23,25]. Table 1 presents a comparison of the functionality of

Table 1 – Comparing S-PC with other similar tools (STOMP, PMC and txt2stop).

Program	Clinician support	Communication channel	Charts	Lists	Medical history	Custom messages	Custom alerts	Templates
STOMP	Yes	SMS	No	No	No	Yes	No	No
PMC	–	E-mail	Yes	Yes	Yes	Yes	No	No
TEXT2STOP	Yes	SMS	No	No	No	Yes	No	Yes
S-PC	Yes	SMS	Yes	Yes	Yes	Yes	Yes	Yes

[1] Hesoft Group. http://www.hesoftgroup.com.

these tools with respect to that of S-PC. S-PC is the application with the more complete set of functionalities. The physical medium used by each of the applications varies. PMC uses e-mail to exchange messages and information with the patients. *STOMP* and *test2stop* use mobile text messaging for the same effect. S-PC was set-up in this study to use mobile text messaging. Nevertheless, it can also use e-mail if that is required by the clinicians. Only S-PC and PMC create customized lists of patients, and graphically represent clinical history and treatment progression. In addition, S-PC sends warning messages to clinicians when a patient at risk is identified by the program. It also permits customizing messages at will. These two features are exclusive to S-PC.

3. Materials and methods

3.1. *Smoking cessation program at University of Lleida/Santa Maria Hospital*

The smoking cessation program of the detoxification unit at the University of Lleida/Santa Maria Hospital has been active for 6 years. Currently, the applied smoking cessation protocol has three goals for the patients. The first two, nicotine detoxification and smoking cessation, are easily achieved with appropriate pharmacological treatment. The third goal is that the patient suffers the least possible anxiety during the process, in order not to relapse when the treatment is over. This goal is currently where the clinician focuses most of his/her efforts.

Over the years, the types of patients treated in this unit have changed. Ten years ago, 90% of the patients were referred to the service via medical recommendations and due to pressing health problems. Currently, 90% of the patients seek treatment because they want to lead healthier lives and spend less money. In fact, economic reasons led to a sharp increase in the number of patients wanting to quit smoking with the unit's help in the last couple of years. Given budget restrictions that make it impossible for more personnel to be hired, patients had a three-month waiting period before being admitted to the program. In addition, many of the patients are from neighboring villages, which leads to an increase in their transportation expenses. If the reasons for quitting smoking are economic, this could threaten the continuity of the patient in the smoking cessation program. This situation made it necessary to develop an e-health tool to improve the efficiency of the staff and decrease the overall cost of treatment for the patients.

The S-PC tool was designed with this considerations in mind and benchmarked for a year in a small scale clinical study performed in the smoking cessation program of Santa Maria Hospital. Ethics approval for the study was obtained from CEIC (in catalan: Comitè Ètic d'Investigació Clínica), the Ethics Committee for the Health Region of the province of Lleida (Spain), where the participants were recruited and human experimentation was conducted. All participants signed an informed and written consent before engaging in the study.

The definition of abstinence used in the smoking cessation program of the detoxification unit at the University of Lleida/Santa Maria Hospital is understood as continuous abstinence from the last day of stopping with no relapses allowed. Self-reported abstinence is verified through the co-oximetry levels with the Fagerström test [30]. Although some tobacco guides define non-smokers from 0 to up to 10 ppm in the co-oximetry test [44], in this case and due to the low contamination levels in the city of Lleida, patients were considered to be non-smokers when the test result was below 1. Various Fagerström tests are done throughout the treatment. When patients first arrive to the unit, one test is done. Then, seven days after, the day to stop is set and a second test is performed. A week after the stopping day a third test is done, the result of which should be below 1. Finally, a last test is done at the end of the treatment.

3.2. *Patient selection*

The Tobacco Unit of the Santa Maria Hospital, where S-PC was implemented, provides service to a town of approximately 150,000 inhabitants. For this reason, the absolute numbers of patients that try to quit smoking is small. In order to be able to perform this study with sufficient statistical power, all 229 patients that required the services of the Unit after implementation of S-PC were enrolled in the study, after giving their informed consent.

The patients were divided into two groups with randomly selected patients. The first group, the intervention arm, made up of 104 patients (45.4% of the total, SMS group) followed a treatment that incorporated S-PC. The only requirement to be part of this group was to own a mobile phone and know how to use it. The second group, the control arm, formed by 125 patients (54.6% of the total, No SMS group) followed a treatment that did not include S-PC. Both groups were then tested to ensure that the following variables, which may have significant impact on the outcome of the treatment, were not significantly different between them: mean cigarette consumption (P-value, $P = 0.1$), age at which cigarette consumption started ($P = 0.8$), current age ($P = 0.16$), tobacco dependence [as measured by the Fagerström test [27]] ($P = 0.86$) and co-oximetry [level of CO exhaled] ($P = 0.39$). There are also no significant differences between the two groups regarding the "number of males/number of females" ratio ($P = 0.14$). Both groups are not significantly different regarding these variables ($P > 0.05$). These are important controls to avoid that results of the study are biased by possible confounding factors.

Fig. 1 illustrates the procedure of the study, with the two groups of patients involved, the type of treatment they followed and the types of communication between them and the clinician. Both groups made regular visits to the clinician. The clinician monitored the Control Group manually, whereas, for the SMS Group the clinician monitored the patients with the assistance of the S-PC tool. Additionally, the SMS Group had an additional form of communication through mobile text messages as positive reinforcement and test questions.

All patients that belonged to the SMS group and had an were sent and answered the satisfaction questionnaire about the use of S-PC during their treatment. 69 patients answered this survey. This group was statistically tested to ensure that it was representative of the larger SMS-group.

Table 2 – Patient characterization. Statistical differences between the two groups.

	SMS group	No SMS group	P
Consumption	25.26 ± 11.07	27.98 ± 14.22	0.11
Fagerström Test	5.38 ± 1.95	5.41 ± 2.07	0.86
Co-oximetry	21.36 ± 11.55	20.60 ± 11.70	0.39
Current age	43.04 ± 9.2	44.7 ± 9.5	0.16
Start smoking age	16.16 ± 2.9	16.29 ± 3.5	0.81

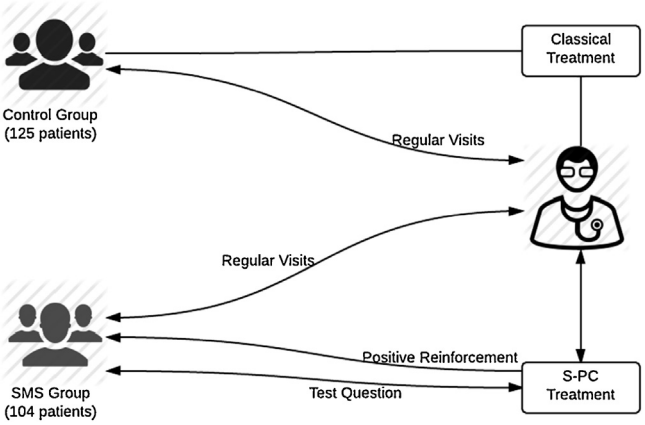


Fig. 1 – Procedure of the study illustrating the two groups of patients and their relations with the clinician.

3.2.1. *Statistical testing*

Mainly we compare patients treated using the S-PC protocol (SMS group) with patients treated using the classical protocol (No SMS group). SPSS [31] was used for statistical analysis. A χ^2 test was used to determine the degree of statistical significance of the differences between patient and control groups with regard to non-numerical variables. For example, comparing how significantly different is the percentage of patients that quit smoking in each group, or how different two groups are with regard to their percent sex composition is done through this test. A Student-*t* test was used to determine the degree of statistical significance of the differences between patient and control groups with regard to numerical variables. For example, differences in co-oximetry, number of cigarettes, ages, etc. The significance level or *p*-value (*P*) was set to 0.05. In our case, this measures the probability that two sets of patients being compared with respect to a given variable are similar. Table 2 describes the statistical differences between the two groups.

3.3. *S-PC design*

The design of S-PC was made following stringent usability and user-friendliness criteria, after an exhaustive analysis of (a) other applications with partial functional overlap (Table 1), and (b) the clinical requirements of the medical staff that uses S-PC.

There are five criteria that determine usability and user-friendliness [38,39]. First, a usable program must allow users to accomplish basic tasks the first time they use it (Learnability). Second, users that are familiar with the program should be able to quickly perform the tasks for which the program is required (Efficiency). Third, users should quickly re-learn how to use the program after some time without using it (Memorability). Fourth, users should not be able to make serious mistakes by using the program, and recovery from any error ought to be easy (Error rate). Fifth and final, the user should be satisfied with using the program (Emotional Response). Design principles of cloud computing were also incorporated in the design. Therefore, S-PC can be accessed from standard computers, smartphones and tablets. These criteria and the comments of six clinicians were used to design and iteratively improve the graphical user interface (GUI) of the program. Development was finished after eleven iterations. The number of clinicians and iterations was determined using Virzi's formula [40–42]. According to this formula, the number of users, n, needed to uncover N% of all problems in the interface is given by $n = \mathrm{Log}(N)/\mathrm{Log}(1-(1-p))$, assuming that the probability p of existing a problem is 0.33 [40]. Finding more than 90% of all problems requires 6 users, which is the number of clinicians that test the program through 11 iterations of change. Deployment of the application was made in Santa Maria Hospital and in two town surgeries.

3.4. *Implementation details for S-PC*

S-PC is a multiplatform, multi-language, application with a user-friendly graphical user interface (GUI) that enables easy access and utilization of all its functions by the clinician, implemented by using Javascript, CSS, JSP and XHTML. English, Spanish and Catalan languages are currently available. Additional languages can be easily added upon request. An intuitive help menu is also available, as well as a user manual. S-PC can run on any computer, operating system (Windows, Symbian, Leopard, Linux, etc.), and on any of the major web-browsers (Firefox, Explorer, Chrome, Opera, Safari, etc.). The implementation reported in this paper runs on an Intel Xeon X3430 (2.4 GHz, 8 Mbytes of cache), Main Memory of 4 GBytes, 1333 MHz and with 3 Mbits bandwidth for Internet access. It is implemented by using cloud technology, which diverts the calculations from the core of the computer where the application runs to virtual machines. This ensures greater service quality and decreases maintenance and upgrade costs.

Basic elements (see Fig. 2) are the computer, a modem, mobile text messages, mobile phones and the web server. The computer stores patients' information in a MySQL database and runs S-PC. S-PC sends text messages to the mobile phones of patients and receives their responses. The modem connects S-PC (located at the server) with the mobile phone network. Fig. 2 shows the communication flow chart between the patients and the clinicians. The patient interface with S-PC via SMS messages received in and sent from their mobile phones. Clinicians interface with SP-C via a computer and use it to communicate with the patients via SMS messaging

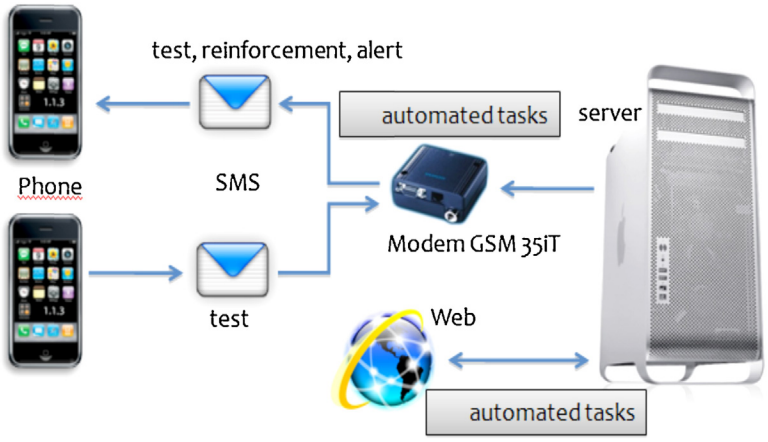


Fig. 2 – S-PC flow chart. This Figure shows both, the basic elements of the S-PC program and the flow of information between them. Basic elements are the computer, a GSM 35iT modem, mobile text messages, mobile phones and the web server. The computer stores patient information in a MySQL database and runs S-PC. The GSM 35iT modem connects S-PC to the mobile phone network. S-PC sends text messages to the mobile phones of patients and receives their responses. Clinicians interact with SP-C via web browser and use it to communicate with the patients via mobile text messaging and to analyze patient progression in the treatment program.

and to analyze patient progression in the treatment program. Patient–Clinician communication is described in more detail in Fig. 2.

S-PC has a client–server architecture (see Fig. 3). In its implementation, different technologies were applied. The presentation layer is implemented using Java Server Pages (JSP) for the structure and CSS (Cascading Style Sheets) is used in the presentation. JavaScript is ideal for verifying web forms. Therefore, we use it for validation of submitted information to the server. Java Servlets has been used in the controller layer and Java in the model layer. CSS has been used for defining the presentation of a web document (HTML, XHTML, etc.). JDBC (Java Database Connectivity) allows the connection to the database. The database is implemented in MySQL because of its performance and wide range of Application Programming Interfaces (APIs) available for it. The database stores

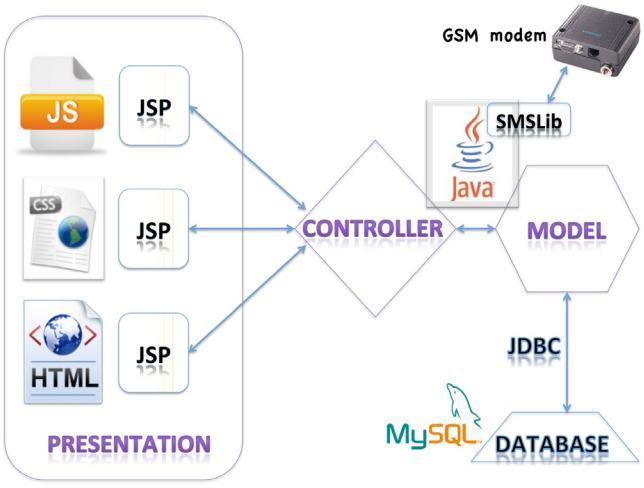


Fig. 3 – S-PC Architecture. Main technology and their relationship. All the used software is free.

information about clinical history of patients of each center, messages and messaging, treatments and clinicians, etc.

3.5. *Patient–clinician communication*

S-PC enables clinicians to pre-define, edit, adapt, and send three types of SMS messages to the patients, following guidelines that were established to improve impact of messaging [27]:

1. The same Test question is sent once a week to follow each patient's progression through the treatment: How is the treatment going? A mandatory text messaging reply is required. Possible replies are 0 (Bad), 1 (Not too Bad), 2 (Good) and 3 (Very Good). This format facilitates automated processing, storage, graphing, and analysis of the answers by the server. Missing replies and "0" answers are flagged as risk patients and forwarded to the clinicians for personalized follow-up. The system relies on the honesty of the patient's answers. Having a single question repeated weekly avoids patient's confusion. Given that the information is organized in the database, a clinician can access the list of patients at risk, and of patients that are not following the treatment appropriately, enabling a personalized treatment of each type of profile. Patients at risk are the patients who answered Bad or Not too Bad to the test question.Once the weekly test response from the patients is received, messages are processed in the following way. We recommend contacting directly with the patients at risk via mobile text message or a phone call (method used in section "An evaluation of S-PC in the context of the smoking cessation program") inviting them to visit the health center. However, as the S-PC allows any configuration, the final decision is up to the clinician.
2. Positive Reinforcement messages that support the patients and reinforce their resolution to quit smoking, assuring

Table 3 – Examples of Positive Reinforcement SMS messages.

Order	Message
1st	Welcome to the smoking cessation program. You will now start receiving messages from your clinician
2nd	Not smoking is easier than it looks. Congratulations on your decision to quit
3rd	You can now resist without smoking for a few more days
4th	Quitting smoking was an excellent decision!!
5th	Think about what you have already achieved!
6th	Breathing better? Getting less tired?
7th	Think about how much you have already improved!!!
8th	You are a great example for other people that are quitting smoking!!!
9th	You have been smokeless for 3 moths!!! Congratulations!!!
10th	Four months without smoking! Well done!
11th	More than 6 months without smoking a cigarette! What an achievement!
12th	Please remember not to smoke, OK?
13th	You'd better not smoke, alright?
14th	Almost a year without smoking!! Congratulations!!!

the patient that he/she is thoroughly followed. A variety of messages can be defined and used, in order to avoid that the patient feels that he/she is being routinely attended by a machine, which would have a negative effect on the treatment. These messages were written by the clinicians, based on previous consultation with patients regarding the positive reinforcement they would like to hear during their treatment. This approach comes from the group of expert patients as a therapeutic strategy, which has been successfully used [43], and which consists of a small group of patients who succeeded in giving up smoking and want to help other patients to do the same. Table 3 shows the 14 first messages used for positive reinforcement.

3. Alert messages that notify risk patients that they are close to relapse (by observing a worsening of the patient's responses to the Test messages), with the purpose of correcting the patient's behavior. Clinicians contact patients with this kind of message in order to have them come to the hospital for consultation.

A Patient–Clinician communication example can be seen in Fig. 4, which describes the communication flow between the two parts. Over the course of a week, various messages are sent from the clinician (S-PC) to the patient. In this example, two reinforcement messages are sent on days 2 and 4 respectively. One test question is sent on day 7 and answered by the patient with one of a range of possible options.

All messages can be defined or modified at any time in the database of S-PC. By default S-PC automatically sends messages on predefined days and schedules. Test and Positive Reinforcement messages are typically sent this way. Alternatively, a clinician can manually send a message to a patient through the system. Alert messages are typically sent this way. No patients were told that S-PC could automatically send the messages at predefined schedules.

3.6. *Treatment protocol*

We are now able to explain the e-treatment implemented by S-PC in which patient and clinician interchange SMS in a predefined protocol. The first message (Positive Reinforcement) welcomes the patient into the program, notifying that SMS monitoring will start. When and how many Positive Reinforcement SMS are sent to each patient is a decision of the professional that follows that patient. The frequency of the messages can be fully automated. However, the clinician must consider that a constant frequency could cause a negative psychological effect on the patient, derived from routine. To avoid this, a protocol that varies the number of Positive Reinforcement messages that are sent to the patients over time was developed and implemented (Table 4). Test messages are sent

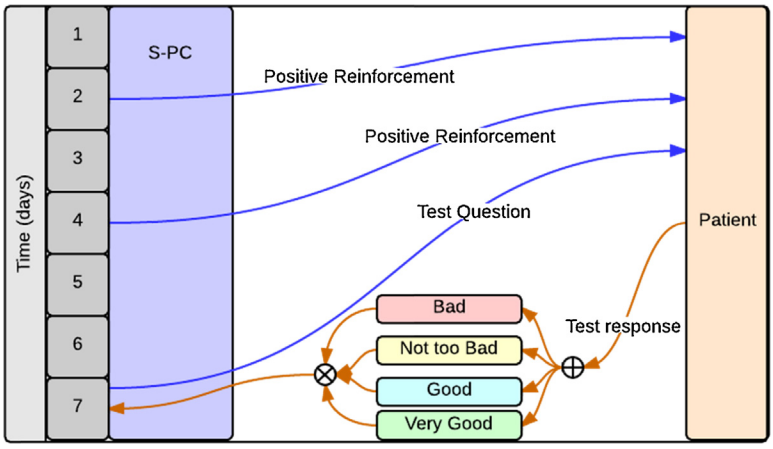


Fig. 4 – Information flow diagram between S-PC and a patient.

Table 4 – Scheduling of the positive reinforcement SMS messages. Positive reinforcement messages are delivered according to schedule shown in this table. The number of delivered positive reinforcement SMS messages sent to a patient during each week of the treatment is shown here.

Month	Week	#SMS	Month	Week	#SMS	Month	Week	#SMS	Month	Week	#SMS
1	1	2	4	13	1	7	25	0	10	37	0
	2	2		14	0		26	0		38	0
	3	2		15	1		27	1		39	1
	4	2		16	0		28	0		40	1
2	5	1	5	17	1	8	29	0	11	41	0
	6	1		18	0		30	0		42	0
	7	1		19	1		31	1		43	0
	8	1		20	0		32	0		44	1
3	9	1	6	21	1	9	33	1	12	45	0
	10	1		22	0		34	0		46	0
	11	1		23	1		35	0		47	1
	12	1		24	0		36	1		48	1

once a week. As was said before, sending an Alert message depends on the progress of each patient.

The number of SMS depends on the therapy duration. To begin with, the psychologist of the team ascertained that 2 Positive Reinforcement messages per week were sufficient to support the patient's progression. If progression of the treatment is favorable, message frequency will decrease to avoid saturation. For stronger psychological effect, it is important that test and Positive Reinforcement messages arrive to the patients at unexpected times (for example Sunday at lunch). This leads to further discussion of the treatment between the patient and his/her local support system, increasing motivation. However, care is taken so that messages are not delivered at inconvenient times (for example at work or at night, while patients are sleeping), because such delivery could contribute to treatment rejection. This strategy is designed to reinforce the sensation that the professional is following the patient at all times with great interest.

Cost of SMS messaging must also be taken into account. The scheduling for positive reinforcement shown in Table 4 takes this factor into account and was decided upon by the clinicians of the Santa Maria Hospital after some testing. The duration of the whole program is 1 year.

4. S-PC operation

S-PC can be accessed through the web-browser, using a secure login window. It is available from anywhere in the world, as long as an active Internet connection is available. The user (clinician) logs in by entering its username and password, choosing the language in which s/he wishes to work (Catalan, Spanish and English). Fig. 5 shows the access screen of the application.

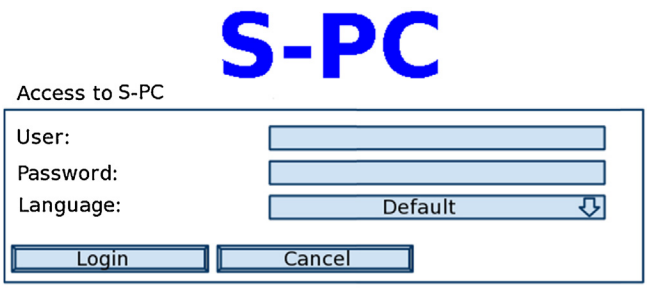


Fig. 5 – S-PC Login. This is the authentication window. Catalan, Spanish or English languages can be chosen.

Next, the user (clinician) accesses the main screen (Fig. 6), where s/he can search for the status of patients, append notes to clinical histories (yellow) or registered new patients. The main S-PC functionalities are (1) registering a new patient, (2) identification and management of lists of different types of patients, (3) create/modifying the templates of "test" and "positive reinforcement" messages, (4) deliver "alert" messages to the mobile phone of patients, (5) change user properties (as password), (6) exit, (7) return to the login screen, (8) show patient status, (9) save private clinician notes, and (10) send user feedback to improve the application. Next, mainly windows are presented.

Patient responses of the "test" messages can be visualized in pie charts (Fig. 7) and progress charts (Fig. 8). Answers are codified as follows: 0 Bad, 1 Regular, 2, Good and 3 Very Good. These charts facilitate identifying patients at risk that need personal attention:

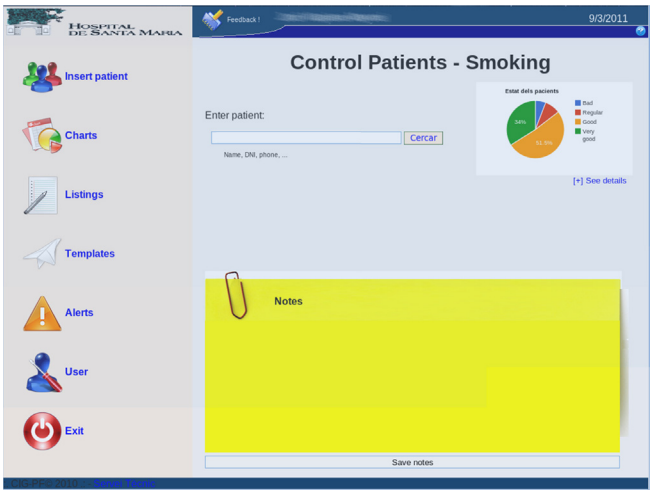


Fig. 6 – Main screen window. It can be appreciated all the functionality provided by S-PC. (For interpretation of the references to color in text near the reference citation, the reader is referred to the web version of this article.)

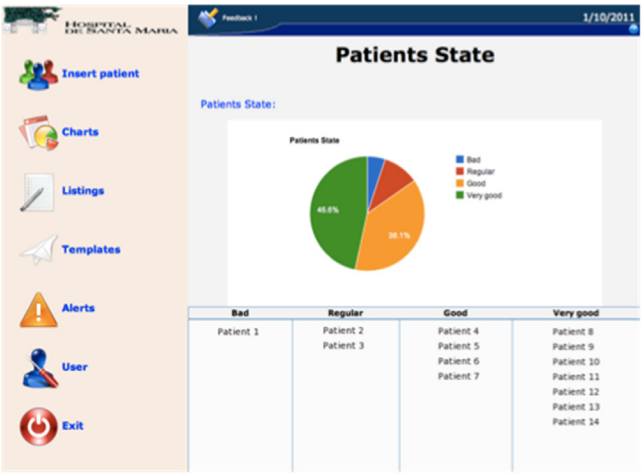


Fig. 7 – Patients State. Graphical and schematic (in %) representation of the patients state. A list of the patients in each corresponding state is also shown.

Fig. 9 – Listing patients at risk. Only the patients at risk (no response or Bad state) are listed.

1. Patients State (Fig. 7). It shows the general graphical percentiles of the current status of patients. The individual status of each patient is also displayed.
2. Patients Evolution (Fig. 8). This table shows the responses of individual patients to test questions over their treatment. In this case the number of weeks are twenty. Users can modify the number of weeks to be represented.

Many listings can be created, as for example patients at risk (see Fig. 9).

Creating and modifying templates for all kind of messages (test, positive reinforcement and alert) can be done as shown in Fig. 10.

The system also allows the clinician to select a group of patients to whom a given alert message must be send (Fig. 11).

5. Results

5.1. *S-PC effectiveness*

The effectiveness of S-PC was evaluated with respect to (a) its effect in assisting the patients to quit smoking, and (b) its effect on clinician time management, as measured by the time that patients spend in the waiting list before being admitted in the program and by the number of personal visits each patient requires.

Table 5 summarizes the results and the differences between the SMS and the No-SMS group. 74% of the SMS group completed the treatment without relapses and remained abstinent three months after the completion of the treatment. In contrast only 45.6% of the No-SMS group complete the treatment without relapses and remained abstinent three month after completion of the treatment. The groups were significantly different with respect to this issue and the treatment

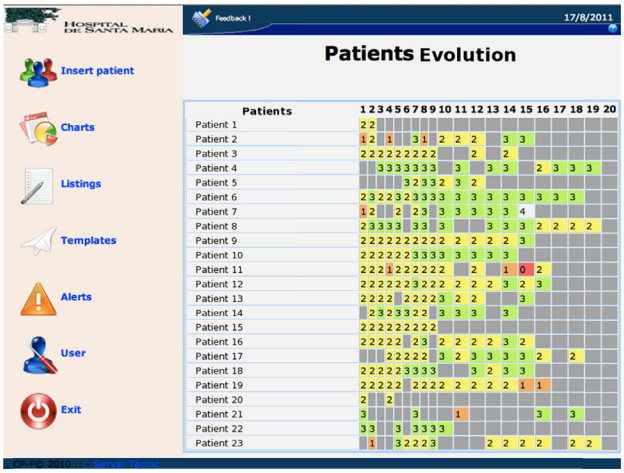


Fig. 8 – Patients evolution. It shows the evolution of the patients according to the codified state: 0 Bad, 1 Regular, 2, Good and 3 Very good.

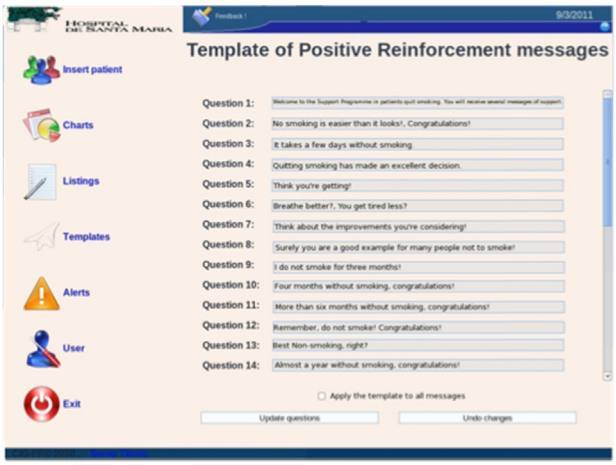


Fig. 10 – Creating template messages. We show in this window the creation of the list of Positive Reinforcement SMS messages.

Table 5 – Contingency table comparing relapses between patients of different groups. Patients in the SMS group more strictly adhered to NRT and had significantly less relapses than those in the No SMS group (χ^2 test shows significance of $P < 0.001$ in both cases). NRT differences are seen by comparing the first five rows of the table, while relapsing differences can be seen by comparing rows 6–9. There was no significant difference in the relapsing frequency with respect to sex, where $P = 0.1$, (see rows 10–13). This can be seen by analyzing the last four rows of the table.

Patient types	SMS	No SMS	Total
Strictly following NRT	52	48	100
Following NRT	34	21	55
Left NRT	18	56	74
Total	104	125	229
Strictly follow treatment	77	57	134
Smoking relapses during treatment	27	68	95
Total	104	125	229
	Men	Women	Total
Strictly follow treatment	63	71	134
Smoking relapses during treatment	55	40	95
Total	118	111	229

is significantly more successful in the SMS group than in the No-SMS group ($P < 0.001$). We found no significant influence of sex on treatment success in either of the groups ($P = 0.1$), discarding sex as a confounding factor for the effect of S-PC in the treatment of patients that are quitting tobacco.

We also note that S-PC has positive effects on the first two goals of treatment (nicotine detoxification and smoking cessation). Records of adherence by the SMS and No SMS group to Nicotine Replacement Therapy (NRT), as reported by the patients, were kept and analyzed. Comparing the SMS and the No SMS group shows that adherence to NRT was significantly greater in the former than in the later ($P < 0.001$).

To evaluate the effect of S-PC on the time management of clinicians, we compared clinician patient load before and after S-PC implementation. We also compared the time that patients spent in the waiting list and the average number of visits per patient before and after S-PC was implemented. The effect of S-PC on clinician time management was strong. 100 new patients per clinician were enrolled in the program during 2010 without S-PC. In comparison, S-PC allowed 200 new patients per clinician to be enrolled and treated during 2011. Before using S-PC, each patient attended 10 visits/year. S-PC allowed this number to decrease to an average of only 7 visits/year per patient. In addition, more time is now spent with the new patients (the ones that require more attention) and significantly less clinician time is dedicated to already enrolled patients, which are effectively accompanied through S-PC. Even though the rate of admittance to the program has doubled in one year, patients went from having to wait for 3 month to be admitted in the program to immediate admittance.

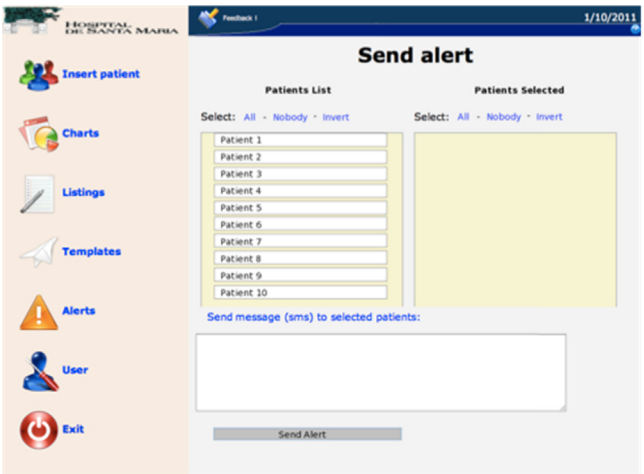


Fig. 11 – Sending Alerts. This window shows the mean for sending SMS messages to patients at risk.

5.2. *Patient satisfaction*

To further evaluate the role of S-PC during the treatment, we prepared an electronic satisfaction survey that was answered by all patients of the SMS group that used The questions were designed to evaluate the opinion of patients on the appropriateness and effectiveness of using S-PC in their treatment. The group contains representative proportions of age, sex and physical condition with respect to the complete treatment group.

95% of all patients used the application throughout the full duration of their treatment. 92% of the patients were either satisfied or completely satisfied with the support given by S-PC. However, 20% felt that they would still have managed without the system. Thus, 70% of the patients strongly agreed that the S-PC system helped them remain smoke-free, 10% had no opinion about this issue and 20% disagreed or strongly disagreed that S-PC helped them remain smoke-free. 60% of the patients agree that S-PC decreases the number of medical appointments needed during the treatment, with 25% of the patients disagreeing with this statement. When comparing S-PC to other methods used in the smoking cessation program, 70% of the patients are satisfied with this method, 25% are unsure about the comparison and less than 5% considers S-PC is not a good methodology. 96% of all patients feel that S-PC is either an adequate or a perfectly adequate tool to aid them quit smoking. Less than 10% of the users reported problems receiving or sending messages. Approximately 70% of all patients do not consider S-PC usage monotonous, with 8% having no opinion on the subject. Less than 5% of the patients felt harassed by the messaging generated by S-PC. 95% of all patients using the system felt that the clinician who took care

of them was permanently following them and taking notice of their messages and answers. Finally, 98% would accept similar systems in other health treatments.

6. Discussion

6.1. *Main findings*

S-PC is effective, useful and perceived as an added value to treatment by patients and by clinicians. Its use increases compliance to NRT to levels of 95%, reduces the probability of relapsing and smoking during treatment by more than half. In addition, more than 90% of patients regard S-PC as having added value in the treatment and see possibilities for applying similar tools in other health treatments.

The time management of clinicians was also significantly improved. The same number of clinicians is now processing, treating and following twice as many patients. The waiting list was reduced from three months to 2 days, which is the time it takes to process and enter patient information into the S-PC central database. In addition, the time dedicated to personalized clinical attention to patients that are at risk and require closer follow-up, has been significantly increased and clinicians more effectively schedule and manage that time. S-PC also avoids unnecessary travel while allowing patients to feel closely followed up by the clinician.

These results are encouraging and complement the results found in earlier studies. Abstinence levels were 14.3% higher in the SMS group than in the control group over a twelve-month period. Previous studies reached similar positive outcomes, although none of them lasted so long and the usual treatment period was six months. Abstinence rates tend to decay over time, and a longer study implies higher levels of relapse among patients. However, our trial had some limitations regarding the number of patients involved in the study. Some of the previous studies had larger participant samples, thus reducing the margin of error. Further larger studies should be performed to verify the current results.

Our findings reassert that the use of mobile phone interventions may be effective in increasing the long-term abstinence rates in smoking patients, and that positive reinforcements messages and the usage of technology-based interventions can be an effective complements to the current smoking cessation programs.

6.2. *Perspectives*

We believe that S-PC could also be successfully adapted to other chronic diseases, such as hypertension. We are working on such an application for following high blood pressure patients in the cardiology service of Hospital Santa Maria. The rationale for doing so is two-fold. Firstly, given that hypertensive patients are on the rise there is a possibility that the service will be overwhelmed and unable to provide appropriate care for all of its patients. Secondly, home blood pressure (HBP) monitoring "should become a routine component" of blood pressure measurement in the majority of patients with known or suspected hypertension [35,36], given that such readings may be better predictors of cardiovascular and renal outcomes than office readings [37].

7. Conclusion

The results indicate that S-PC was successfully designed, implemented and used in the context of the quit smoking treatment. They also support the usefulness of text messaging in improving (a) the outcome likelihood of smoking cessation interventions, (b) the management of time and patients by clinicians, and thus the optimization of health care resources and the reduction of waiting lists, and (c) the patient's perception of constant psychological support by the clinician.

Conflicts of interest

The authors claim no conflicts of interest.

Acknowledgments

This work was supported by the MEyC-Spain under contracts BFU2010-17704 and TIN2011-28689-C02-02, by Generalitat de Catalunya, through research groups 2009SGR809 and 2009SGR145 and the CUR of DIUE of GENCAT, and by the European Social Fund.

REFERENCES

[1] L. Erhardt, Cigarette smoking: an undertreated risk factor for cardiovascular disease, Atherosclerosis 205 (2009) 23–32.
[2] S.R. Orth, S.I. Hallan, Smoking: a risk factor for progression of chronic kidney disease and for cardiovascular morbidity and mortality in renal patients–absence of evidence or evidence of absence? Clin. J. Am. Soc. Nephrol. 3 (2008) 226–236.
[3] P. Boffetta, et al., Tobacco smoking as a risk factor of bronchioloalveolar carcinoma of the lung: pooled analysis of seven case-control studies in the International Lung Cancer Consortium (ILCCO), Cancer Causes Control 22 (2011) 73–79.
[4] P. Bravo, J. del Rey Calero, J. Sanchez, M. Conde, Tobacco as a risk factor in cancer of the bladder, Arch. Esp. Urol. 39 (1986) 237–240.
[5] A.S. Furber, R. Maheswaran, J.N. Newell, C. Carroll, Is smoking tobacco an independent risk factor for HIV infection and progression to AIDS? A systemic review, Sex. Transm. Infect. 83 (2007) 41–46.
[6] J.S. Sandhu, Smoking–a renal risk factor, J. Assoc. Physicians India 51 (2003) 900–902.
[7] S.S. DeBlack, Cigarette smoking as a risk factor for cataract and age-related macular degeneration: a review of the literature, Optometry 74 (2003) 99–110.
[8] O.P. Almeida, G.K. Hulse, D. Lawrence, L. Flicker, Smoking as a risk factor for Alzheimer's disease: contrasting evidence from a systematic review of case-control and cohort studies, Addiction 97 (2002) 15–28.
[9] B.B. Love, J. Biller, M.P. Jones, H.P. Adams, Bruno, A cigarette smoking. A risk factor for cerebral infraction in young adults, Arch. Neurol. 47 (1990) 693–698.

[10] T. Flensborg-Madsen, et al., Tobacco smoking as a risk factor for depression. A 26-year population-based follow-up study, J. Psychiatr. Res. 45 (2011) 143–149.
[11] L.F. Stead, R. Perera, C. Bullen, D. Mant, T. Lancaster, Nicotine replacement therapy for smoking cessation, Cochrane Database Syst. Rev. (1) (2008), http://dx.doi.org/10.1002/14651858.CD000146.pub3 (Art. N. CD000146).
[12] H. Alpert, G. Connolly, L. Biener, A prospective cohort study challenging the effectiveness of population-based medical intervention for smoking cessation, Tob. Control (2012), http://dx.doi.org/10.1136/tobaccocontrol-2011-050129.
[13] J. Etter, S. Stapleton, Nicotine replacement therapy for long-term smoking cessation: a meta-analysis, Tob. Control 15 (2006) 280–285, http://dx.doi.org/10.1136/tc.2005.015487.
[14] M.C. Fiore, A clinical practice guideline for treating tobacco use and dependence: 2008 update. A U. S. Public Health Service report, Am. J. Prev. Med. 35 (158) (2008) e76.
[15] R. West, A. McNeill, M. Raw, Smoking cessation guidelines for health professionals: an update Health Education Authority, Thorax 55 (987) (2000) e99.
[16] N. Zwar, R. Richmond, R. Borland, et al., Smoking cessation guidelines for Australian general practice, Aust. Fam. Physician 34 (461) (2005) e6.
[17] C. Fong-ching, H. Teh-wei, L. Shu-ying, Y. Po-tswen, C. Kun-yu, H. Mei-ling, Quit smoking advice from health professionals in Taiwan: the role of funding policy and smoker socioeconomic status, Tob. Control 19 (2010) 44–49.
[18] T. Lancaster, L.F. Stead, Individual behavioural counseling for smoking cessation, Cochrane Database Syst. Rev. (2) (2005), http://dx.doi.org/10.1002/14651858.CD001292.pub2 (Art. N. CD001292).
[19] L.F. Stead, R. Perera, T. Lancaster, Group behaviour therapy programmes for smoking cessation, Cochrane Database Syst. Rev. (2) (2005), http://dx.doi.org/10.1002/14651858 (Art. N. CD001007. CD001007.pub2).
[20] L.F. Stead, R. Perera, T. Lancaster, Telephone counseling for smoking cessation, Cochrane Database Syst. Rev. (3) (2006), http://dx.doi.org/10.1002/14651858.CD002850.pub2 (Art. N. CD002850).
[21] J.M. Ramon, I. Nerin, A. Comino, C. Pinet, F. Abella, J.M. Carreras, M. Banque, A. Baena, S. Morchon, A. Jimenez-Muro, A. Marqueta, A. Vilarasau, R. Bullon, C. Masuet-Aumatell, A multicentre randomized trial of combined individual and telephone counselling for smoking cessation, Prev. Med. 57 (September (3)) (2013) 183–188.
[22] R. Mermelstein, S. Cohen, E. Lichtenstein, J.S. Baer, T. Kamarck, Social support and smoking cessation and maintenance, J. Consult. Clin. Psychol. 54 (4) (1986) 447–453, http://dx.doi.org/10.1037/0022-006X.54.4.447.
[23] C. Free, R. Whittaker, R. Knight, T. Abramsky, A. Rodgers, I.G. Roberts, Txt2stop: a pilot randomised controlled trial of mobile phone-based smoking cessation support, Tob. Control 18 (2009) 88–91.
[24] F. Tzelepis, C.L. Paul, J. Wiggers, R.A. Walsh, J. Knight, S.L. Duncan, C. Lecathelinais, A. Girgis, J. Daly, A randomised controlled trial of proactive telephone counselling on cold-called smokers' cessation rates, Tob. Control 20 (2011) 40–46.
[25] C. Free, R. Knight, S. Robertson, R. Whittaker, P. Edwards, W. Zhou, A. Rodgers, J. Cairns, M.G. Kenward, I. Roberts, Smoking cessation support delivered via mobile phone text messaging (txt2stop): a single-blind, randomised trial, Lancet 378–9785 (2011) 49–55.
[26] Institut Català del consum de tabac (Generalitat de Catalunya). Guies de pràctica clínica. Detecció i tractament del consum de tabac. http://www20.gencat.cat/portal/site/canalsalut/ (accessed 15.10.11), 2009.
[27] K.C. Davis, J.M. Nonnemaker, M.C. Farrelly, J. Niederdeppe, Exploring differences in smoker's perceptions of the effectiveness of cessation media messages, Tob. Control 20 (2011) 26–33.
[28] R. Whittaker, H. McRobbie, C. Bullen, R. Borland, A. Rodgers, Y. Gu, Mobile phone-based interventions for smoking cessation, Cochrane Database Syst. Rev. (11) (2012), http://dx.doi.org/10.1002/14651858.CD006611.pub3 (Art. No.: CD006611).
[29] W.R. Hersh, M. Helfand, J. Wallace, D. Kraemer, P. Patterson, S. Shapiro, et al., Clinical outcomes resulting from telemedicine interventions: a systematic review, BMC Med. Inform Decis. Making 1 (1) (2001) 5.
[30] T.F. Heatherton, L.T. Kozlowski, R.C. Frecker, K.O. Fagerström, The Fagerström test, for nicotine dependence: a revision of the Fagerstrom Tolerance Questionnaire, Br. J. Addict. 86 (1991) 1119–1127.
[31] L. Fiddler, L. Hecht, E.E. Nelson, E.N. Nelson, J. Ross, SPSS for Windows Version 13.0: A Basic Tutorial. http://www.doocu.com/pdf/read/15204 (accessed 09.04.14).
[32] https://nihi.auckland.ac.nz/page/completed-research/stomp-stop-smoking-mobile-phones-trial (accessed 09.04.14).
[33] L. Lenert, R.F. Muñoz, J. Stoddard, K. Delucchi, A. Bansod, S. Skoczen, E.J. Pérez-Stable, Design and pilot evaluation of an internet smoking cessation program, J. Am. Med. Inform. Assoc. 10 (January–February (1)) (2003) 16–20.
[34] R. Whittaker, R. Borland, C. Bullen, R.B. Lin, H. McRobbie, A. Rodgers, Mobile phone-based interventions for smoking cessation (Review), Cochrane Libr. 4 (2009) (Art. N. CD006611).
[35] T.G. Pickering, N.H. Miller, G. Ogedegbe, et al., Call to action on use and reimbursement for home blood pressure monitoring: a joint scientific statement from the American Heart Association, American Society of Hypertension, and Preventive Cardiovascular Nurses Association, Hypertension 52 (2008) 10–29.
[36] T. Ohkubo, Y. Imai, I. Tsuji, et al., Home blood pressure measurement has a stronger predictive power for mortality than does screening blood pressure measurement: a population-based observation in Ohasama, Jpn. J. Hypertens. 16 (1998) 971–975.
[37] G. Bobrie, G. Chatellier, N. Genes, et al., Cardiovascular prognosis of "masked hypertension" detected by blood pressure self-measurement in elderly treated hypertensive patients, J. Am. Med. Assoc. 291 (2004) 1342–1349.
[38] J. Nielsen, Usability 101: Introduction to Usability. http://www.useit.com/alertbox/20030825.html (accessed 09.04.14).
[39] W. Karwowski, M.M. Soares, N.A. Stanton, Human Factors and Ergonomics in Consumer Product Design: Methods and Techniques (Handbook of Human Factors in Consumer Product Design): Needs Analysis: Or, How Do You Capture, Represent, and Validate User Requirements in a Formal Manner/Notation before Design, CRC Press, 2011 (Chapter 26 by K. Tara Smith).
[40] J. Nielsen, T. Landauer, A mathematical model of the finding of usability problems, in: Proceedings of ACM INTERCHI'93 Conference, Amsterdam, Netherlands, ACM Press, 1993, pp. 206–213.
[41] R. Virzi, Refining the test phase of usability evaluation: how many subjects is enough? Hum. Factors (34) (1992) 457–468.

[42] C. Turner, J. Lewis, J. Nielsen, Determining Usability Test Sample Size. International Encyclopedia of Ergonomics and Human Factors, CRC Press, Boca Raton, FL, 2006, pp. 3084–3088.

[43] F. Abella, A. Vilarasau, A. Perera, J.L. Cruz, The group of expert patients as a therapeutic strategy in tobacco cessation, Rev. Clin. Med. Fam. 6 (n.2) (2013) 118–119, http://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S1699-695X2013000200009

[44] C. Pereiro, E. Becoña, R. Córdoba, J. Martínez, C. Pinet, Tabaquismo. Guías Clínicas SOCIDROGOALCOHOL, 2008, http://www.socidrogalcohol.org/index.php?option=com_docman&task=doc_details&gid=72&Itemid=19

Chapter 4

Global discussion of results

A model based on queuing theory to study the Quality of Service (QoS) in Cloud computing was presented. Cloud platforms were modeled with an open Jackson network that could be used to determine and measure the QoS guarantees the Cloud can offer regarding the response time. The analysis was performed according to different parameters, such as the arrival rate of customer services and the number and service rate of processing servers, among others. When scaling the system and depending on the types of bottleneck, the model provided the best option to guarantee QoS. Moreover, the response time of the system was consistent with extreme situations, and the similarities in shapes between the simulation (obtained with Sage 5.3 mathematical software [1]) and real results (obtained with OpenStack) proved the good behavior of the model. These results confirmed the usefulness of the model presented for designing real Cloud computing systems.

A power-aware scheduling policy algorithm called Green Cloud (GreenC) for heterogeneous Cloud systems was presented. An initial test case was performed by modelling the policies to be executed by a solver that demonstrated the applicability of this proposal for saving energy and also guaranteeing the QoS. The proposed policy was implemented using the OpenStack Cloud platform and the obtained results showed that energy consumption was significantly lowered by applying GreenC to allocate virtual machines to physical hosts.

[1] Sage http://www.sagemath.org

Moreover, two eHealth applications were presented. The Hypertension Patient Control (H-PC) tool was designed to improve current hypertension treatments by means of telemonitoring. A novel and efficient cloud-based platform managing H-PC with QoS was also proposed in this thesis. The results showed that H-PC was successfully designed, implemented, and used in the context of telemonitoring. It also supported the usage of text messaging and e-mailing to telemonitor the patients to improve the communication between them and clinicians. The H-PC computing boundaries were measured and proved its good behavior and novel Cloud design. Good results in scalability, performance, reliability, and variability were also proved.

The Smoking Patient Control (S-PC) tool was also presented. This tool was aimed to improve current smoking treatments. A study with real patients was performed in order to test its effectiveness in a real scenario. A total of 229 patients were enrolled in the study, randomly divided into two groups, although some variables were tested to ensure that there were no significant differences between the groups that could have an impact on the outcome of the treatment. The first group was made up of 104 patients (45.4% ofthe total) and followed a treatment that incorporated the S-PC tool, while the second one had 125 patients without the S-PC tool. S-PC was evaluated for its effectiveness at assisting the patients to give up smoking, and its effect on clinician time management. 74% of the S-PC group completed the treatment without relapses and remained abstinent three months after the completion of the treatment. In contrast only 45.6% of the No S-PC group completed the treatment without relapses and remained abstinent three months after completion of the treatment.

Chapter 5

General conclusions and future directions

This chapter summarizes the research work on Cloud systems and architectures applied to eHealth that are presented in this thesis and highlights the main findings. It also discusses open research problems in the area and outlines a number of future research directions.

5.1 Conclusions

On the one hand, this thesis proposed and investigated multiple Cloud architectures and models and implemented them using different techniques and scenarios, such as queueing theory, nonlinear programming and Cloud simulation. Having these models, a Cloud infrastructure was developed using OpenStack. Moreover, a scheduling mechanism called GreenC was developed to achieve energy efficiency while maintaining the performance of the Cloud system. This policy obtained a reduction in energy consumption of up to 23% compared with the default policy.

On the other hand, two eHealth telemonitoring applications were implemented and deployed on the developed Cloud infrastructure. The first of these, Smoking Patient Control (S-PC), was deployed and tested in two hospitals with a total of 229 real patients, where 104 patients were treated using the S-PC tool and 125 without it.

The results showed that 74% of the patients treated using the S-PC tool successfully completed the treatment without relapses, while only 45.6% of those who followed the conventional treatment successfully achieved long-term abstinence. The second application, Hypertension Patient Control (H-PC), was successfully designed and implemented, and the computing boundaries were measured, proving its good behavior in terms of *performance* and *scalability*. These performance results are also directly applicable to the S-PC tool.

5.2 Future research directions

Despite the contributions of the current thesis in the Cloud and eHealth fields, there are a number of open research challenges to address in order to further advance in these areas.

Regarding the telemedicine applications, they will be deployed in more hospitals and healthcare centers in order to test their effectiveness with a larger test group. On the one hand, the *Sant Joan de Déu*[1] hospital in Manresa has agreed to test the S-PC tool with its patients. More hospitals and healthcare centres are being contacted to expand the testing both in the number of patients and the geographic location. On the other hand, the H-PC tool has entered the testing phase with real patients in three different centers: the *Arnau de Vilanova*[2] and *Santa Maria*[3] hospitals in Lleida and the *Clinic*[4] Hospital in Barcelona. Moreover, a smartphone app is being developed for both tools. This way, communication can be done directly using mobile devices and it also allows much more information to be displayed. The eHealth field applied to mobile devices, referred to as mHealth, offers great opportunities for improving current healthcare treatments and processes, becoming an interesting research field. One example would be that patients may sometimes provide wrong information when manually entering the data. New devices can directly send the measurements through the mobile phone without the need to be manually introduced by the patient, avoiding

[1]Sant Joan de Déu hospital webpage: http://www.althaia.cat/
[2]Arnau de Vilanova hospital webpage: http://www.icslleida.cat/
[3]Santa Maria hospital webpage: http://www.gss.cat/
[4]Clinic Hospital webpage: http://www.hospitalclinic.org/

these errors.

Another future research path is to explore additional treatments and diseases, such as diabetes, dietary treatment or alcoholism, where these and similar techniques could be applied. Expanding the number of healthcare centers, patients and treatments entails an increase in the complexity of the underlying infrastructure. Hence, additional performance and scalability tests will be undertaken in order to avoid performance issues.

Another different, but related, direction involves the analysis of social networks that are increasingly used by young people. The work done during the doctoral stay (see Appendix A) showed that studying and analyzing social networks like Twitter can greatly help in major health-related issues like cyber-bullying. Moreover, additional techniques and studies can be performed, such as using social networks to group together patients that are following the same treatment, preventive actions associated with social networks and social behaviour linked to the use of new technologies.

5.2.1 Internet of Things

Increasingly, more "smart" devices or "things" are being manufactured and available to the population. The network of electronic devices with embedded software and sensors is called the *Internet of Things* (IoT). The IoT can generate a new paradigm where data can be gathered much more easily, frequently and accurately. Hence, it can have a great impact in fields like healthcare, where these devices could greatly help to monitor patients, prevent and detect emergency situations and provide valuable feedback to practitioners.

This new paradigm will also bring with it new research challenges and issues. One of the most direct challenge will be how to process, store and analyze the huge amounts of data provided by these devices, where the data will not be strictly structured due to coming from a wide range of devices, and where the data will need to be processed and analyzed quickly or even in real time.

5.2.2 Big Data techniques applied to eHealth

Cloud systems bring the possibility to aggregate several healthcare centers, managing thousands or millions of patients from different areas. Big Data techniques offer the possibility of analyzing large sets of data and extracting valuable information. These techniques could potentially allow to perform in depth analyses of current treatments to be performed, sharing useful data between healthcare centers and professionals and improving current medical workflows.

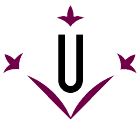
Big Data techniques could allow an efficient management of all the data produced by IoT devices, where the information arriving in a variety of ways and formats must be processed quickly. These techniques will be applied to the two eHealth applications, S-PC and H-PC, when their databases grow beyond a certain threshold both in volume and variability. Moreover, data visualization techniques like the ones used during the doctoral stay (see Appendix A), are also required to display the information in a useful and engaging way.

5.3 Final remarks

Cloud systems have powered a new paradigm that has revolutionized many sectors, including healthcare. Research, such as that presented in this thesis, combined with the new technological advances in the industry, will undoubtedly drive further innovation in eHealth and the development and improvement of healthcare treatment.

Appendix A

Doctoral stay at the Ulster University: Research diary



Doctoral stay at Ulster University

Jordi Vilaplana, PhD student

PhD advisor: Prof. Francesc Solsona

Doctoral stay supervisor: Prof. Yaxin Bi

Abstract

This research diary presents a summary of the work that I have done during my three-month doctoral stay at the University of Ulster (Belfast) under the supervision of Prof. Yaxin Bi.

There are three major aspects to be considered:

- ***What I am expected to contribute:*** *Become involved in the "Identifying cyberbullying from social media" project and develop a client application for the existing sentiment analysis backend server. The initial idea is to develop a web-based user interface that allows Twitter's datasets to be queried and sent to the existing backend server and display the results in a visually engaging way.*
- ***What I expect to get:*** *Get to know an existing project that analyses Twitter data, its architecture and inner working. Become familiar with Twitter's APIs and data visualization tools and apply this knowledge to the #eMOVIX project. Moreover, build a research relationship with Prof. Yaxin Bi and the Ulster University, exploring possible further collaboration and research opportunities between both institutions.*
- ***What I am expected to do:*** *Mainly, the following three points:*
  - *Work on the cyberbullying project.*
  - *Work on the #eMOVIX project.*
  - *Work on the writing of my PhD thesis.*

1 February 09 - February 13

1.1 Initial preparation: Workspace setup and initial meeting

I have discussed with Prof. Yaxin Bi what the course of action during my stay at Ulster University will be.

We discussed the #eMOVIX and the cyberbullying projects.

The main objective of my work will be to develop a client application that will access Twitter datasets through its APIs based on some search queries and hashtags that need to be defined. Once the data is retrieved, the client will send it to the previously-developed server application through its APIs. Then, the results will be fetched from the server in JSON format and presented to the user through a web-based dashboard.

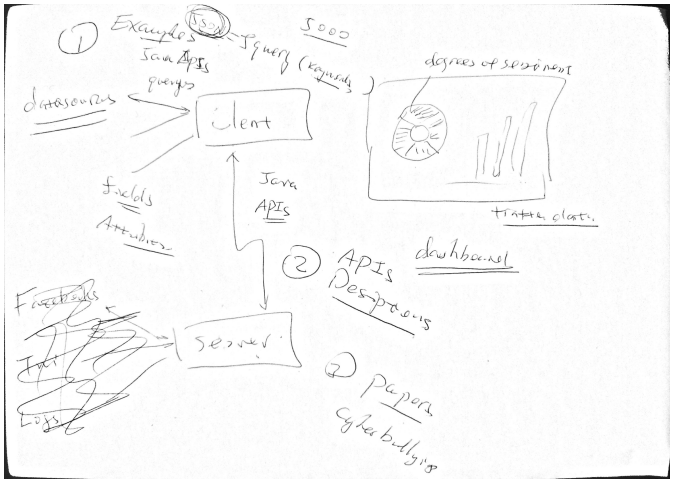


Figure 1: Initial project sketch.

A first sketch of the overall process can be seen in Figure 1. The work in this first week is focused on defining the work to be done and becoming familiar with the existing server's API, the Twitter APIs and reading some cyberbullying-related papers in order to gain a better insight into the subject.

1.2 TO-DO list

- @DONE: Send a summarized and translated version of the #eMOVIX project definition to Prof. Yaxin Bi.
- @DONE: Explore the different Twitter APIs.
- @DONE: Read cuberbullying-related papers [2, 1, 9, 3, 7, 5].

2 February 15 - February 19

2.1 Initial design: Twitter APIs and web application

This week I have been using the Twitter4J [8] library in order to search and retrieve tweets from Twitter based on predefined queries.

The client web application has been started using the Grails [6] framework. This web-based interface will allow users to execute queries, send the data to the server engine and display the results.

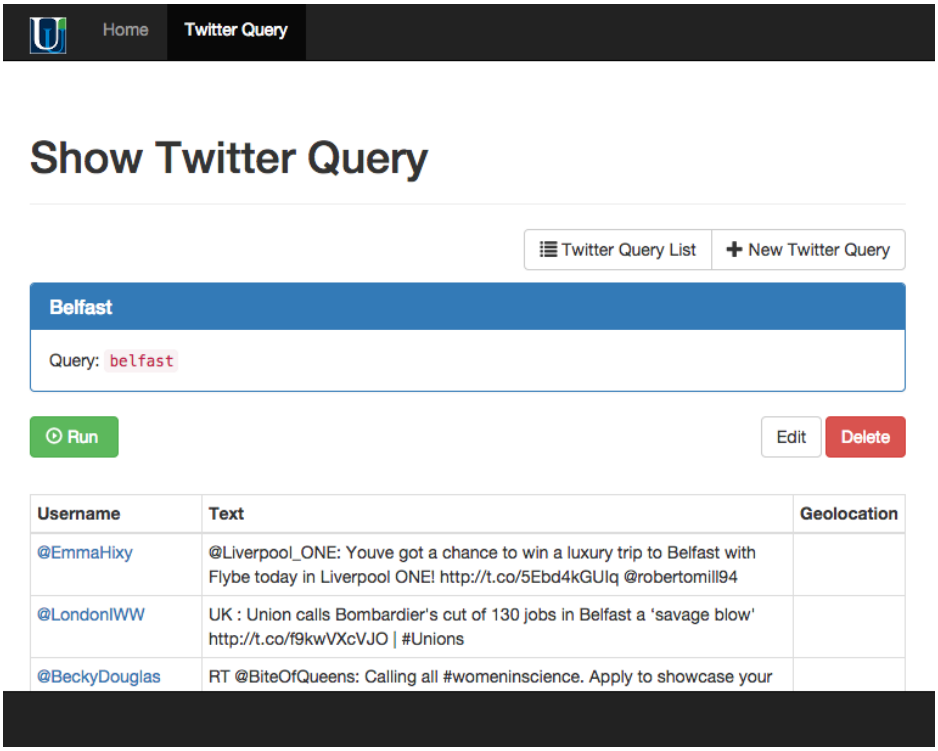


Figure 2: Initial web application design.

So far, I have worked on the initial phase. Figure 2 shows an initial user interface where users can run Twitter queries and see the results.

Early next week, I should meet Prof. Yaxin Bi in order to review this initial work and further develop the required and desired features.

2.2 TO-DO list

- @DONE: Install and test the Twitter4J library.
- @DONE: Develop a first sketch of the web platform.
- @TODO: Explore the engine API.
- @DONE: Start writing my PhD thesis.

3 February 23 - February 27

3.1 Engine API: Connecting Twitter to the backend server

This week Prof. Yaxin Bi showed me the server API and I have been able to use it to analyze tweets successfully. There is a sample web application that shows its basic usage [1].

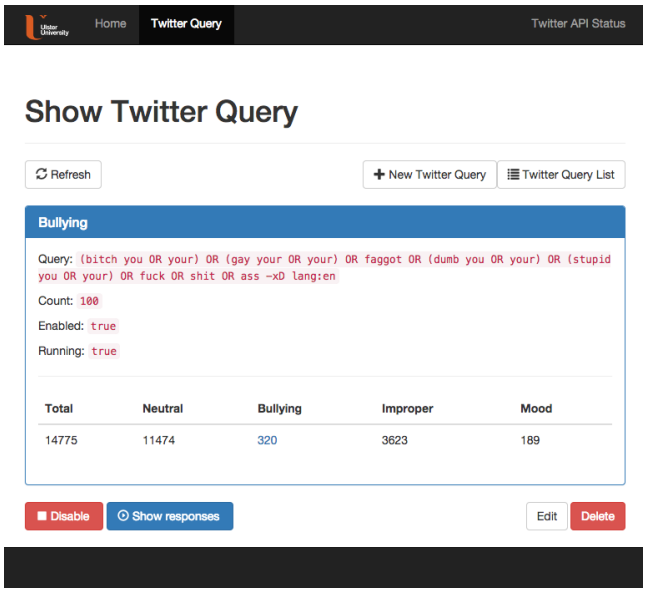


Figure 3: Twitter query interface.

We also reviewed the current work. Figure 3 shows the current twitter query interface. This interface should be enhanced to be a dashboard with relevant statistical measures and charts. Prof. Yaxin Bi has sent me a new paper so I can explore more keywords in order to build the search queries. The queries should be wide in order to obtain the maximum number of tweets. Moreover, I should look into other types of queries like geolocation-based queries, and further search parameters like filtering by date.

There should also be a way to manually mark tweets as bullying or not bullying. To ease this process, bullying tweets should be ordered by the score returned by the engine (where a higher score is more likely to be a bullying message).

Prof. Yaxin Bi also provided me with the Ulster University logo to include on the interface. I will also think up a name for the current project. Depending on the results, this could be further developed in a joint research project between both universities.

3.2 TO-DO list

- @DONE: Explore the engine API.
- @DONE: Save tweets to database.
- @DONE: Add the new Ulster University logo.
- @TODO: Think of possible project names.
- @DONE: Read cyberbullying-related paper [4].
- @DONE: Explore Twitter search options and use a wide range of keywords.
- @TODO: Develop a dashboard with statistics and some chart.
- @TODO: Develop a feature to manually verify tweets as bullying.
- @DONE: Start writing the introduction section of my PhD thesis.

[1] Treze Bullying Servlet Request: `http://restful.scm.ulster.ac.uk:8080/Treze/SentilyticIndex.html`

4 March 02 - March 06

4.1 Interface development: Review and initial visualization

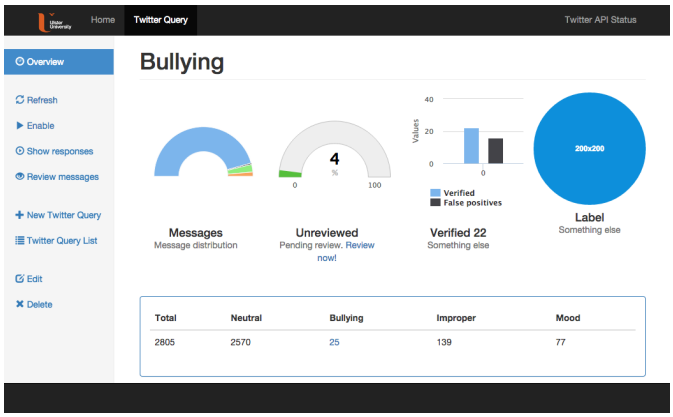


Figure 4: Initial dashboard interface.

This week has been focused on developing and improving the web application:

- Enhancing the current interface and transforming it into a dashboard. Figure 4 shows an initial mock-up.
- Manual review. Users can now manually review tweets the engine marks as bullying in order either to verify them or to mark them as false positives (see Figure 5).
- User and role authentication. Some parts and features of the application can be private or for registered users only. Tweet review is only available for authenticated users.

After reading [4], the current predefined query has also been improved with more keywords added. Also, the current dashboard should be further improved and additional data visualization scenarios should be explored.

Regarding the technologies behind the current server engine, Prof. Yaxin Bi will meet with the industrial partner that holds its intellectual property [2] to see how much information they could provide us so we can replicate it for the #eMOVIX project.

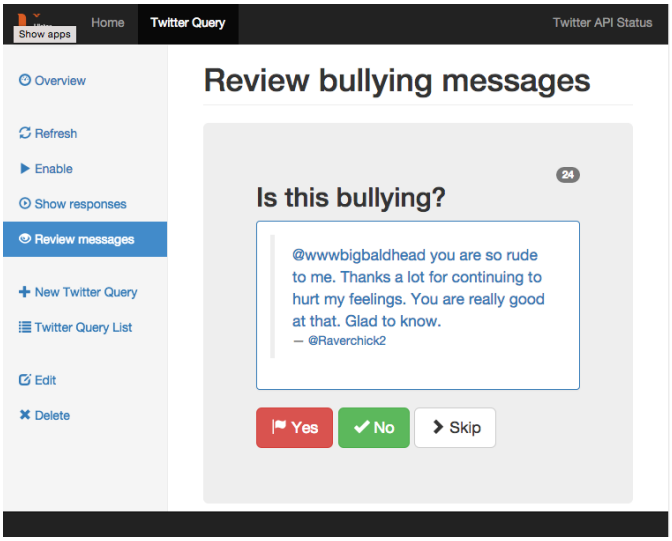


Figure 5: Initial manual review interface.

4.2 TO-DO list

- @TODO: Learn about the technologies involved in the server engine.
- @DONE: Develop an initial dashboard.
- @DONE: Manual tweet verification.
- @DONE: Expand the current search query.
- @TODO: Think of possible project names.

[2] Treze: `http://treze.co.uk/`

5 March 09 - March 13

5.1 Visualizing the data: Exploring data visualization techniques

The first month is over, and at this point, the development of the cyber-bullying application has successfully achieved two of the three main goals:

- ✓ Fetch data from Twitter.
- ✓ Send the data to the back-end engine and obtain the results.
- ✗ Visualize the results.

As for the schedule, during this second month, the visualization part should be fully developed and the application should be thoroughly debugged and tested. During the third month, the project should be validated to ensure everything is correct and as expected, and plan the writing of a research paper describing this work.

So this week has been focused on researching data visualization tools and techniques that could be applied to this project. Also, the human reviewing process has been enhanced. Now, when marking a message as bullying, the user name of the victim must be also provided. This way, we will have more information regarding the relationships between the bullies and their victims.

Regarding the technologies behind the current server engine, I will have a meeting with Prof. Francesc Solsona and Prof. Yaxin Bi next week in order to discuss this topic.

5.2 TO-DO list

- @TODO: Learn about the technologies involved in the server engine.
- @DONE: Improve the current reviewing process
- @DONE: Start researching data visualization tools.

6 March 16 - March 20

6.1 Data visualization tools: D3.js

This week I have been researching different data visualization tools that could fit the project. Among all the explored possibilities the chosen one is the D3.js tool. D3.js is a JavaScript library for manipulating documents based on data, and allows almost any type of chart and visualization to be created. Figure 6 shows a sample graph developed using the D3.js visualization tool.

Figure 6: D3.js sample graph.

Unfortunately, due to the agreements between the Ulster University and its industry partner, it is not allowed to disclose the inner working of the server engine. However, Prof. Yaxin Bi will discuss possible future collaborations in this area with the industry partner.

6.2 TO-DO list

- @DONE: Select a visualization tool.

7 March 23 - March 27

7.1 Visualizing the data: Getting started with D3.js

During this week, the D3.js tool has been researched in order to familiarize myself with its workings and to ensure that it is a valid tool for our purpose. Although getting started with D3.js has proved somewhat tricky, it is a very powerful tool that can offer great results once mastered.

Moreover, we have decided to develop a directional graph based on the reviewed data from the application. This way, we will have a highly intuitive interface that will show the relationships between bullies and their victims.

7.2 TO-DO list

- @DONE: Get familiar with the D3.js tool.
- @TODO: Develop a directional graph.

8 March 30 - April 03

8.1 Visualizing the data: Developing the directional graph

This week the directional graph has been developed using the D3.js visualization tool. Figure 7 shows the resulting graph, where bullies and victims are connected by arrows going from the former to the latter.

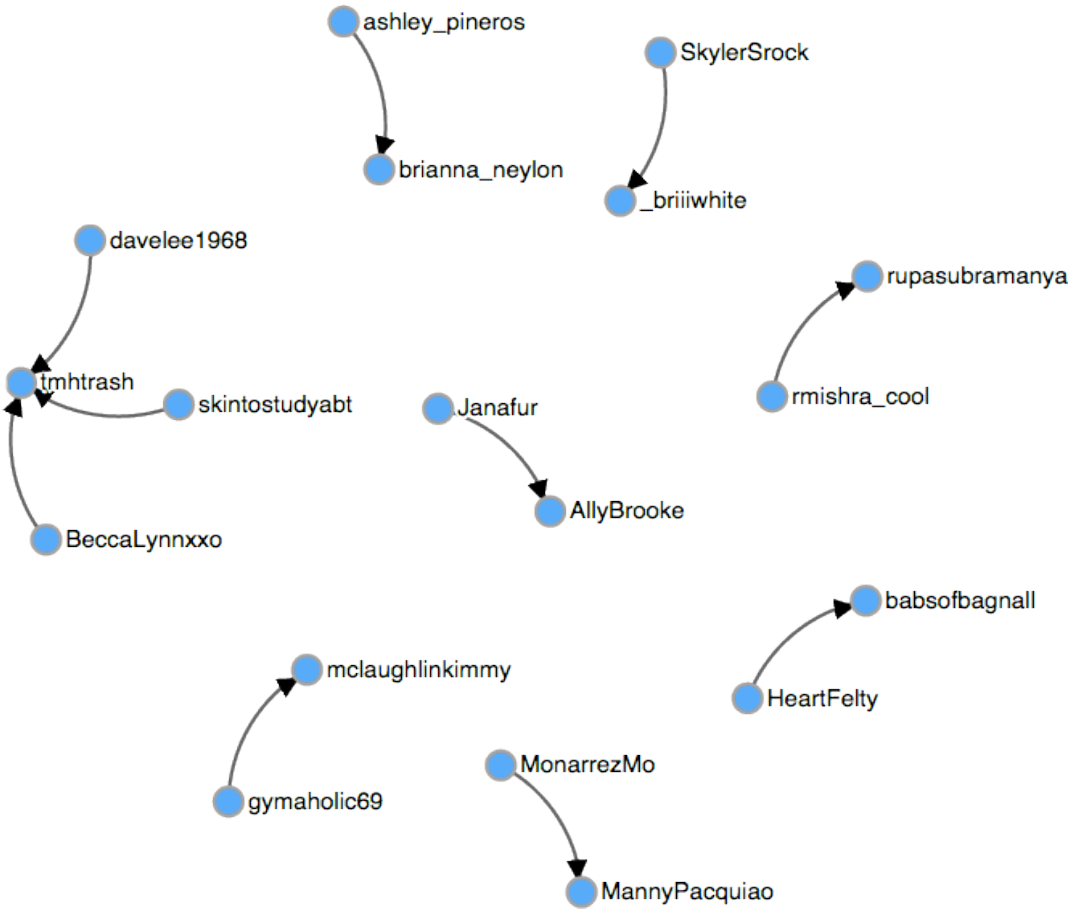


Figure 7: Directional graph developed with D3.js.

The next step should be to verify the good behaviour of this graph and add some more statistical visualizations to understand the data gathered.

8.2 TO-DO list

- @DONE: Develop a directional graph.
- @TODO: Add further statistics and graphs to the visualize data section.

9 April 06 - April 10

The Ulster University is closed during this week for holidays.

The last four weeks will be aimed at finishing the results visualization section and testing and deploying the application.

10 April 13 - April 17

10.1 Visualizing the data: Data statistics

This week, the final development of the data visualization section has been carried out. Now the data from the human-verified bullying messages can be visualized in the following ways:

- Overview: Statistical information regarding the total number of gathered messages, the number of suspected bullying messages, the ones that were verified as bullying and the total number of false positives (messages marked as bullying by the server engine but discarded by a human verifier).
- Map: A world map where those messages are displayed with attached geo-location.
- Words: A chart showing the most frequently used words within the bullying messages.
- Tweets: A table that shows all the bullying messages with the author and date.
- Relationship graph: The directed graph described in 7.

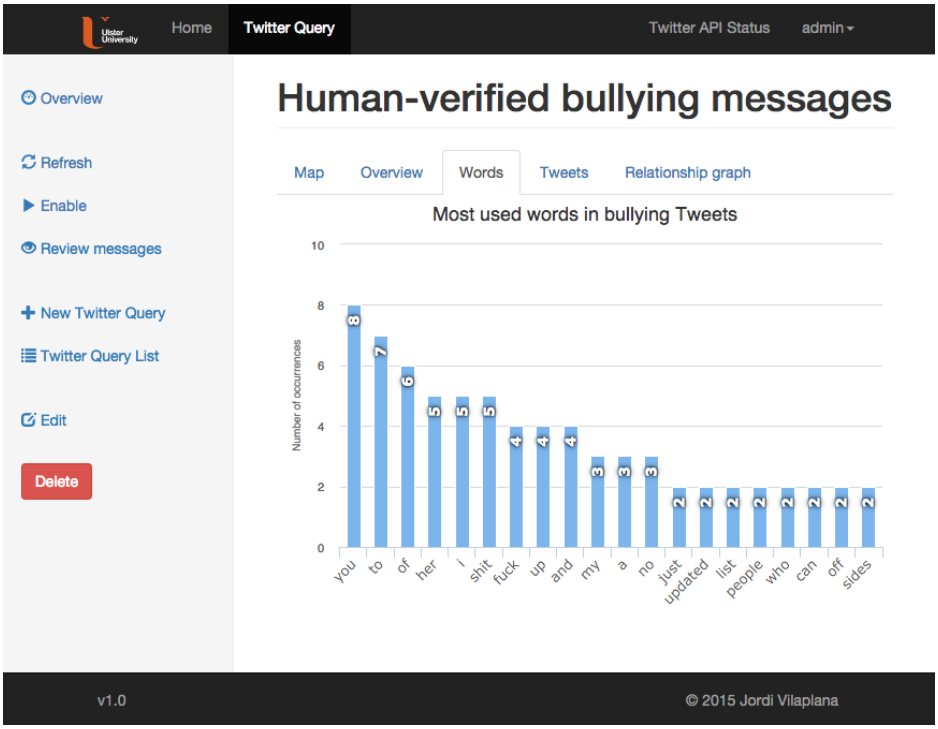


Figure 8: Directional graph developed with D3.js.

Figure 8 shows the final result of the described section, where the Words section can be seen.

10.2 TO-DO list

- @DONE: Add further statistics and graphs to the visualize data section.

11 April 20 - April 24

11.1 Enhancing the application: Usability and display testing

This week has been focused on improving the visual feeling of the application and testing the good rendering of the interface in different screen sizes and resolutions.

11.2 TO-DO list

- @DONE: Visual testing.
- @TODO: Functional testing.

12 April 27 - May 01

12.1 Final testing: Functional testing

This week the application has been thoroughly tested to ensure that all the functionalities and features are working smoothly and error-free.

12.2 TO-DO list

- @TODO: Deploy the application in the test server at the Ulster University.

13 May 04 - May 08

13.1 Final arrangements: Verification and deployment

This last week has been devoted to test all the interface and arrange the deployment of the application within a test server at the Ulster University.

There has been some trouble deploying the database due to minor issues regarding the character set, although these have been successfully solved.

Finally, the application has been deployed and Prof. Yaxin Bi has said that he will present it to the industry partner to explore possible further collaborations.

References

[1] Eytan Bakshy et al. "The role of social networks in information diffusion". In: *Proceedings of the 21st international conference on World Wide Web - WWW '12* (2012), p. 519. DOI: 10.1145/2187836.2187907. URL: http://dl.acm.org/citation.cfm?id=2187907$\backslash$nhttp://dl.acm.org/citation.cfm?doid=2187836.2187907.

[2] Maral Dadvar and Franciska De Jong. "Cyberbullying detection: a step toward a safer Internet yard". In: *Proceedings of the 21st international conference ...* (2012), pp. 121–125. DOI: 10.1145/2187980.2187995. URL: http://dl.acm.org/citation.cfm?id=2187995.

[3] Karthik Dinakar, Roi Reichart, and Henry Lieberman. "Modeling the Detection of Textual Cyberbullying". In: *Association for the Advancement of Artificial Intelligence* (2011), pp. 11–17.

[4] Karthik Dinakar et al. "Common Sense Reasoning for Detection, Prevention, and Mitigation of Cyberbullying". In: *ACM Transactions on Interactive Intelligent Systems* 2.3 (2012), pp. 1–30. ISSN: 21606455. DOI: 10.1145/2362394.2362400.

[5] Samaneh Nadali et al. "A Review of Cyberbullying Detection : An Overview". In: (2013), pp. 326–331. URL: http://www.mirlabs.net/isda13/proceedings/pdf/paper81.pdf.

[6] Pivotal. *Grails Open Source, full stack, web application framework for the JVM*. 2013. URL: http://grails.org/ (visited on 02/18/2015).

[7] Huascar Sanchez and Shreyas Kumar. "Twitter bullying detection". In: *UCSC ISM245 Data Mining course report* (2011).

[8] Yusuke Yamamoto. *Twitter4J Twitter Java API*. 2007. URL: http://twitter4j.org/ (visited on 02/18/2015).

[9] Dawei Yin et al. "Detection of Harassment on Web 2.0". In: *In Proceedings of the Content Analysis in the WEB 2.0 (CAW2.0) Workshop at WWW2009, .* (2009).

Appendix B

Paper: A performance model for scalable cloud computing

A performance model for scalable cloud computing

Jordi Vilaplana[1], Francesc Solsona[1] and Ivan Teixidó[1]

[1] Department of Computer Science. University of Lleida, Jaume II 69, 25001 Lleida, Spain.
Emails: jordi@diei.udl.cat,francesc@diei.udl.cat,iteixido@diei.udl.cat

Abstract

Cloud computing is a new paradigm that offers several advantages in terms of scalability, maintainability, high availability, efficiency and data processing. Infrastructure and applications are moved to the cloud and provided as a pay-for-use service.

When designing a cloud infrastructure, it is critical to assure beforehand that the system will be able to offer the desired level of QoS (Quality of Service). Our attention is focused here on simulation environments for cloud computing systems.

This paper presents a model based on queuing theory for service performance in cloud computing. More complicated cloud-based proposals are then presented. Such models, based on event-driven simulation, possess scheduling capabilities for heterogeneous and non-dedicated clouds.

The results demonstrate the usefulness of the simulation models presented for the design of cloud computing systems with guarantees of QoS under ideal conditions and when scaling the system.

Keywords: SLA, QoS, cloud computing, scalability, simulation

1 Introduction

Clouds [1] aim to power the next-generation datacenters as the enabling platform for dynamic and flexible application provisioning. This is motivated by the fact that datacenters offer their computational resources as a network of virtual services, so that users are able to access and deploy applications from anywhere in the Internet, driven by the demand and Quality of Service (QoS) requirements [2]. As a consequence, by using clouds, IT companies are no longer required to make high investments in computing infrastructures [1].

Cloud systems can be served on three different levels [3]. The first of these is Infrastructure as a Service (IaaS), which means offering hardware, storage and physical devices over the Internet. The second layer is Software as a Service (SaaS), which means offering software and hosted applications over the Internet. Then, combining both of these, there is Platform as a Service (PaaS), which means offering the capability to deploy applications created using programming languages, libraries, services, and tools supported by the provider.

The main contribution of this paper consists of an IaaS model proposal with QoS for clouds providing processing and database service. The cloud is modeled as a single access point for the computing needs of the customers being served [4]. The service center is a collection of computing resources offered to customers (or users) by a service provider. A service request sent by a user is usually transmitted to a web server running a service application [5], which is associated with an SLA (Service-Level Agreement). An SLA is a contract negotiated and agreed between a customer and a service provider for which a customer pays only for the resources and services used according to negotiated QoS requirements at a given price. Job response time is perhaps the most important performance index in a cloud computing context [6]. For this reason, it was also the QoS parameter chosen in this work.

Quantifying the computing resources in servicing the required QoS in a real cloud computing environment (Amazon EC2 [7], Microsoft Azure [8], Google App Engine [9]) is a research challenge because clouds exhibit a high variability in their demands on computing resources. Users also have dynamic and different QoS requirements. The use of real infrastructure for benchmarking the application performance under variable conditions (availability, task arrival rate, falling, scalability, occupancy), is often constrained by the rigidity of the infrastructure. Therefore, this makes it extremely difficult to reproduce the results that it can generate [2]. Thus, it is not possible to perform benchmarking experiments using real-world cloud environments. A more viable and widely-used alternative is to use simulation tools.

Some authors rely on queuing theory to simulate real cloud behavior [3, 4, 10] while others rely more on the possibilities of event-driven simulators, although there is only a small set of these for cloud computing. The best-known ones are GreenCloud [11], MDCSim [12], iCanCloud [13], NetworkCloudSim [14] and CloudSim [2], the one chosen in this paper.

First of all, we develop a theoretical model based on queuing theory with ideal conditions for a cloud proposal designed for process and data servicing. The main drawback is the difficulty of reproducing it in real clouds. Then, the same model is taken to an event-driven simulation environment. We also extend the model by providing it with a two-level scheduler to map tasks to processing nodes according to their processing power and availability. The top level assigns virtual machines to hosts, and the bottom one maps tasks to virtual machines. Finally, the CloudSim implementation of the even-driven simulator is also presented in this article.

The remainder of this paper is organized as follows. Section 2 details the related work. Section 3 presents the design of a cloud architecture and Section 4 its queuing theory model. Next, a more realistic scheduling model based on event-driven simulation is presented in Section 5. Extensive experimentation (Section 6) shows the performance and good behavior of our proposals. Finally, Section 7 outlines the main conclusions and future work.

2 Related Work and Motivation

Simulation-based cloud approaches offer significant benefits by allowing cloud services to be tested in repeatable and controllable environments; correct framework bottlenecks before deploying on real clouds; and experiment with different scheduling polices and workload to determine resource performance on simulated infrastructures [14]. We investigate two kinds of simulators, ones based on queuing theory and the others on discrete events.

The problem of computer service performance modeling subject to such QoS metrics as response time, throughput, network utilization and so on, have been extensively studied in the literature [4, 23, 24]. For instance, in [23], Karlapudi proposed and validated a web application performance tool for the performance prediction of web applications between specified end-points. In [24], Mei addressed the problem of an end-to-end QoS guarantee for VoIP services.

In [29], authors present a modeling technique of using Jackson's network theorem to characterize the performance of mashup multiple servers in cloud computing environments. Their proposed model aims to predict the breakpoint where the waiting time in mashup cloud servers would sharply increase. In [30], authors evaluate the possibility of using Microsoft Windows Azure as a platform for HPC applications and outline the challenges encountered during porting applications and their resolutions. Furthermore, they introduce a metric to measure the efficiency of Cloud Computing platforms in terms of performance and price. In [31], authors present a continuum of four scheduling polices along with an analytical resource prediction model for each policy to estimate the level of resources needed to operate an efficient, responsive, and reliable virtual cluster system.

In [4], the authors obtained the response time distribution of a cloud system modeled on a classical open network $M/M/m$, assuming an exponential density function for the inter-arrival and service times. By using the response time distribution, they determined the optimum level of service and the relationship between the maximum number of tasks and the minimum number of resources (virtual machines). For a given computing service, the authors obtained the level of QoS services in terms of response time that can be guaranteed. The complexity of other queues ($G/M/m$, $M/G/m$, $G/G/m$) comes from the impossibility of obtaining a closed formula to represent the probability distributions of the response or waiting time of customers in the queue [3]. Our work does not focus on research into specific queuing theory challenges but rather on the use of existing models for designing and testing the performance of cloud systems. In addition, problem arises when trying to incorporate scheduling, heterogeneity, user workload and occupancy emulation or sophisticated network connections between their forming computational resources that make them up.

To deal with this problem, data-driven simulators have been taken into account, thus allowing research into the behavior of large scale distributed systems. There are a great range of simulators focused on simulating grid computing systems, such as MDCSim [12], GridSim [25], GangSim [26]. Grid simulators have been used to evaluate costs of executing distributed applications in grid/cluster infrastructures [25]. The most popular among these is GridSim. The GridSim toolkit is a Java-based simulation toolkit that supports the modeling and simulation of heterogeneous grid resources and users spread across multiple organizations with their own policies for scheduling applications. It supports multiple application models and provides primitives for creating and deleting application tasks, the mapping of tasks to resources and the managing of tasks and resources [25]. However, none of the current grid system simulators can be directly used for modeling cloud-computing environments [2].

The CloudSim framework was originally built on top of the GridSim toolkit [25]. CloudSim is the most advanced ([14]) among the cloud simulation environment, more so than GreenCloud [11], MDCSim [12], iCanCloud [13] and NetworkCloudSim [14]. It scales well and has a low simulation overhead and is also the most widely used in current research publications [15, 16, 17]. NetworkCloudSim is capable of simulating real cloud datacenter networks and applications with communicating tasks, such as MPI, with a high degree of accuracy compared to the real ones [14]. As the use of complex MPI applications falls outwith the scope of this work, the good accuracy of the CloudSim (the base platform of NetworkCloudSim) for simulating embarrassingly parallel applications justifies it being chosen for this work. Embarrassingly parallel applications, the ones used in this work, require no synchronization or communication between tasks. They are much simpler than the more complicated MPI distributed applications, which need to interchange intermediate results between tasks. So our efforts have been centered on CloudSim. No other additional features were needed to support complex communication, as in the case of NetworkCloudSim. Thus, we can state that the behavior of our models is in line with real clouds.

3 Cloud Architecture

As stated in [4], most current cloud computing infrastructures consist of services that are offered and delivered through a service center, that can be accessed from a web browser anywhere in the world. The two most significant components of a cloud-computing framework are the Front-end and the Back-end. They are also the main components of our cloud architecture, the base platform on which we develop our proposals.

3.1 Front-end

The Front-end is the gateway to the cloud and consists of the software components and the interfaces needed to connect to the platform using remote client applications [5]. Normally, these applications use standard web protocols to access the system and an authentication protocol that allows access to authorized users. All requests are processed by the scheduler, which sends the selected tasks to the queue of the Back-end. For simplicity, a First-Come-First-Serve (FCFS) scheduling policy was assumed.

Furthermore, user requests are usually received by the web server, which is also in charge of tasks running in the Back-end. This application/service is associated with an SLA. The interface and negotiation

processes are usually performed at the Front-end.

As we are proposing a generic system, application workflows other than embarrassingly parallel applications will not be considered in our model, thus avoiding deadlock situations. All arriving tasks forming the applications will consist of web requests served by the Front-end.

3.2 Back-end

The Back-end is really the part of the cloud were the main HPC processing is performed. The Back-end functions include management of the job queue, the servers and their virtual machines and the storage servers. For clarity purposes, we have only considered one storage (i.e. database) server. Note that, internally, this server could be composed by a clustered database.

The Back-end is made up of two different kinds of server:

***Processing servers*:**
Virtual machines responsible for performing most of the computation.

***Data servers*:**
Virtual machines whose main task is to perform database transitional operations.

We are interested on modeling this part of the cloud, namely the Back-end. In doing so, we first deal with a model based on queuing theory (Sec. 4). Next, we propose a more realistic way of modeling the cloud Back-end, based on event-driven simulation.

4 Cloud Architecture Modeling

To model the Back-end, we propose a multi-server system based on the queuing model (see Fig. 1), representing an Open Jackson network [20, 21].

This network has a single Entry point E. This server acts as a load balancer, which forwards the user requests to one of the processing nodes N_i, where $i = 1..m$. The load balancer is represented by an *M/M/1* queue with an arrival and service rate modeled by an exponential pdf with parameters λ and l respectively, where $\lambda < l$.

A processing node N_i is a node, core or processor representing the physical computational resources of our cloud architecture where the services are computed. The selected N_i node performs the service required by the user request. All N_i nodes are identical (homogeneous) and are modeled as an *M/M/m* queueing system. Each N_i node has the same service rate and is equal to μ, this is $\mu = \mu_i, i = 1..m$.

Each N_i node accesses D (Database) with a probability δ. D represents a database server and serves to model the access to files, directories and databases or to any kind of I/O access to secondary memory during the service in the cloud architecture. This is an important consideration to be taken into account because I/O interference between virtual machines is an important issue. D is modeled by an *M/M/1* queue with exponential arrival and service rates of $\delta\lambda$ (according to the rules of the open Jackson network) and d respectively.

Connecting servers with exponential arrival and service distributions in a feedforward (without feedback paths) are independent from each other and preserve pdf distributions [22]. So, λ is the feeding distribution for the customers leaving the Entry node (E).

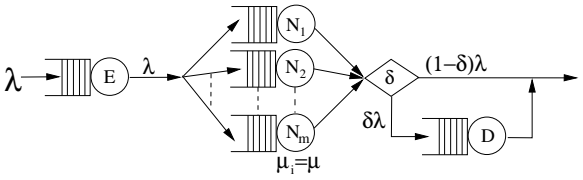


Figure 1: Queuing theory architectural model.

As a result of being considered as an open Jackson network, the response time (T) of the global cloud architecture is the following:

$$T = T_E + T_N + T_D \tag{1}$$

T_E represents the response time of the Entry node, modeled as an $M/M/1$ queue. Thus, T_E is defined as (see [18] for more details):

$$T_E = \frac{1/l}{1 - \lambda/l}, \tag{2}$$

T_N represents the response time of the servicing nodes that actually process the user requests. As these servers are modeled as an $M/M/m$ queue, according to [19], the response time of such a queue is defined as:

$$T_N = \frac{1}{\mu} + \frac{C(m,\rho)}{m\mu - \lambda}, \tag{3}$$

The term $C(m,\rho)$ represents Erlang's C formula, which gives the probability of a new client joining the $M/M/m$ queue. Erlang's C formula is defined as [18]:

$$C(m,\rho) = \frac{\left(\frac{(mp)^m}{m!}\right)\left(\frac{1}{1-\rho}\right)}{\sum_{k=0}^{m-1}\frac{(mp)^k}{k!} + \left(\frac{(m\rho)^m}{m!}\right)\left(\frac{1}{1-\rho}\right)}, \tag{4}$$

where $\rho = \lambda/\mu$.

Finally, T_D represents the response time of the Database server, D. As D is modeled as an $M/M/1$ queue, similarly to T_E, T_D is obtained as follows:

$$T_D = \frac{1/d}{1 - \delta\lambda/d}, \tag{5}$$

5 Scheduling

The Back-end can be divided into two levels, the physical hosts and the virtual machines. Tasks are assigned to virtual machines, which are in turn mapped to a sort of computing clustered resources (cores/processors, main memory, secondary memory and network bandwidth). As a consequence, we extend the functionality of the previous model by taking these issues into account. In doing so, we propose a two-level event-driven scheduler to map tasks to processing nodes according to their processing power and availability. The top-level scheduler assigns virtual machines to hosts (named Virtual Machine Scheduler), and the bottom one assigns tasks to virtual machines (named Job Scheduler).

5.1 Virtual Machine Scheduling

Our virtual machine scheduling criteria is based on minimizing power consumption. Note that, ideally, the total response time of the tasks assigned to virtual machines should not be affected by the virtual

machine scheduling policy. This is because each virtual machine has its own requirements in terms of computing capacity, and regardless of which physical host is selected to allocate the virtual machine, it will always have those minimum requirements. For this reason, the criteria selected for the virtual machine scheduling policy was minimizing the power consumption. Note that this scheduling level has nothing to do with Section 4. Those are really two separate issues.

The way that virtual machines are allocated to physical hosts will determine the energy consumption of the whole system. Energy consumption is not lineal and modeling it would lead to a complex analytical problem [17]. In order to solve that, real data on power consumption obtained from benchmarks is used.

Instead of using an analytical power model for the physical hosts, we utilize real data on power consumption provided by the results of the SPECpower benchmark [1], as in [17]. This is also the power model used in CloudSim. As an example, Fig. 2 shows the power consumption of two HP machines, obtained with the SPECpower and implemented in CloudSim.

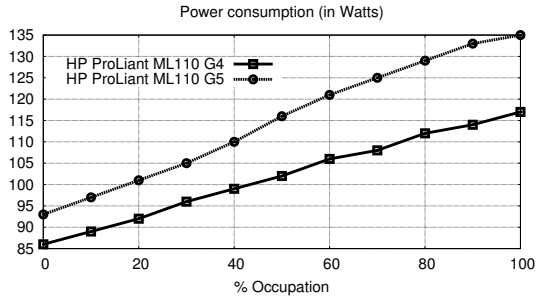


Figure 2: Power consumption for two HP machines.

The following are two policy proposals for the virtual machine scheduler.

5.1.1 Round-robin VM Scheduling

First of all, we present Round-robin, the first policy for scheduling virtual machines (VM) to hosts. Virtual machines are assigned to hosts by following a circular ring ordering. In this first approach, no considerations about power consumption are taken into account. The main aim of this simple *vm* scheduling is to present the policy implemented by the CloudSim. Apparently, a wide margin from improvement is left to the following policy.

5.1.2 Power-aware VM Scheduling

We propose a custom policy, called "*Power-aware Virtual Machine Scheduling*", that assigns VMs to hosts based on the number of free cores (see Alg. 1). For each virtual machine to be allocated, we select the host with the fewest cores available which can host it. This way, hosts with less capacity are used first, decreasing the energy consumption as the more powerful hosts can meanwhile be idle.

For each virtual machine (VM) to be allocated (from list *vmList*), Alg. 1 selects the host ($host_{sel}$) with fewest available cores from the host list (*hostList*) after the assignment (*minAC*) that suits the requirements of the current virtual machine ($vm.requiredCores$). Finally, it assigns the current virtual machine to the selected host ($vm.assignHost(host_{sel})$).

Algorithm 1 Power-aware virtual machine allocation algorithm

```
minAC=∞;
for all vm ∈ {vmList} do
    for all host ∈ {hostList} do
        C = host.availCores − vm.requiredCores;
        if (C > 0 && C < minAC) then
            host_sel = host; minAC = C;
        end if
    end for
    vm.assignHost(host_sel)
end for
```

5.2 Job Scheduler

The minimisation of response time is addressed by the bottom scheduler, or Job Scheduler. The heuristics of this scheduler are focused on reducing the time of parallel jobs (or tasks), thus providing the QoS as agreed at the SLA. However, we will not delve into interface nor negotiation mechanisms. We are only interested in obtaining the best scheduling according to the response time performance.

Two job scheduler proposals are presented. In the first version, Round-robin, we simply transferred the queuing model to an event-driven model as accurately as possible by defining a single scheduling policy for the processing nodes (N_i, from Fig. 1). Note that a processing node refers to a virtual machine as presented in Section 5.1. In the second proposal, we increased the accuracy of real cloud environments by providing the scheduler with the functionality of dealing with additional processing element features, such as workload occupancy and processing capabilities (heterogeneity).

5.2.1 Round-robin Job Scheduler

First of all, we present Round-robin, a policy analogous to the one presented for scheduling virtual machines (Section 5.1.1), but which schedules jobs instead. We suppose the system is entirely dedicated to processing our servicing requests. That means the cloud is entirely dedicated to processing arriving services (tasks making up an embarrassingly parallel application) in any order from one user. This simplifies the scheduling task, that is the selection of the N_i nodes of Fig. 1, in order to process the processing requests.

We also suppose, as in Sec. 4, that each N_i node has the same computing power. The mapping policy of tasks to nodes N_i, implemented at the E node, is quite simple. The nodes are scheduled in a single Round-robin manner, without taking either their occupancy or computing power into account. We also named this scheduling version "*Round-robin*".

Communication costs have not been considered. Network latency in sending tasks from the E to one N_i, and from one N_i to D are negligible. This is the case when all these cloud components are physically located in the same machine, where communicating costs are low enough to be ignored.

Another aim of this first version of the event-driven implementation is to construct the base model to compare with the queuing system designed in Sec. 4 and the improvements added in the Balanced version.

[1] SPECpower benchmark. $www.spec.org/power_ssj2008/$

5.2.2 Balanced Job Scheduler

We provide the E with a scheduling policy which is able to deliver jobs to the best processing node N_i. We named this scheduling version "*Balanced*". Assignments take into account both the occupancy and computing power capacity of each N_i node. This feature will provide the model with the ability to simulate heterogeneous environments. Nowadays, the disposal of a great range of computing power nodes in cloud systems is very common. Over time, cloud centers acquire new computational resources as needed, and these are hot-plugged to the pool of processing nodes without restarting the system. Consequently, the N_i nodes can become quite heterogeneous with time.

Starting as a reference point the work done in [28], we define the *Effective Power* (Γ_i) of each node N_i as a function of the *relative computing power* and the *availability* of such a node. The scheduler will assign a new task entering to the N_i node with the highest Effective Power. Formally, Γ_i is defined as follows:

$$\Gamma_i = P_i \cdot A_i, \tag{6}$$

where P_i is the *relative computing power* of node i (N_i) with regard to the most powerful node in the pool of processing nodes. The values of this parameter are in the range $0 < P_i \leq 1$, where $P_i = 1$ means node i is the most powerful one and $P_i < 1$ otherwise. The P_i value can be obtained by benchmarking applications, as described in [27]. The *availability* (A_i) of a node N_i is defined as the complement to one of its percentage of occupation. $A_i = 0$ when the CPU is fully occupied and $0 < A_i \leq 1$ otherwise.

As a consequence, the Effective Power value is in the range (0..1], where $\Gamma_i \approx 0$ means node i is unable to execute any task and $\Gamma_i = 1$ implies tasks can be executed at full speed in such a node.

Algorithm 2 Balanced job scheduling algorithm

```
Γ_max = 0
for job : jobList() do
    for node : nodeList() do
        A = 1 − (node.workload/sum{jobList.workload})
        P = (node.power/max{nodeList.power})
        Γ = P · A
        if Γ > Γ_max then
            node_sel = node
            Γ_max = Γ
        end if
    end for
    job.assignNode(node_sel)
end for
```

Based on the considerations above, the *Balanced* job scheduling algorithm is presented in Alg. 2. It distributes a job to the node with the highest computed Γ, which depends on its A and P. A is obtained as the complement to one of the workload of such nodes compared to the total workload, the sum of the overall jobs $(1 - (node.workload/sum\{jobList.workload\}))$. There are many ways to estimate the workload. It can be supposed, for example, that jobs are made up of homogeneous tasks, and that a task is the unity workload. So a node with one job with 8 tasks would have a workload equal to 8. The computing power P of a node is obtained as the relation between its power and the maximum $(node.power/max\{nodeList.power\})$. Each job from list *jobList* will be mapped onto the node in the pool of processing nodes *nodeList* with the maximum computed Γ ($job.assignNode(node_{sel})$).

6 Results

We present an analysis of how the response time is affected by modifying some of the metrics in the models presented. First of all, we analyze the queueing theory-based model of the Back-end part of the cloud architecture. Secondly, a set of simulations is performed using the CloudSim framework. Our purpose is to verify the cloud architecture when scaling the computing resources of a cloud site. The scheduling proposals (the Power-aware Virtual Machine and the Job Scheduling policies) are also tested below. Next, the computing resource utilization of a unique cloud site is analyzed. Finally, we show the performance when scaling the number of cloud site's (datacenters).

6.1 Queuing Theory Simulation

The Back-end architecture discussed in Section 3.2, modeled through a mathematical approach (the queuing theory) and presented in Section 4, was implemented using the mathematical software Sage 5.3[2]. The parameters used in the implementation are described below:

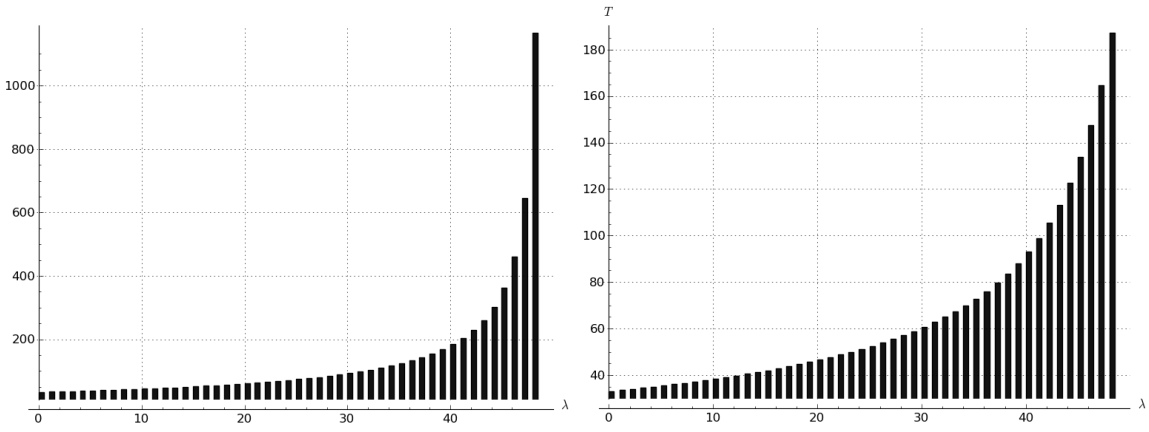
- λ Arrival rate. Average number of requests reaching the system per unit of time. $1/\lambda$ is the mean inter-arrival time. We aimed to show how response time (T) is affected by varying the parameter λ. The number of processing servers (m) represents the total number of processors, cores or nodes dedicated to servicing requests. Changing this parameter will show the impact of adding or removing servers to the system.
- δ Database access probability. The probability that an incoming request needs to access the database node D. Not all requests will always require access to the database server. Note however that this probability will usually be relatively high.
- μ Service rate. This describes the speed at which the web servers handle the requests. $1/\mu$ is the mean service time. The total service rate of the system must always be greater than λ for the system to be stable. This value was modified through the tests in order to check the impact on performance. Although service rates of E (l), N_i $i = 1..m$ (μ_i), and D (d) could be different, in order to simplify the experiments, it was assumed that they all had the same value. Thus, throughout the test, we supposed $l = \mu = \mu_i = d$.

Figs. 3a and 3b show the response time (T) obtained in the queuing simulator by increasing the arrival rate (λ) for one and 1000 servers, respectively.

The response time decreased from one to 1000 servers servers. However, the response time stabilized quickly when the number of servers increased beyond ten. The explanation for this phenomenon can be found in the utilization of the N_i nodes (see Fig. 4), defined as $\rho = \lambda/\mu$.

From Fig. 3 and Fig. 4, we can derive that the response time stabilizes with server utilization (ρ). In Fig. 3a, when the entering arrival rate for requests to the system, which in turn equals the arrival rate in the pool of servers (λ), is high, the utilization (ρ) approaches one. Adding servers to the N_i nodes, led the utilization rate of the $M/M/m$ queue to decrease and the response time to stabilize. For this reason, there is an upper bound above which, adding more N_i nodes means hardly any decrease in the system's

[2]Sage. http://www.sagemath.org/



(a) Response time (T) with 1 server.

(b) Response time (T) with 1000 servers.

Figure 3: Response Time (T) with 1 and 1000 servers.

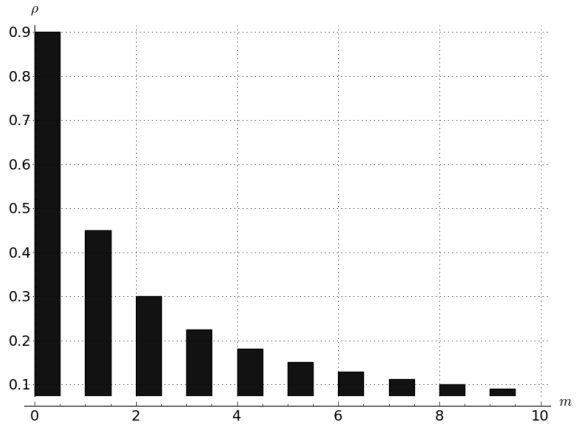


Figure 4: Utilization ($\rho = \lambda/\mu$) rate.

response time (T). In Fig. 4, we can see how utilization (ρ) quickly decreases when servers start to be added to the system. For values above 10, the degree of utilization scarcely decreases, although its value approaches asymptotically to 0.

6.2 Event-driven Simulation

Once the queueing theory-based model had been implemented, we used the CloudSim 3.0.2 software [2] to implement the cloud as an event-driven simulation to verify the behavior consistency of the proposed models. The performance of the Back-end implemented was measured in an event-driven simulator.

From then on, we used a new cloud event-driven simulation environment, based on a real machine with two processors AMD Opteron 6,172 processors, 144 GB of Memory and 4.5 TB of secondary memory storage[3]. Each processor was composed of 12 cores (PE Processing Elements) at 2.1 GHz and 3,150 MIPS each. By using CloudSim, it was possible to emulate the same machine reliably. Virtual machines have a different number of PE (CPU) depending on each test, and all the PEs were configured to 3,150 MIPS and 4,096 MB of Memory. The number of virtual machines depends on the simulation performed below.

[3]http://www.cisco.com/

Prior to presenting the implementation, let us introduce some CloudSim terminology. CloudSim manages a set of entities, described as follows:

Host.
Physical machine which can be divided into several virtual machines.

PE.
Processing Element. This represents a CPU unit, defined in terms of Millions Instructions Per Second (MIPS).

VM.
Virtual machine. An abstraction of a host (physical machine), with its own RAM Memory, CPUs, secondary memory storage and bandwidth. Each virtual machine is assigned to a specific broker.

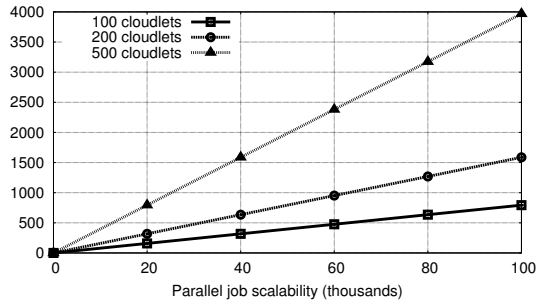
Broker.
Entity responsible for handling the service requests made by the cloud users. A virtual machine list and a cloudlet list are submitted to each broker. This entity is used internally in the simulations.

Cloudlet.
Represents an application service or a cloud task. The work to be done is specified by giving its execution length (number of instructions) to be executed in the virtual machines. Each cloudlet is assigned to a specific broker.

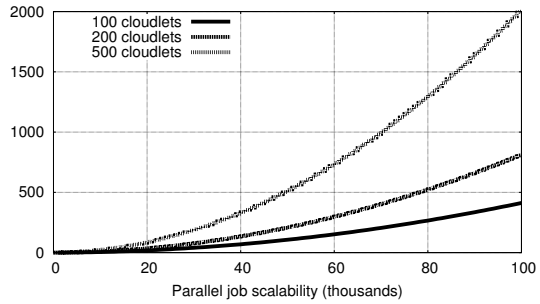
Datacenter.
Set of hosts. Cloudlets (representing applications, tasks or jobs) are delivered to datacenters in order to be executed. At least one datacenter is needed to run a simulation.

We first measured the behavior of a single cloud site by scaling the number of jobs to be processed in the Back-end. In order to measure the scalability of a cloud site, we evaluated the job execution times in function of their size. Fig. 5a shows the performance in the execution of the cloudlets (in time units) for three different application (job) sizes (100, 200 and 500 cloudlets, i.e. services or tasks). We used two equal virtual machines with two PEs. Each service (task) was specified by defining its cloudlet size, in other words, the number of instructions (in thousands). Its size in turn is determined by the number

of instructions required to complete the cloudlet. As can be observed, for each job, times increased linearly with the size of the cloudlets. Measurements were obtained for 0, 20,000, 40,000 ... 100,000 instructions per cloudlet. In addition, for the same task size, the execution times increased proportionally with the number of cloudlets. Although the results are logical in ideal conditions, real clouds are not accurately represented because the shapes of the time lines should behave asymptotically and not so linearly.



(a) Constant job size.



(b) Variable job size.

Figure 5: Execution times for three scenarios (100, 200 and 500 cloudlets).

In order to solve the linearity presented in the previous figure (Fig. 5a), we defined a different policy for workload arrival. Fig. 5b shows the total execution time of all the parallel jobs submitted to the system (and as before, with job sizes of 100, 200 and 500 cloudlets) when the size of each job is not constant. For each kind of job, the number of instructions per cloudlet was also varied from 0..100,000, in increments of $100,000/(number\ of\ cloudlets)$. For example, for the case of 500 cloudlets, the increment was 200. In addition, each job was executed separately and all its component cloudlets were submitted at the same time. Cloudlets were executed in increasing size without delays between consecutive executions. Each cloudlet should be waiting for the execution of the previous one. This way, we simulated two aspects of real cloud computing systems, the sharing of cloud resources by the workload and the non-linear property of the most real cloud systems, a key feature not provided by other simulation systems of the literature. Furthermore, it was very useful in order to perform stress testing, although this kind of arrivals may not represent realistic or probable arrival rates.

Fig. 6 shows the system behavior when scaling it by increasing the number of virtual machines and the number of available hosts. The simulation environment was configured with one datacenter and 1,000 cloudlets with a size of 100,000 instructions each. Further experiments by varying the parameter values gave similar results.

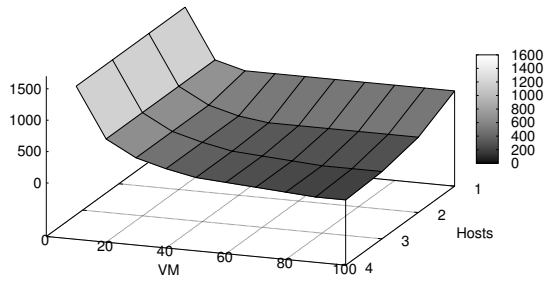


Figure 6: Execution times depending on the number of virtual machines and the number of hosts.

We can appreciate that increasing the number of virtual machines and hosts significantly decreases the total execution time of the jobs. From Fig. 6, it can be seen that by adding virtual machines, the system will approach asymptotically a limit where it does not have enough computational resources (RAM, CPUs, etc.. making up the hosts) to map the cloudlets (i.e. the cloudlet instructions), and so the execution time stabilizes. Similar behavior occurs when adding more hosts without adding more virtual machines.

6.3 Scheduling

We measured the two proposed schedulers and their different versions. Firstly, the performance of the virtual machine allocation policy that maps virtual machines into hosts was tested. Lastly, we checked the job (cloudlet) allocation scheduler, which determines the virtual machine the cloudlets will be executed in.

6.3.1 Virtual Machine Scheduling

In Fig. 7, we applied the Power-aware policy defined in Sec. 5.1.2 and compared it with the default CloudSim policy, which is based on the Round-robin scheduling algorithm, and defined in Sec. 5.1.1. The way virtual machines are allocated to hosts determines the power consumption of each host and eventually the energy efficiency of the whole datacenter.

The simulation environment was configured with a heterogeneous set of 27 VM of three different types: 20 VM with 1 core, 5 VM with 8 cores and 2 VM with 4 cores. Each core had a capacity of 3150 MIPS. The simulation was carried out from 100 to 500 cloudlets with two different policies. All the cloudlets had 10,000 instructions. The datacenter had three different hosts. The first with 12 PE of 5,000 MIPS each, the second one with 24 PE of 6,000 MIPS each, and the third one with 48 PE of 6,300 MIPS each.

The results in Fig. 7 show a significant improvement in the energy consumption of the datacenter when compared to the default Round-robin policy, with a 7,9% reduction of the consumption. This denotes how allocating virtual machines efficiently can lead to a more power-efficient system.

Note that allocating virtual machines to hosts does not affect the response times of the tasks that will be later assigned, as each virtual machines are only assigned to hosts that fulfill its computing resource needs.

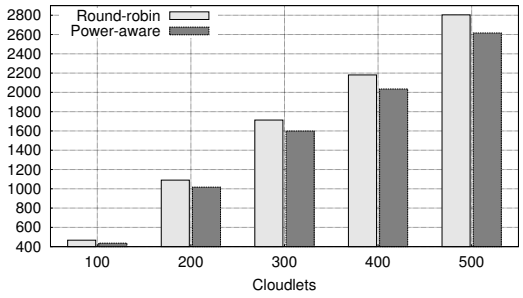


Figure 7: Energy consumption (in kW) of two different VM allocation policies.

6.3.2 Job Scheduling

In the following section, we discuss cloudlet (task) allocation to virtual machine policies (see Section 5.2). As opposed to the previous section (Sec. 5.1), cloudlet mapping has a direct impact on the response time of the system and determines finishing task times.

In Fig. 8, we applied the policy defined in Sec. 5.2.2 and compared it to the Round-robin scheduling policy (described in Sec. 5.2.1). The simulation environment was configured with a heterogeneous set of 6 VM of three different types: 2 VM with 1 core, 2 VM with 2 cores and 2 VM with 4 cores. Each core had a capacity of 3150 MIPS. A total of 6 simulations were performed, for 100 cloudlets, 200 cloudlets and 500 cloudlets with two different policies. All the cloudlets had 100,000 instructions. The first policy was the default policy included in CloudSim, which is a Round-robin algorithm that distributes cloudlets in equal portions and in circular order to the different available VM (as in Sec. 5.2.1). The second policy consisted of the one described in Sec. 5.2.2, based on the Formula 6 ($\Gamma_i = P_i \cdot A_i$).

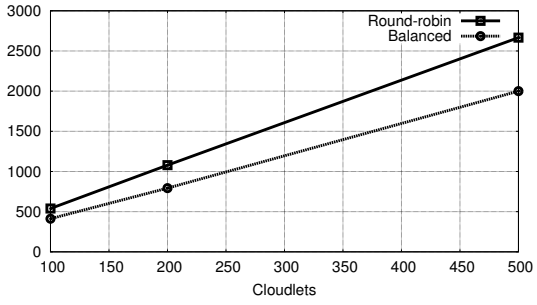


Figure 8: Total response time using two different cloudlet allocation policies.

The results in Fig. 8 show that taking the capacity and the availability of the VMs into account in a heterogeneous environment improved the performance of the system significantly when compared with the default Round-robin policy. The gains reached 25%.

This improvement is due to considering not only the computing capacity of the VMs, but also its current state. Thus avoiding overloading the most powerful VMs while leaving the less powerful ones underloaded.

6.4 Resource utilization

In this section, we are interested in measuring the utilization of the computing resources (CPU, Main Memory and Bandwidth). In the next experiment, we defined two datacenters and three customers. The first and second datacenter held 4 and 200 hosts respectively. Fig. 9 shows the resource utilization of the first datacenter. The utilization behavior of the second datacenter is very similar.

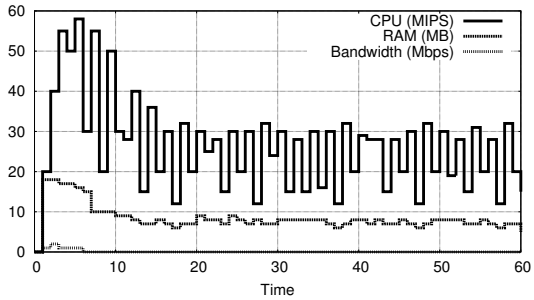


Figure 9: Resource utilization. Datacenter 1.

We can appreciate how the datacenter analyzed experienced a peak of CPU utilization during the beginning of the simulation that then stabilized with time. This is explained by CloudSim first having to schedule and allocate all the customer's jobs in the respective virtual machines. Next, the amount of resources needed to keep executing the jobs decreased until it stabilized. Note that the Memory and CPU utilization are closely correlated. However, the CPU was in an upper scale of magnitude. It can be appreciated that no network bandwidth was used. There was no communication between job tasks (cloudlets). This is the distinctive feature of the parallel applications we used throughout the experimentation, namely the embarrassingly parallel ones

6.5 Cloud Site Scalability

In this section, we measure the impact of having multiple sites (datacenters). In doing so, we scale the number of sites. The concept of doing this is depicted visually in Fig. 10.

In Fig. 11a, two datacenters were added to the one used previously in the simulation in order to test the scalability of the system. These two datacenters had the same characteristics as the one previously defined. The simulation environment was configured using 500 cloudlets, each requiring 1,000,000 instructions and a total of 60 VM with 2 CPU and 4,096GB RAM each.

We can appreciate in Fig. 11a that the total execution time significantly decreases when one more datacenter is added. In this case, CloudSim used a Round-robin filling-up scheduling policy. When a datacenter is at its full capacity, incoming tasks are submitted to the next one, following a ring order. We can see that execution times tend to stabilize.

In Fig. 11b, further testing was done to verify the scalability. A total of seven datacenters with the same characteristics as the ones used in the previous test were used. The simulation environment was configured using 130 VM and 1,000 cloudlets each with 1,000,000 instructions. We can verify how the behavior from Fig. 11a is still valid when scaling in both senses, the amount of computing resources and the application size.

Having multiple datacenters also ensures a higher level of robustness. Even if one or more datacenters falls or becomes unavailable, the workflow of incoming tasks can be redirected to other datacenters by

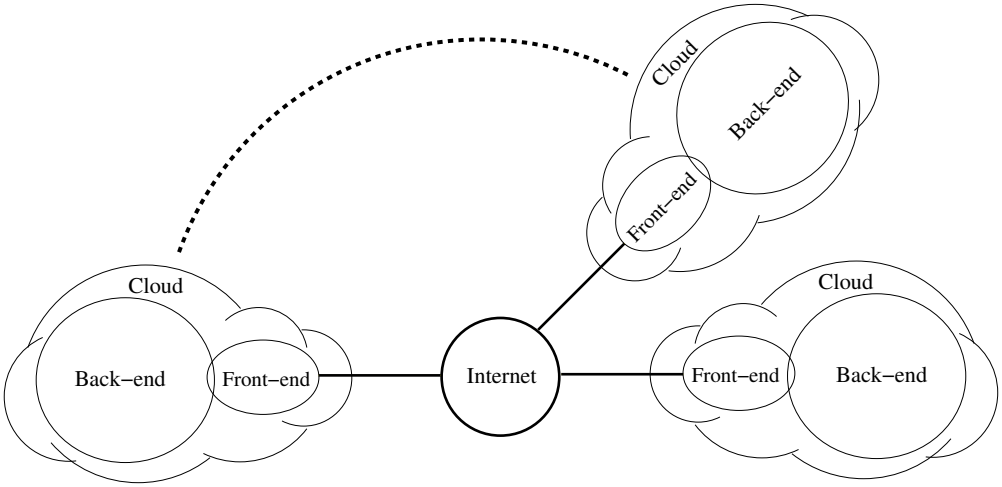
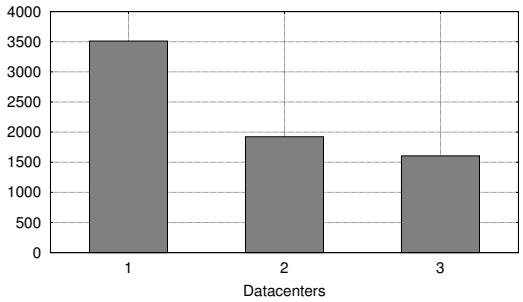
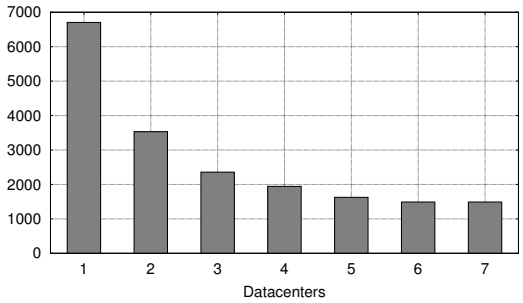


Figure 10: Cloud site scaling.



(a) Up to 3 datacenters.



(b) Up to 7 datacenters.

Figure 11: Execution time.

applying a heuristic policy, increasing the fault tolerance of the whole system enormously.

7 Conclusion and Future Work

In this paper, a model for designing cloud-computing architectures with QoS is presented. We presented two main alternatives for doing so. The first one, based on queuing theory and the open Jackson networks, was selected as the basic means of guaranteeing a certain level of performance according to the response times of such networks. Secondly, the need to design more accurate schedulers for real sites, that is, non-dedicated and heterogeneous clouds, led us to develop new event-driven simulation policies. It is demonstrated that our proposal, a two-level scheduling algorithm, improved performance significantly. The mapping of virtual machines into physical hosts was demonstrated to be important from the energy consumption point of view (the savings reached 7.9%), whilst job scheduling into virtual machines improved response time by up to 25%.

As future work, we plan to develop new scheduling policies taking into account more complicated jobs, with communication and synchronization between their component tasks. Moreover, we want to develop further models that also consider data replication and multiple storage servers. We also are working on the design and deployment of our proposals into real clouds by means of the OpenStack[4] platform.

Acknowledgments

This work was supported by the MEyC under contract TIN2011-28689-C02-02. The authors are members of the research group 2009 SGR145, funded by the Generalitat de Catalunya.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. *A view of cloud computing.* Communications of the ACM, vol. 53(4), pp. 50-58. 2010.

[2] R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, R. Buyya. *CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms.* Software: Practice and Experience, vol. 41, no. 1, pp. 23-50. 2011.

[3] H. Khazaei, J. Misic, V. Misic. *Performance Analysis of Cloud Computing Centers Using M/G/m/m+r.* Queuing Systems. IEEE Transactions on parallel and distributed systems, vol. 23, n. 5. 2012.

[4] K. Xiong, H. Perros. *Service Performance and Analysis in Cloud Computing.* Proc. IEEE World Conf. Services, pp. 693-700. 2009.

[4] OpenStack. http://www.openstack.org/software

[5] J. Martin, A. Nilsson. *On service level agreements for IP networks*. In Proceedings of the IEEE INFOCOM. 2002.

[6] R. Aversa, B. Di Martino, M. Rak, S. Venticinque, U. Villano. *Performance Prediction for HPC on Clouds*. Cloud Computing: Principles and Paradigms. 2011.

[7] Amazon Elastic Compute Cloud (EC2). http://www.amazon.com /ec2/. 2013.

[8] D. Chappell. *Introducing the Azure services platform*. White Paper. 2008.

[9] Google App Engine. http://appengine.google.com. 2013.

[10] J. Vilaplana, F. Solsona, F. Abella, R. Filgueira, J. Rius. *The Cloud Paradigm Applied to e-Health*. BMC Med. Inf. & Decision Making vol. 13. 2013.

[11] D. Kliazovich, P. Bouvry, S. Khan. *GreenCloud: a packet-level simulator of energy-aware cloud computing data centers*. The Journal of Supercomputing. 2010.

[12] S. Lim, B. Sharma, G. Nam, E. Kim, C. Das. *MDCSIM: A Multi-tier Data Center Simulation Platform*. Proceedings of IEEE International Conference on Cluster Computing. 2009.

[13] A. Nuez, J. Vzquez-Poletti, A. Caminero, G. Casta, J. Carretero, I. Llorente. *iCanCloud: A Flexible and Scalable Cloud Infrastructure Simulator*. J. Grid Computing 10:185-209. 2012.

[14] S.K. Garg, R. Buyya. *NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations*. Fourth IEEE International Conference on Utility and Cloud Computing. 2011.

[15] R. Calheiros, R. Buyya, C. De Rose. *Building an automated and self-configurable emulation testbed for grid applications*. Softw. Pract. Exper. 2010, 40405-429. 2010.

[16] K. Kim, A. Beloglazov, R. Buyya. *Power-aware Provisioning of Cloud Resources for Real-time Services*. Proc. of the 7th Intl. Workshop on Middleware for Grids, Clouds and e-Science. 2009.

[17] A. Beloglazov, R. Buyya. *Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers*. Concurrency and Computation: Practice and Expererience, vol. 24 pp. 1397-1420. 2012.

[18] L. Kleinrock. *Queueing Systems: Theory, vol. 1*. Wiley-Interscience. 1975.

[19] M. Barbeau, E. Kranakis. *Principles of Ad-hoc Networking*. John Wiley & Sons. 2007.

[20] J.R. Jackson. *Networks of waiting lines*. Operations Research, vol. 5, 518-521. 1957.

[21] J.R. Jackson. *Jobshop-Like Queueing Systems*. Management Science, vol. 10, pp. 131-142. 1963.

[22] P.J. Burke. *The Output of a Queuing System*. Operations Research, vol. 4, pp. 699-704. 1956.

[23] H. Karlapudi, J. Martin. *Web application performance prediction*. In Proceedings of the IASTED International Conference on Communication and Computer Networks, pp. 281-286. 2004.

[24] R.D. Mei, H.B. MeeuwissenB. *Modelling end-to-end Quality-of-Service for transaction-based services in multidomain environement*. In Proceedings of the 19th International Teletraffic Congress (ITC19), pp. 1109-1121. 2005.

[25] R. Buyya, M. Murshed. *GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing*. Concurrency Computat.: Pract. Exper. vol. 14 pp. 1175-1220. 2002.

[26] C. Dumitrescu, I. Foster. *GangSim: a simulator for grid scheduling studies*. Proc. of the 5th International Symposium on Cluster Computing and the Grid (CCGrid 05), IEEE Computer Society. 2005.

[27] X. Du, X. Zhang. *Coordinating parallel processes on networks of workstations*. Journal of Parallel and Distributed Computing vol. 46, pp. 125-135. 1997.

[28] J.L. Lerida, F. Solsona, F. Giné, J.R. Garcia, M. Hanzich, P. Hernndez. *Enhancing Prediction on Non-dedicated Clusters*. Lecture Notes in Computer Science EuropPar'2008, vol 5168 pp. 233-242. 2008.

[29] Wei-ping Yang, Li-Chun Wang, Hung-Pin Wen. *A queueing analytical model for service mashup in mobile cloud computing*. Wireless Communications and Networking Conference (WCNC). 2013.

[30] E. Roloff, F. Birck, M. Diener, A. Carissimi, P. Navaux. *Evaluating High Performance Computing on the Windows Azure Platform*. IEEE 5th International Conference on Cloud Computing (CLOUD). 2012.

[31] T.J. Hacker, K. Mahadik. *Flexible resource allocation for reliable virtual cluster computing systems*. High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for. 2011

- Hacker, T.J., Mahadik, K., Flexible resource allocation for reliable virtual cluster computing systems, in International Conference for High Performance Computing, Networking, Storage and Analysis (SC), IEEE 2011

Bibliography

[1] Patrick R. Amestoy, Iain S. Duff, Jean-Yves L'Excellent, and Jacko Koster. A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling, 2001.

[2] M Armbrust, M Armbrust, A Fox, A Fox, R Griffith, R Griffith, AD Joseph, AD Joseph, RH, and RH. Above the clouds: A Berkeley view of cloud computing. *University of California, Berkeley, Tech. Rep. UCB*, pages 07–013, 2009.

[3] Michael Armbrust, Ion Stoica, Matei Zaharia, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, and Ariel Rabkin. A view of cloud computing, 2010.

[4] Algirdas Avižienis, Jean Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[5] Xiaoying Bai, Muyang Li, Bin Chen, Wei Tek Tsai, and Jerry Gao. Cloud testing tools. In *Proceedings - 6th IEEE International Symposium on Service-Oriented System Engineering, SOSE 2011*, pages 1–12, 2011.

[6] Anton Beloglazov and Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. In *Concurrency Computation Practice and Experience*, volume 24, pages 1397–1420, 2012.

[7] Guillaume Bobrie, Gilles Chatellier, Nathalie Genes, Pierre Clerson, Laurent Vaur, Bernard Vaisse, Joël Menard, and Jean-Michel Mallion. Cardiovascular prognosis of "masked hypertension" detected by blood pressure self-measurement in elderly treated hypertensive patients. *JAMA : the journal of the American Medical Association*, 291(11):1342–1349, 2004.

[8] Emma P Bray, Roger Holder, Jonathan Mant, and Richard J McManus. Does self-monitoring reduce blood pressure? Meta-analysis with meta-regression of randomized controlled trials. *Annals of medicine*, 42(5):371–386, 2010.

[9] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A F De Rose, and Rajkumar Buyya. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software - Practice and Experience*, 41(1):23–50, 2011.

[10] Yuan-shun Dai, Bo Yang, Jack Dongarra, and Gewei Zhang. Cloud Service Reliability : Modeling and Analysis. *Science And Technology*, pages 1–17, 2009.

[11] Kevin C Davis, James M Nonnemaker, Matthew C Farrelly, and Jeff Niederdeppe. Exploring differences in smokers' perceptions of the effectiveness of cessation media messages. *Tobacco control*, 20(1):26–33, 2011.

[12] Heather O Dickinson, James M Mason, Donald J Nicolson, Fiona Campbell, Fiona R Beyer, Julia V Cook, Bryan Williams, and Gary A Ford. Lifestyle interventions to reduce raised blood pressure: a systematic review of randomized controlled trials. *Journal of hypertension*, 24(2):215–233, 2006.

[13] Truong Vinh Truong Duy, Yukinori Sato, and Yasushi Inoguchi. Performance evaluation of a green scheduling algorithm for energy savings in cloud computing. In *Proceedings of the 2010 IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum, IPDPSW 2010*, 2010.

[14] Leif Erhardt. Cigarette smoking: An undertreated risk factor for cardiovascular disease, 2009.

[15] C Free, R Whittaker, R Knight, T Abramsky, A Rodgers, and I G Roberts. Txt2stop: a pilot randomised controlled trial of mobile phone-based smoking cessation support. *Tobacco control*, 18(2):88–91, 2009.

[16] Caroline Free, Rosemary Knight, Steven Robertson, Robyn Whittaker, Phil Edwards, Weiwei Zhou, Anthony Rodgers, John Cairns, Michael G. Kenward, and Ian Roberts. Smoking cessation support delivered via mobile phone text messaging (txt2stop): A single-blind, randomised trial. *The Lancet*, 378(9785):49–55, 2011.

[17] Saurabh Kumar Garg and Rajkumar Buyya. NetworkCloudSim: Modelling parallel applications in cloud simulations. In *Proceedings - 2011 4th IEEE International Conference on Utility and Cloud Computing, UCC 2011*, pages 105–113, 2011.

[18] Alfredo Goldman and Yanik Ngoko. A MILP approach to schedule parallel independent tasks. In *Proceedings of the 7th International Symposium on Parallel and Distributed Computing, ISPDC 2008*, pages 115–122, 2008.

[19] Iulia Ion, Niharika Sachdeva, Ponnurangam Kumaraguru, and Srdjan Capkun. Home is Safer than the Cloud ! Privacy Concerns for Consumer Cloud Storage. *Security*, pages 13:1–13:20, 2011.

[20] Alexandru Iosup, Nezih Yigitbasi, and Dick Epema. On the performance variability of production cloud services. In *Proceedings - 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2011*, pages 104–113, 2011.

[21] Alexander Keller and Heiko Ludwig. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, 11(1):57–81, 2003.

[22] Mohammed Faisal Khan, Zaki Anwar, and Qazi Shoeb Ahmad. Assignment of personnels when job completion time follows gamma distribution using stochastic programming technique.

[23] Hamzeh Khazaei, Jelena Misic, and Vojislav B. Misic. Performance analysis of cloud computing centers using M/G/m/m+r queuing systems. *IEEE Transactions on Parallel and Distributed Systems*, 23:936–943, 2012.

[24] Leonard Kleinrock. *Theory, Volume 1, Queueing Systems*. Wiley-Interscience, 1975.

[25] Dzmitry Kliazovich, Pascal Bouvry, and Samee Ullah Khan. GreenCloud: A packet-level simulator of energy-aware cloud computing data centers. *Journal of Supercomputing*, 62(3):1263–1283, 2012.

[26] Kurt Kroenke, Dale Theobald, Jingwei Wu, Kelli Norton, Gwendolyn Morrison, Janet Carpenter, and Wanzhu Tu. Effect of telecare management on pain and depression in patients with cancer: a randomized trial. *JAMA : the journal of the American Medical Association*, 304(2):163–171, 2010.

[27] Alex Mu-Hsing Kuo. Opportunities and Challenges of Cloud Computing to Improve Health Care Services, 2011.

[28] T Lancaster and L F Stead. Individual behavioural counselling for smoking cessation. *Cochrane database of systematic reviews (Online)*, (2):CD001292, 2005.

[29] M R Law, J K Morris, and N J Wald. Use of blood pressure lowering drugs in the prevention of cardiovascular disease: meta-analysis of 147 randomised trials in the context of expectations from prospective epidemiological studies. *BMJ (Clinical research ed.)*, 338:b1665, 2009.

[30] Leslie Lenert, Ricardo F. Muñoz, Jackie Stoddard, Kevin Delucchi, Aditya Bansod, Steven Skoczen, and Eliseo J. Pérez-Stable. Design and pilot evaluation of an internet smoking cessation program. *Journal of the American Medical Informatics Association*, 10(1):16–20, 2003.

[31] Agathe León, César Cáceres, Emma Fernández, Paloma Chausa, Maite Martin, Carles Codina, Araceli Rousaud, Jordi Blanch, Josep Mallolas, Esteban Martinez, Jose L. Blanco, Montserrat Laguno, Maria Larrousse, Ana Milinkovic, Laura Zamora, Neus Canal, Josep M. Miró, Josep M. Gatell, Enrique J. Gómez, and Felipe García. A new multidisciplinary home care telemedicine system to monitor stable chronic human immunodeficiency virus-infected patients: A randomized study. *PLoS ONE*, 6, 2011.

[32] Josep L. Lerida, Francesc Solsona, Porfidio Hernandez, Francesc Gine, Mauricio Hanzich, and Josep Conde. State-based predictions with self-correction on Enterprise Desktop Grid environments. *Journal of Parallel and Distributed Computing*, 73(6):777–789, 2013.

[33] Wenjuan Li and Lingdi Ping. Trust model to enhance security and interoperability of cloud environment. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5931 LNCS, pages 69–79, 2009.

[34] Seung Hwan Lim, Bikash Sharma, Gunwoo Nam, Eun Kyoung Kim, and Chita R. Das. MDCSim: A multi-tier data center simulation platform. In *Proceedings - IEEE International Conference on Cluster Computing, ICCC*, 2009.

[35] B. N. W. Ma and J. W. Mark. Approximation of the Mean Queue Length of an M/G/c Queueing System, 1995.

[36] Rahul Malhotra and Prince Jain. Study and Comparison of Various Cloud Simulators Available in the Cloud Computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(9):347–350, 2013.

[37] Magnos Martinello, Mohamed Kaâniche, and Karama Kanoun. Web service availabilityimpact of error recovery and traffic model, 2005.

[38] Peter Mell and Timothy Grance. The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. *Nist Special Publication*, 145:7, 2011.

[39] R Mermelstein, S Cohen, E Lichtenstein, J S Baer, and T Kamarck. Social support and smoking cessation and maintenance. Technical Report 4, 1986.

[40] M. Mezmaz, N. Melab, Y. Kessaci, Y. C. Lee, E. G. Talbi, A. Y. Zomaya, and D. Tuyttens. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. *Journal of Parallel and Distributed Computing*, 71(11):1497–1508, 2011.

[41] G Ogedegbe and A Schoenthaler. A systematic review of the effects of home blood pressure monitoring on medication adherence. *Journal of Clinical Hypertension (Greenwich, Conn.)*, 8(United States PT - Journal Article PT - Research Support, N.I.H., Extramural PT - Review LG - English OVID MEDLINE UP 20081216):174–180, 2006.

[42] T Ohkubo, Y Imai, I Tsuji, K Nagai, J Kato, N Kikuchi, A Nishiyama, A Aihara, M Sekino, M Kikuya, S Ito, H Satoh, and S Hisamichi. Home blood pressure measurement has a stronger predictive power for mortality than does screening blood pressure measurement: a population-based observation in Ohasama, Japan. *Journal of hypertension*, 16(7):971–975, 1998.

[43] Stephan R Orth and Stein I Hallan. Smoking: A Risk Factor for Progression of Chronic Kidney Disease and for Cardiovascular Morbidity and Mortality in Renal PatientsAbsence of Evidence or Evidence of Absence? *Clinical Journal of the American Society of Nephrology*, 3:226–236, 2008.

[44] Guy Paré, Mirou Jaana, and Claude Sicotte. Systematic Review of Home Telemonitoring for Chronic Diseases: The Evidence Base, 2007.

[45] Birju Patel, Sharon Turban, Cheryl Anderson, Jeanne Charleston, Edgar R. Miller, and Lawrence J. Appel. A comparison of web sites used to manage and present home blood pressure readings. *Journal of Clinical Hypertension*, 12(6):389–395, 2010.

[46] Cuong Pham, Phuong Cao, Zbigniew Kalbarczyk, and Ravishankar K. Iyer. Toward a high availability cloud: Techniques and challenges. In *Proceedings of the International Conference on Dependable Systems and Networks*, 2012.

[47] Thomas G. Pickering, Nancy Houston Miller, Gbenga Ogedegbe, Lawrence R. Krakoff, Nancy T. Artinian, and David Goff. Call to action on use and reimbursement for home blood pressure monitoring: A joint scientific statement from the american heart association, american society of hypertension, and preventive cardiovascular nurses association. *Hypertension*, 52(1):10–29, 2008.

[48] Josep M. Ramon, Isabel Nerin, Araceli Comino, Cristina Pinet, Francesc Abella, José Ma Carreras, Marta Banque, Antoni Baena, Sergio Morchon, Adriana Jimenez-Muro, Adriana Marqueta, Assumpcio Vilarasau, Raquel Bullon, and Cristina Masuet-Aumatell. A multicentre randomized trial of combined individual and telephone counselling for smoking cessation. *Preventive Medicine*, 57(3):183–188, 2013.

[49] A Rodgers, T Corbett, D Bramley, T Riddell, M Wills, R-B Lin, and M Jones. Do u smoke after txt? Results of a randomised trial of smoking cessation using mobile phone text messaging. Technical Report 4, 2005.

[50] Arnon Rosenthal, Peter Mork, Maya Hao Li, Jean Stanford, David Koester, and Patti Reynolds. Cloud computing: A new business paradigm for biomedical information sharing, 2010.

[51] Farzad Sabiha. Cloud Computing Reliability, Availability and Serviceability (RAS): Issues and Challenges, 2012.

[52] L F Stead and T Lancaster. Group behaviour therapy programmes for smoking cessation. *Cochrane database of systematic reviews (Online)*, (2):CD001007, 2005.

[53] L F Stead, R Perera, and T Lancaster. Telephone counselling for smoking cessation. *Cochrane database of systematic reviews (Online)*, 3:CD002850, 2006.

[54] Lindsay F Stead, Rafael Perera, Chris Bullen, David Mant, Jamie Hartmann-Boyce, Kate Cahill, and Tim Lancaster. Nicotine replacement therapy for smoking cessation. *Cochrane database of systematic reviews (Online)*, 11:CD000146, 2012.

[55] Dawei Sun, Guiran Chang, Changsheng Miao, and Xingwei Wang. Building a high serviceability model by checkpointing and replication strategy in cloud computing environments. In *Proceedings - 32nd IEEE International Conference on Distributed Computing Systems Workshops, ICDCSW 2012*, pages 578–587, 2012.

[56] Dawei Sun, Guiran Chang, Lina Sun, and Xingwei Wang. Surveying and analyzing security, privacy and trust issues in cloud computing environments. In *Procedia Engineering*, volume 15, pages 2852–2856, 2011.

[57] Flora Tzelepis, Christine L Paul, John Wiggers, Raoul A Walsh, Jenny Knight, Sarah L Duncan, Christophe Lecathelinais, Afaf Girgis, and Justine Daly. A randomised controlled trial of proactive telephone counselling on cold-called smokers' cessation rates. *Tobacco control*, 20(1):40–46, 2011.

[58] Kashi Venkatesh Vishwanath and Nachiappan Nagappan. Characterizing Cloud Computing Hardware Reliability. In *Proceedings of the 1st ACM symposium on Cloud computing - SoCC '10*, page 193, 2010.

[59] Lizhe Wang, Gregor Laszewski, Andrew Younge, Xi He, Marcel Kunze, Jie Tao, and Cheng Fu. Cloud Computing: a Perspective Study, 2010.

[60] Bhathiya Wickremasinghe, Rodrigo N. Calheiros, and Rajkumar Buyya. CloudAnalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, pages 446–452, 2010.

[61] Kaiqi Xiong and Harry Perros. Service Performance and Analysis in Cloud Computing. In *2009 Congress on Services - I*, pages 693–700, 2009.

[62] Dimitrios Zissis and Dimitrios Lekkas. Addressing cloud computing security issues. *Future Generation Computer Systems*, 28(3):583–592, 2012.