A queuing theory model for cloud computing

Article *in* The Journal of Supercomputing · July 2014

DOI: 10.1007/s11227-014-1177-y

CITATIONS
140

READS
5,364

6 authors, including:

Jordi Vilaplana
Universitat de Lleida
47 PUBLICATIONS **299** CITATIONS
SEE PROFILE

Francesc Solsona
Universitat de Lleida
154 PUBLICATIONS **651** CITATIONS
SEE PROFILE

Ivan Teixidó
Universitat de Lleida
35 PUBLICATIONS **247** CITATIONS
SEE PROFILE

Jordi Mateo fornes
Universitat de Lleida
50 PUBLICATIONS **221** CITATIONS
SEE PROFILE

J Supercomput
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

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

Published online: 09 April 2014

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

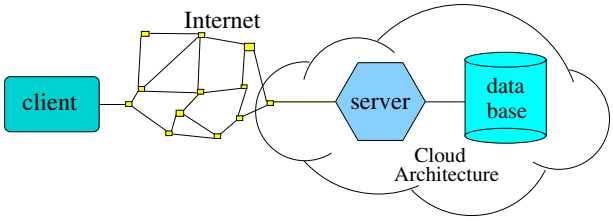


Fig. 1 Cloud computing paradigm

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.

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

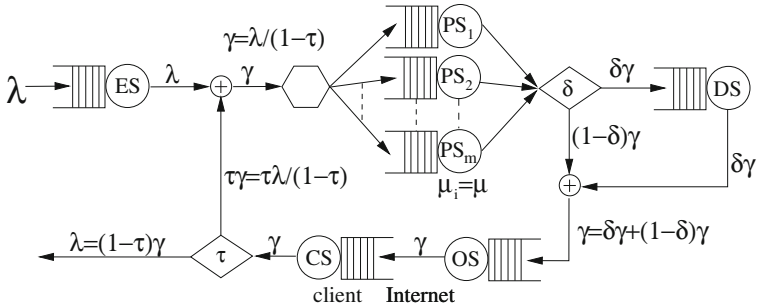


Fig. 2 Model of the cloud architecture

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:

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

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):

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

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.

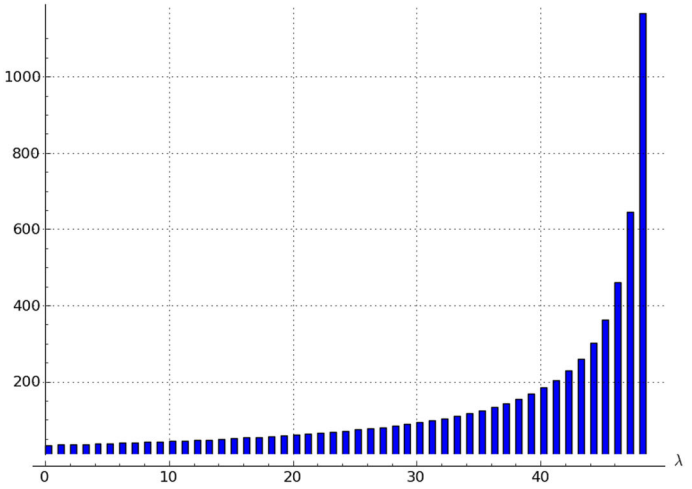
First, we give the results obtained in the simulation. Then, we validate the model by comparing the simulated results with those obtained in a real cloud system.

4.1 Simulation

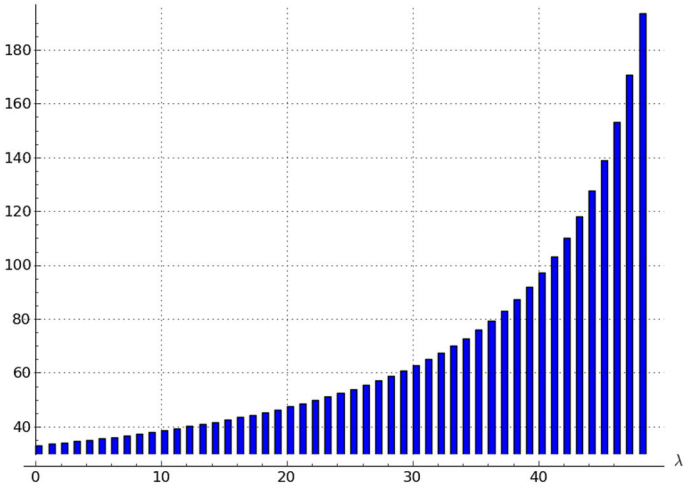
The model was implemented using Sage 5.3 mathematical software[1]. All the results (y-axis) are in simulation units of time, so no Y-labels were placed in the figures. The parameters used in the implementation are described below:

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 $\rho = \gamma/\mu$.

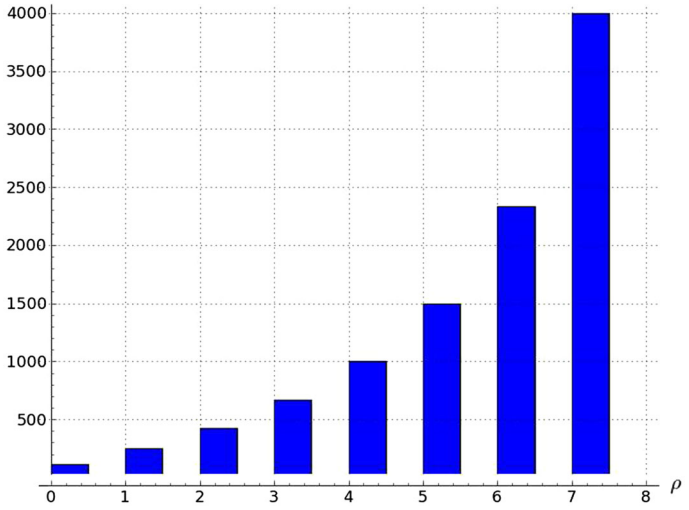


Fig. 4 Total response time (T) of the cloud model in function of ρ

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.

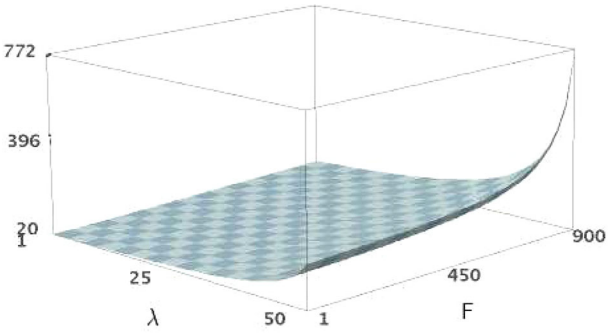
Figure 4 illustrates how the response time of the PS node (T_{PS}) lengthens exponentially as the utilization or traffic intensity (ρ) of the server increases. When the utilization is close to zero, the response time is almost equal to the service time. The response time increases with the utilization 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.

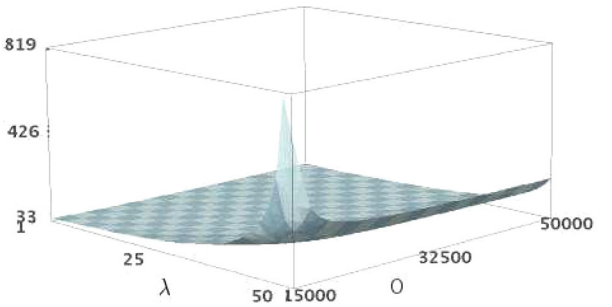
Total response time (T) grows exponentially when the mean file size (F) is increased or when the server bandwidth (O) is decreased. This behavior was expected. When files are too large, the output server OS needs more bandwidth to transfer data to clients across the network (i.e., Internet) efficiently. 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

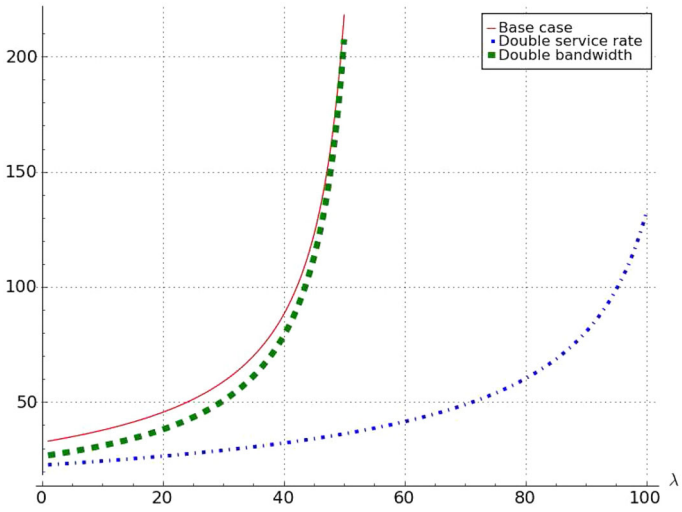


Fig. 6 Total response time (T). Servers are the bottleneck

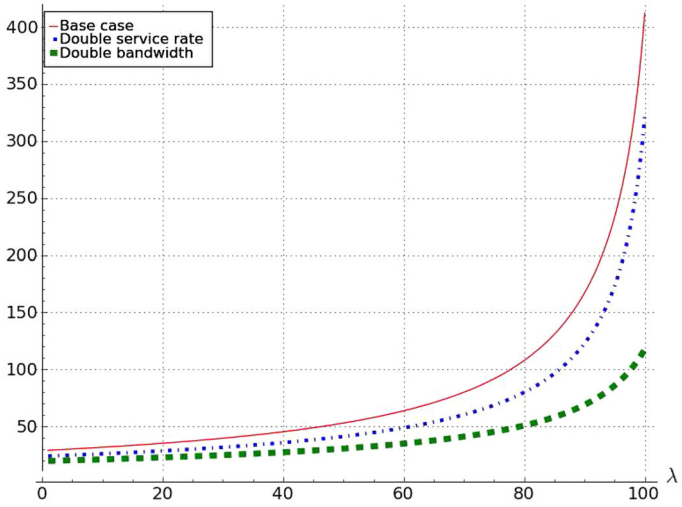


Fig. 7 Total response time (T). Bandwidth is the bottleneck

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 %.

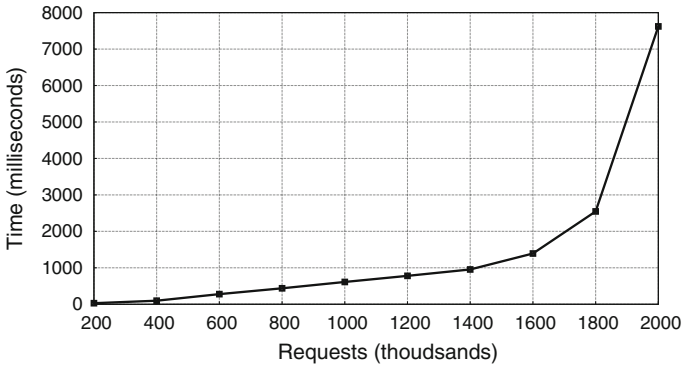
From the given tests, we can confirm that our model represents a cloud accurately. The response time of the system is consistent with extreme situations. We showed that, depending on the system configuration, some parameters become more important or can be rather insignificant for the overall system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

4.2 Model validation

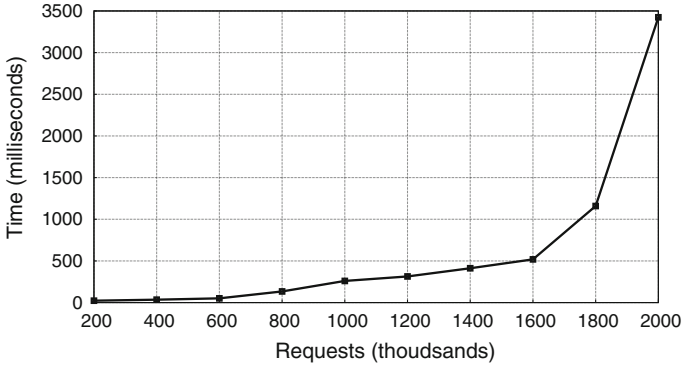
To validate the usefulness of this model in predicting the cloud behavior, we compared the above simulation results with the results from the experimentation in a real environment system.

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