Studying the Impact of the User Subscription Times in Different Cloud Configurations

Hernán-Indibil De la Cruz[1]
HernanIndibil.Cruz@uclm.es

María-Emilia Cambronero[1]
MEmilia.Cambronero@uclm.es

Valentín Valero[1]
Valentin.Valero@uclm.es

Pablo C. Cañizares[2]
pablo.cerro@uam.es

Adrián Bernal[1]
Adrian.Bernal@uclm.es

Alberto Núñez[3]
Alberto.Nunez@pdi.ucm.es

[1]Albacete Research Institute of Informatics, Computer Science Department, Universidad de Castilla-La Mancha
[2]Computer Science Department, Autonomous University of Madrid
[3]Software Systems and Computation Department, Complutense University of Madrid

Abstract

In this paper, we model cloud systems and the user interactions with the cloud provider using the `UML2Cloud` profile. In general, users request virtual machines according to their needs, but they can also subscribe to the cloud provider and wait to be notified when the requested resources are not available. In this case, users indicate a maximum subscription time, so once this time elapses without being notified, users leave the system unattended. In this paper, then, we present an exhaustive research study to measure how the user subscription times affect the overall system responsiveness. In this study, three different cloud configurations are analyzed. Each cloud processes several workloads, which are generated using two distribution functions for the user arrivals, namely a normal and a cyclic normal distribution. The purpose of this study is to find out the inflection point for the waiting time of the users, from which the cloud responsiveness and its performance do not improve. The obtained information is therefore useful for the cloud provider to improve the configuration of the cloud.

1. Introduction

Cloud computing is experiencing important growth nowadays. Cloud service providers need tools that allow them to better manage their resources, with the goal of maintaining the Quality of Service offered to a growing number of customers, agreed in the so-called Service Level Agreements (SLAs). One of these tools is the simulation and, particularly, cloud simulators, which allow us to simulate workloads that are executed in virtual environments. With these tools, we can predict behaviors in the real cloud systems, even before these systems are built and deployed so that they allow cloud providers to anticipate some problems that could arise once the system is running.

In addition, modeling the cloud infrastructure and the users' interactions with the cloud providers allow us to have a better understanding of the behavior of all the roles in these systems. With this purpose in mind, we defined the `UML2Cloud` profile [1]. The main features of the cloud infrastructure, that is, CPUs, storage, and network bandwidth, among others, are considered in this parameterized profile, as well as the exchange of messages between the users and the cloud provider, with parameters such as the specification of the virtual machines required, the applications to be executed on them, and the maximum subscription time when the requested machines are not available at the time of the request.

This paper aims at studying the behavior of simulated cloud environments modeled with the `UML2Cloud` UML profile. In essence, this study focuses on the abandon-rate and waiting time of the users, in order to help finding the best configurations and workloads for the analyzed cloud systems. We investigate the relationships between the number of users trying to be served by the cloud and the waiting time due to users' subscriptions, which is a quality of service-related metric defined in the `UML2Cloud` profile.

There are several works in the cloud literature studying different resource allocation policies with the goal to meet the quality of service (QoS) features. For instance, Kouki et al. [6] present an analytical performance model to predict cloud service performance taking into account the last values for abandon rate, latency, and cost. Following the same line, Wu et al. [12] propose several resource allocation algorithms for SaaS providers to minimize SLA violations and infrastructure costs by managing the workloads. Mateo-Fornés et al. [7] present an analytic model, called CART, for studying cloud availability and response time to improve several QoS items, as performance, cost, and availability in SaaS. There are significant differences with our work, in which we consider the publish-subscribe paradigm and we study the impact of user subscription times on QoS parameters, like response time and performance.

DOI reference number: 10.18293/SEKE2021-095

Vinodhini [11] analyzes a cloud system based on a queueing model with possible failures and cloud repairs, instead of the publish-subscribe paradigm used in this study.

There are other works that analyze different metrics related to performance evaluation. For instance, Yang et al. [13] evaluate cloud performance taking into account the service response time in an environment with fault recovery to improve cloud reliability and considering a queueing system to conduct the performance analysis. Similarly, Khazaei et al. [5] propose an approach also focused on the response time, using other queueing system model.

In general, these works are based on theoretical models and the obtained results are based on the assumptions that need to be established for the analysis of these models. An alternative is the usage of cloud simulation, which is a widely adopted technique that allows us to reproduce the behavior of real cloud environments. Furthermore, simulation allows mitigating some problems related to these environments, such as the experiments reproducibility and the high costs of renting real cloud systems.

In the current literature, we can find multiple proposals based on simulation tools to study different aspects of the cloud [3]. Some simulators focus on resource provisioning algorithms, such as *CloudSim* [2], and *NetworkCloudSim* [4]. Another simulator, *SimIC*, is focused on the management of large-scale resources in inter-cloud environments. Finally, *iCanCloud* [9] helps users of a cloud deciding the best starting conditions on pay-as-you-go scenarios.

In this paper, we focus on the use of cloud simulation. Specifically, we use the Simcan2Cloud simulator [1], which is a simulation tool of parallel and distributed architectures and applications. We use a different perspective, our approach focuses on the users waiting time analysis when they subscribe to the cloud provider. Thus, users are notified when the resources they need are available. This metric is measured in a cloud environment close to saturation, i.e., in a cloud system where a high number of users are requesting resources, in comparison with the cloud size.

In this study, each cloud processes different workloads, which have been generated using two distribution functions for the users' arrival, a normal and a cyclic normal distribution. Thus, we analyze the impact of the maximum subscription times in the cloud behavior, in terms of the requests that are finally served, the average waiting times for users, and the number of unattended users. Subscription time has been chosen as a key parameter in this study because it influences the trade-off between the waiting time and the number of unattended users. If users have a long subscription time, then the queue for resources is enlarged and the average waiting time increases as well. In contrast, users with a short subscription time will leave earlier unattended, keeping the queues with a lower number of users and the average waiting times will decrease.

The paper is structured as follows. Section 2 presents the background. Section 3 shows the methodology used to conduct the experimental phase of our study. In Section 4, a complete study about the impact of the user waiting time in different cloud configurations. And finally, Section 5 presents the conclusions and future work.

2. Background

In this section, we present an overview of the `UML2Cloud` profile [1], which has been created using UML, for the modeling of both cloud systems and the behavior of the users when they interact with the cloud provider. We also describe the *Simcan2Cloud* cloud simulator [1] that we use in the experiments.

2.1. The `UML2Cloud` UML profile

In this profile, a cloud system consists of a cloud provider, one or more data centers, and clients (also called cloud users) requesting resources to the cloud. The cloud provider manages a catalog of Virtual Machines (VMs) and hardware resources provided by the data centers. Each data center consists of a collection of physical machines, also called nodes, which are grouped by racks. Thus, each rack contains a set of nodes with the same hardware features, that is, CPU, memory, and storage. The whole cloud infrastructure is described in a component diagram, which can be found in our previous work [1].

The interactions between the users and the cloud provider are modeled using a sequence diagram (SD).

Figure 1 shows a new version of the SD presented in [1]. The interaction starts with a *request* message from the user, containing a list of all the VMs needed to execute its apps. Each VM is defined as a tuple: *VM*=(number, VM_type, renting_time) where we indicate the *number* of VMs of a certain type (*VM_type*) that we request, and the *renting time*.

The user then enters into a loop to handle the messages received from the cloud provider, until no requested VM is in execution or no subscription is active for this user. The answer to a *request* is a *response* message that contains the set of IPs corresponding to the physical machines containing all the requested VMs, which can be empty if this request cannot be attended to. If the set of IPs is not empty, the request can be served, and the user sends an *execute* message containing the list of applications (APPs) to execute and the list of IPs in which each APP is executed. Otherwise, when the set of IPs is empty, that is, at least one of the VM requested cannot be provided, the user can subscribe to the cloud provider indicating the VMs required and a maximum subscription time (*subscribe* message). The latter is the maximum time that the user is willing to wait for being

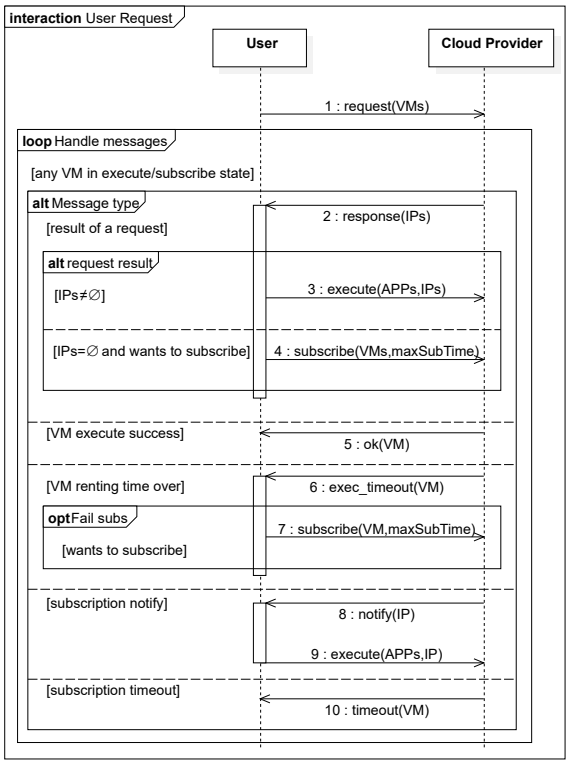


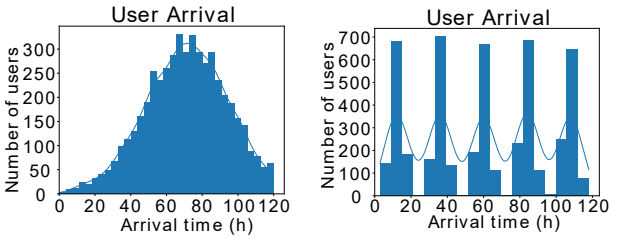
Figure 1: Cloud provider and user interaction SD.

served. When the cloud provider receives an *execute* message, it starts the execution of the APPs in their corresponding VMs. In the case that all the APPs running in a VM finish within the agreed *renting time* period, an *ok* message is sent to the user, indicating the ended VM. These VMs are marked as *finished*.

It can be the case that the APPs running in some VMs do not finish in the agreed *renting time*. An *exec_timeout* message is then sent to the user for each VM that was not able to complete its workload. In this case, the user can decide to subscribe to these VM characteristics in order to be notified when a VM fulfilling these features allows resuming the APPs execution. For this purpose, the user sends a *subscribe* message containing the VM characteristics and the maximum subscription time (*maxSubTime*) that she is willing to wait. A *notify* message will then be sent to the user as soon as a VM fulfilling these features is available. However, it can also be the case that the maximum waiting time elapses and no VM is available to resume the execution. In this case, a *timeout* message is sent to the user indicating the VM that could not be resumed.

2.2. Simcan2Cloud Simulator

`Simcan2Cloud` [1] is a cloud simulator written in C++ using OMNeT++ [10], which is a cloud extension of `SIMCAN` [8], a tool for the simulation of parallel and distributed architectures and applications. The implementation of `Simcan2Cloud` fulfills the cloud specifications and the user interactions defined in the `UML2Cloud` profile. `Simcan2Cloud` is designed to provide a high level of flexibility, allowing the user to set up the cloud configuration in a modular way, in terms of data centers, computing and storage nodes, and network connections, among other components. Thus, `Simcan2Cloud` allows us to model and analyze different cloud scenarios.



(a) Normal user arrivals. (b) Cyclic normal user arrivals.

Figure 2: Distributions of number of user arrivals per time intervals.

3. Methods

In this section, we describe the methodology used to study the impact of the maximum user subscription time in different cloud configurations. In this study, we consider three cloud configurations, consisting of 64, 96, and 128 physical machines, respectively, where all machines have the same configuration (CPU and storage). These configurations have been chosen to analyze the impact of subscription times as we increase the number of nodes. Each cloud system processes several workloads, which are generated by establishing the inter-arrival time for the users, who execute the same application on the VMs. Each workload consists of 5000 users that request services to the cloud provider for a period of 5 days, where each VM is rented for 2 hours.

The first group of experiments analyzes the cloud responsiveness when users come at a normal distribution basis, with a single peak in the workload. In this case, a normal distribution with a mean of 3 days and a standard deviation of 1 day has been considered (see Figure 2a), where the x-axis represents the user arrival time and the y-axis shows the number of users. The second group of experiments analyzes the impact of daily burst user arrivals and, thus, a cyclic normal distribution has been considered (see Figure 2b). This figure represents the repetition of the same normal distribution in cycles of 24 hours of duration, with strong daily peaks at midday, where the x-axis represents the user ID and the y-axis shows the arrival time (in hours). In this case, therefore, we consider a normal distribution with a mean of 12 hours and a standard deviation of 3 hours.

The main goal of this study is to analyze the impact of the maximum subscription time that users establish when they

subscribe to the cloud provider. For simplicity, all the users assign the same value for the maximum subscription time, so the experiments are repeated using different values for this parameter. Thus, we put the focus on the waiting time obtained for the users when they intend to execute their applications in the cloud. The results obtained when different values for the maximum subscription time are used, provide us with valuable information about the responsiveness of the cloud and allows us to conclude the best configurations according to the submitted workload.

4. Results and Discussion

In this section, we show the results obtained from the empirical study described in Section 3. First, in Section 4.1 we show an experiment in which the cloud processes a user workload generated using a normal distribution. Next, in Section 4.2, we conduct an experiment in which the cloud processes a user workload generated using a cyclic normal distribution. Finally, we present a discussion of the obtained results in Section 4.3.

4.1. Case Study 1: Normal Distribution

In this scenario, users arrive by following a normal distribution with a mean of 3 days and a standard deviation of 1 day.

Figure 3 shows the results obtained for a cloud consisting of 64 physical machines, considering the following values for the maximum subscription time: 30, 50, and 70 hours. In these charts, the x-axis shows the user IDs, while the y-axis shows the waiting time for the users to be attended to. Black dots represent users that were fully served, while the red ones represent users that left the system without being served. The latter situation occurs when the maximum subscription time elapses and the cloud is not able to provide the user the requested resources. The first chart (left) shows the results obtained for a maximum subscription time of 30 hours. In this case – approximately – the first 1000 users are immediately served, i.e. their waiting time is 0. However, as more users arrive at the system, the cloud becomes more saturated and, approximately, when 2500 users are processed, the cloud cannot serve the new users' requests, so they leave the system without being served (red dots at the upper area). Finally, once the user arrivals slow down after the peak, we can see that the final users are again served, but with a high waiting time.

When the subscription time is set to 50 hours (central figure) similar results are obtained. However, the point at which users leave the cloud is obtained when – approximately – 3700 users are attended. When the subscription time is set to 70 hours (right figure), we can see that the cloud can attend to all the requests, and the maximum waiting time is about 63 hours. This is the inflection point for the *user waiting time*, that is, the point at which the cloud responsiveness reaches the worst value and, at the same time, it can attend to all the users' requests, so it should be the maximum subscription time for the users if they wish their works to be executed.

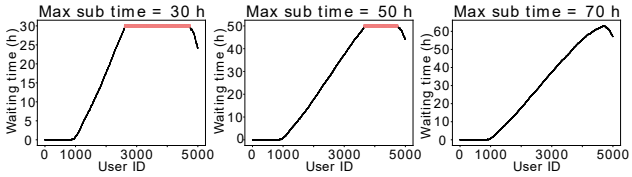


Figure 3: Case study 1 with 64 computing nodes.

Figure 4 shows the results for a cloud consisting of 96 physical machines, where the maximum subscription time ranges from 10 to 30 hours. This figure shows similar results to those obtained in the previous case. However, it is important to note that we have considered smaller values for the maximum subscription time. Thus, in this case, the inflection point is of 21.32 hours, so this should be the maximum subscription time for users that want their works to be executed.

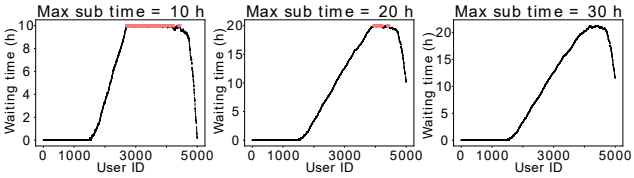


Figure 4: Case study 1 with 96 computation nodes.

Figure 5 shows the results obtained for a cloud with 128 physical machines. In this case, we consider 2, 4, and 6 hours for the maximum subscription time. The first chart (left) shows how the cloud saturation appears at about user 2800, from which some users leave the cloud without being served. We notice that some users can actually be served in the upper part of the normal distribution as a consequence of the specific random numbers that were generated (see Figure 2a). However, in general, we would obtain a red line in the upper part of the figure. Finally, the final users can be attended to with better waiting times as in the previous cases. In the central chart, we can see the results for a maximum subscription time of 4 hours. In this case, most of the users can be served, and only a few of them must leave the cloud being unattended. When 6 hours are considered as maximum subscription time, all the users are served. In fact, the inflection point for the *user waiting time* is of 5.16 hours (see Table 1).

4.2. Case Study 2: Cyclic Normal Distribution

In this experiment, we consider a workload in which the users arrive following a cyclic normal distribution. Thus,

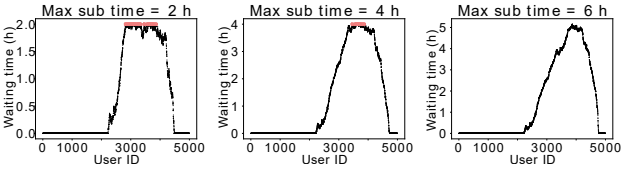


Figure 5: Case study 1 with 128 computation nodes.

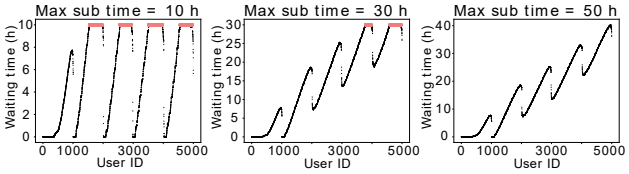


Figure 6: Case study 2 with 64 computation nodes.

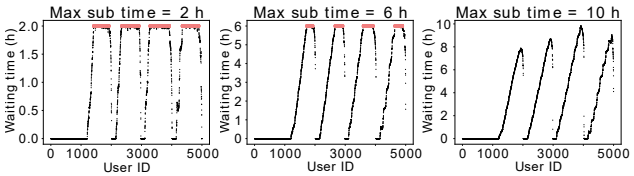


Figure 7: Case 2 with 96 computation nodes.

the workload has 5 peaks, so every day at midday we have – approximately – 1000 users requesting services to the cloud, with a peak of approximately 700 users.

Figure 6 shows the results of a cloud consisting of 64 physical machines processing the workload, using 10, 30, and 50 hours as maximum subscription times. The first chart (left) shows that the cloud can attend to all the users arriving during the first day, although some of them have to wait. However, in the following days, we have that many users must leave the cloud without being served. Using a maximum subscription time of 30 hours (central chart), users are served during the first 3 days. However, as the waiting time increases, they start to leave during the fourth day. Finally, considering 50 hours as maximum subscription time, the cloud can attend to all the users. Their waiting times reach up to 40.29 hours for some users, which is the inflection point in this case.

Figure 7 shows the results obtained for a cloud with 96 physical machines and a maximum subscription time of 2, 6, and 10 hours. The inflection point, in this case, is about 9,82 hours. The first chart of this figure (left) refers to the cloud processing the workload using a maximum subscription of 2. In this case, we observe again that many users leave the cloud from the second day onwards, because we still have in execution the applications from previous users, even from previous days. If we consider a maximum subscription time of 6 hours, only a few users leave the system (central figure) and taking 10 hours (right figure) all users are served, and the waiting times tend to stabilize in the peaks.

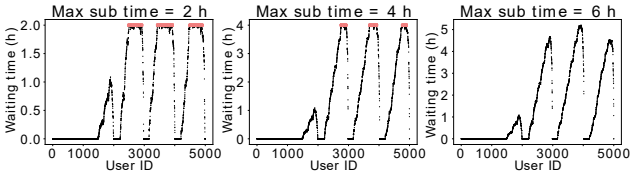


Figure 8: Case study 2 with 128 computation nodes.

The results for a cloud configuration with 128 nodes are presented in Figure 8, using 2, 4, and 6 hours as maximum subscription times. In this case, the inflection point is about 5.20 hours. This system is now able to serve the users' requests for the first two days, with small waiting times, and from the third day on-wards the waiting times increase above 2 hours. Thus, we have reduced the inflection point in a factor close to 8 in comparison with the 64-nodes configuration, and in a factor of 2 with respect to the 96-nodes configuration.

4.3. Discussion of the results

This section provides a brief discussion of the obtained results. Table 1 shows the values for the inflection points in all the experiments.

Regarding the first set of experiments, where the studied clouds process a workload generated using a normal distribution, we observe that, in general, the maximum subscription time has a significant impact on the overall system performance. Increasing this parameter allows more users to be attended to by the system. Additionally, increasing the number of physical machines also impacts positively in the cloud performance, which allows us to reduce the maximum subscription time in order to process the same amount of users. In particular, a cloud with 64 physical machines is unable to attend to all the user's requests when short subscription times are considered, and the users must set up subscription times of several days if they want their applications to be executed. In contrast, with a cloud with 128 nodes we have seen that the inflection point has been reduced by a factor of 12, and with the configuration with 96 nodes the reduction is about 1/3.

The results obtained in the next set of experiments, that is, where the clouds process a workload consisting of daily blurts, render similar results. A cloud with 96 nodes would offer responses below 10 hours for the users' requests, and an investment to improve the cloud infrastructure up to 128

Table 1: Inflection points for the user waiting times (hours).

	64 nodes	96 nodes	128 nodes
Case 1	63.02	21.32	5.16
Case 2	40.29	9.82	5.20

nodes would produce a gain of one half in the waiting times. Obviously, the final decision strongly depends on the applications and the users, who usually pay for the use of the cloud, so the cloud provider should take into account all of these aspects to make a final decision.

Broadly speaking, these results provide relevant and valuable information for the cloud provider, so as to improve the cloud configuration with the goal of increasing the overall income by adapting the physical resources and the internal configuration parameters.

5. Conclusions

In this paper, we have studied the impact of the users' maximum subscription time in three different cloud configurations, considering an infrastructure consisting of 64, 96, and 128 physical machines, respectively. Two models of workload were analyzed, taking two different distribution functions for the users' arrivals, namely, a normal and a cyclic normal. Thus, in the first case, we analyzed the impact of a single peak in the users' arrivals, and in the second case, we considered daily peaks at midday. In this study, the responsiveness of the cloud was then analyzed, to conclude which configurations provide better results according to the workload submitted and the maximum subscription times indicated by the users. We concluded that increasing the number of physical machines and the maximum subscription time produce better responsiveness. However, the cloud provider must take the final decision, that is, to make an investment by including more resources to the cloud, or to reduce the overall cloud performance by increasing the maximum subscription time for the users.

As future work, we will extend this study by considering other parameters, such as the offered VMs, the storage system and the communication network. We will also consider some other distribution functions for the user arrivals, such as the exponential and Erlang distributions. Furthermore, we plan to include costs in the use of the cloud by the users, to make a deeper analysis of the profits.

ACKNOWLEDGEMENTS

This work was supported by the Spanish Ministry of Science and Innovation (co-financed by European Union FEDER funds) project references RTI2018-093608-B-C32 and RTI2018-095255-B-I00. There was also support from the Junta de Comunidades de Castilla-La Mancha project SBPLY/17/180501/000276/01 (cofunded with FEDER funds, EU), the Region of Madrid (grant number FORTE-CM, S2018/TCS-4314), and the Madrid Government (Comunidad de Madrid-Spain) under the Multiannual Agreement with the Complutense University as part of the Program to Stimulate Research for Young Doctors in the context of the V PRICIT (Regional Programme of Research and Technological Innovation) under grant PR65/19-22452.

References

[1] A. Bernal, M. E. Cambronero, A. Núñez, P. C. Cañizares, and V. Valero. Improving cloud architectures using UML profiles and M2T transformation techniques. *The Journal of Supercomputing*, 75(12):8012–8058, 2019.

[2] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[3] F. Fakhfakh, H. H. Kacem, and A. H. Kacem. Simulation tools for cloud computing: A survey and comparative study. In *IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS'17)*, pages 221–226, 2017.

[4] S. K. Garg and R. Buyya. NetworkCloudSim: Modelling parallel applications in cloud Simulations. In *4th IEEE International Conference on Utility and Cloud Computing (UCC'11)*, pages 105–113, 2011.

[5] H. Khazaei, J. Misic, and V. B. Misic. Performance analysis of cloud centers under burst arrivals and total rejection policy. In *IEEE Global Telecommunications Conference (GLOBECOM'11)*, pages 1–6, 2011.

[6] Y. Kouki and T. Ledoux. SLA-driven capacity planning for Cloud applications. In *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings (CLOUDCOM'12)*, pages 135–140, 2012.

[7] J. Mateo-Fornés, F. Solsona-Tehàs, J. Vilaplana-Mayoral, I. Teixidó-Torrelles, and J. Rius-Torrentó. Cart, a decision sla model for saas providers to keep qos regarding availability and performance. *IEEE Access*, 7:38195–38204, 2019.

[8] A. Núñez, J. Fernández, R. Filgueira, F. García, and J. Carretero. SIMCAN: A flexible, scalable and expandable simulation platform for modelling and simulating distributed architectures and applications. *Simulation Modelling Practice and Theory*, 20(1):12–32, 2012.

[9] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente. iCanCloud: A flexible and scalable cloud infrastructure simulator. *Journal of Grid Computing*, 10(1):185–209, 2012.

[10] A. Varga and R. Hornig. An overview of the OMNeT++ simulation environment. In *1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops (Simutools '08)*, pages 1–10, 2008.

[11] G. A. F. Vinodhini. Cloud computing as a queue model with server breakdown. *Advances in Mathematics: Scientific Journal*, 9(10):8217–8225, 2020.

[12] L. Wu, S. K. Garg, and R. Buyya. SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments. In *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID '11)*, pages 195–204, 2011.

[13] B. Yang, F. Tan, and Y.-S. Dai. Performance evaluation of cloud service considering fault recovery. *The Journal of Supercomputing*, 65(1):426–444, 2013.