A CLOUD BROKER APPROACH WITH QOS ATTENDANCE AND SOA FOR HYBRID CLOUD COMPUTING ENVIRONMENTS

Mário Henrique de Souza Pardo, Adriana Molina Centurion,
Paulo Sérgio Franco Eustáquio, Regina Helena Carlucci Santana,
Sarita Mazzini Bruschi and Marcos José Santana

Institute of Mathematical and Computer Sciences (ICMC)
University of São Paulo (USP)
São Carlos, Brazil
{mhpardo,amolina,psfe,rcs,sarita,mjs}@icmc.usp.br

ABSTRACT

Cloud Computing is the industry whose demand has been growing continuously since its appearance as a solution that offers different types of computing resources as a service over the Internet. The number of cloud computing providers grows into a run, while the end user is currently in the position of having many pricing options, distinct features and performance for the same required service. This work is inserted in the cloud computing task scheduling research field to hybrid cloud environments with service-oriented architecture (SOA), dynamic allocation and control of services and QoS requirements attendance. Therefore, it is proposed the QBroker Architecture, representing a cloud broker with trading features that implement the intermediation services, defined by the NIST Cloud Computing Reference Model. An experimental design was created in order to demonstrate compliance to the QoS requirement of maximum task execution time, the differentiation of services and dynamic allocation of services. The experimental results obtained by simulation with CloudSim prove that QBroker has the necessary requirements to provide QoS improvement in hybrid cloud computing environments based on SOA.

KEYWORDS

Cloud Broker, Cloud Computing, SOA, QoS, dynamic service allocation, deadline, task scheduling algorithm, intermediation, NIST Reference Model.

1. INTRODUCTION

The growing adoption of cloud computing as a solution to infrastructure, platform or software offering as a service has grown so much (about 32.8% increase, according to a forecast by the Gartner Group [1] for the year 2015) that the market and the cloud computing environments are becoming increasingly crowded and complex.

David C. Wyld et al. (Eds) : CCSEA, CLOUD, DKMP, SEA, SIPRO - 2016
pp. 63–82, 2016. © CS & IT-CSCP 2016 DOI : 10.5121/csit.2016.60207

This complexity goes beyond the physical infrastructure of data centers, as currently the major trend has been the multiplicity of providers and the construction of complex organizations involving multiple data centers, such as cloud federations [2], the inter-clouds [3] [4], and hybrid clouds [5] [6], among others. In these approaches, the complexity is revealed when we try to provide resources for a range of users with different needs of applications and services [4], bearing in mind the possibility that the solution to the user request may be in an environment with multiple suppliers with infrastructure managed in completely different forms, i.e., it is a highly heterogeneous computing environment [3] [7].

To tackle problems arising from the allocation of cloud resources and meet the demands of users based on quality of service (QoS) requirements, there is now one of the most discussed topics in cloud computing research field: the intermediation process and task scheduling to cloud computing environments [3].

The recent works which focus their efforts on solving specific problems inherent in cloud environments, such as energy efficient consumption, allocation and migration of virtual machine instances, optimizations in data communication through computer networks within data centers [6] [8] [9] [10], among many other issues, implement, in their methodology, cloud brokers created with strict scheduling policies focused on system balancing for seeking specific goal. However, the new reality of brokering activity for cloud systems is the use of an intermediary architecture represented by a broker that may be multi-objective.

This work relates to the task scheduling and intermediation activity research field, proposing a new Cloud Broker architecture, implemented as simulation entity for CloudSim, working this way as an extension to this cloud computing simulation toolkit. The Broker implemented has the characteristic of openness, i.e., is designed to be coupled to various modes of operation, using as a basis for such implementation the NIST Cloud Computing Reference Model [11] and the operation mode of intermediation services for the experiments.

The remainder of this paper is organized as follows: Section 2 presents the related work reviewed and discussed; Section 3 presents in detail the new Cloud Broker Architecture implemented; Section 4 introduces the design of experiments and the simulation scenario designed to test the Cloud Broker; Section 5 consists of the discussion of the experimental results; Section 6 presents the final conclusion of the work; in Section 7 are presented the acknowledgements and the last section is a list of references.

2. RELATED WORK

The CloudSim Toolkit became an adopted framework for evaluating the test environments of many recent jobs published on the Cloud Computing research field, which mention the tool as relevant and capable of providing the necessary resources for modeling and simulation [12] [13] [14] [15] [16].

In [17], the authors propose a cloud broker architecture for selecting a cloud provider from multiple providers' instances. The cloud broker designed measures the QoS of each provider and sorts them according to the client's request requirements. For differentiation of cloud providers there is the Service Measurement Index (SMI), a relative index calculated to provide the requester a perception gap between the services of different providers. Proper provider selection

technique called TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) is based on the establishment of a ranking for selecting an appropriate cloud provider. The experimental results of this work were obtained from experiments on simulation CloudSim. It conducted a set of experiments considering 6 providers and the authors' conclusion was that the application of the chosen set of techniques allowed an efficient selection of cloud providers based on customer requirements.

In another recent work which deals with the problem of service selection in cloud environments with multiple providers [18], the authors propose a project through a solution approach with a multi agent broker. The Jumper Firefly Algorithm was used in the implementation to reduce the execution time of make span time (response time) through a status table which records past behavior. The validation of all the propositions made at work was carried out with the aid of CloudSim simulation environment. In the experimental results, according to statements of the authors, the Firefly Jumper Mechanism is more effective than the standard Firefly Algorithm and other heuristics that were tested.

In another related work that employ their efforts on rapid and effective execution of jobs sent by users to a cloud computing environment [19], the authors propose a communication framework between the broker elements and the virtual machines (VMs), seeking cost and execution optimal results, that was named Broker Virtual Machine Communication Framework (BVCF). The testing environment was constructed with assistance from CloudSim simulator and its API, creating VM scheduling policies based on cost. In the context of the simulated environment programming were also considered cloudlets scheduling and cloudlets relay, and the review of the implementation of the tasks execution was carried out through the Round Robin and FCFS policies. According to the results obtained in testing and analysis conducted by the authors of work, cost factors and task runtime are always the primary components of the constraints of service quality required by customer requests.

In a job that believes in the growth of the computer market demand and the evolution of the industry into the era of cloud federations and inter-clouds [20], the authors state that the aggregate values to cloud services that will be most valued by customers will be pricing or ticketing policy, the allocation scheme of resources to provide the best performance as the signed service level agreements (SLA). The implementation of the work was carried out with the aid of CloudSim Toolkit version 3.0.3, whereby the authors implemented a broker for cloud federations, which works with the intermediation process, interoperability and negotiation of service requests. According to the authors and the experimental results, it is concluded that the resource allocation model based on QoS and reimbursement worked and successfully demonstrated the applicability and necessity of observation of the QoS degradation in complex environments inter-cloud.

In a work that implements a new scheduling model for cloud computing environments called ICMS (Inter-Cloud Meta-Scheduling) [21], the researchers also created an extension of CloudSim Toolkit which was named SimIC (Inter-Cloud). The goal was to meet the complex simulation scenarios in which inter-clouds contexts are considered and the process of intermediation requests (cloudlets) is done by multiple cloud meta-brokers running dynamic management and real-time workloads received using a standard decision-making to made tasks scheduling. The metrics used for the analysis were Execution Time and RTT (Round Trip Time) and as modification factors of simulated environments were used different user submissions and computational requirements. From the comparative experimental results between the values

returned for the original CloudSim Toolkit and for the SimIC, it was possible to verify and conclude that there were considerable gains in the new algorithms implemented by the ICMS module, especially in the graphs comparing results of execution time metrics.

All related work carried out have important features and contributions related to task scheduling to cloud computing systems using CloudSim. From the observation of all cloud broker implementations made in related work, it is possible to see the existing gap on the issue of standardization of a broker architecture that can be used in order to mix and permit the development and application of various types of scheduling strategies considering multiple service quality factors considered in the related articles. In this paper, the simulation environment includes a QBroker Entity with QoS negotiation for incoming requests, adding a set of desirable characteristics in simulation scenarios that want to provide more realistic and similar results to the real-world cloud systems.

3. CLOUD BROKER PROPOSED ARCHITECTURE

This section will present the cloud broker architecture designed in this work, which was named *QBroker* (QoS Broker). The goal of the implementation was to add features to existing *DatacenterBroker* class in CloudSim API. The version of CloudSim considered in the implementation of the extension was to 3.0.3.

As already mentioned, the main implementation consists of a subclass of *DatacenterBroker* class, which is in *org.cloudbus.cloudsim* package, which was called *QBroker*. It is important to note that *DatacenterBroker* class also has an inheritance relationship with *SimEntity* class belonging to *org.cloudbus.cloudsim.core* package. Through inheritance it was possible to harness and hone, in QBroker class, methods previously inherited from *SimEntity* and *DatacenterBroker* classes.

3.1 QBroker Operation Modes

One of the major new features implemented in the QBroker class is related to the operating modes of this component in cloud architecture. According to the reference model of the NIST [11], the operating modes are the directives that guide how cloud brokers entities must meet customer requests and relate to the resources of service providers. Thus, NIST defines three main models of operation: intermediation, aggregation, and arbitrage. The definition of each of the operation modes of a cloud broker, according to direct reference to NIST [11] model, is presented below:

- **Intermediation:** A Cloud Broker can increase the performance of a given service increasing any specific capacity and providing value-added services to customers. Such performance improvement can be achieved with the management of services, identity management, performance reporting, enhanced security, among others.

- **Aggregation:** A Cloud Broker can combine and integrate multiple services in one or more services. The Broker provides data integration and ensures secure data communication between client and provider.

- **Arbitration**: the arbitration operation mode is similar to the Services Aggregation, with the exception that the services that are grouped are not fixed. In services arbitration, a cloud broker has the flexibility to choose services from multiple providers' services. To perform such activity, for example, the broker can use a credit scoring service to evaluate and select the provider with the best reputation for that type of service requested by the customer.

The new QBroker entity was developed seeking the implementation of all the above operating modes, however, for this specific paper, a version of QBroker is presented in which only the services intermediation operation mode has been developed.

3.2 QBroker Services Intermediation

The process of services intermediation defines some actions for cloud broker in its task as mediator between customers and cloud providers. Increase one or more capabilities of a given service mean improving the quality of service. Therefore, this increase in the providers' service QoS can be achieved in many ways, so that the NIST reference model left open the possibility for the cloud brokers developers.

In this work, the mode of operation of intermediation services was designed to allow that QBroker negotiates the execution of individual requests (cloudlets) with one or more cloud service providers, giving priority to the QoS parameters required by the client and also ensuring the quality of the services, so that, by detecting a degradation of service, the Broker acts allocating new resources (VMs and/or services instances), in order to maintain the satisfactory execution performance and the compliance with other requirements in the requests.

The operating procedure for activity flows related to QBroker Services Intermediation Algorithm is shown in Figure 1, formatted as an UML Activity Diagram (Unified Modeling Language).

Adjustments were made in *Cloudlet* class from *org.cloudbus.cloudsim* package, in which the following class attributes have been added:

- *maxExecutionTime*: variable type *double* in which is stored the maximum execution time or execution deadline.
- *service:* variable type *int* to mark the requested service id.
- *arrivalTime:* variable type *double* that hosts the arrival time of cloudlet at the broker.
- *clientID:* variable type *int* used to identify the source client of a request.
- *sendTime e receiveTime:* are variables of type *double* that are used to store the time of submission of the request by a client and the receipt of cloudlet executed on the client.

It is interesting to notice that this intermediation mode of operation in QBroker is always looking to accomplish the QoS requirement of maximum execution time. This makes the implementation of the operation mode fairly close to the services intermediation definition of NISTCloud Computing Architecture [22].

This characteristic also allows customers to get the results of your requests with quality of service in a hybrid cloud computing environment, always giving priority to the allocation of resources in private cloud and, when needed, allocating resources in the public cloud.

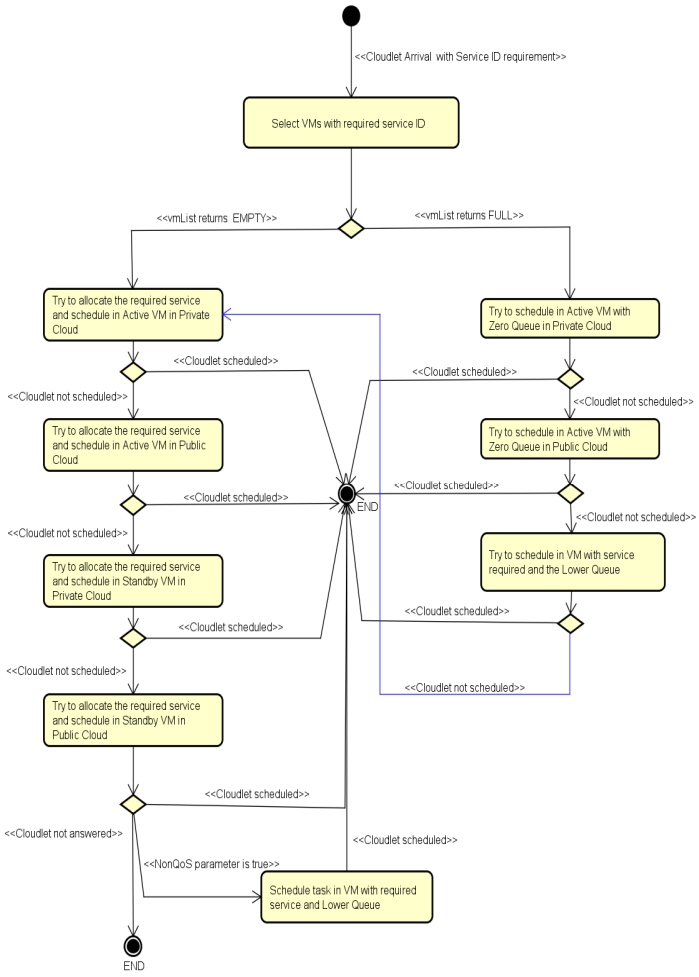


Figure 1: Activity Diagram of QBroker Service Intermediation Operation Mode

3.3 QBroker Class Simulation Events

The simulation entity QBroker has some specific events that were created in addition to support several actions that should occur during the simulation time. For receiving individual requests (cloudlets) the event NEW_CLOUDLET_ARRIVAL was created, through which the cloud broker may receive individual cloudlets during the simulation. It is responsible for receiving task routines, booking and forwarding to the scheduling function and subsequent job submission to a datacenter.

3.4 QBroker class Relationship with other simulation components

To perform its functions during the execution of the simulations, the *QBroker* entity works together with other two important classes implemented in addition: *MetaCloudletScheduler* class and *RequestMonitor* class (which is also an extension of *SimEntity* class). These three classes coexist in the same package named *br.icmc.usp.lasdpc.BeQoS.classes*.

The QBroker class has an instance of RequestMonitor class, in this way, whenever an event of arrival of individual or in group request occurs, the QBroker signals the event of arrival so that the *RequestMonitor* entity receives such notification and account the requests received in cloud broker. The MetaCloudletScheduler class serves as support for QBroker, having all methods that implement the desired scheduling strategies for the cloud computing environment. It is through this class that QBroker is no longer a cloud broker with a rigid systematic task scheduling, offering now the possibility of implementing other scheduling methods. In *MetaCloudlet Scheduler* are methods that allow different types of verification related to resources, whether VMs or services, so that the mediation process is successful.

4. DESIGN OF EXPERIMENTS

In this work were planned three sets of experiments in order to test and demonstrate the features implemented in the intermediation process performed by *QBroker*. All experiments were repeated 10 times, each repetition during 9000 seconds (simulation time based on the clock tick of CloudSim) with 95% confidence interval according to the *T-Student Table*.

4.1 Datacenter and Virtual Machine Configuration

The characterizations adopted for cloud computing simulated scenario were standardized to the three sets of experiments. The scenarios are set up with private cloud or hybrid cloud. The datacenter configurations for private cloud are demonstrated in Table 1.

Table 1: Settings for Private Cloud Infrastructure.

Private Datacenter – Host Configuration	
MIPS/Core:	10000
Cores/Host:	4
RAM:	8000 Mb
Network Bandwidth:	1000 Mbps
Storage:	500000 Mb
OS:	Linux
VMM:	Xen
Total Number of Hosts:	5

The settings of the VMs from private cloud datacenter are shown in Table 2.

Table 1: Settings for Private Cloud VMs.

Private Datacenter – VM Configuration	
MIPS/Core:	10000
PEs Number(Core):	1
RAM:	2000 Mb
Network Bandwidth:	100 Mbps
Image Size:	125000 Mb
VMM:	Xen
Total Number of VMs:	20

The settings used in the simulation scenario with hybrid cloud computing are designed with a public cloud datacenter with settings as demonstrated in Table 3.

Table 3: Settings for Public Cloud Infrastructure.

Public Datacenter – Host Configuration	
MIPS/Core:	20000
Cores/Host:	8
RAM:	32000 Mb
Network Bandwidth:	10000 Mbps
Storage:	1000000 Mb
OS:	Linux
VMM:	Xen
Total Number of Hosts:	2

In the implemented simulation scenario, a total number of 10 VMs on public cloud datacenter was created. The settings for Public VMs are shown in Table 4.

Table 4: Settings for Public Cloud VMs.

Public Datacenter – VM Configuration	
MIPS/Core:	20000
PEs Number(Core):	1
RAM:	4000 Mb
Network Bandwidth:	1000 Mbps
Image Size:	250000 Mb
VMM:	Xen
Total Number of VMs:	10

Also related to cloud computing simulated scenario, the client layer settings were implemented considering a systematic of service demand generation and a fixed amount of customers.

4.2 Service Demand and Client Settings

With regard to service demand generating, a table of service identifiers and their demands in MI (millions of instructions) has been implemented. The service demand for each cloudlet is assigned based on the requested service ID as a specific exponential distribution for each service. The exponential distribution considered has average value of 70000 MI. The total number of possible services, which were considered in the scenario, is 5. It is important to remember that the demand for MI is applied to the length field of each cloudlet, which specifies the size of each task. The services demand values considered in the experiments are listed in Table 5.

Table 2: Service Demand Settings.

Service ID	***Demand (MI)***
S1	30000
S2	50000
S3	70000
S4	90000
S5	110000

The amount of client entities was set to 150 units for all scenarios. The client type configuration, which sets the simulation time client entity operating mode, it was sending requests in real time, meaning that the requests are sent by clients during the course of CloudSim logical clock, creating a more realistic and reliable arrival process to the real world. The generation of service

IDs to be inserted into each request was also made in a random manner considering only 5 services.

To make the heterogeneous service demand, a method in the Client class generates random values that are associated with a service ID as a distribution in percentage. This distribution created can be seen in Table 6.

Table 3: Distribution of services random generation to the requests of client entities

Service ID	*Distribution (%)*
S1	5.0
S2	15.0
S3	60.0
S4	19.0
S5	1.0

Still referring to the configuration of client entities, it is important to note that the QoS attribute considered in each cloudlet was the maximum execution time (*maxExecutionTime*). To obtain the value of QoS constraint field was developed a method in the Client class to ensure that the generation of the maximum execution times are proportional to the size of each cloudlet.

Based on common settings that were explained, it was possible to obtain meaningful simulation results, influencing the response variables considered in the experiments, which will be detailed in the next section,

4.3 Considered Response Variables

For obtaining feedback values in sets of experiments, were selected three response variables that are described below:

- **Response time:** measured in seconds considering the amount of time expended in sending a request to the VM from one provider and its return back to the client.
- **Percentage of Processed Requests**: consider the requests that were processed with *Success* status.
- **Percentage of Unanswered Requests:** consider the requests which could not be met by the cloud broker because not meet the QoS requirement of maximum execution time (*maxExecutionTime*).

5. RESULTS AND DISCUSSION

This section presents information regarding the results of the three sets of executed experimental plans.

5.1 Disclosure of QoS Scenario

The first scenario that will be discussed is the disclosure of QoS. Table 7 summarizes the experimental design created for the scenario in question. Abbreviations found in tables 7, 8 and 9

on the number of VMs field whose acronyms are PRV and PUB, refer, respectively, Private Cloud and Public Cloud.

As can be seen by observing Table 7, experiments with Round Robin Algorithm were compared with experiments using QBroker Services Intermediation Algorithm dealings with or without QoS.

Table 7: Experimental design for disclosure of QoS scenario.

Experiment ID	*Task Scheduling Algorithm*	*Cloud Type*	*Number of VMs*	*Number of Allocated Services*
A	Round Robin	Private	PRV=20	-
B	Round Robin	Hybrid	PRV=20+PUB=10	-
C	Intermediation with QoS	Private	PRV=10	5
D	Intermediation with QoS	Hybrid	PRV=20+PUB=10	5
E	Intermediation without QoS	Private	PRV=20	5
F	Intermediation without QoS	Hybrid	PRV=20+PUB=10	5

In the experiments with intermediation were allocated the five services considered the environment in all VMs in order to make a fair comparison with the Round Robin, which does not have the service selection policy. The results concerning the variable average response time set out in Figure 2.

The obtained results for average response time variable (Figure 2) show that, in private cloud scenarios (experiments A, C and E), QBroker intermediation algorithm proved to be efficient, since in experiment A with Round Robin, the response time was 49.08 seconds while in the experiment E, with intermediation without QoS, obtained better performance with an average time of 45.13 seconds (about 5.8% faster). Still by comparing experiment A with the experiment C, i.e., considering the intermediation with QoS, the performance was even better against the two other experiments, obtaining the value of 18.12 seconds (about 63.08% faster than the experiment A and 59.85% faster than the experiment E).

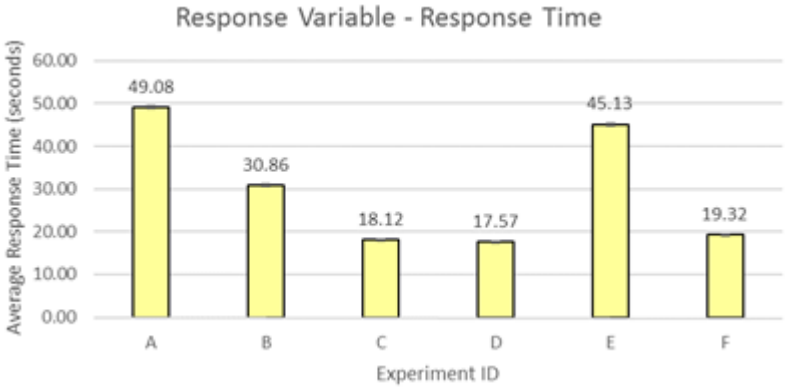


Figure 1: Average response time graph for disclosure of QoS scenario

Although the results with average response time (Figure 2), considering the experiments with hybrid cloud scenarios (experiments B, D and F), the QBroker intermediation algorithm also showed gains in efficiency and performance. The experiment B, which considered using Round Robin had the average response time of 30.86 seconds, while the experiment F considering intermediation without QoS, got 19.32 seconds, which means better performance (about 37.40%

more fast). In experimental examination of experiment D, considering intermediation with QoS, the average value obtained was better than the other two experiments, resulting in 17.57 seconds (about 43.07% faster than Experiment B and 8.9% faster compared to experiment F).

These results corroborate the premise of this paper that the new QBroker Architecture provides performance gains for a major response variables observed by end users of cloud computing systems, that is, the response time for service requests.

It is also possible to visualize differences in how the task scheduling algorithms behave in the simulation scenarios according to the variables of percentage of processed requests and percentage of unanswered requests. According to the results presented by the response variables relating to percentages of processed and missed requests (Figures 3 and 4) stand out from the experiments C and D, which considered scenarios with private and hybrid cloud respectively, using intermediation algorithm with QoS, because it was the only restrictive scenarios on the issue of rejection of requests because of violation of the maximum execution time (*maxExecutionTime*) QoS parameter.

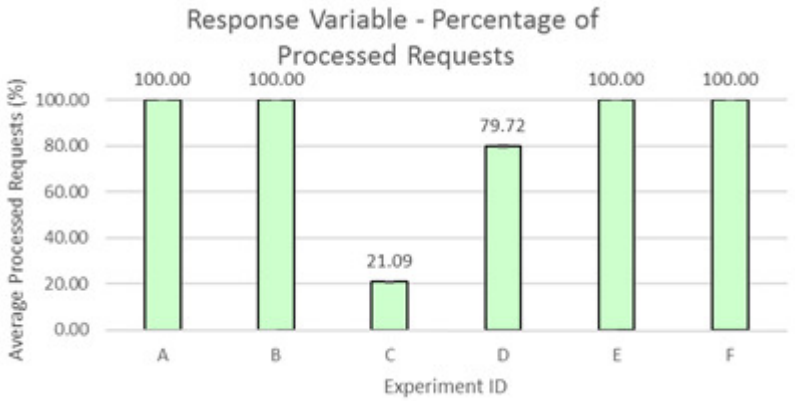


Figure 2: Average percentage of processed requests graph for disclosure of QoS scenario.

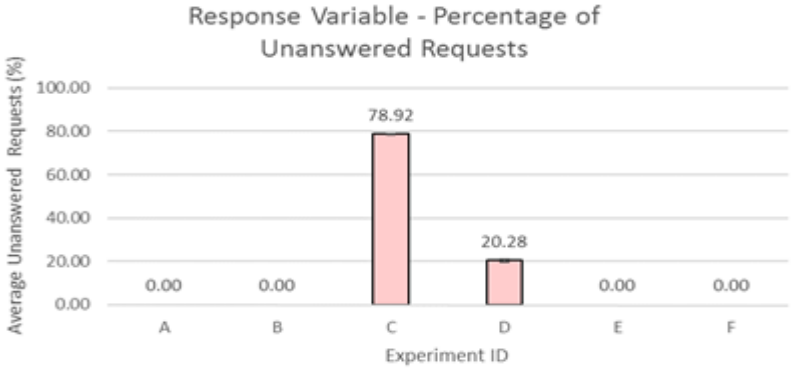


Figure 3: Average percentage of unanswered requests for disclosure of QoS scenario.

In experiment C (according to Figures 3 and 4), as the need arises to keep QoS deadline informed as attribute for each request (cloudlet), QBroker processed 21.09% of the requests sent by clients and rejected others 78.92%. In the experiment D, using the same premise, the QBroker processed 79.72% of the requests and rejected others 20.28%. In other experiments there was no rejection of

requests registered and the response variable percentage of processed requests obtained the constant value of 100%.

5.2 Service Differentiation Scenario

In the second experiment scenario, the objective was to evidence the service differentiation by varying the amount of allocated services in the virtual machines. The characteristic to differentiate services by the use of identifiers approaches QBroker Architecture of cloud brokers compatible with service-oriented architectures (SOA). Table 8 shows the planning of the current scenario of experiments.

According to the experiments plan (Table 8), it is possible to check that the setting of experiments is a variation of the experimental design originally done in disclosure of QoS scenario. The experiments C', D', E' and F' have the same scenario characteristics as, respectively, experiments C, D, E and F, however, the number of services allocated in the machines is different. In the experiments C, D, E and F are allocated 5 services in all instantiated VMs while in experiments C', D', E' and F' the amount of allocated services in the VMs is 2. It should be remembered that in all scenarios where the QBroker used intermediation algorithm, existing services use identifiers numbered from 1 to 5.

Table 8: Design of experiments for service differentiation scenario.

Experiment ID	*Task Scheduling Algorithm*	*Cloud Type*	*Number of VMs*	*Number of Allocated Services*
C	Intermediation with QoS	Private	PRV=20	5
D	Intermediation with QoS	Hybrid	PRV=20+PUB=10	5
E	Intermediation without QoS	Private	PRV=20	5
F	Intermediation without QoS	Hybrid	PRV=20+PUB=10	5
C'	Intermediation with QoS	Private	PRV=20	2
D'	Intermediation with QoS	Hybrid	PRV=20+PUB=10	2
E'	Intermediation without QoS	Private	PRV=20	2
F'	Intermediation without QoS	Hybrid	PRV=20+PUB=10	2

The information of the results of the services differentiation scenario regarding the average response time are shown in Figure 5.

It is possible to see, through the table 8, that the number of services for each VM in this scenario is preset at the beginning of simulation, so there is no occurrence of attempted allocation of new services. In the specific case of the experiments C', D', E' and F', the instantiated services in each VM uses a method of normal distribution for the 5 considered services.

According to Figure 5, for this disclosure of service differentiation scenario, it is possible to note that experiments C and D have the very close results, although not statistically equivalent. Comparing experiments C and C', it can see that C' got an average response time faster with 15.13 seconds. The same situation occurs with the experiments D and D', in which case the experiment D' performed better response time, which value was 14.02 seconds.

Figure 4: Average response time graph for service differentiation scenario.

The justification for these values is precisely the question of the distribution of services, as in the case of experiments C and D, all possible services are instantiated on all scenario's VMs, so that while it may offer more scheduling possibilities for requests, end up making higher the size of average queue, and in this situation, there is a decrease in response time variable and there is also a discard percentage slightly higher.

Experiments with only two services and use of intermediation with QoS (as Figure 5), i.e., C' and D', although become the most restrictive scenario for scheduling options of requests for VMs, generate an average queue time differentiated of a VM to another, because those services whose demand exponential function are larger are not instantiated on all VMs, leading to this situation in particular, a better performance in response time variable.

Also relating to information from experiments in Figure 5, in experiments E, F, E' and F', the results have another positioning. As in experiments E and F has all instantiated services in all VMs of the scenarios and the availability ends thus being wider, and, as already explained, considering that the last activity of intermediation without QoS is schedule the request to the VM that has the service requested instantiated with the lower queue, in such cases, scenarios with more services offer more scheduling opportunities, which makes the values of average times of E and F the experiments, i.e., 45.13 seconds and 19.32 seconds respectively, perform better than the experiments E' and F' having two instantiated services in all scenario's VMs.

The figures 6 and 7 have the performance graphs of percentage of processed and missed requests to the current experiments scenario.

To disclosure a little more the argumentation for the average response time variable, it is possible to observe, as figures 6 and 7, that experiments C and D gave a lower value in terms of processed requests and in turn, higher percentage of unanswered requests (figure 7) as arguments already provided on considerations involving the response time variable.

According to figures 6 and 7, in other experiments (E, F, E' and F') which do not consider the QoS parameter *maxExecutionTime*, always get 100.0% of processed requests, so that there are no unanswered requests.

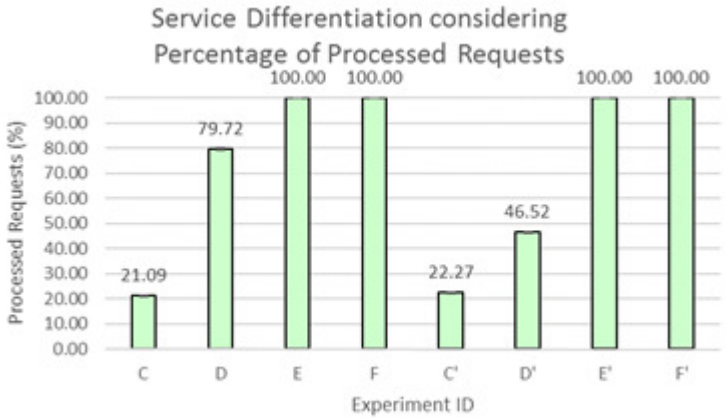


Figure 6: Average percentage of processed requests graph for service differentiation scenario

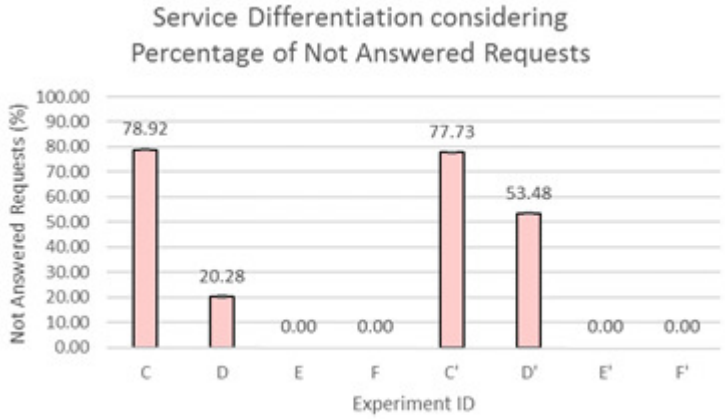


Figure 7: Average percentage of unanswered requests graph for service differentiation scenario

5.3 Dynamic Service Allocation Scenario

In the third experiments scenario the main objective was to highlight the dynamic allocation of services at runtime conducted by QBroker. Table 9 has the experiments planning information created for the experimental scenario explained.

Table 9: Experimental design of dynamic services allocation scenario.

Experiment ID	*Task Scheduling Algorithm*	*Cloud Type*	*Number of VMs*	*Number of Allocated Services*
C'	Intermediation with QoS	Private	PRV=20	2
D'	Intermediation with QoS	Hybrid	PRV=20+PUB=10	2
E'	Intermediation without QoS	Private	PRV=20	2
F'	Intermediation without QoS	Hybrid	PRV=20+PUB=10	2
C''	Intermediation with QoS	Private	PRV=(10 + 10 Stdby)	2
D''	Intermediation with QoS	Hybrid	PRV=(10 + 10 Stdby) + PUB=(10+10 Stdby)	2
E''	Intermediation without QoS	Private	PRV=(10 + 10 Stdby)	2
F''	Intermediation without QoS	Hybrid	PRV=(10 + 10 Stdby) + PUB=(10+10 Stdby)	2

According to the experiments plan (Table 9), it is possible to note the fact that were made a combination of experiments with fixed number of services (C', D', E' and F') with four other experiments that perform dynamic allocation of services. It can also to note that in the private cloud experiments, only 5 VMs have 2 instantiated services while the other 15 VMs remain in standby state. In the scenario with hybrid cloud, private cloud is initialized with the same previous configuration and the public cloud is initialized with all the VMs in standby state.

The results concerning the average response time variable for current scenario are shown in Figure 8.

From graph analysis, it can be observed that the experiments which consider intermediation algorithm with QoS (C', D', C'' and D'') have a difference in performance, is noted that the experiments with dynamic service allocation the response time was longer.

The response time in experiment C', which considered static service allocation and private cloud was 14.66% faster than C'', with dynamic service allocation. A similar situation occurs between experiments with hybrid cloud in the scenarios, i.e., the experiment D', considering static service allocation, obtained response time of 17.67% faster than the experiment D'', which used dynamic service allocation. This result was expected because, at the beginning of the execution of simulation experiments, the experiments C'' and D'' has only 5 VMs available for task scheduling, so the dynamic allocation of services is executed when there is real necessity due to the breach of QoS parameter maximum execution time.

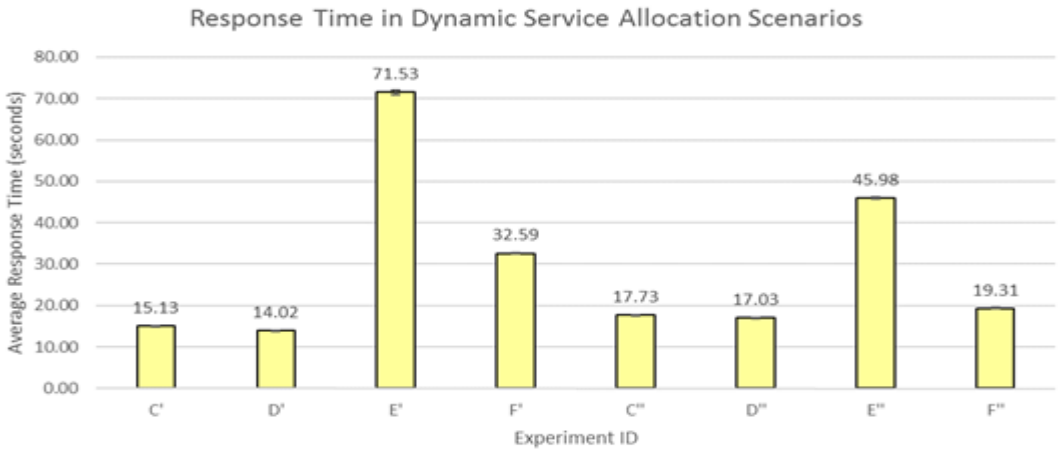


Figure 5: Average response time graph for dynamic service allocation scenario.

Still referring to Figure 8, the experiments that have been configured with intermediation without QoS (E', F', E'' and F'') have a different result because, in this particular case, the experiments with dynamic allocation of services have outstanding difference, with better performance. Experiments E' and F' start with 2 services using a normal distribution. Due to this justified reason, the experiments E 'and F' end up having a lower performance for response time variable because the arrangement of services is predefined at the start of the simulation.

The experiments E'' and F'', have only 5 VMs that are initially initialized with services using the same uniform distribution method. Thus, by effecting on demand service allocation, they have

significant advantage, since the services are allocated on the basis of real need and as services are required in requests.

Figures 9 and 10 present the result of information of variable percentage of processed and missed requests. The experiments in which used the intermediation algorithm without QoS (E', F', E" and F") have a similar behavior, i.e., the variable percentage of processed requests in these experiments was 100.0% and there was no unanswered request.

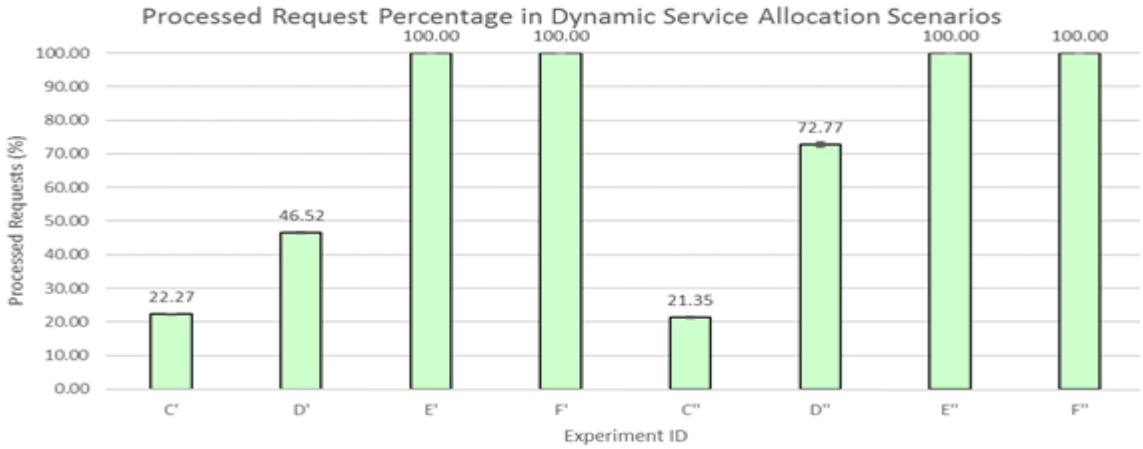


Figure 6: Average percentage of processed requests graph for dynamic service allocation scenario.

Already in the experiments with intermediation with QoS, in the case of experiments considering private cloud C' and C", their values have percentages of processed and unanswered requests next, revealing a similar behavior in the restricted environment of private cloud resources. As for experiments D' and D", which consider hybrid cloud, the experiment D" achieved a better result because, processed a higher percentage of requests, this takes place, as already explained, because of the dynamic service allocation at runtime, what revealed a QBroker feature, that makes the attendance to virtual clients more profitable and causes almost an adaptive effect when you look at the records of the allocation of services performed during the execution of the experiment in CloudSim output report.

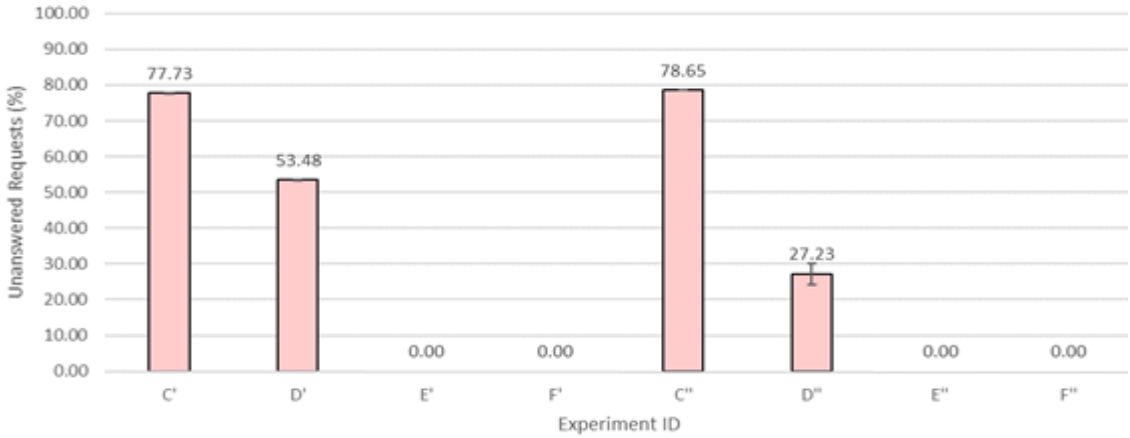


Figure 7: Average Percentage of unanswered requests for dynamic service allocation scenario.

The results of this scenario show that the resource of dynamic service allocation, present in QBroker service intermediation operation mode, is an important differential in the reproduction of real situations of task scheduling to cloud computing systems.

6. CONCLUSION

In this work was presented a cloud broker architecture that provides several features to obtain QoS in hybrid cloud computing environments. To this end, it was implemented, based on the service intermediation definition of NIST cloud computing reference model, a task scheduling policy that considers maximum deadlines for execution of service requests, the allocation control and management of the amount number of services in each VM and the dynamic service allocation on-demand during the execution of simulations. These three key features help the intermediator component of the architecture, that is, help the QBroker to increase the QoS of the services requested on demand, a fact that has been proven through design of experiments performed and presented in three scenarios.

It is worth noting that the Broker is a component that is part of a cloud computing architecture called CloudSim BEQoS (Bursting Energy and Quality of Service), developed by the Laboratory of Distributed Systems and Concurrent Programming (LaSDPC), which is linked to the ICMC University of São Paulo Campus of São Carlos. The results presented in this work highlight the functionality of QBroker operation mode named as service intermediation (with or without QoS). As the information presented from experimental results, it is possible to see the interesting contributions on the simulation of hybrid cloud computing environments through CloudSim coupled to QBroker, MetaCloudletScheduler and other components of BEQoS Architecture.

ACKNOWLEDGEMENTS

The authors acknowledge the financial support from the Brazilian Foundations FAPESP, CNPq and CAPES for the projects under development at the Distributed System and Concurrent Program Group of the Computes Systems Department at ICMC - USP.

REFERENCES

[1] S. Moore, “Gartner Says Worldwide Cloud Infrastructure-as-a-Service Spending to Grow 32.8 Percent in 2015,” 2015. [Online]. Available: http://www.gartner.com/newsroom/id/3055225. [Accessed: 07-Oct-2015].

[2] M. Salama and A. Shawish, “A QoS-Oriented Inter-cloud Federation Framework,” 2014 IEEE 38th Annu. Comput. Softw. Appl. Conf., no. Cc, pp. 642–643, Jul. 2014.

[3] S. Sotiriadis, N. Bessis, and N. Antonopoulos, “Towards Inter-cloud Schedulers: A Survey of Meta-scheduling Approaches,” 2011 Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput., pp. 59–66, Oct. 2011.

[4] M. Aazam and E. N. Huh, “Inter-cloud Media Storage and Media Cloud Architecture for Inter-cloud Communication,” 2014 IEEE 7th Int. Conf. Cloud Comput., pp. 982–985, Jun. 2014.

[5] M. H. Sqalli, M. Al-saeedi, F. Binbeshr, and M. Siddiqui, “UCloud: A simulated Hybrid Cloud for a university environment,” 2012 IEEE 1st Int. Conf. Cloud Netw., pp. 170–172, Nov. 2012.

[6] V. Bagwaiya and S. K. Raghuwanshi, "Hybrid approach using throttled and ESCE load balancing algorithms in cloud computing," 2014 Int. Conf. Green Comput. Commun. Electr. Eng., pp. 1–6, Mar. 2014.

[7] M. Aazam and E.-N. Huh, "Broker as a Service (BaaS) Pricing and Resource Estimation Model," 2014 IEEE 6th Int. Conf. Cloud Comput. Technol. Sci., pp. 463–468, Dec. 2014.

[8] M. Nir, A. Matrawy, and M. St-Hilaire, "An energy optimizing scheduler for mobile cloud computing environments," 2014 IEEE Conf. Comput. Commun. Work. (INFOCOM WKSHPS), pp. 404–409, Apr. 2014.

[9] R. S. Moorthy, T. S. Somasundaram, and K. Govindarajan, "Failure-aware resource provisioning mechanism in cloud infrastructure," 2014 IEEE Glob. Humanit. Technol. Conf. - South Asia Satell., pp. 255–260, Sep. 2014.

[10] E. Hwang and K. H. Kim, "Minimizing Cost of Virtual Machines for Deadline-Constrained MapReduce Applications in the Cloud," 2012 ACM/IEEE 13th Int. Conf. Grid Comput., pp. 130–138, Sep. 2012.

[11] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, "NIST Cloud Computing Reference Architecture Recommendations of the National Institute of Standards and."

[12] R. N. Calheiros, R. Ranjan, A. Beloglazov, and A. F. De Rose, "CloudSim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," no. August 2010, pp. 23–50, 2011.

[13] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications," 2010 24th IEEE Int. Conf. Adv. Inf. Netw. Appl., pp. 446–452, 2010.

[14] X. Li, X. Jiang, P. Huang, and K. Ye, "DARTCSIM : AN ENHANCED USER-FRIENDLY CLOUD SIMULATION SYSTEM BASED ON CLOUDSIM WITH," 1857.

[15] S. Long and Y. Zhao, "A Toolkit for Modeling and Simulating Cloud Data Storage: An Extension to CloudSim," 2012 Int. Conf. Control Eng. Commun. Technol., pp. 597–600, Dec. 2012.

[16] S.-M. Jung, N.-U. Kim, and T.-M. Chung, "Applying Scheduling Algorithms with QoS in the Cloud Computing," 2013 Int. Conf. Inf. Sci. Appl., pp. 1–2, Jun. 2013.

[17] R. Achar and P. S. Thilagam, "A broker based approach for cloud provider selection," 2014 Int. Conf. Adv. Comput. Commun. Informatics, pp. 1252–1257, Sep. 2014.

[18] N. G. and J. G., "A Multi-agent Brokering Approach and Jumper Firefly Algorithm for Job Scheduling in Cloud Computing," 2014 Int. Conf. Intell. Comput. Appl., pp. 52–58, Mar. 2014.

[19] G. Raj and S. Setia, "Effective Cost Mechanism for Cloudlet Retransmission and Prioritized VM Scheduling Mechanism over Broker Virtual Machine Communication Framework," vol. 2, no. 3, pp. 41–50, 2012.

[20] M. Aazam and S. Korea, "Advance Resource Reservation and QoS Based Refunding in Cloud Federation," pp. 139–143, 2014.

[21] S. Sotiriadis, N. Bessis, and N. Antonopoulos, "Towards Inter-cloud Simulation Performance Analysis: Exploring Service-Oriented Benchmarks of Clouds in SimIC," 2013 27th Int. Conf. Adv. Inf. Netw. Appl. Work., pp. 765–771, Mar. 2013.

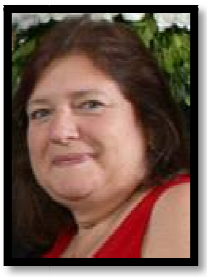
[22] R. B. Bohn, J. Messina, F. Liu, J. Tong, and J. Mao, "NIST cloud computing reference architecture," Proc. - 2011 IEEE World Congr. Serv. Serv. 2011, pp. 594–596, 2011.

AUTHORS

Mário Henrique de Souza Pardo received bachelor's degree in Systems Analysis from the University of the Sacred Heart (USC), Bauru / SP, Brazil, in 2001. concluded a master's degree in Computer Science from UNIVEM, Marilia / SP, Brazil in 2006. He is currently a PhD student in research line of Distributed Systems and Concurrent Programming from the University of São Paulo (USP), São Carlos / SP, Brazil, supervised by Professor Dr. Regina H. C. Santana. His current research interest is focused on the study of Cloud Computing, specifically for task scheduling with QoS for complex cloud computing environments. Also deals with computer systems performance evaluation and simulation of cloud computing through discrete event simulation.

Adriana Molina Centurion received the BS degree in Computer Science from Marilia University (UNIMAR) in 1995; the MS degree (in 1998) and PhD degree (in 2015) in Computer Science and Computational Mathematics from the University of Sao Paulo (USP). She has experience in Computer Science with emphasis in Distributed Computer Systems and Performance Evaluation and 10 years of experience in management of projects and services in the area of Information Technology. She currently is professor at Institute of Education, Science and Technology of Sao Paulo (IFSP). Her research interests include Performance Evaluation, Distributed Systems, Service Oriented Architecture, Cloud Computing, Simulation, Workload Modeling and Burstiness Phenomenon.

Paulo Sérgio Franco Eustáquio graduated Bachelor in Computer Science from the Pontifical Catholic University of Minas Gerais (PUC), Pocos de Caldas / MG, Brazil, holds a Master degree in Computer Science from the University of São Paulo (USP), São Carlos / SP, Brazil. It is a PhD student at Institute of Mathematics and Computer Sciences (ICMC) at USP of São Carlos / SP, Brazil, supervised by Professor Dr. Sarita M. Bruschi. His research interests are: Cluster, Grid and Cloud Computing, Web Servers Architecture, intake systems and task scheduling for distributed computing systems, client-side QoS and provider-side QoS considering Green Computing and efficient energy consumption for Computing Cloud environments.



Regina Helena Carlucci Santana graduated in Electrical Electronic Engineering from the School of Engineering of São Carlos (1980), Master degree in Computer Science from the Institute of Mathematical Sciences of São Carlos (1985) and PhD in Electronics and Computing - University of Southampton (1989). She is currently Associate Professor at the University of São Paulo. She has expertise in Computer Science, with emphasis on Performance Evaluation, acting on the following topics: performance measurement, simulation, distributed simulation, tasks and process scheduling and parallel computing. Other topic of her interest in research is Distributed Computational Systems Architecture involving Cluster, Grid, Cloud Computing and others.

Sarita Mazzini Bruschi graduated in Bachelor of Computer Science from Paulista State University "Julio de Mesquita Filho" (1994), Master degree in Computer Science from the University of São Paulo (1997) and PhD in Computer Science from the University of São Paulo (2002). She is currently Doctor Professor MS3 RDIDP at the University of São Paulo. She has expertise in Computer Science, with emphasis on Performance Evaluation, acting on the following topics: performance evaluation, simulation, tasks and process scheduling in Cloud Computing, Green Computing, Educational Environments and Operating System.

Marcos José Santana graduated in Electrical Electronic Engineering from the School of Engineering of São Carlos (1980), Master degree in Computer Science from the Institute of Mathematical Sciences of São Carlos (1985) and PhD in Electronics and Computing - University of Southampton (1989). He is currently Associate Professor at the University of São Paulo. He has expertise in Computer Science, with emphasis on performance evaluation, acting on the following topics: performance evaluation, web services, cluster computing, grid computing, cloud computing, process scheduling, parallel computing, simulation and load balancing for distributed systems. Coordinator of Computer Engineering at ICMC since 2002 to 2011 and Chief of the Computer Systems Department since 2010.