Future Generation Computer Systems 29 (2013) 1777–1785

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

# Analyzing the impact of elasticity on the profit of cloud computing providers

Rostand Costa [a,b,*], Francisco Brasileiro [a], Guido Lemos [b], Dênio Sousa [b]

[a] Federal University of Campina Grande, Systems and Computing Department, Distributed Systems Lab, Av. Aprígio Veloso, 882 - Bloco CO, Bodocongó Campina Grande, Paraíba, Brazil

[b] Federal University of Paraíba, Informatics Department, Digital Video Applications Lab, Campus I - Cidade Universitária, João Pessoa, Paraíba, Brazil

ARTICLE INFO

*Article history:*
Received 1 September 2012
Received in revised form
22 December 2012
Accepted 31 December 2012
Available online 11 January 2013

*Keywords:*
Cloud computing
Elasticity
Availability
Capacity planning
BoT

ABSTRACT

Bag-of-tasks (BoT) is an important class of scientific applications. These applications are typically comprised of a very large number of tasks that can be executed in parallel in an independent way. Due to its cost associativity property, a public cloud computing environment is, theoretically, the ideal platform to execute BoT applications, since it could allow them to be executed as fast as possible, yet without implying any extra costs for the rapid turnaround achieved. Unfortunately, current public cloud computing providers impose strict limits on the amount of resources that a single user can simultaneously acquire, substantially increasing the response time of large BoT applications. In this paper we analyze the reasons why traditional providers need to impose such a limit. We show that increases in the limit imposed have a severe impact on the profit achieved by providers. This leads to the conclusion that new approaches to deploy cloud computing services are required to properly serve BoT applications.



## 1. Introduction

Cloud computing is an evolving paradigm that allows the provision of information technology (IT) in the form of a service that can be purchased on-line and on-demand by clients [1]. The resources that are bought from a public cloud computing provider can be rapidly provisioned to customers, and later released by them, offering a theoretically unlimited elasticity in the way service consumption increases and decreases with time. Moreover, clients are charged using a pricing model where they pay only for the amount of service that they actually consume.

More formally, let the service requested by a client from a cloud computing provider over time be defined by a sequence of tuples $s_1, s_2, \ldots$, with $s_i = \langle \rho_i, \sigma_i, \delta_i \rangle$, where $\rho_i$ is the amount of resources that was requested in service request $s_i$, $\sigma_i$ is the time when the client wants to start using the resources, and $\delta_i$ is the duration of the time interval for which the $\rho_i$ resources were requested. The elasticity property defines that no restrictions are imposed on $\rho_i - \rho_{i-1}$ for any $i, i > 1$, while the pay-as-you-go property defines that the fee charged to the client for any request $s_i$ is a function of $\rho_i \cdot \delta_i$. The conjunction of the elasticity and pay-as-you-go properties leads to a third property, named cost associativity [2], which defines that clients are charged the same fee for any two requests $s_i$ and $s_j$, whenever $\rho_i \cdot \delta_i = \rho_j \cdot \delta_j$.

The cost associativity property of the cloud computing model is particularly interesting for an important class of scientific applications which has become increasingly popular nowadays — the so-called "embarrassingly" parallel" or simply "bag-of-tasks" (BoT) applications [3]. They are comprised of a very large number of tasks that can be independently executed. Thus, they can be trivially parallelized using a simple workqueue scheduler that keeps dispatching tasks to be executed in any available processing resource until all tasks are executed. It is easy to see that, considering homogeneous resources, such as those available in cloud infrastructures, the more resources that are available to run BoT applications, the faster they will terminate.

Due to the cost associativity property, clients who need to run a BoT application could fully explore the elasticity property of cloud computing providers, and request as many computing resources as needed in order to maximize the level of parallelization of the execution of their applications, allowing them to run their applications in the shortest possible time without any additional cost, when compared to other scheduling options.

Obviously, the elasticity of a cloud computing provider is limited by its capacity. However, compared with their speculated capacity, public cloud providers currently in operation impose a very low limit on the amount of resources that each client can simultaneously request; for instance, currently, one of the major

* Corresponding author at: Federal University of Campina Grande, Systems and Computing Department, Distributed Systems Lab, Av. Aprígio Veloso, 882 - Bloco CO, Bodocongó, Campina Grande, Paraíba, Brazil. Tel.: +55 83 99710037; fax: +55 83 3216 7093.
*E-mail addresses:* rostand@lavid.ufpb.br, rostand.costa@gmail.com (R. Costa).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2012.12.021

players in the market limits this to 20 resources in most cases [4]. Although the limits currently imposed by cloud providers do not prevent most clients from seeing the service provided as an infinite source of resources, this is not the case for most clients who need to execute BoT applications, which may require the instantiation of thousands of computing resources in a single request [5]. In this paper, we analyze the reasons why public cloud providers impose limits that restrict the usefulness of their services for the execution of large BoT applications. Our results show that a high level of service availability can only be provided with acceptable cost, if providers somehow regulate their clients' demand. This is currently achieved by imposing the above-mentioned limits, and the limits currently used are unlikely to change in the future. Our results also shed light on the impact that different demand patterns have on the profitability of providers. They indicate that users with highly elastic and ephemeral workloads can only be appropriately served if a new model for the provisioning of public IaaS is developed.

Our approach is based on the use of simulation. We start by defining a simplified model for IaaS providers (Section 2), and an appropriate synthetic workload generator for the proposed model (Section 3). Then, we describe the experiments that we conducted (Section 4). To instantiate the simulation model adequately, we performed an experimental design to identify the random variables of the model that had a major impact on the response variable. In the absence of data from real IaaS providers, we performed a parameter sweep over the main factors identified. The results and analysis of the simulations performed are presented in Section 5. Related work is reviewed in Section 6. Finally, Section 7 gives our concluding remarks.

## 2. A simplified model for an IaaS provider

Among the many QoS properties that a cloud computing provider must consider, in this paper we will focus on *service availability*, *i.e.*, the probability that a client requesting a service has its request fully met.[1] This property should not be confused with *resource availability*, which is represented by the probability that the virtual resource provided will not fail while the client is using it. In other words, the service availability is affected when a client requests a new virtual machine and the provider is unable to instantiate the requested resource, while the resource availability is affected when a virtual machine that is being used by a client fails. Note that the SLA established between client and provider is normally focused on virtual machine availability. Nevertheless, service availability is an important metric for the IaaS provider, since a client whose demand is denied will probably seek another provider to fulfill his request and may never return to ask for service from a provider that has a reduced service availability.

Since the resources of a provider are allocated to each client for a minimum time interval, for example, 1 h, we assume that the time is discretized into intervals of fixed size (slots), and we model an IaaS provider $P$ in a given observation period $\Delta T$ as a tuple:

$$P = \langle K, L, U, D, A, C_i, C_u, V, E \rangle,$$

where:

- $K$ is the amount of resources available on the provider, *i.e.* its capacity;
- $L$ is the maximum amount of resources that can be allocated to a single client in each time slot;
- $U$ is the set of users registered at the provider;
- $D$ is the distribution of demand for these users;
- $A$ is the strategy used by the provider to allocate resources;
- $C_i$ is the cost incurred by the provider to provide each individual resource for a slot duration, which is obtained by prorating the amortization of the total cost of ownership of the available resources over all time slots that comprise the amortization period[2];
- $C_u$ is the additional cost incurred by the provider whenever a resource is effectively used in a time slot, spent only when each individual resource is being effectively used, based on the concept of *utilization cost* proposed by Li et al. [6] and considering that some level of energy efficiency is practiced [7];
- $V$ is the value that is charged to users for the actual use of a resource during one time slot or fraction;
- $E$ is the cost to the provider for each violation committed on the service availability; it can be tangible (*e.g.* contractual compensation paid to the client) or intangible (*e.g.* damage to the image of the provider). In this paper we consider only the tangible aspect of the charges for violations.

In the next section we present in detail how the demand $D$ of users $U$ of a provider $P$ is described. For now, we just assume that $d(u,t), 0 \leq d(u,t) \leq L, \forall u \in U, 1 \leq t \leq \Delta T$, is the amount of resources demanded by user $u$ in a slot $t$. Depending on the demand pattern ($D$) of users, the allocation strategy ($A$), the limit of allocation per client ($L$), and the capacity ($K$) of the provider, each user $u$ that requests $d(u,t)$ will receive an associated allocation of resources that is expressed by $a(u,t), 0 \leq a(u,t) \leq d(u,t)$. When $a(u,t) < d(u,t)$ we have a violation on the service availability of the provider. Thus, the total number of violations in a slot $t$ is given by

$$v(t) = \sum_{u \in U} 1 - \left\lfloor \frac{a(u,t)}{d(u,t)} \right\rfloor.$$

Let $\alpha(t)$ be the allocated capacity of the provider in the time slot $t$. $\alpha(t) = \sum_{u \in U} a(u,t)$. One way to gauge the efficiency of the provider is to measure its profit in the period of time considered, represented in our model by

$$\Lambda = \sum_{t=1}^{\Delta T} [(V - C_u) \cdot \alpha(t) - v(t) \cdot E] - K \cdot C_i \cdot \Delta T. \quad (1)$$

## 3. Generation of synthetic workloads for an IaaS provider

Due to the unavailability of traces from actual executions or even characterizations of the workload of IaaS providers, we had to create a generator of synthetic workloads to define the demand imposed on the provider in our simulations.

The total utilization of the system in each time slot $t$, represented by $\alpha(t)$, results from the usage profile of each individual user. In principle, all users can, on demand and without additional charge, take advantage of the inherent elasticity of the service and, in any time slot, use any number of resources, from zero to the limit $L$ imposed by the provider.

Considering the behavior of the system in the time interval of duration $\Delta T$, some categories of users will emerge. An initial classification of users is related to the level of demand during the period considered. Active users are those who have made a demand for system resources in a given interval, *i.e.*, $d(u,t) > 0$ for some value of $t$, $1 \leq t \leq \Delta T$. The other users are said to be inactives.

[1] The focus on availability was a simplification to make the model tractable, other dimensions can be addressed following a similar approach.

[2] Although the described costs have a linear behavior, and are a simplification of actual costs, which have a more complex profile, this simplification provides a good approximation, and meets the needs of our model.

Let $U_a$ be the set of active users:

$$U_a = \{u | u \in U \wedge \exists t, 1 \leq t \leq \Delta T, d(u, t) > 0\}.$$

The behavior of each category of active user is described through the use of distributions traditionally associated in the literature with classes of users and usage sessions [8–10]. For the creation of the workload we applied the hierarchical generation approach, through a user-based modeling [8]. This technique is based on the separation of users' behavior into three levels: population profile, session duration, and activity within the session, covering aspects such as locality of sampling and auto-similarity [8]. Thus, it is possible to include long term stays and absences (long tail [10]) and also regular behaviors in the generated workload. The modeled system is closed, with a known and finite number of users ($|U_a|$).

The population of active users can be divided into two groups, considering the regularity of their demand. Active regular users are those with uninterrupted use. The set of regular users is described as follows:

$$U_r = \{u | u \in U_a \wedge \forall t, 1 \leq t \leq \Delta T, d(u, t) > 0\}.$$

The set of eventual users ($U_e$) contains the active users that are not regular:

$$U_e = U_a - U_r.$$

We assume that regular users have only one session, whose duration comprises at least all the interval $\Delta T$ considered. On the other hand, the session time of eventual users is governed by the following random variables:

- $\tilde{o}$: duration (in slots) of each session of an eventual user, following a discrete uniform distribution with lower bound $l_o$ and upper bound $u_o$ [10]; and
- $\tilde{i}$: interval between sessions, following a discretized Pareto distribution with parameters $k_i$ and $s_i$ [10,8].

In each session, the user may be "in activity" or "in standby" (*think time*), which indicate, respectively, whether the user is effectively using resources, or not. The behavior of each user in activity can be defined by the amount of resources it uses, the duration of use, and also by the time it is not using system resources. Thus, each activity can be characterized by the tuple

$$\mathcal{A} = \langle r, n, e \rangle$$

where $r$ and $n$ represent the amount of resources required per time slot, and the duration of the activity in number of slots, respectively, and $e$ represents the waiting time until the next slot when the user will be in activity. A change in the amount of resources, although possible, implies the start of another activity.

Next, we describe the utilization profiles of each user category of our population.

The usage profile of regular users is modeled in a simplified form. Regular users have uninterrupted activities lasting one time slot. In each session, the number of demanded resources is based on the random variable $\tilde{m}$ with normal distribution, mean $\tau$ and variance $\sigma$, where $\tau$ is the *average ticket* of the regular users, given by

$$\tau = \frac{\sum_{t=1}^{\Delta t} \sum_{u \in U_r} a(u, t)}{\Delta T \cdot |U_r|}.$$

The activity profile of regular users is defined as

$$\mathcal{A}_{regular} = \langle \tilde{m} \sim N(\tau, \sigma), 1, 0 \rangle.$$

This approach models the possible increases or decreases in the individual requests. However, statistical multiplexing of the demand of regular users leads to negligible variations in the total use of regular users in each time slot. More abrupt changes in the behavior of regular users that affect this relation will be addressed later.

The "in activity" behavior of eventual users, in turn, is based on three random variables:

- $\tilde{s}$: amount of resources allocated to each activity, following a discrete uniform distribution between 1 and $L$ [10,8];
- $\tilde{d}$: duration (in slots) of each activity, following a two stage hyper-exponential distribution with rates $\lambda_{d_1}$ and $\lambda_{d_2}$ and associated probabilities $\varphi_{d_1}$ and $\varphi_{d_2}$ [10,8]; and
- $\tilde{t}$: interval (in slots) between activities (*think time*), following a two stage hyper-exponential distribution with rates $\lambda_{t_1}$ and $\lambda_{t_2}$ and associated probabilities $\varphi_{t_1}$ and $\varphi_{t_2}$ [10,8].

The activity profile of eventual users is defined as

$$\mathcal{A}_{eventual} = \langle \tilde{s} \sim U(1, L), \tilde{d} \sim H_2(\lambda_{d_1}, \varphi_{d_1}, \lambda_{d_2}, \varphi_{d_2}), \tilde{t} \sim H_2(\lambda_{t_1}, \varphi_{t_1}, \lambda_{t_2}, \varphi_{t_2}) \rangle.$$

Two particular profiles of eventual users were also modeled to cover the following situations: (a) regular users presenting an unusual demand for resources driven by *flurries* or *flashmobs* in their services, with varying intensity [11], and (b) eventual users with intensive and time sensitive utilization (*e.g.* BoT application users) [5] that always consume all available resources. These profiles are defined as follows:

$$\mathcal{A}_{flashmob} = \langle \tilde{s} \sim U(\tau + 1, L), \tilde{d} \sim H_2(\lambda_{d_1}, \varphi_{d_1}, \lambda_{d_2}, \varphi_{d_2}), \tilde{t} \sim H_2(\lambda_{t_1}, \varphi_{t_1}, \lambda_{t_2}, \varphi_{t_2}) \rangle$$

$$\mathcal{A}_{BoT} = \langle L, \tilde{d} \sim H_2(\lambda_{d_1}, \varphi_{d_1}, \lambda_{d_2}, \varphi_{d_2}), \tilde{t} \sim H_2(\lambda_{t_1}, \varphi_{t_1}, \lambda_{t_2}, \varphi_{t_2}) \rangle.$$

## 4. Description of experiments

The main objective of the simulation experiments is to observe (i) the minimum capacity required to meet all client requests for a given level of service availability, (ii) the idleness of the system in each scenario, and (iii) the provider's operational income with different limit values.

### 4.1. Implementation of the simulation model

To be resolved by simulation, the proposed model was implemented using the Möbius tool [12]. This platform allows the realization of discrete event simulation and numerical or analytical resolution of system models that can be described in a variety of formalisms.

One of the supported formalisms allows the composition of models in a tree structure, in which each leaf of the tree can be either an atomic model, described in one of the other supported formalisms, or another composed model. Each node of the tree that is not a leaf is classified as either a *Join* node or a *Replicate* node. A *Join* node is used to compose two or more submodels through the sharing of state, while a *Replicate* node is used to construct a model consisting of a number of identical copies of its child submodel.

To represent the active users of an IaaS provider we used this formalism for the creation of the composed model *ActiveUsers* (Fig. 1). This model contains four atomic submodels modeled using the *Stochastic Activity Network* (SAN) formalism, representing the four user profiles described: *Regular*, *Eventual*, *Flashmob*, and *BoT*. The use of *Replicate* nodes allows the creation of the desired number of instances of each user profile defined, and also state sharing between instances of the same type of submodel. The *Join* node, in turn, allows state sharing between instances of submodels of different types. Thus, the synthetic workload was built through the autonomous and combined activity of an instance of the submodel Regular, whose demand in each slot ($\tau$) is multiplied by

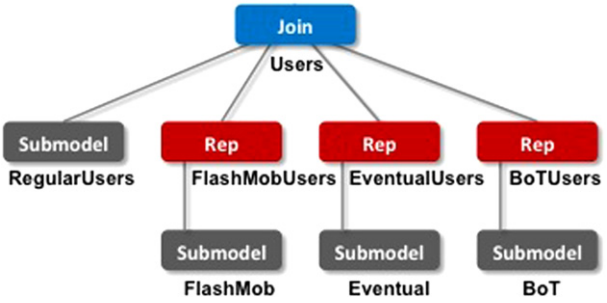



**Fig. 1.** The composed model of active users of an IaaS provider.

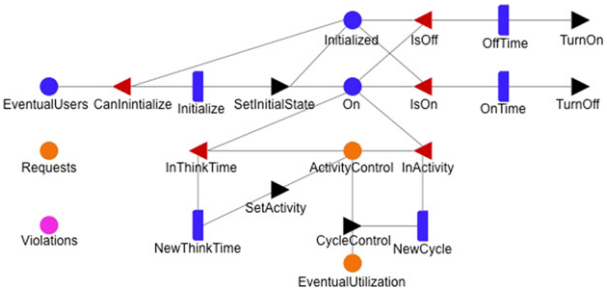



**Fig. 2.** The SAN model for a user with the *Eventual* profile.

$|U_r|$, and a total of $|U_e|$ instances of submodels *Eventual*, *Flashmob*, and *BoT*, according to the distribution of each type of profile.

For example, the submodel *Eventual*, shown in Fig. 2, represents the behavior of a user with the *Eventual* profile. As described in Section 3, a user consumes cloud resources through a series of stages. These stages were modeled in an SAN submodel as *places* and *extended places* (blue and orange circles, respectively). Each *place* maintains a counter (represented by *tokens*) that expresses the user's current state at that stage. The *input gates* (red triangles) are used to inspect these states, and enable (or not) the transition of the system through the implementation of timed activities (vertical bars). Each timed activity has a duration that impacts on the dynamics of the modeled system, and also a distribution (and associated parameters) that regulates its behavior. The *output gates* (black triangles) are executed after the duration of a timed activity has elapsed, and allow change of the system state by changing the number of *tokens* in the *places*. The arches (black lines) signal the flow of transitions between stages.

Each user of the *Eventual* profile is initialized randomly in one of the possible stages (*OnSession* or *OffSession*), which is controlled by the place *On*. After initialization, the activities *OffTime* and *OnTime* begin to regulate user sessions alternating usage and periods of inactivity, controlled by the random variables $\tilde{o}$ and $\tilde{i}$, respectively. A new activity for the user session is assigned (as described in the *Eventual* profile, and using the random variables $\tilde{d}$ and $\tilde{s}$) through the output gate *SetActivity* after a waiting period (*think time*) has elapsed. The expected duration of each waiting period is managed by the timed activity *NewThinkTime* (random variable $\tilde{t}$). The place *ActivityControl*, in turn, controls the duration of each individual activity, slot by slot, through the timed activity *NewCycle*.

The other submodels — *Regular* (Fig. 3), *Flashmob* (Fig. 4) and *BoT* (Fig. 5) — have similar modeling and due to space limitations will not be discussed here. However, the complete Möbius model used in this work can be found at http://www.lsd.ufcg.edu.br/~rostand/IaaSModelH.zip.

The dynamics of the configured population of users drives the allocation of resources of the IaaS provider. We assume a very simple first-come-first-served allocation algorithm that always allocates the amount of resources that are demanded by each user

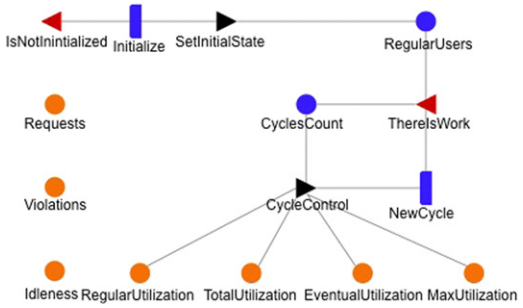



**Fig. 3.** The SAN model for a user with the *Regular* profile.

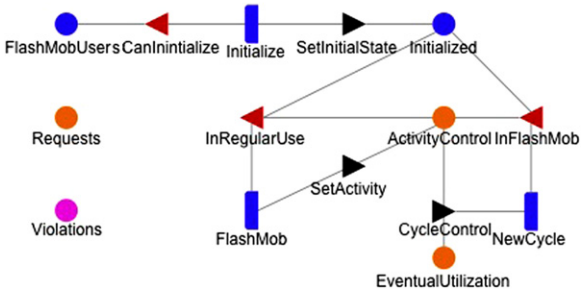



**Fig. 4.** The SAN model for a user with the *Flashmob* profile.

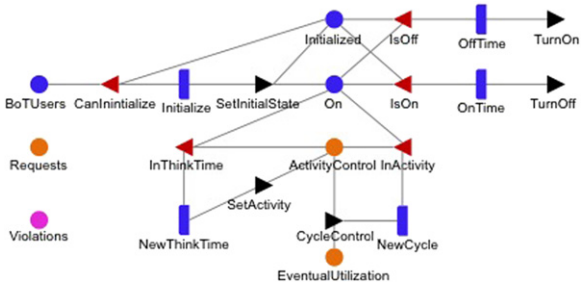



**Fig. 5.** The SAN model for a user with the *BoT* profile.

request, as long as there is enough free capacity available. The response variables accounted for by the simulation model are the allocated capacity at each slot ($\alpha(t)$), and the number of violations per slot ($v(t)$).

Simulation experiments can be executed using the Möbius simulator by simply providing appropriate configurations for the several system parameters, including those required by the workload modeling just presented.

### 4.2. System parameters

To set the system parameters we have used two strategies, namely design of experiment (DoE), and parameter sweep. The part of the parameters related to the generation of the synthetic workload and associated with the distributions described in Section 3 has been addressed through a $2^k$ factorial DoE [10]. Through the DoE it was possible to analyze the effects of the random variables $\tilde{o}$ (session duration), $\tilde{i}$ (interval between sessions), $\tilde{s}$ (activity duration), $\tilde{t}$ (*think time*) and also the value of $L$ on one of the response variables of the system: the maximum utilization of the system in a given interval ($\max(\alpha(t))\forall t, 1 \leq t \leq \Delta T$). The levels assigned to the DoE are shown in Table 1.[3]

[3] $\lambda_{d_2}$ and $\lambda_{t_2}$ are set to be half of $\lambda_{d_1}$ and $\lambda_{t_1}$, respectively.

**Table 1**
Factors, levels and effects of the DoE $2^k$ factorial ($k = 5$).

| Factor | Low | High | Effect estimated | Sum of squares | % Contribution |
|---|---|---|---|---|---|
| A: upper bound $u_o$ (in slots) for $\tilde{o}$ | 36 | 108 | 2744.56 | 60,260,986 | 4.25 |
| B: lower bound $k_i$ (in slots) for $\tilde{i}$ | 120 | 360 | −4365.29 | 152,445,880 | 10.75 |
| C: rate $\lambda_{d_1}$ (in slots) for $\tilde{d}$ | 0.0625 | 0.1875 | 5464.65 | 238,899,197 | 16.85 |
| D: rate $\lambda_{t_1}$ (in slots) for $\tilde{t}$ | 0.125 | 0.375 | −1695.91 | 23,008,954 | 1.62 |
| **E: $L$ (in amount of resources)** | 20 | 100 | 9273.94 | 688,047,334 | 48.54 |

We conducted several repetitions of the 32 experiments to obtain means with a confidence level of 95%, and errors not larger than 5%. The contribution of each factor is shown in Table 1, highlighting the dominant factor, $L$, which had a contribution of 48.54%. The significant interactions (above 0.5%) were AE (1.84%), BE (4.65%) and CE (7.29%), all of them related to $L$. As a result of analyzing the effects through ANOVA [10], an $F$-value of 65.6448 implies that the model is significant. The adjusted $R^2$indicates that the model explains 94.34% of the variation observed, and the $R^2$ of the prediction is inside 0.20 of the adjusted $R^2$, representing a good predictive capacity for the model.[4]

To carry out the simulation, the values of the four parameters with lower impact were adjusted to the averages of the values for the "Low" and "High" levels used in the DoE. For the parameters *Percentage of Eventual Activity*, *Percentage of Users with BoT Profile*, *Amount of Active Users*, and $L$, a parameter sweep strategy was applied. Table 2 shows how the system was configured for the experiments.

## 5. Results and analysis

In the first experiment, the objective was to observe how the provider's profitability was impacted by increasing the limit imposed by the provider ($L$). In this experiment we consider a situation where the service availability of the provider must be kept at 100%. To this end, the capacity ($K$) was set in such a way that for any slot $t$, it is always possible to allocate resources to a user $u$ that has a positive demand ($d(u, t) > 0$) and, therefore,

$$a(u, t) = d(u, t), \quad \forall u \in U \wedge 1 \leq t \leq \Delta t.$$

Thus, considering Eq. (1), as the penalties will be void and the net income of running the same workload is constant, the provider's profit is affected only by the capacity that needs to be maintained to meet the desired level of availability. To ensure a similar system load, the number of active users was kept constant for this experiment at 5000 users. However, we investigated several values for *Percentage of Eventual Activity* and *Percentage of Users with BoT Profile* to simulate different scenarios of regular and eventual activity, and different shares of users with *BoT* profiles. This class of users is especially interesting for this analysis because they have high volume and time sensitive workloads, and always consume the maximum allowed allocation of resources ($L$).

To cover all combinations of input parameters, 288 simulations were executed. Each scenario was repeated until the required confidence levels were achieved (95%, and errors not larger than 5%). The response of interest was the maximum allocated capacity ($\max(\alpha(t))$) observed at all time slots for each setting of the simulated system, since this value defines the minimum capacity necessary to ensure 100% service availability during the simulation period. Some of the results obtained are shown graphically in Fig. 6.

As can be seen, even assuming a population of constant size, the minimum capacity required increases as the limit is incremented.

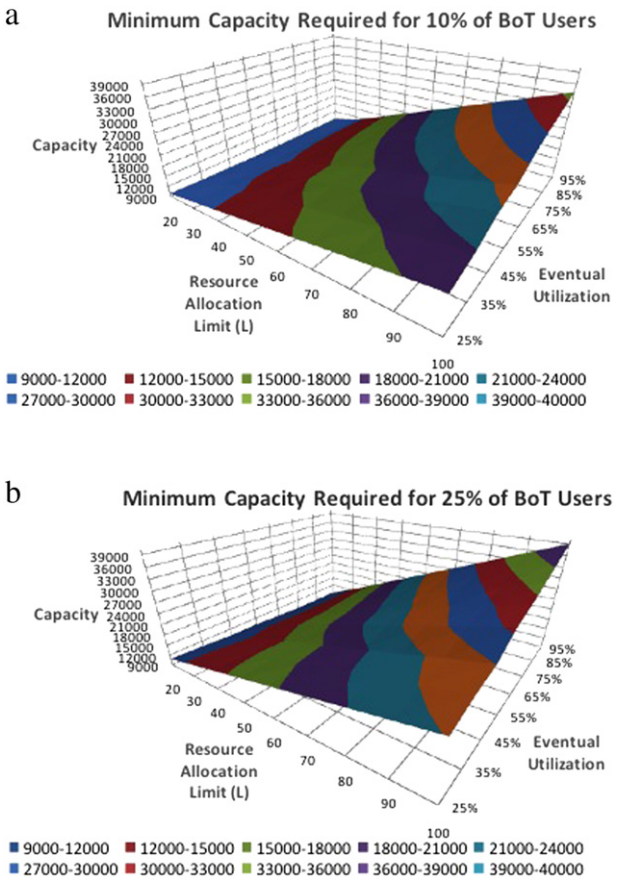



**Fig. 6.** Minimum capacity required to meet 100% service availability when varying the limit ($L$) and *Eventual* activity for two scenarios of users having a *BoT* profile (10% and 25%).

This demand for more capacity is already present even in settings where regular activity is dominant with 25% of eventual users, of which only 10% have *BoT* profiles (Fig. 6(a)). When eventual activity is more prevalent, with 95% of all users, the required installed capacity increases more than three times, as the limit increases from 20 to 100. Considering a scenario with 25% of users with profile *BoT* (Fig. 6(b)), the minimum capacity necessary is tripled with 75% of eventual activity, reaching peaks of fourfold increase when such activity reaches 95% and the limit is set at 100.

It is interesting to note that when the limit is set at 20 in the scenario with 10% of *BoT* users, an increase in the percentage of eventual users leads to a decrease in the capacity required, which is the opposite of what happens for larger values of the limit imposed (light blue area in Fig. 6(a)). A closer inspection on the simulation results reveals that this happens because in this particular case the demand distribution of the 10% of *BoT* users ends up being diluted within the large mass of eventual users. When the percentage of users with *BoT* profiles increases, this phenomenon is no longer relevant and the pressure caused by this type of user begins to be felt in the capacity required even for low values of the limit (Fig. 6(b)).

[4] Further details about this study, including the diagnostic, cube and interaction graphs, can be found at http://www.lsd.ufcg.edu.br/~rostand/IaaSModelH.zip.

**Table 2**
The parameters used in the simulation.

| Parameter | Value |
|---|---|
| Session duration ($\tilde{o}$) | $l_o = 1$ h and $u_o = 72$ h |
| Interval between sessions ($\tilde{i}$) | $k_i = 240$ h and $s_i = 2$ |
| Activity duration ($\tilde{d}$) | $\lambda_{d_1} = 0.125$ (8 h) and $\lambda_{d_2} = \lambda_{d_1}/2$ (16 h)<br>$\varphi_{d_1} = 0.7$ and $\varphi_{d_2} = 1 - \varphi_{d_1}$ |
| Waiting between activities or *think time* ($\tilde{t}$) | $\lambda_{t_1} = 0.25$ (4 h) and $\lambda_{t_2} = \lambda_{t_1}/2$ (8 h)<br>$\varphi_{t_1} = 0.7$ and $\varphi_{t_2} = 1 - \varphi_{t_1}$ |
| $\Delta T$ | 8.760 h (1 year) |
| Amount of Active Users ($\|U_a\|$) | {625; 1250; 2500; 5000} |
| Percentage of Eventual Activity | {25%; 35%; 45%; 55%; 65%; 75%; 85%; 95%} |
| Percentage of Users with *FlashMob* Profile | 1% |
| Percentage of Users with *BoT* Profile | {10%; 15%; 20%; 25%} |
| Limit ($L$) | {20; 30; 40; 50; 60; 70; 80; 90; 100} |
| Average Ticket ($\tau$) | 2 resources |

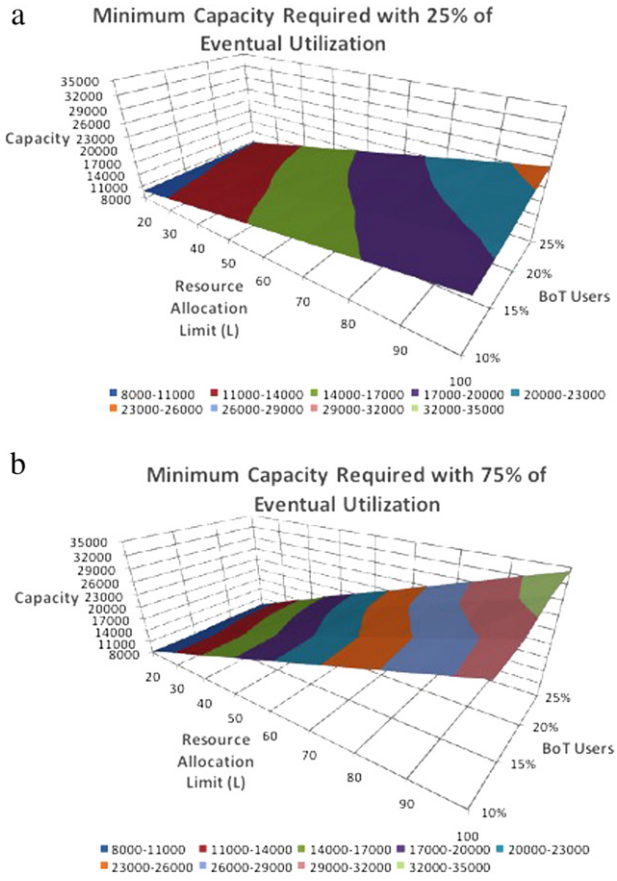



**Fig. 7.** The minimum capacity required for 100% service availability when varying the limit ($L$) and the percentage of *BoT* users for different scenarios of *Eventual* utilization.

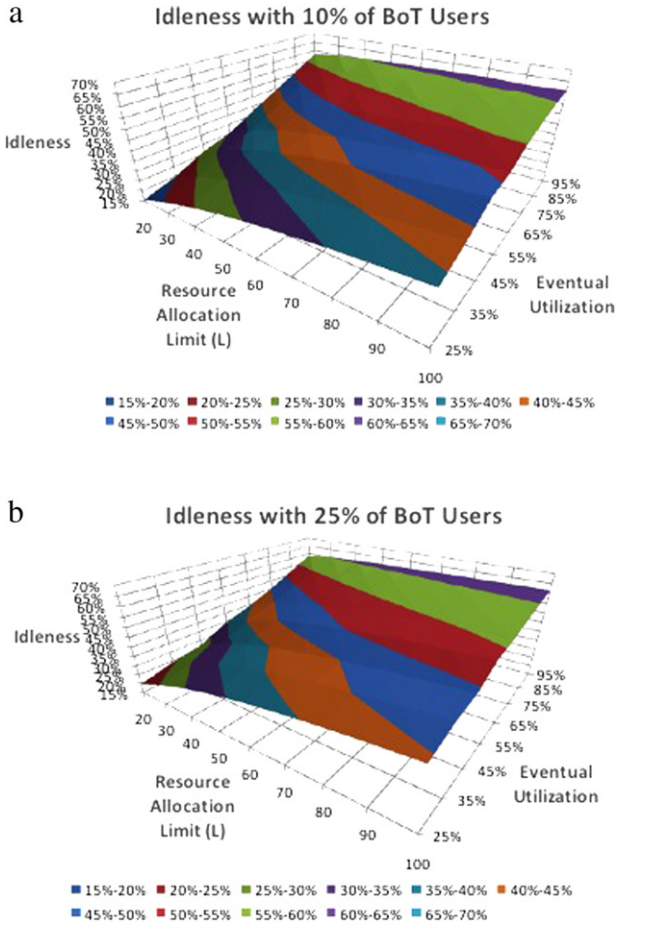



**Fig. 8.** The idleness observed when varying the limit ($L$) and the percentage of *Eventual* users for different scenarios of users with *BoT* profiles.

Fig. 7 shows a different perspective, in which the percentage of users with *BoT* profiles ranges from 10% to 25% in two scenarios of Eventual Utilization percentage (25% and 75%). Again, it was possible to observe a consistent increase of the minimum capacity required in both scenarios, influenced both by increase in the value of the limit and increase in the number of *BoT* users. It is possible to see that the percentage of *Eventual* users has a stronger impact on the minimal capacity required when combined with both the percentage of users with *BoT* profiles, and increase in the limit of resources that can be simultaneously allocated by a client.

A second analysis allowed us to observe how an increase in the installed capacity affects the level of system utilization. Using the values of $\max(\alpha(t))$ obtained in the previous experiment as the provider's capacity ($K$), we obtained the idleness presented by the system. The idleness is represented by the ratio between the total amount of resources used during the period $\Delta T$ and the total capacity available for the same period:

$$\frac{\sum_{t=1}^{\Delta T} \alpha(t)}{K \cdot \Delta T}.$$

Fig. 8 illustrates the idleness obtained in two scenarios: 10% and 25% of users with *BoT* profiles.

The results indicate a variation of idleness proportional to the variation of the limit, and the percentage of eventual users, ranging from 20% to 65% of idle capacity in all combinations of eventual activity and *BoT* profiles simulated.

Fig. 9 shows, for a scenario with 10% of *BoT* users and different levels of *Eventual* users, the evolution of the percentage increase

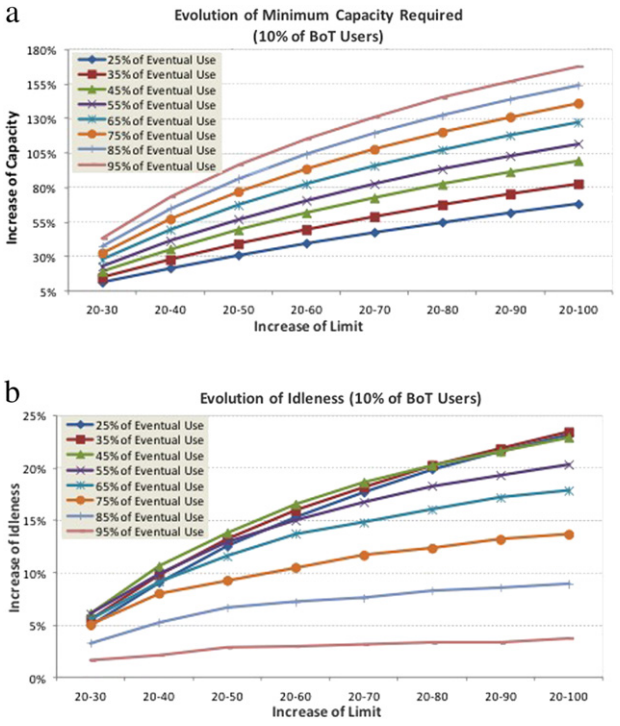



**Fig. 9.** The evolution of the minimum capacity required and the idleness observed when varying the limit ($L$) and percentage of *Eventual* users for a scenario with 10% of users with *BoT* profiles.

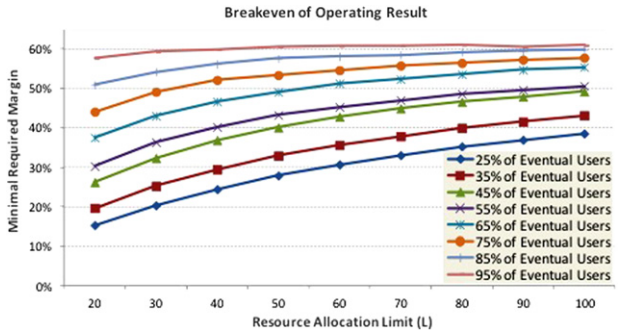



**Fig. 10.** Breakeven of the operating result when varying the limit ($L$) and the percentage of *Eventual* users for a scenario with 10% of *BoT* users.

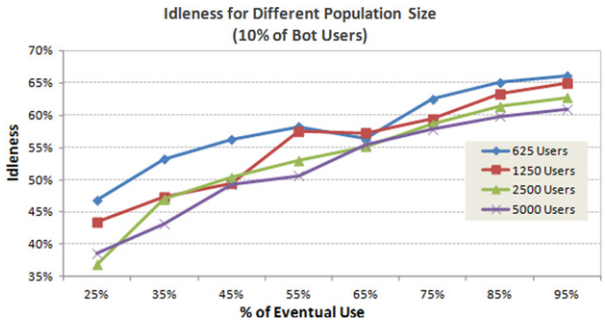



**Fig. 11.** The idleness for different user population sizes.

of the minimum capacity required to avoid violations, and of the corresponding idleness observed, when the limit value was increased in the experiments performed. As can be seen in Fig. 9(a), the minimum capacity necessary maintains an almost constant expansion in percentage terms in response to an increase in the percentage of *Eventual* users and in the limit imposed. On the other hand, as can be seen in Fig. 9(b), the percentage of idleness increase follows a different evolution pattern: the larger the percentage of *Eventual* users is, the smaller the percentage increase of idleness levels attained when the limit value increases is. In the case of 95% of *Eventual* users, the percentage increase observed in idleness is below 1% in each step, which leads to a total increase of below 5% when the limit varies from 20 to 100. The same behavior was also observed in scenarios with other percentages of *BoT* users. This happens because when the number of *Eventual* users is large, the idleness is already high, even for small values of the limit (as can be seen in Fig. 8). Moreover, this behavior shows that although an increase in the limit leads to a sizable impact on the idleness level, an increase in the number of *Eventual* users has an even stronger impact on the idleness level.

The need to increase the minimum capacity required impacts the initial investment (CAPEX) for the provider, while a corresponding increase in idleness level impacts its operational costs (OPEX). Considering the price charged by the current market leading IaaS provider [13], and using the expression for calculating the profit (Eq. (1)), a third analysis was performed. We applied different profit margins to the values obtained in the previous experiments to identify the point at which the operation of the provider becomes balanced, *i.e.*, no profit or loss, in each configuration. It was observed that as the limit was increased, the balance point of operation was achieved only when the profit margin was also increased, with direct consequences on the competitiveness of the provider. In Fig. 10, it can be seen that the profit margin necessary to equalize revenues and expenses ranges from 40% to almost 60% for the largest value considered for the limit, for a variation of 25%–75% of eventual activity, and with only 10% of users with *BoT* profiles.

In the previous experiments we fixed the population size to 5000 users (the maximum number of instances of the model that the tool used could simulate). In order to assess the impact that the population size may have on the results, the same experiments were repeated for different numbers of active users. With the same conditions of limit and activity profiles maintained, the observed curves are very similar for all numbers of active users simulated (Fig. 11). This is an indication that economies of scale will not play a role in improving the profitability of IaaS providers for the same value of $L$.

The results presented so far have considered a scenario in which no violations occur. Although service availability should always be very high, it is seldom cost effective to keep it at 100%. Given this fact, we also preformed experiments to assess how a more relaxed service availability level would impact the idleness level of the system and, as a result, its operational cost. In these experiments, we gradually reduced the capacity to below the minimum capacity required so that no violations would occur, identified in previous experiments, and for each reduction performed, we measured the violations introduced.

The service availability for various limit values in a population with only 35% of *Eventual* users is illustrated in Fig. 12(a). It can be observed that the capacity reduction has more dramatic effects on the service's availability for lower limit values. This is explained by the fact that these are the settings that present lower idleness and, hence, have less flexibility for reductions of the installed capacity. The idle capacities calculated for the same scenarios are illustrated in Fig. 12(b), where the effect just discussed can be better visualized.

Note that these simulations allow a service provider to perform an inverted analysis to identify the most suitable value for the limit $L$ required to meet a desired level of profit margin. For that, the provider must choose the value for $L$ that best balances its resulting idle capacity (availability costs), and the level of service availability (penalty costs).

Our experiments show that while the demand of regular users is permanent and predictable, its growth is beneficial to the

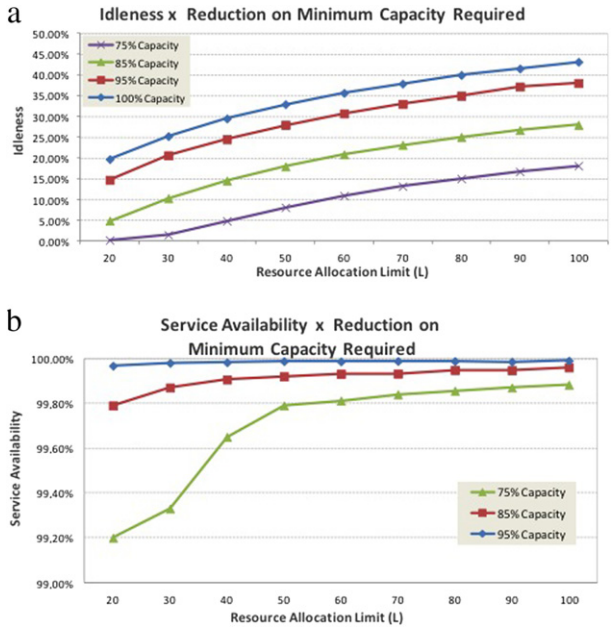



**Fig. 12.** The service availability level and idleness after a reduction of the capacity to below the minimum capacity required to achieve 100% service availability.

profitability of the provider, since it does not impose a risk of oversizing the infrastructure. Thus, the profit of the provider is negatively affected only by the demand that comes from eventual users, which can result in increased inactivity of the infrastructure, if not controlled. This is only exacerbated when eventual users are heavy consumers of resources and make very large sporadic demands. It was observed that users with eventual and intense utilization force the minimum capacity necessary and increase the idleness of the system, increasing the operational costs for the provider. Due to this, not only is the attribution of a limit for the allocation of resources necessary, but also the assigned value has a significant impact on the investment in infrastructure to ensure an adequate level of service availability for the provider.

## 6. Related work

The work by Menascé and Ngo [14] discusses how traditional methods of capacity planning were impacted by the advent of cloud computing, and how the risks and costs involved are migrating from customers to providers. The further investigation we conducted in this paper on the aspects of availability and demand regulation on the part of providers confirmed this condition.

The study by Greenberg et al. [15] shows that the typical costs associated with the construction of cloud data centers are distributed in four categories. A framework for detailed analysis of these investments, and how they make up the total cost of ownership (TCO) [16] for providers, was proposed by Li et al. [6]. Their study expanded the classification to eight categories of costs, and introduced the concept of Utilization Cost — the cost associated with the effective consumption of resources by users. We used the depreciation and utilization costs proposed by Li et al. to compute the profit obtained by an IaaS provider.

Anandasivam et al. [17] introduced a version of the concept of bid price tailored for cloud computing, in which the provider uses an auction system that acts as an influence on the behavior of price-sensitive users and regulates the use of the available resources. Our study shows that the limit imposed by providers is an alternative way to regulate users' demands. Indeed, an observation of the current state of affairs in the IaaS market shows that this is the option that is practiced by virtually all IaaS providers.

## 7. Conclusions

In this paper we analyzed the reasons that lead current IaaS providers to impose very restrictive limits on the number of resources that any client can simultaneously acquire. Our assessment uses a simulation model for an IaaS provider that is fed with a synthetic workload, allowing the simulation of a wide range of scenarios. The use of a closer to reality model seemed the most appropriate option for this study. To mitigate the complexity of the model and the lack of field data, we used techniques such as design of experiments, to identify the most important independent variables, and parameter sweep, for instantiation of a wide range of scenarios. We obtained consistent results in all simulated scenarios.

The analysis shows that it is mandatory to assign a limit for the amount of resources that can be simultaneously allocated to any user, in order to keep service availability sufficiently high and at a reasonable cost for the provider. The actual value for this limit will vary from provider to provider depending on its own assessment of this trade-off, but our results indicate that it tends to be not much larger than the current practiced values that fall in the range of a couple of dozen. We also observed that users with *Eventual* and *BoT* utilization pressure the minimum capacity necessary, and increase the idleness of the system, increasing the operating costs for the provider. Moreover, if the same profile of the population and the same limit value are maintained, the system dynamics is independent of the number of users and does not constitute, therefore, a context where economy of scale may lead to cost reductions.

The results helped in the understanding of the need to use a limit and how its impact on the profitability of the provider is directly related to the utilization pattern of the user population, making us conclude that some categories of users/applications that would benefit from a wider elasticity will continue to be underserved if an alternative model of public IaaS resource provisioning is not devised.

The next steps of our research include an investigation of alternative ways to minimize the costs involved with increase of the capacity of public cloud computing providers to appropriately deal with the demand of eager eventual users, such as those that need to run large scientific BoT applications. These costs are a major obstacle to the provision of elasticity in more flexible conditions, allowing these users to fully benefit from the advantages of the cloud computing model. The discovery, federation and resale of resources already amortized in other contexts may represent an alternative path, because these rely on the existence of idle capacity in contexts where the costs of availability have already been absorbed by other businesses or purposes.

## Acknowledgments

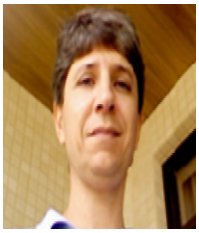
The authors are grateful to Prof. Jacques Sauvé for his suggestions on the simulation model and synthetic workload generator. This work was partially supported by CNPq/Brazil (grant 305858/2010-6), CAPES/Brazil, and RNP/CTIC through the funding of the JiT Clouds project. Francisco Brasileiro is a CNPq/Brazil researcher.

## References

[1] K. Stanoevska-Slabeva, T. Wozniak, Cloud basics-an introduction to cloud computing, in: K. Stanoevska-Slabeva, T. Wozniak, S. Ristol (Eds.), Grid and Cloud Computing, Springer Berlin Heidelberg, 2010, pp. 47–61. http://dx.doi.org/10.1007/978-3-642-05193-7_4.
[2] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, A view of cloud computing, Communications of the ACM 53 (2010) 50–58.
[3] W. Cirne, D. Paranhos, L. Costa, E. Santos-Neto, F. Brasileiro, Running Bag-of-Tasks Applications on Computational Grids: the MyGrid Approach, IEEE, 2003.

[4] AWS, Amazon Web Services, Available in http://aws.amazon.com, 2011.
[5] M. Sevior, T. Fifield, N. Katayama, Belle Monte-Carlo production on the Amazon EC2 cloud, Journal of Physics: Conference Series 219 (2010) 012003.
[6] X. Li, Y. Li, T. Liu, J. Qiu, F. Wang, The method and tool of cost analysis for cloud computing, in: 2009 IEEE International Conference on Cloud Computing, 2009.
[7] L.A. Barroso, U. Hölzle, The case for energy-proportional computing, Computer 40 (2007) 33–37.
[8] D.G. Feitelson, Workload Modeling for Computer Systems Performance Evaluation, 0.30 ed., Hebrew University of Jerusalem (Online Book), 2009.
[9] D. Talby, User Modeling of Parallel Workloads by User Modeling of Parallel Workloads, Hebrew University of Jerusalem (Ph.D. Thesis), Available in http://www.cs.huji.ac.il/labs/parallel/stud/Talby-PhD.pdf, 2006.
[10] R. Jain, The Art of Computer Systems Performance Analysis, John Wiley and Sons, 1991.
[11] J. Jung, B. Krishnamurthy, M. Rabinovich, Flash Crowds and Denial of Service Attacks, ACM Press, New York, 2002.
[12] D. Deavours, G. Clark, T. Courtney, D.e.a. Daly, The Mobius framework and its implementation, IEEE Transactions on Software Engineering 28 (2002).
[13] EC2, Amazon Elastic Compute Cloud Pricing, Available in http://aws.amazon.com/en/ec2/#pricing, 2012.
[14] D.A. Menascé, P. Ngo, Understanding cloud computing: experimentation and capacity planning, in: 2009 Computer Measurement Group Conference, Dallas, Texas, USA, 2009, p. 11.
[15] A. Greenberg, J. Hamilton, D.a. Maltz, P. Patel, The cost of a cloud, ACM SIGCOMM Computer Communication Review 39 (2008) 68.
[16] L. Mieritz, B. Kirwin, Defining Gartner Total Cost of Ownership, Gartner Group, Available in http://www.gartner.com, 2005.
[17] A. Anandasivam, S. Buschek, R. Buyya, A heuristic approach for capacity control in clouds, in: IEEE International Conference on E-Commerce Technology, 2009, pp. 90–97.

**Rostand Costa** received his bachelor and master degrees in Computer Science from the Federal University of Paraíba, Brazil, in 1988 and 1999, respectively. He is currently a Ph.D. candidate at the Federal University of Campina Grande, Brazil. He is also a researcher at UFPB's Digital Video Applications Laboratory and UFCG's Distributed Systems Laboratory. His research interests include distributed systems in general, with focus on cloud computing systems. He has been working as a software engineering and computational applications consultant for more than 25 years, with experience in government departments and in educational and health care companies. He can be contacted at rostand@lavid.ufpb.br.



**Francisco Brasileiro** is a Professor at the Federal University of Campina Grande, Brazil. He received a B.S. degree in Computer Science from the Federal University of Paraíba, Brazil in 1988, an M.Sc. degree from the same University in 1989, and a Ph.D. degree in Computer Science from the University of Newcastle upon Tyne, UK in 1995. His research interests include distributed systems in general, with focus on federated and cooperative systems. He is a member of the Brazilian Computer Society, the ACM, and the IEEE Computer Society. He can be contacted at fubica@dsc.ufcg.edu.br.

**Guido Lemos** de Souza Filho is a Professor of the Computer Engineering Department, Director of the Informatics Center and leader of the Digital Video Advanced Applications Laboratory (LAViD) at the Federal University of Paraíba, Brazil. He received a B.S. degree in Computer Science from the Federal University of Paraíba, Brazil in 1988, an M.Sc. in 1991, and a Ph.D. degree in Computer Science from the Pontifícia Universidade Católica do Rio de Janeiro, Brazil in 1997. He is member of the Digital Television Brazilian System Forum Board, which is in charge of the definition of Brazilian Digital TV standards. He worked as one of the designers of Ginga Middleware, adopted as standard by ITU and used in the Brazilian Digital TV System and DTV systems of 12 countries in Latin America and Africa. At LAViD, he works on research in distributed multimedia systems, digital television, digital cinema and accessibility. He can be contacted at guido@lavid.ufpb.br.

**Dênio Sousa** received a degree in Computer Science from the Federal University of Paraíba (UFPB), Brazil (1989), specialization in Information Technology in Education from the Federal University of Minas Gerais, Brazil (1993), a Masters in Computer Science from UFPB, Brazil (1998) and a Ph.D. in Computer Science from the Federal University of Pernambuco, Brazil (2004). He has been a Professor at the Federal Institute of Education Science and Technology of Paraíba since 1991, where, among other duties, he teaches in the Graduate Program in Electrical Engineering. He is also a researcher at UFPB's Digital Video Applications Laboratory, and a Professor of UFPB's Graduate Program in Computer Science. His research focus includes resource allocation and traffic analysis in computer networks. He can be contacted at denio@ifpb.edu.br.