A Game Theoretic Formulation of the Service Provisioning Problem in Cloud Systems

Danilo Ardagna*, Barbara Panicucci*, Mauro Passacantando**
*Politecnico di Milano, **Università di Pisa

ABSTRACT

Cloud computing is an emerging paradigm which allows the on-demand delivering of software, hardware, and data as services. As cloud-based services are more numerous and dynamic, the development of efficient service provisioning policies become increasingly challenging. Game theoretic approaches have shown to gain a thorough analytical understanding of the service provisioning problem.

In this paper we take the perspective of Software as a Service (SaaS) providers which host their applications at an Infrastructure as a Service (IaaS) provider. Each SaaS needs to comply with quality of service requirements, specified in Service Level Agreement (SLA) contracts with the end-users, which determine the revenues and penalties on the basis of the achieved performance level. SaaS providers want to maximize their revenues from SLAs, while minimizing the cost of use of resources supplied by the IaaS provider. Moreover, SaaS providers compete and bid for the use of infrastructural resources. On the other hand, the IaaS wants to maximize the revenues obtained providing virtualized resources. In this paper we model the service provisioning problem as a Generalized Nash game, and we propose an efficient algorithm for the run time management and allocation of IaaS resources to competing SaaSs.

1. INTRODUCTION

Cloud Computing has been a dominant IT news topic over the past few years. It is essentially a way for IT companies to deliver software/hardware on-demand and for costumers to store and access data over the Internet. Cloud computing applications are generally priced on a subscription model, so end-users may pay a yearly usage fee, for example, rather than the more familiar model of purchasing software to run on desktop. The Cloud-based services are not only restricted to software applications (Software as a Service – SaaS), but could also be the platform for the development and deployment of cloud applications (Platform as a Service – PaaS) and the hardware infrastructure (Infrastructure as a Service – IaaS). Many Companies, e.g. Google and Amazon, are offering Cloud computing services such as Google's App Engine and Amazon's Elastic Compute Cloud (EC2). Large data centers provide the infrastructure behind the Cloud and virtualization technology makes Cloud computing resources more efficient and cost-effective both for providers and customers. Indeed, end-users obtain the benefits of the infrastructure without the need to implement and administer it directly adding or removing capacity almost instantaneously on a "pay-as-you-use" basis. Cloud providers can, on the other hand, maximize the utilization of their physical resources also obtaining economies of scale.

The development of efficient service provisioning policies is among the major issues in Cloud research. Indeed, modern Clouds live in an open world characterized by continuous changes which occur autonomously and unpredictably. In this context, game theoretic methods allow to gain a in-depth analytical understanding of the service provisioning problem. Game Theory has been successfully applied to diverse problems such as Internet pricing, flow and congestion control, routing, and networking [5]. One of the most widely used "solution concept" in Game Theory is the Nash Equilibrium approach: A set of strategies for the players constitute a Nash Equilibrium if no player can benefit by changing his/her strategy while the other players keep their strategies unchanged or, in other words, every player is playing a *best response* to the strategy choices of his/her opponents.

In this paper we take the perspective of SaaS providers which host their applications at an IaaS provider. Each SaaS provider wants to maximize its profit while complying with QoS requirements, specified in Service Level Agreement (SLA) contracts with the end-users, which determine the revenues and penalties on the basis of the achieved performance level. The profit of the SaaS is given by the revenues from SLAs minus the cost sustained for using the resources supplied by the IaaS. However, each SaaS competes with others SaaS and bids for the use of infrastructural resources. The IaaS, in his turn, wants to maximize the revenues obtained providing the resources. To capture the behavior of SaaSs and IaaS in this conflicting situation in which the best choice for one depends on the choices of the others, we recur to the Generalized Nash equilibrium concept [12], which is an extension of the classical Nash equilibrium. In this paper the service provisioning problem will be modelled as a Generalized Nash game. We then use Game Theory results for developing an efficient algorithm for the run time allocation of IaaS resources to competing SaaSs.

The remainder of the paper is organized as follows. Section 2 introduces the reference system under study. In Sec-

tion 3 we introduce our model based on the concept of Generalized Nash equilibrium and we prove its existence. In Section 4 we provide analytical results for a simple case study, while a general solution method is proposed in Section 5. The experimental results discussed in Section 6 demonstrate the efficiency of our method. Other approaches proposed in the literature are discussed in Section 7. Conclusions are finally drawn in Section 8.

2. PROBLEM STATEMENT AND ASSUMPTIONS

We consider SaaS providers using Cloud computing facilities to offer services, according to the IaaS paradigm. We assume that a SaaS provider offers multiple transactional Web services (WSs) and each service represents a different application. The hosted WSs can be heterogeneous with respect to resource demands, workload intensities and QoS requirements. The set of WS applications offered by the p-th SaaS provider are denoted with $\mathcal{A}_p$.

An SLA contract, associated with each WS application, is established between the SaaS provider and its end-users. In particular, as in other approaches [8, 9, 30], for each WS application $k \in \mathcal{A}_p$, a linear utility function specifies the per request revenue (or penalty) $\mathcal{V}_k = \nu_k + m_k R_k$ incurred when the end-to-end response time R_k assumes a given value. The slope of the utility function is $m_k = -\nu_k/\overline{R}_k < 0$ and $\overline{R}_k$ is the threshold that identifies the revenue/penalty region, that is, if $R_k > \overline{R}_k$ the SLA is violated and the SaaS incurs in penalties. Linear utility functions are a flexible mechanism to rank different applications (e.g., assigning higher slopes to more important applications), and allow also to implement soft constraints on response times since the SaaS goal is to keep the infrastructure working in a profitability region, i.e., to provide an average response time lower than $\overline{R}_k$ looking for the trade-off between the SLA revenues and infrastructural costs [8].

Applications are hosted in virtual machines (VMs) which are dynamically instantiated by the IaaS provider. We make the simplifying assumption that each VM hosts a single WS application. Multiple VMs implementing the same WS application can also run in parallel. In that case, we further assume that the running VMs are homogeneous in terms of RAM and CPU capacity C and evenly share the workload.

IaaS providers usually charge software providers on an hourly basis [6]. Hence, the SaaS has to face the problem of determining every hour the optimal number of VMs for each WS class in order to maximize the net revenues.

The SaaS performs resource allocation on the basis of a prediction of future WS workloads [9]. The SaaS needs also an estimate of the future performance of each VM in order to determine application average response time. In the following we model each WS class hosted in a VM as an M/G/1 queue [10], as done in [25, 1] and we assume, as common among Web service containers, that requests are served according to the processor sharing scheduling discipline.

For the IaaS provider we consider a model similar to Amazon EC2 [6] and we assume that the IaaS provider offers: (i) *flat* VMs, for which SaaS providers applies for a one-time payment (currently every one or three years) for each instance they want to reserve, (ii) *on demand* VMs, which allows SaaS to access computing capacity with no long-term commitments, and (iii) *on spot* VMs, for which SaaS providers bid and compete for unused IaaS resources.

The VM instances are charged with the on spot cost σ_k for application k, which is set by the IaaS and fluctuates periodically depending on the IaaS provider time of the day energy costs and also on the supply and demand from SaaS for on spot VMs [6, 19]. Indeed, SaaS providers compete among them for the use of on spot VMs and specify the maximum cost σ_k^U for each application k they are willing to pay per instance hour. The on spot cost σ_k is fixed by the IaaS provider which can also decide to do not allocate any on spot instance to a SaaS. On the other hand, each SaaS provider is guaranteed to have access up to f_p^U flat VMs he reserved by applying to the one-time payment. The number of on spot VMs available at the IaaS cloud service center is denoted by s^U.

For example in the Amazon case on spot costs are available via the EC2 API [6] or by third party sites [28]. On spot costs fluctuate according to the time of the day and on the Cloud site region, and could be less or greater than the time unit cost φ for flat VMs. Finally, we denote with δ the cost for on demand instances. With the current pricing models, δ is strictly greater than φ and we assume $\delta > \sigma_k^U$ for all k. Indeed, since the IaaS provider can arbitrarily terminate on spot instances from a SaaS resource pool [6], no one is willing to pay for a less reliable resource a time unit cost higher than on demand instances, which provide a higher availability level.

On spot instances have been traditionally adopted to support batch computing intensive workloads [6]. However, since nowadays IaaS providers allow specifying autonomic policies which can dynamically allocate VM instances in few minutes as a reaction to failures, we advocate the use of on spot instances also to support the execution of traditional Web applications.

3. GENERALIZED NASH EQUILIBRIUM APPROACH

Hereafter we introduce the Generalized Nash Equilibrium Problem arising in the Cloud computing system under study and we prove the existence of at least an equilibrium.

3.1 Problem Formulation

The goal of SaaS provider p is to determine every hour the number of flat f_k, on demand d_k and on spot s_k VMs to be devoted for the execution of all WS applications $k \in \mathcal{A}_p$, in order to maximize its profits and, at the same time, so as to to satisfy the prediction Λ_k for the arrival rate of the WS application k. Let us denote with μ_k the maximum service rate for the requests of application k on a VM of capacity one. If the workload is evenly shared among the VMs, then the average response time for the execution of application k requests is given by:

$$E[R_k] = \frac{1}{C\mu_k - \frac{\Lambda_k}{f_k+d_k+s_k}}, \tag{1}$$

under the assumption that VMs are not saturated, i.e. guaranteeing the equilibrium conditions for the M/G/1 queues $C\,\mu_k\,(f_k + d_k + s_k) - \Lambda_k > 0$.

The average per time unit revenues for application k requests are given by $\mathcal{V}_k\,\Lambda_k = (\nu_k + m_k\,E[R_k])\,\Lambda_k$.

Considering the infrastructural costs to access flat, on demand, and on spot VM instances the goal of a SaaS provider is to maximize its profits given by:

$$\sum_{k\in\mathcal{A}_p} \left(\nu_k\,\Lambda_k \; + \frac{m_k\,\Lambda_k(f_k+d_k+s_k)}{C\,\mu_k\,(f_k+d_k+s_k) - \Lambda_k} + \right.$$

$$-\varphi\, f_k - \delta\, d_k - \sigma_k\, s_k)\,.$$

With this setting in mind, the problem of the generic SaaS provider p becomes:

$$\max\ \Theta_p = \sum_{k\in\mathcal{A}_p} \frac{m_k\,\Lambda_k(f_k+d_k+s_k)}{C\,\mu_k\,(f_k+d_k+s_k)-\Lambda_k} + - \sum_{k\in\mathcal{A}_p} \varphi\, f_k - \sum_{k\in\mathcal{A}_p} \delta\, d_k - \sum_{k\in\mathcal{A}_p} \sigma_k\, s_k$$

$$\sum_{k\in\mathcal{A}_p} f_k \le f_p^U \tag{2}$$

$$f_k + d_k + s_k > \frac{\Lambda_k}{C\,\mu_k} \qquad \forall\ k\in\mathcal{A}_p \tag{3}$$

$$\sum_{k\in\mathcal{A}} s_k \le s^U \tag{4}$$

$$f_k, d_k, s_k \ge 0 \qquad \forall\ k\in\mathcal{A}_p. \tag{5}$$

Note that the terms $\sum_{k\in\mathcal{A}} \nu_k\,\Lambda_k$ can be dropped in the SaaS objective function since they are independent of the decision variables.

Constraint (2) entails that the flat VMs allocated to applications are less or equal to the one available, while constraint (3) guarantees that resources are not saturated. Finally constraint (4) guarantees than the on spot VMs allocated to competing SaaS providers are lower than the one available at the IaaS cloud service center s^U.

We would like to remark that, in the formulation of the problem, we have not imposed variables f_k, d_k, s_k to be integer, as in reality they are. In fact, requiring variables to be integer makes the solution much more difficult. However, preliminary experimental results have shown that if the optimal values of the variables are fractional and they are rounded to the closest integer solution, the gap between the solution of the real integer problem and the relaxed one is very small, justifying the use of a relaxed model. We therefore decide to deal with continuous variables, actually considering a relaxation of the real problem.

On the other side, the IaaS provider's goal is to determine the time unit cost σ_k for on spot VM instances for all applications $k\in\mathcal{A}_p$ and every SaaS provider p, in order to maximize its total revenue:

$$\max\ \Theta_I = \sum_{k\in\mathcal{A}} (\varphi\, f_k + \delta\, d_k + \sigma_k\, s_k)$$

$$\sigma_k^L \le \sigma_k \le \sigma_k^U \qquad \forall\ k\in\mathcal{A}, \tag{6}$$

where $\mathcal{A}$ denotes the set of indexes of all WS applications (i.e., $\mathcal{A}=\cup_p\mathcal{A}_p$, $\mathcal{A}_{p_1}\cap\mathcal{A}_{p_2}=\oslash$ if $p_1\neq p_2$).

Note that the on spot instance cost lower bound σ_k^L is fixed according to the time of the day and includes the energy costs for running a single VM instance for one hour and the amortized cost of the hosting physical machine [19]. For the sake of clarity, the notation adopted in this paper is summarized in Table 1.

If the maximum time unit costs of an application is lower than the minimum set by the IaaS, i.e. $\sigma_k^U < \sigma_k^L$, formally the SaaSs and IaaS problems have no solution. In that case we can set $s_k = 0$ and consider a simplified problem where the capacity allocation problem for application k is limited to determine the number of flat and on demand instances. Hence in the following we will always assume that $\sigma_k^L \le \sigma_k^U$ for all k. Note that, if the on spot instances are terminated by the IaaS provider, then the SaaS can dynamically start the execution of an equal number of on demand instances.

In this framework, SaaS providers and the IaaS provider are making decisions at the same time, and the decisions of a SaaS depend on those of the others SaaS and the IaaS. Vice versa, the IaaS objective function depends on SaaS decisions. In this setting, we can not analyze decision in isolation, but we must ask what a SaaS would do, taking into account the decision of the IaaS and other SaaSs. To capture the behavior of SaaSs and IaaS in this conflicting situation (game) in which what a SaaS or the IaaS (the players of the game) does directly affects what others do, we consider the Generalized Nash equilibrium concept [12], which is broadly used in Game Theory and other fields. We remind the reader that the generalized Nash equilibrium problem (GNEP) differs from the classical Nash equilibrium problem (NEP) since, not only the objective functions of each player (called *payoff functions*) depend upon the strategies chosen by all the other players, but also each player's strategy set may depend on the rival players' strategies. In our setting the constraint of each problem involving other player's variables (joint constraint) comes from (4).

Following the Nash equilibrium concept, SaaS and IaaS providers adopt a strategy such that none of them can improve its revenue by changing its strategy unilaterally (i.e., while the other players keep their strategies unchanged). The service provisioning problem results therefore in a GNEP where the players are the SaaS providers and the IaaS provider, the strategy variables of SaaS provider p are f_k, d_k, and s_k, for $k\in\mathcal{A}_p$, while the strategy variables of the IaaS are the costs for *on spot* VMs, σ_k, for all $k\in\mathcal{A}$. Within this setting, the IaaS's strategy is simple. In fact, if a SaaS provider decides not to use on spot VMs for application k, that is $s_k = 0$, the value of the IaaS payoff does not depend on the choice for σ_k, that can be any feasible value. Whereas, if $s_k \neq 0$, regardless its value, the best response of the IaaS is to play $\sigma_k = \sigma_k^U$. When one player has a strategy that yields a higher revenue, no matter which choice the other players makes, that player is said to have a *dominant strategy*, and he will play that strategy in each of the Nash equilibria. Therefore, whenever $s_k \neq 0$, the IaaS will play its dominant strategy $\sigma_k = \sigma_k^U$. Another important feature of the derived GNEP is that it satisfies the *Convexity Assumption*: the payoff functions of both SaaS providers and IaaS, are concave in its own variables (Θ_p is concave [7] being the sum of linear and concave function, and Θ_I is linear) and the set of strategies are convex. Moreover, even if the decision of a SaaS depends on the decisions of the other SaaSs and the IaaS, the only constraint of each problem involving other player's variables (coming from (4) in each SaaS problem), is the same for all players: we refer to this special class of GNEP as *jointly convex* GNEP [14].

3.2 Existence of equilibria

Using the model introduced in the previous sections, we now prove that there exists an equilibrium for service provisioning on the Cloud. The proof is based on the equivalence between generalized Nash equilibria and fixed points of the best-response mapping, and on the Kakutani's fixed point theorem [20].

To simplify the discussion we introduce the following notations. Let $x^p = (f_k, d_k, s_k)_{k\in\mathcal{A}_p}$ denotes the strategies vec-

Decision Variables

f_k	Number of *flat* VMs used for application k
d_k	Number of *on demand* VMs used for application k
s_k	Number of *on spot* VMs used for application k
σ_k	Time unit cost for *on spot* VMs used for application k

System Parameters

n	number of SaaS providers
$\mathcal{A}_p$	Set of applications of the p SaaS provider
$\mathcal{A}$	Set of applications of all the SaaS providers
f_p^U	Maximum number of flat computational resources IaaS can provide for provider p
s^U	Maximum number of on spot computational resources IaaS can provide for all the SaaS providers
C	Capacity of computational resources
Λ_k	Prediction of the arrival rate for application k
μ_k	Maximum service rate of a capacity 1 server for executing class k application
m_k	Application k utility function slope
φ	Time unit cost for *flat* VMs
δ	Time unit cost for *on demand* VMs
σ_k^L	Minimum time unit cost for *on spot* VMs used for application k, set by the IaaS provider
σ_k^U	Maximum time unit cost for *on spot* VMs used for application k, set by the SaaS provider

Table 1: Parameters and decision variables.

tor of SaaS provider p, $x = (x^p)_{p=1}^n$, x^{-p} the vector formed by the strategies of all SaaS providers different from p and $\sigma = (\sigma_k)_{k\in\mathcal{A}}$. Moreover we indicate by $X_p(x^{-p})$ the set of strategies for provider p, and X_I the set of strategies of the IaaS provider.

Theorem 1 *There exists at least one generalized Nash equilibrium for the game.*

Proof Let consider any SaaS provider p. For any feasible strategy x^{-p} of the other SaaS providers we have that $X_p(x^{-p})$ contains the set:

$$X_p^L := \{x^p \geq 0 : \quad \sum_{k\in\mathcal{A}_p} f_k \leq f_p^U, \quad s_k = 0 \quad \forall\, k \in \mathcal{A}_p \\ f_k + d_k > \Lambda_k/(C\,\mu_k) \quad \forall\, k \in \mathcal{A}_p\}.$$

Moreover, for any feasible strategy σ of IaaS provider we have the following relations:

$$\Theta_p^L(x^p) \leq \Theta_p(x^p,\sigma) \leq \Theta_p^U(x^p),$$

where $\Theta_p^L(x^p)$ is:

$$\sum_{k\in\mathcal{A}_p}\left[\frac{m_k\,\Lambda_k(f_k+d_k+s_k)}{C\,\mu_k\,(f_k+d_k+s_k)-\Lambda_k} - \varphi\,f_k - \delta\,d_k - \sigma_k^U\,s_k\right]$$

and $\Theta_p^U(x^p)$ is:

$$\sum_{k\in\mathcal{A}_p}\left[\frac{m_k\,\Lambda_k(f_k+d_k+s_k)}{C\,\mu_k\,(f_k+d_k+s_k)-\Lambda_k} - \varphi\,f_k - \delta\,d_k - \sigma_k^L\,s_k\right].$$

We remark that if $f_k + d_k + s_k \to \frac{\Lambda_k}{C\,\mu_k}$ for some $k \in \mathcal{A}_p$, then $\Theta_p^U(x^p) \to -\infty$ and if $d_k \to +\infty$ then $\Theta_p^U(x^p) \to -\infty$ as well.

If we denote by $M^p := \max_{x^p\in X_p^L} \Theta_p^L(x^p)$, then the set:

$$\widetilde{X_p} := \{x^p :\ \Theta_p^U(x^p) \geq M^p\}$$

is nonempty, convex and compact. Therefore, for any feasible x^{-p} and σ we obtain that:

$$\max_{x^p\in X_p(x^{-p})} \Theta_p(x^p,\sigma) \geq \max_{x^p\in X_p^L} \Theta_p(x^p,\sigma) \geq \max_{x^p\in X_p^L} \Theta_p^L(x^p) = M^p$$

thus:

$$\arg\max_{x^p\in X_p(x^{-p})} \Theta_p(x^p,\sigma) \subseteq \widetilde{X_p},$$

that is the sets of best responses of player p to the strategies of the rivals are uniformly bounded by $\widetilde{X_p}$.

Finally, we consider the convex compact set

$$\widetilde{X} := \widetilde{X_1} \times \ldots \widetilde{X_n} \times X_I$$

and the best response set-valued mapping B defined as follows:

$$B(x,\sigma) := \left[\arg\max_{y^1\in X_1(x^{-1})} \Theta_1(y^1,\sigma)\right] \times \ldots \\ \times \left[\arg\max_{y^n\in X_n(x^{-n})} \Theta_n(y^n,\sigma)\right] \times \arg\max_{\sigma'\in X_I} \Theta_I(x,\sigma').$$

From the above discussion it follows that the set valued map $B : \widetilde{X} \rightrightarrows \widetilde{X}$ has nonempty and convex values, and its graph is closed by continuity of payoff functions. Therefore, by Kakutani's theorem there exists a fixed point of B, that is a strategy $(x,\sigma) \in B(x,\sigma)$, which is a generalized Nash equilibrium of the game.

4. A SINGLE APPLICATION CASE STUDY

In order to gain insight into the properties of the equilibria in our setting, let us focus on the case of a single SaaS provider with a single application class.

In the following we will also assume that the IaaS is overprovisioned and there is no an upper bound s^U on the number of on spot VM instances available and, hence, the constraint (4) is relaxed. Indeed, it is not reasonable that a single SaaS will be able to saturate the on spot instances capacity available in a real system. In that case, each player's strategy (SaaS and IaaS) belong to a set which is fixed and does not depend on the rival players' strategies: hence the GNEP reduces to a NEP which is much more simple to solve.

The aim of the SaaS provider, given the IaaS strategy σ, is to choose f, d, and s that maximize the payoff:

$$\Theta_S = \frac{m\,\Lambda\,(f+d+s)}{C\,\mu\,(f+d+s)-\Lambda} - \varphi\,f - \delta\,d - \sigma\,s,$$

over the set $X_S = \{(f, d, s) \in \mathbb{R}^3_+ : \ f + d + s > \frac{\Lambda}{C\,\mu},\ f \leq f^U\}$. The aim of the IaaS provider, given the SaaS strategies (f, d, s), is to choose σ that maximize the payoff:

$$\Theta_I = \varphi\, f + \delta\, d + \sigma\, s,$$

over the set $X_I = \{\sigma \in \mathbb{R} : \ \sigma^L \leq \sigma \leq \sigma^U\}$.

An analytic study can be obtained writing down the KKT conditions for the SaaS and the IaaS optimization problems and concatenating them and is omitted here for space limitation. See [7] for further details. We can observe that the Nash equilibria depend on the application workload conditions and on the relation between the cost φ of the flat VMs and the upper bound σ^U for on spot instances cost. The obtained equilibria and the corresponding value of the payoff functions Θ_S and Θ_I are reported in Table 2.

The first important remark we get is that, in general, there is no a unique Nash equilibrium (e.g., when $\varphi = \sigma^U$). However, if multiple equilibria exist they are equivalent, that is they are characterized by the same values of the payoffs. Furthermore, assuming $\delta > \sigma^U_k$ for all k we always have $d = 0$ (i.e., the SaaS provider always adopt flat or on spot VMs which are always cheaper than on demand instances).

When $\varphi \geq \sigma^U$ (Table 2, first and second rows) then it is more convenient for the SaaS provider to use only on spot instances which number can be obtained by a closed formula under any workload conditions. Remind that the value of σ^L is set by the IaaS provider, and SaaS provider can not set the value of σ^U too low otherwise the IaaS will not provide on spot resources. Let us now examine the case $\varphi < \sigma^U$ and assume initially that $f^U \leq \frac{\Lambda}{C\,\mu}$. In these hypotheses, the SaaS provider is under heavy load conditions and requests cannot be executed by using only the flat VMs. The SaaS provider will use all of its flat VMs and will buy on spot VMs at the maximum cost. Again, the number of on spot VMs can be determined by a closed formula (see Table 2, third row). Since $\varphi < \sigma^U$, in these conditions the payoff Θ_S is greater than the value obtained when $\varphi \geq \sigma^U$ (Table 2 first and second rows). Hence these conditions are more profitable for the SaaS provider which determines the optimal trade-off between the cost of the flat and on spot VMs and the revenues which can be achieved by the SLA contract. With the same arguments we can derive that, vice versa, the payoff Θ_I is lower than the value obtained when $\varphi \geq \sigma^U$, hence these conditions are less profitable from the IaaS point of view even if on spot instances are sold at the maximum cost.

Let us now examine the case when $\varphi < \sigma^U$ and $f^U > \frac{\Lambda}{C\,\mu}$. The SaaS provider is under light load conditions and, therefore, the incoming requests could be executed by using only the flat VMs. Under these hypotheses, the equilibria are set according to the marginal value of the payoff function of the SaaS player evaluated in $f = f^U$ and $s = 0$, that is $\frac{-m\,\Lambda^2}{(C\,\mu\,f^U - \Lambda)^2}$. This marginal value represents the change in the revenues of SaaS per unit change in s, i.e., the increase in the SaaS provider revenues deriving from SLA contracts obtained providing better performance to end-users by adopting an additional on spot VM. We identify two cases:

- if $\varphi < \sigma^U < \frac{-m\,\Lambda^2}{(C\,\mu\,f^U - \Lambda)^2}$ (see Table 2, fourth row), the marginal value is greater than the cost of individual on spot instance sold at the maximum cost ($\sigma = \sigma^U$), therefore it is convenient for the SaaS provider to buy on spot instances. Note that under these conditions, the payoff Θ_S is greater than the value obtained when $\varphi \geq \sigma^U$ as in the previous case. Vice versa, for the IaaS provider these circumstances are less profitable.
- if $\varphi < \sigma^U$ and $\frac{-m\,\Lambda^2}{(C\,\mu\,f^U - \Lambda)^2} \leq \sigma^U$ (see Table 2, fifth and sixth row), then the marginal value is less or equal to the cost of on spot VMs and hence the SaaS provider has no incentives to buy additional on spot VMs. Also in these cases, the equilibrium value of the payoff of SaaS provider is greater than the one obtained when $\varphi \geq \sigma^U$, while this again does not happen for the IaaS provider. Finally, it is worth noticing that if $\frac{-\sigma\,\Lambda^2}{(C\,\mu\,f^U - \Lambda)^2} < \varphi$ the SaaS provider adopts a number of VMs lower than f^U. Hence, in that case the costs incurred by using a larger number of flat VMs cannot be counterbalanced by the revenues which can be obtained by improving application performance.

5. GENERAL SOLUTION METHOD

In the previous sections, we showed that the service provisioning problem in a Cloud Computing environment represents a GNEP and more precisely a jointly convex GNEP. Concerning solution algorithms for general GNEPs, the literature is still very limited. More interesting results have been obtained for GNEPs with a jointly convex structure as our model is. In fact, a jointly convex GNEP can be solved introducing a variational inequality (VI) problem[1] reformulation: given a jointly convex GNEP with N players, whose payoffs Θ_i are continuously differentiable and convex with respect to x_i, then every solution of the VI, where $F = -(\nabla_{x_i}\Theta_i(x))_{i=1}^N$ and X is the set of individual and joint constraints, is also a solution of the jointly convex GNEP (see, e.g., [14]). We remark that the GNEP has usually multiple or even infinitely many solutions and it is not true that any solution of the jointly convex GNEP is also a solution of the VI. A solution of the jointly convex GNEP which is also a solution of the VI is called a *variational equilibrium*. Our approach is to calculate, among all the equilibria, a variational equilibrium which is more socially stable than other equilibria (see [11]) and it is a valuable target for an algorithm.

There are plenty of methods for solving VI problems. In our setting we have:

$$F = \begin{bmatrix} \frac{m_1\,\Lambda_1^2}{(C\,\mu_1\,(f_1 + d_1 + s_1) - \Lambda_1)^2} + \varphi \\ \frac{m_1\,\Lambda_1^2}{(C\,\mu_1\,(f_1 + d_1 + s_1) - \Lambda_1)^2} + \delta \\ \frac{m_1\,\Lambda_1^2}{(C\,\mu_1\,(f_1 + d_1 + s_1) - \Lambda_1)^2} + \sigma_1 \\ \cdots \\ \cdots \\ -s_1 \\ \cdots \end{bmatrix}$$

whose Jacobian is:

$$JF = \left[\begin{array}{ccc|c} \begin{bmatrix} a_1 & a_1 & a_1 \\ a_1 & a_1 & a_1 \\ a_1 & a_1 & a_1 \end{bmatrix} & & 0 & \\ & \ddots & & B \\ 0 & & \begin{bmatrix} a_{|\mathcal{A}|} & a_{|\mathcal{A}|} & a_{|\mathcal{A}|} \\ a_{|\mathcal{A}|} & a_{|\mathcal{A}|} & a_{|\mathcal{A}|} \\ a_{|\mathcal{A}|} & a_{|\mathcal{A}|} & a_{|\mathcal{A}|} \end{bmatrix} & \\ \hline -B^T & & & 0 \end{array}\right]$$

[1]Given $X \subseteq \mathbb{R}^n$ and $F\colon \mathbb{R}^n \to \mathbb{R}^n$, the VI problem consists in finding a vector $x^* \in X$ such that $\langle F(x^*), x - x^* \rangle \geq 0$, for all $x \in X$.

Conditions	SaaS Equilibrium and value	IaaS Equilibrium and value
$\varphi > \sigma^U$	$f = 0 \quad d = 0 \quad s = \frac{\Lambda}{C\,\mu}\left(1 + \sqrt{\frac{-m}{\sigma^U}}\right)$ $\Theta_S = -\frac{\Lambda}{C\,\mu}(\sqrt{\sigma^U} + \sqrt{-m})^2$	$\sigma = \sigma^U$ $\Theta_I = \frac{\Lambda}{C\,\mu}(\sqrt{-m\,\sigma^U} + \sigma^U)$
$\varphi = \sigma^U$	$0 \le f \le f^U \quad d = 0 \quad s \ge 0$ $f + s = \frac{\Lambda}{C\,\mu}\left(1 + \sqrt{\frac{-m}{\sigma^U}}\right)$ $\Theta_S = -\frac{\Lambda}{C\,\mu}(\sqrt{\sigma^U} + \sqrt{-m})^2$	$\sigma = \sigma^U$ $\Theta_I = \frac{\Lambda}{C\,\mu}(\sqrt{-m\,\sigma^U} + \sigma^U)$
$\varphi < \sigma^U$ $f^U \le \frac{\Lambda}{C\,\mu}$	$f = f^U \quad d = 0 \quad s = \frac{\Lambda}{C\,\mu}\left(1 + \sqrt{\frac{-m}{\sigma^U}}\right) - f^U$ $\Theta_S = -\frac{\Lambda}{C\,\mu}(\sqrt{\sigma^U} + \sqrt{-m})^2 + f^U\,(\sigma^U - \varphi)$	$\sigma = \sigma^U$ $\Theta_I = \frac{\Lambda}{C\,\mu}(\sqrt{-m\sigma^U} + \sigma^U) + f^U\,(\varphi - \sigma^U)$
$\varphi < \sigma^U$ $f^U > \frac{\Lambda}{C\,\mu}$ $\sigma^U < \frac{-m\,\Lambda^2}{(C\,\mu\,f^U - \Lambda)^2}$	$f = f^U \quad d = 0 \quad s = \frac{\Lambda}{C\,\mu}\left(1 + \sqrt{\frac{-m}{\sigma^U}}\right) - f^U$ $\Theta_S = -\frac{\Lambda}{C\,\mu}(\sqrt{\sigma^U} + \sqrt{-m})^2 + f^U\,(\sigma^U - \varphi)$	$\sigma = \sigma^U$ $\Theta_I = \frac{\Lambda}{C\,\mu}(\sqrt{-m\sigma^U} + \sigma^U) + f^U\,(\varphi - \sigma^U)$
$\varphi < \sigma^U$ $f^U > \frac{\Lambda}{C\,\mu}$ $\frac{-m\,\Lambda^2}{(C\,\mu\,f^U - \Lambda)^2} < \varphi$	$f = \frac{\Lambda}{C\,\mu}\left(1 + \sqrt{\frac{-m}{\varphi}}\right) \quad d = 0 \quad s = 0$ $\Theta_S = -\frac{\Lambda}{C\,\mu}(\sqrt{\varphi} + \sqrt{-m})^2$	$\max\{\sigma^L, \varphi\} \le \sigma \le \sigma^U$ $\Theta_I = \frac{\Lambda}{C\,\mu}(\sqrt{-m\,\varphi} + \varphi)$
$\varphi < \sigma^U$ $f^U > \frac{\Lambda}{C\,\mu}$ $\varphi \le \frac{-m\,\Lambda^2}{(C\,\mu\,f^U - \Lambda)^2} \le \sigma^U$	$f = f^U \quad d = 0 \quad s = 0$ $\Theta_S = \frac{m\,\Lambda\,f^U}{C\,\mu\,f^U - \Lambda} - \varphi\,f^U$	$\max\{\sigma^L, \frac{-m\,\Lambda^2}{(C\,\mu\,f^U - \Lambda)^2}\} \le \sigma \le \sigma^U$ $\Theta_I = \varphi\,f^U$

Table 2: Single Application Single SaaS Equilibria.

where the generic term $a_i = -\frac{2\,m_i\,\Lambda_i^2\,C\,\mu_i}{(C\,\mu_i\,(f_i + d_i + s_i) - \Lambda_i)^3} > 0$, and B is a matrix of dimension $3\,|\mathcal{A}| \times |\mathcal{A}|$.

Its symmetric part is:

$$\left[\begin{array}{cccc|c} \begin{bmatrix} a_1 & a_1 & a_1 \\ a_1 & a_1 & a_1 \\ a_1 & a_1 & a_1 \end{bmatrix} & & 0 & & \\ & \ddots & & & 0 \\ & & \begin{bmatrix} a_{|\mathcal{A}|} & a_{|\mathcal{A}|} & a_{|\mathcal{A}|} \\ a_{|\mathcal{A}|} & a_{|\mathcal{A}|} & a_{|\mathcal{A}|} \\ a_{|\mathcal{A}|} & a_{|\mathcal{A}|} & a_{|\mathcal{A}|} \end{bmatrix} & & \\ 0 & & & & \\ \hline & 0 & & & 0 \end{array}\right]$$

Therefore, in the feasible set X, its non-zero eigenvalues (i.e., $3\,a_1, \cdots, 3\,a_{|\mathcal{A}|}$) are positive (being $m < 0$ and $f + d + s > \frac{\Lambda}{C\,\mu}$), and hence F results to be monotone[2] (not strictly). Hence we can calculate a variational equilibrium of the GNEP associated to the service provisioning problem by solving a monotone VI. Concerning the solution methods for monotone VI, the projection type methods are among the simplest ones. These methods found on the well known fixed point reformulation of a VI:

$$x^* \in X \text{ solves } VI(X, F) \text{ iff}$$

$$x^* = \text{Proj}_X(x^* - \alpha\,F(x^*)) \text{ for any } \alpha > 0,$$

where Proj_X denotes the orthogonal projection onto X.[3]

[2] F is monotone on X if, for any $x, y \in X$, it holds that $\langle F(x) - F(y), x - y\rangle \ge 0$.

[3] $\text{Proj}_X(z) = argmin\{\|w - z\|, w \in X\} \quad \forall\, z \in \mathbb{R}^n$.

The important feature of these methods is that there is not need of using derivative of F and they do not involve any computation besides the function evaluation and the projection onto X. When the projection is easily computable, as is our case, projection methods are extremely simple and cheap. Among the projection type methods we consider the hyperplane projection method [18], where two projections per iteration are executed. The method is simple and admits a geometric interpretation. Given the current point (x^t, σ^t), first we compute $\text{Proj}_X((x^t, \sigma^t) - \beta_t\,F(x^t, \sigma^t))$ and then we search on the line segment between those points, for (y^t, δ^t) such that the hyperplane

$$\{(x, \sigma) : \langle F(y^t, \delta^t), (x, \sigma) - (y^t, \delta^t)\rangle = 0\}$$

strictly separates (x^t, σ^t) from any solution of the problem. The next iterate (x^{t+1}, σ^{t+1}) is computed by projecting (x^t, σ^t) onto the hyperplane and then onto X. The scheme is formally stated in Figure 1. We set as initial solution of the algorithm x^0 the one obtained by the best reply among SaaS according to the Gauss-Siedel scheme [14] and setting $\sigma^0 = \sigma_k^U$ for all k.

We emphasize that constructing the hyperplane requires a single projection onto the feasible set and employs an Armijo-type line search. During the line search, moreover, no projection onto X are required, but only function evaluations. Regarding the choice of the parameters, following [18] we take $\beta_{t+1} = median(\hat{\beta}, \theta\,\alpha_t\,\beta_t, \tilde{\beta})$, where $\theta > 1$ but not too large, e.g. $\theta = 2$.

The algorithm for the VI solution can be executed efficiently by the IaaS, under the assumption that the SaaSs

STEP 0. Select parameters $\xi \in (0,1)$, $\hat{\beta}, \tilde{\beta}$ s.t. $0 < \hat{\beta} \leq \tilde{\beta}$ and a sequence $\{\beta_t\} \subset [\hat{\beta}, \tilde{\beta}]$. Let $(x^0, \sigma^0) \in X$ and set $t = 0$.

STEP 1. Compute $(\overline{x^t}, \overline{\sigma^t}) = (x^t, \sigma^t) - \beta_t F(x^t, \sigma^t)$

STEP 2. If $(x^t, \sigma^t) = \mathrm{Proj}_X(\overline{x^t}, \overline{\sigma^t})$ then STOP

STEP 3. Set $j(t)$ the minimum $j \geq 0$:

$$\langle (x^t, \sigma^t), F(2^{-j}\mathrm{Proj}_X(\overline{x^t}, \overline{\sigma^t}) + (1-2^{-j})(x^t, \sigma^t))\rangle \geq \frac{\xi}{\beta_t}\|(x^t, \sigma^t) - \mathrm{Proj}_X(\overline{x^t}, \overline{\sigma^t})\|^2.$$

Let $\alpha_t = 2^{-j(t)}$ and

$$(y^t, \delta^t) = \alpha_t \,\mathrm{Proj}_X(\overline{x^t}, \overline{\sigma^t}) + (1-\alpha_t)\,(x^t, \sigma^t),$$

$$\gamma_k = \frac{\langle F(y^t, \delta^t), (x^t, \sigma^t) - (y^t, \delta^t)\rangle}{\|F(y^t, \delta^t)\|^2},$$

$$(x^{t+1}, \sigma^{t+1}) = \mathrm{Proj}_X((x^t, \sigma^t) - \gamma_t F(y^t, \delta^t))$$

STEP 4. Set $t = t + 1$ and go to STEP 1.

Figure 1: Algorithm for VI.

provide to the IaaS also the the incoming workload prediction Λ_k for the next hour. Indeed, for the problem under analysis the SaaS utility function slopes are advertised to the cloud end-users and hence are known also by the IaaS.

6. EXPERIMENTAL RESULTS

The resource management algorithm proposed has been evaluated for a variety of system and workload configurations. The application performance parameters have been varied as considered in the literature (see e.g. [24, 1, 8] and references therein). Cloud providers time unit costs have been varied according to the commercial fees currently adopted [6]. Section 6.1 is devoted to quantitatively analyse the single application case study presented in Section 4. Section 6.2 illustrates the variational equilibria properties on a medium size system. Finally, the scalability of the algorithm reported in Figure 1 is discussed in Section 6.3.

6.1 Single Application Analysis

For the numerical analysis reported in this Section we set $\Lambda = 10$ req/sec, $C = 1$, $\mu = 1$ req/sec, and $m = -1$. Figures 2 and 3 report the plots of the SaaS and IaaS payoff functions Θ_S an Θ_I where we set $\sigma^U = 0.09$\$, and we varied φ and f^U under the assumption that $f^U \leq \Lambda/(C\,\mu)$ (i.e., which corresponds to rows 1-3 of Table 2). Under this hypothesis the SaaS provider is under heavy load conditions since he cannot serve the overall incoming workload by using only his flat resources. The plots show that under the condition $\varphi > \sigma^U$ the payoff functions are constant, while when $\varphi \leq \sigma^U$ Θ_S (Θ_I) increases (decreases) linearly with f^U.

Figures 4 and 5 plot Θ_S and Θ_I as function of σ^U (Table 2 rows 4-6) under light workload conditions for the SaaS ($\Lambda < C\,\mu\, f^U$) where we set $\varphi = 0.03$\$ and $f^U = 50$. In this case the behaviour of the payoff functions changes crossing the marginal value $\frac{-m\,\Lambda}{(C\,\mu\, f^U - \Lambda)^2}$ which with the considered setting is equal to 0.0625\$: When σ^U is greater than the marginal value both Θ_S and Θ_I are constant, while increase for lower values of σ^U. Indeed, the SaaS provider acquires additional on spot instances to profitably serve incoming end

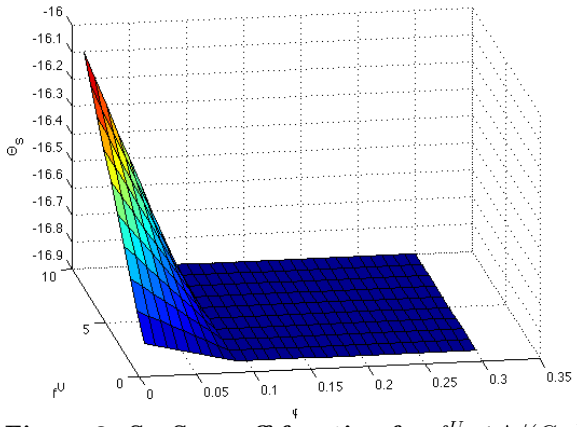

Figure 2: SaaS payoff function for $f^U \leq \Lambda/(C\,\mu)$.

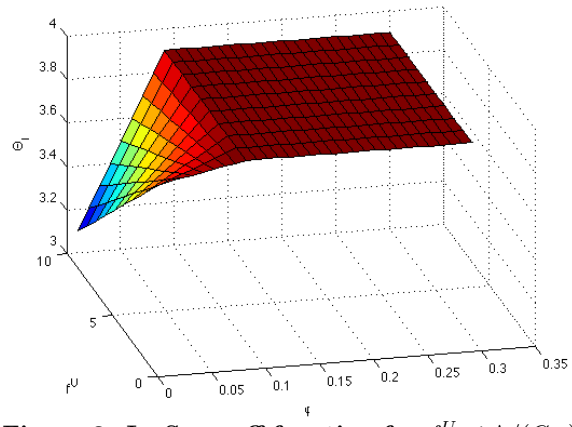

Figure 3: IaaS payoff function for $f^U \leq \Lambda/(C\,\mu)$.

users requests, while the IaaS obtains higher revenues selling on spot instances.

6.2 Equilibria Sharing Analysis

The aim of this Section is to analyse how the on spot VMs are shared among competing SaaS changing the game parameters. The analysis results have been obtained by the algorithm described in Section 5. In particular we considered two SaaS offering five heterogeneous applications each. If not differently stated we set $s^U = 40$, $C = 1$, $\varphi = 0.1$\$, $\delta := 0.11$\$, $f_p^U = 20$ ($p \in \{1, 2\}$), $\Lambda_k = 1$ req/s, $\mu_k = k$ req/s, $m_k = -1$, $\sigma_k^L = 0.03$\$, and $\sigma_k^U = 0.09$ for all $k \in \{1, 10\}$. In the following we will vary one parameter at the time for the first application $k = 1$, while the parameters of the remaining ones will be held fixed. Figures 6-9 show how the number of resources devoted to the first application (in terms of flat, on demand, and on spot instances) and the overall capacity allocated to the remaining classes change as a function of the varying parameter. In particular, in Figure 6 the incoming workload Λ_1 varies between 1 and 14 req/s. As the Figure shows, all of the on spot instances ($s^U = 40$) available at the IaaS are always used but, as the workload increases, they are migrated from the other applications to application 1. In order to profitably sustain the workload, the number of flat instances used is also increased, but on demand VMs are not used until Λ_1 reaches 11 req/s. When Λ_1 is further in-

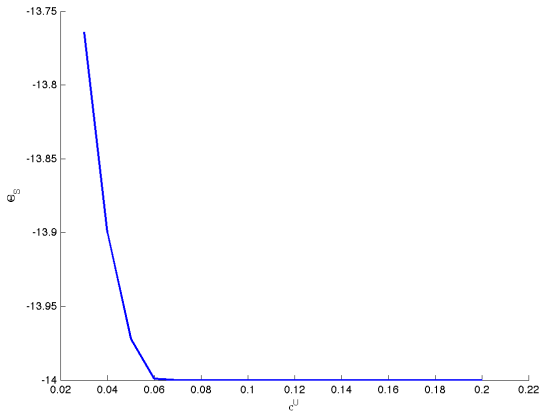

Figure 4: SaaS payoff function for $\Lambda < C\,\mu\,f^U$.

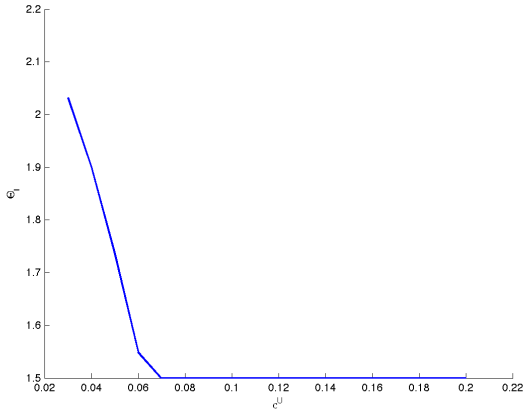

Figure 5: IaaS payoff function for $\Lambda < C\,\mu\,f^U$.

creased the system starts allocating on demand VMs which are more expensive but are needed to serve the incoming requests. In general the resource allocation trends are linear with Λ_1, the discontinuities in the plots are due to the fact that the equilibrium is not unique and hence the same performance and revenues can be obtained with multiple values of (f_k, d_k, s_k).

Figure 7 shows the resource sharing at the equilibrium changing the slope of application 1 utility function (which has been varied in the range $[-15, -1]$). As in the previous analysis, the on spot capacity is migrated to application 1 which becomes more sensible to response time variations and hence requires additional capacity. However, in this case the adoption of on demand instances is never profitable.

Figure 8 analyses how the variational equilibrium changes by varying application 1 maximum service rate (the range $[0.05, 1]$ req/s has been considered). If the maximum service rate increases the service time required to process each application 1 request decreases and the overall capacity required to process application 1 decreases accordingly. Hence, in this case on spot instances are migrated from application 1 to the other classes and on demand instances are used only when application 1 requests are very CPU intensive ($\mu_1 < 0.1$ req/s).

Finally, Figure 9 shows how the equilibrium changes by varying the maximum time unit cost for application 1 (σ_1^U has been varied in the range $[0.1, 1]$\$; we set $\varphi = 0.03$\$, while for the remaining classes $\sigma_k^L = 0.01$\$ and $\sigma_k^U = 0.02$\$). As σ_1^U increases the number of on spot VMs allocated to application 1 decreases since the IaaS set $\sigma_1 = \sigma_1^U$ and the SaaS provider can use in a more cost efficient way the on spot VMs to serve his remaining applications, while application 1 is supported by flat instances. Also in this scenario on demand VMs are never used and the trends are linear. This is very unintuitive, since increasing the maximum time unit cost one is willing to pay for a given application implies that the number of on spot instances devoted to the same application is reduced.

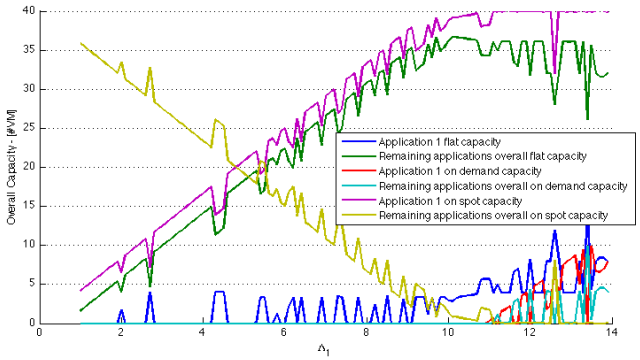

Figure 6: Resource allocation with varying application 1 incoming workload.

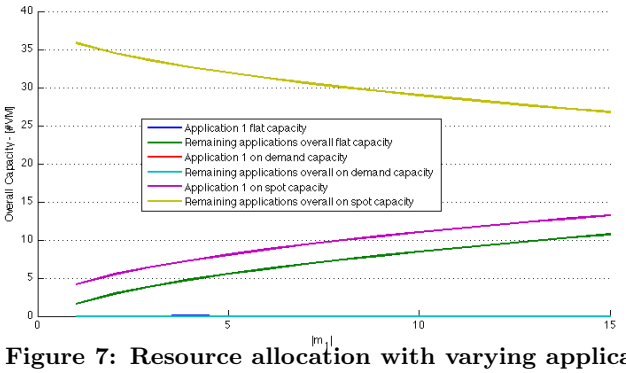

Figure 7: Resource allocation with varying application 1 utility function slope.

6.3 Scalability Analysis

To evaluate the scalability of our resource allocation algorithm we have considered a very large set of randomly generated instances. All tests have been performed on VMWare virtual machine based on Ubuntu 9.10 server running on an Intel Nehalem dual socket quad-core system with 32 GB of RAM. The virtual machine has a physical core dedicated with guaranteed performance and 4 GB of memory reserved. MINOS 5.51 has been use as non linear optimization solver.

The number of SaaS provider has been varied between 10 and 80, the number of applications (evenly shared among SaaSs) between 100 and 800.

The performance parameters of Web applications and infrastructural resources costs have been randomly generated uniformly in the ranges reported in Table 3 as in other literature approaches [24, 1, 8] and according to commercial fees applied by IaaS cloud providers [6].

Table 4 reports, for problem instances of different sizes, the average computational time in seconds as well as the

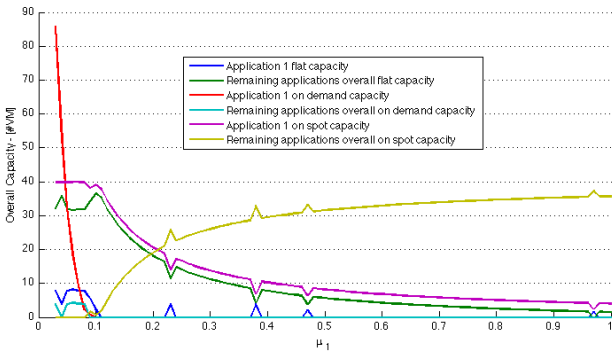

Figure 8: Resource allocation with varying application 1 maximum service rate.

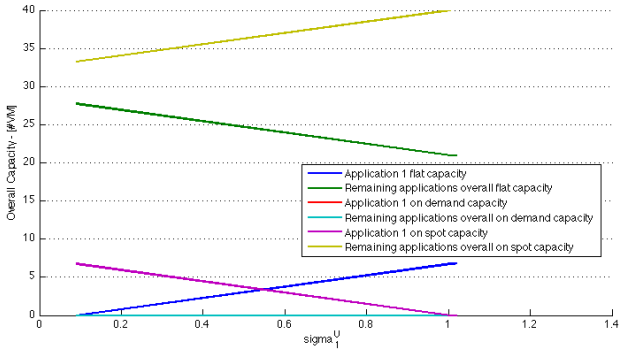

Figure 9: Resource allocation with varying application 1 on spot maximum time unit cost.

average number of iterations performed by the algorithm reported in Figure 1 from the initial best reply solution (the means are computed on ten different runs). Since problems with a size comparable with real systems [27] including thousands of VM instances and hundreds of SaaS providers can be solved in less than one hour, our approach can be used to support the run time management of real cloud infrastructures.

7. RELATED WORK

The recent development of Cloud systems and the rapid growth of the Internet have led to a remarkable development in the use of the Game Theory tools. Problems arising in the ICT industry, such as resource or quality of service allocation problems, pricing, and load shedding, can not be handled with classical optimization approaches. Interaction across different players is non-negligible: Each player can be affected by the actions of all players, not only her own action. In this setting, a natural modelling framework involves seeking an equilibrium, or stable operating point for the system. A survey of different modelling and solution concepts of networking games, as well as a number of different applications in telecommunications and wireless networks, based on Game Theory, can be found in [5].

With respect to telecommunication applications, a rich literature exists which includes solutions for flow and congestion control [3], network routing [4], file allocation [23], load balancing [21], resource allocation [16] and quality of

s^U	[100,1000]	C	[1,3]
Λ_k	[1,100] req/s	μ_k	[1,10] req/s
m_k	[-10,-1] req/s		
φ	[0.03,0.24]$	δ	[0.08,1.24]$
σ_k^L	[0.02,0.08]$	σ_k^U	[0.09,0.30]$

Table 3: Performance parameters and time unit cost ranges.

N-SaaS,N-Appl.	Exe. Time (s)	N-It.
10,100	322	20
20,200	748	32
30,300	1051	47
40,400	1627	51
50,500	1922	37
60,600	2127	42
70,700	2966	53
80,800	3450	38

Table 4: VI algorithm average execution time and number of iterations.

service provisioning [13].

In [17] a Markovian queueing network model is used to derive decentralized flow control mechanisms in computer communication networks with multiple controllers.

In the setting of optimal routing strategies, [22] investigates the existence of Nash equilibria in noncooperative flow control in a general product-form network shared by multiple end-users introduced in [17]. The goal is to study the existence of Nash equilibria for decentralized control schemes. This approach is based on directly proving the existence of a fixed point of the best response correspondence of the underlying game.

In [26] the authors examine the problem of communication delays for two main types of heterogeneous systems: (i) systems where all the nodes have the same processing rates and capabilities but the arrival rate of local jobs at nodes may not be the same, and (ii) systems where different nodes may process jobs at different rates.

In [15] the static load balancing problem in heterogeneous distributed systems is formulated as a noncooperative game among users. Based on the Nash equilibrium concept, the authors derive a distributed load balancing algorithm, whose performance are compared with that of other existing schemes. The main advantages of the proposed approach are the distributed structure, low complexity and optimality of allocation for each user.

Regarding Cloud computing, the use of Game Theory for the resource allocation problem is investigated in [29]. Here, the authors start from a bid proportional auction resource allocation model and propose an incomplete common information model where one bidder does not know how much the others would like to pay for the computing resource. To this end a Bayesian learning mechanism is introduced.

In [2], the authors consider centralized and decentralized load balancing strategies in a system with multiple and heterogeneous processor sharing servers. Each server has an associated service capacity and a holding cost per unit time. The requests arrive as a Poisson process, and the service time of incoming jobs is assumed to be know. For such system, the load balancing problem is investigated.

In [31] the authors propose a pricing mechanism for allocation capacity in a utility computing system among competing end-users requests. The fixed available service capacity is allocated among the different flows proportionally to their monetary bids. The paper studies the resulting equilibrium point, establishes convergence of a best-response algorithm, and bounds the efficiency loss (price of anarchy) of this distributed mechanism.

Differently from our point of view, in [31] the problem of the capacity allocation is considered for a single virtualized server among competing user requests, while in this paper we consider the infrastructure data center at a higher granularity (i.e., VMs).

8. CONCLUSIONS

We proposed a game theory based approach for the run time management of a IaaS provider capacity among multiple competing SaaSs. The model includes infrastructural costs and revenues deriving form cloud end-users which depend on the achieved level of performance of individual requests. Future work will validate of our solution by performing experiments in real cloud environments. Furthermore, a comparison with the heuristic solutions adopted by SaaS and IaaS providers for the run time cloud management will be also performed.

9. REFERENCES

[1] J. M. Almeida, V. A. F. Almeida, D. Ardagna, I. S. Cunha, C. Francalanci, and M. Trubian. Joint admission control and resource allocation in virtualized servers. *J. Parallel Distrib. Comput.*, 70(4):344–362, 2010.

[2] E. Altman, U. Ayesta, and B. Prabhu. Load balancing in processor sharing systems. In *ValueTools '08 Proc.*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2008.

[3] E. Altman and T. Basar. Multi-user rate-based flow control. *IEEE Trans. on Communications*, 46(7):940–949, 1998.

[4] E. Altman, T. Basar, T. Jimenez, and N. Shimkin. Competitive routing in networks with polynomial cost. *IEEE Trans. on Automatic Control*, 47(1):92–96, 2002.

[5] E. Altman, T. Boulogne, R. El-Azouzi, T. Jiménez, and L. Wynter. A survey on networking games in telecommunications. *Comput. Oper. Res.*, 33(2):286–311, 2006.

[6] Amazon Inc. Amazon Elastic Cloud. http://aws.amazon.com/ec2/.

[7] D. Ardagna, B. Panicucci, and M. Passacantando. A Game Theoretic Formulation of the Service Provisioning Problem in Cloud Systems. Politecnico di Milano, Technical Report 2010.22. http://home.dei.polimi.it/ardagna/CloudTechRep.pdf.

[8] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang. Energy-Aware Autonomic Resource Allocation in Multi-tier Virtualized Environments. IEEE Trans. on Services Computing. To appear, available on line.

[9] M. Bennani and D. Menascé. Resource Allocation for Autonomic Data Centers Using Analytic Performance Models. In *IEEE Int'l Conf. Autonomic Computing Proc.*, 2005.

[10] G. Bolch, S. Greiner, H. de Meer, and K. Trivedi. *Queueing Networks and Markov Chains*. J. Wiley, 1998.

[11] E. Cavazzuti, M. Pappalardo, and M. Passacantando. Nash equilibria, variational inequalities, and dynamical systems. *J. of Optimization Theory and Applications*, 114(3):491–506, 2002.

[12] G. Debreu. A social equilibrium existence theorem. *Proc. of the National Academy of Sciences of the USA*, 38:886–893, 1952.

[13] R. El-Azouzi and E. Altman. Constrained traffic equilibrium in routing. *IEEE/ACM Trans. on Automatic Control*, 48(9):1656–1660, 2003.

[14] F. Facchinei and C. Kanzow. Generalized Nash equilibrium problems. *Ann. Oper. Res.*, 175:177–211, 2010.

[15] D. Grosu and A. Chronopoulos. Noncooperative load balancing in distributed systems. *J. Parallel Distrib. Comput.*, 65(9):1022–1034, 2005.

[16] M. Haviv. The aumann-shapely pricing mechanism for allocating congestion costs. *Operations Research Letters*, 29(5):211–215, 2001.

[17] T. Hsiao and A. Lazar. Optimal decentralized flow control of markovian queueing networks with multiple controllers. *Performance Evaluation*, 13(3):181–204, 1991.

[18] A. N. Iusem and B. F. Svaiter. A variant of korpelevich's method for variational inequalities with a new search strategy. *Optimization*, 42(4):309–321, 1997.

[19] J. Hamilton. Using a Market Economy. http://perspectives.mvdirona.com/2010/03/23/UsingAMarketEconomy.aspx.

[20] S. Kakutani. A generalization of Brouwer's fixed point theorem. *Duke Mathematical Journal*, 8:457–459, 1941.

[21] H. Kameda, E. Altman, T. Kozawa, and Y. Hosokawa. Braess-like paradoxes in distributed computer systems. *IEEE Trans. on Automatic Control*, 45(9):1687–1691, 2000.

[22] A. Korilis and A. Lazar. On the existence of equilibria in noncooperative optimal flow control. *J. ACM*, 42(3):584–613, 1995.

[23] J. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Trans. on Computers*, 38(5):705–717, 1989.

[24] D. Kusic, J. O. Kephart, N. Kandasamy, and G. Jiang. Power and Performance Management of Virtualized Computing Environments Via Lookahead Control. In *ICAC 2008 Proc.*, 2008.

[25] D. A. Menascé and V. Dubey. Utility-based QoS Brokering in Service Oriented Architectures. In *IEEE ICWS Proc.*, pages 422–430, 2007.

[26] R. Mirchandaney, D. Towsley, and J. Stankovic. Adaptive load sharing in heterogeneous distributed systems. *J. Parallel Distrib. Comput.*, 9(4):331–346, 1990.

[27] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the electric bill for internet-scale systems. In *SIGCOMM '09 Proc.*, pages 123–134, New York, NY, USA, 2009. ACM.

[28] SpotHistory.com. Spot Instance Price History Graphs. http://www.spothistory.com/.

[29] F. Teng and F. Magoules. A new game theoretical resource allocation algorithm for cloud computing. In *Advances in Grid and Pervasive Computing*, pages 321–330, 2010.

[30] B. Urgaonkar and P. Shenoy. Sharc: Managing CPU and Network Bandwidth in Shared Clusters. *IEEE Trans. on Parallel and Distr. Systems*, 15(1):2–17, 2004.

[31] B. Yolken and N. Bambos. Game based capacity allocation for utility computing environments. In *ValueTools '08 Proc.*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2008.