Optimal Pricing and Service Provisioning Strategies in Cloud Systems: A Stackelberg Game Approach

Valerio Di Valerio
University of Roma "Tor Vergata"
di.valerio@ing.uniroma2.it

Valeria Cardellini
University of Roma "Tor Vergata"
cardellini@ing.uniroma2.it

Francesco Lo Presti
University of Rome "Tor Vergata"
lopresti@info.uniroma2.it

***Abstract*—In this paper we consider several Software as a Service (SaaS) providers, that offer a set of applications using the Cloud facilities provided by an Infrastructure as a Service (IaaS) provider. We assume that the IaaS provider offers a *pay only what you use* scheme similar to the Amazon EC2 service, comprising flat, on demand, and spot virtual machine instances. We propose a two stage provisioning scheme. In the first stage, the SaaS providers determine the number of required flat and on demand instances by means of standard optimization techniques. In the second stage the SaaS providers compete, by bidding for the spot instances which are instantiated using the unused IaaS capacity. We assume that the SaaS providers want to maximize a suitable utility function which accounts for both the QoS delivered to their users and the associated cost. The IaaS provider, on the other hand, wants to maximize his revenue by determining the spot prices given the SaaS bids. We model the second stage as a Stackelberg game, and we compute its equilibrium price and allocation strategy by solving a Mathematical Program with Equilibrium Constraints (MPEC) problem. Through numerical evaluation we study the equilibrium solutions as function of the system parameters.**

I. Introduction

Cloud computing has recently been experiencing a high rate of growth, mainly due to the ability of realizing highly scalable and reliable infrastructures for running software applications in an efficient and cost-effective way. Industrial companies and research communities have started using commercially available infrastructures provided by Infrastructure as a Service (IaaS) providers to run their applications, that can scale up or down as demand changes by allocating or deallocating virtual computing and storage resources almost instantaneously. In their turn, customers of IaaS providers can rapidly offer their innovative applications, thus becoming Software as a Service (SaaS) providers, but without needing to own and maintain development or production infrastructures.

Among the different methods to deliver Cloud services, an IaaS provider can sell resources in the form of Virtual Machine (VM) instances to customers, that generally rent the resources by using a pay-as-you-go model on a per-hour basis at a fixed price (also called *on demand* price). When resource utilization can be planned in advance, IaaS customers can also reserve *flat* resources in advance, paying a long-term reservation fee plus a per-hour price which depends on the effective resource usage and is lower than the on demand price. In order to achieve high utilization in data centers that are often under-utilized, IaaS providers can also sell their spare capacity in form of *spot* instances by organizing an auction where customers bid, providing a maximum per-hour price they are willing to pay. On the basis of the bids and his spare capacity, the IaaS provider sets the spot instances price. For IaaS customers spot instances represent an attractive and cost-effective solution to deal with unexpected load spikes and run compute-intensive applications but at the risk of a lower reliability than flat and on demand instances, since the IaaS provider can revoke spot instances without notice due to price and demand fluctuations.For example, Amazon's Elastic Cloud Computing (EC2) service offers three types of VM instances (i.e., flat, on demand, and spot VMs) [1], with different pricing and reliability.

From an IaaS perspective, selling resources to multiple customers requires to determine efficient service provisioning and pricing strategies, in order to maximize the IaaS resources usage and the provider profit and to satisfy the customers. On the other hand, the customers compete among them to acquire the resources and are interested in saving money. Developing such strategies in the Cloud environment is a challenging task, mostly because the Cloud environment is inherently competitive and dynamic. In this context, game theoretic methods can help to analytically model and understand the service provisioning and pricing problem and to devise adequate strategies. Game theory has been already successfully applied to networking problems like Internet congestion control and pricing, e.g., [2] and in the Cloud computing environment, e.g., [3], [4], [5]. In most works, the solution concept of Nash equilibrium has been extensively applied (a set of players' strategies is a Nash equilibrium if no player one can improve his revenue by changing its strategy unilaterally, i.e., while the other players keep theirs unchanged).

In this paper, we consider a Cloud scenario where an IaaS provider sells his resources to several SaaS providers, offering flat, on demand, and spot VM instances. In their turn, SaaS providers offer to their end users Web applications with Quality of Service (QoS) guarantees, using the IaaS facilities to host and run the provided applications.

Revenues and penalties of each SaaS provider depend on the provisioning of an adequate performance level, which is specified in a Service Level Agreement (SLA) contract that each SaaS stipulates with his end users. Therefore, each SaaS provider has to face the problem of determining the optimal number of VMs to satisfy the SLA with his end users while maximizing his revenue. However, a proper solution cannot be accomplished in isolation, since the SaaS providers compete among them and bid to acquire the spot IaaS resources. On the other hand, the IaaS provider aims to maximize his revenue and therefore wants to properly choose the price of his resources. In other words, each player strategy influences what the other players do.

To model this conflicting situation we recur to a Stackelberg game [6]. In this class of games, one player (i.e., the leader, in our case the IaaS provider) moves first and commits his strategy to the remaining players (i.e., the followers, for us the SaaS providers), that consider the action chosen by the leader before acting simultaneously to choose their own strategy in a selfish way through a standard Nash game. Stackelberg games are commonly used to model attacker-defender scenarios, e.g. [7]. For the considered Cloud scenario, the adoption of a leader-follower strategy sounds feasible, since we can reasonably assume that the IaaS provider fixes the price before the SaaS providers compete to acquire the VM resources. Furthermore, a Stackelberg game help us to devise a revenue-maximizing pricing scheme for the IaaS provider.

The main contributions of the present work are as follows. 1) We devise a *two-stage* service provisioning and pricing strategy. In the *first stage*, each SaaS provider independently determines the optimal number of flat and on demand VMs (which have a fixed price) in such a way to guarantee the performance level offered in the SLA to his end users while maximizing his profit. In the *second stage*, the SaaS providers bid and compete for the unused IaaS provider capacity. The goal of each SaaS provider is to determine the number of spot instances to allocate which maximizes his profit, given the number of flat and on demand instances bought in the first stage. The goal of the IaaS provider is to determine the price of the spot VMs in order to maximize his profit. While the first stage involves the solution of standard optimization problems, the second stage requires to compute the equilibria of the SaaS/IaaS Stackelberg game on spot instances. 2) We address the solution of the challenging Stackelberg game by solving a suitable Mathematical Program with Equilibrium Constraint (MPEC) problem. 3) We study through numerical investigation the behavior of the proposed provisioning and pricing strategy under different workload and bidding configurations and compare it with the policy proposed in [3]. Using our proposed strategy, the IaaS provider can set a price of the spot instances lower than the maximum price in the bids in order to incentivize SaaS providers to buy more instances, thus increasing his revenue with a higher volume of sold instances.

The rest of this paper is organized as follows. In Section II we define the system model.In Section III we define the two-stage service provisioning and pricing strategy. We discuss the solution method of the Stackelberg game that arises from our problem formulation in Section IV. In Section V we analyze through numerical experiments the behavior of the proposed strategy and compare its results to those achieved by the formulation in [3]. In Section VI we discuss related works. Finally, in Section VII we conclude the paper.

II. System Model

We consider a set $\mathcal{U}$ of SaaS providers that offer a set of Web applications $\mathcal{A}_u$, $u \in \mathcal{U}$, using the cloud facilities offered by a IaaS provider. We assume that each application $k \in \mathcal{A}_u$ is characterized by a SLA which stipulates the application QoS levels, *i.e.*, response time, and the associated cost/penalty for its use.

Web applications are hosted on virtual machines instantiated by the IaaS provider. For the sake of simplicity, we assume that the IaaS provider offers only one type of VMs, *i.e.*, all the VMs have the same capacity. Each VM hosts only one application; on the other hand, each application can be distributed on multiple VMs and in that case we assume the workload to be evenly split among them.

The IaaS provider manages an infrastructure which can provide to his users up to $\mathcal{S}$ VMs which are offered to users as flat, on demand, and/or spot instances. Flat instances are characterized by one-time payment plus a payment of φ unit per hour of actual use. On demand instances have no one-time payment and are charged at a price δ, which we assume to be strictly larger than φ. Spot instances are charged at a price $\sigma_{u,k}$, which we assume it may vary from SaaS provider to SaaS provider and from application to application, and which depends on the users bids and competition for the unused resources and the IaaS provider optimal pricing strategy.

Given the number of flat $f_{u,k}$, on demand $d_{u,k}$, and spot instances $s_{u,k}$ and their prices $\sigma_{u,k}$ allocated to application $k \in \mathcal{A}_u$, $u \in \mathcal{U}$, the associated per-hour IaaS revenue is:

$$\Theta_I = \sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{A}_k} (\varphi f_{u,k} + \delta d_{u,k} + \sigma_{u,k} s_{u,k}) \qquad (1)$$

Each SaaS provider determines for each application k the number of flat $f_{u,k}$, on demand $d_{u,k}$, and spot $s_{u,k}$ VMs to be allocated which maximizes his revenue, given the predicted arrival rate $\Lambda_{u,k}$ and the application SLA.

We assume that SLA takes the form of an upper bound on the application response time $R_{u,k}^{max}$. The SLA also specifies the user per-request cost $\mathcal{C}_{u,k} = C_{u,k}(1 - \frac{R_{u,k}}{R_{u,k}^{max}})$, which we assume to be a linear function of the application response time $R_{u,k}$. We denote with $m_{u,k} = -\frac{C_{u,k}}{R_{u,k}^{max}}$ the slope of this function. Observe that the application cost is a decreasing function of the response time and it becomes

negative (hence, the SaaS provider incurs a penalty) when $R_{u,k} > R^{max}_{u,k}$. We adopt such a simple model since linear costs allow to implement a soft constraint on the response time, which enables the SaaS provider to trade-off revenues and infrastructural costs [4].

We model each Web application hosted on a VM as an M/G/1/PS queue with an application dependent service rate $\mu_{u,k}$. Under the assumption of perfect load sharing among multiple VMs assigned to the same application, the application k average response time is given by:

$$E[R_{u,k}] = \frac{f_{u,k} + d_{u,k} + s_{u,k}}{\mu_{u,k}(f_{u,k} + d_{u,k} + s_{u,k}) - \Lambda_{u,k}}$$

provided the stability condition $\frac{\Lambda_{u,k}}{\mu_{u,k}(f_{u,k}+d_{u,k}+s_{u,k})} < 1$ holds. Taking into account the infrastructural per hour cost for the allocated VMs, the per-hour SaaS profit is:

$$\Theta_u = \sum_{k \in \mathcal{A}_u} \Lambda_{u,k} \mathcal{C}_{u,k} - \sum_{k \in \mathcal{A}_u} (\varphi f_{u,k} + \delta d_{u,k} + \sigma_{u,k} s_{u,k}) \tag{2}$$

where the first term is the sum of the average per application revenues $\Lambda_{u,k}\mathcal{C}_{u,k} = \Lambda_{u,k} C_{u,k}(1 - \frac{E[R_{u,k}]}{R^{max}_{u,k}})$ and the remaining terms the VMs costs.

III. Service Provisioning: a Stackelberg Game Approach

We assume that SaaS providers every hour allocate and deallocate VMs relying on a prediction of the next-hour future workload. In this paper we consider a two-stage allocation procedure. In the first stage, each SaaS provider independently determines, for each offered application, the number of flat and on demand instances[1] which guarantee the performance level defined in the SLA to its prospective users and maximize his profit. In the second stage, the IaaS provider sells the unused capacity as spot instances. The SaaS providers compete for these additional resources by submitting, to the IaaS provider, a bid specifying the maximum per VM price they are willing to pay. The IaaS provider, given his residual capacity and the submitted bids, determines the spot instances prices which maximize his profit.

A. First Stage: Flat and on Demand VM Allocation

In the first stage, each SaaS provider independently determines the optimal number of flat and on demand VMs necessary to sustain the predicted load for the next hour which maximizes his profit. We assume that the IaaS provider always has enough resources to accomodate all flat and on demand instances the users may require. For each SaaS provider $u \in \mathcal{U}$ we have the following optimization problem:

$$\textbf{max } \Theta_u \tag{3}$$

$$\textbf{subject to: } \sum_{k \in \mathcal{A}_u} f_{u,k} \leq f_u \tag{4}$$

$$\frac{\Lambda_{u,k}}{\mu_{u,k}(f_{u,k} + d_{u,k})} \leq U^{max}_u, \quad \forall k \in \mathcal{A}_u \tag{5}$$

$$f_{u,k}, d_{u,k} \geq 0, \quad \forall k \in \mathcal{A}_u \tag{6}$$

Constraint (4) ensures that the flat instances allocated to SaaS provider u are less than or equal to the number of reserved ones f_u. Constraint (5) guarantees that resources are not saturated, in particular that their utilization is less than a threshold U^{max}_u. As in [3], [4], we do not impose to the variables to be integers as in reality they are, because the problem would have been much more difficult to solve (NP-hard). Therefore, throughout the paper we deal with a relaxation of the real problem. We believe that, nonetheless, our findings apply to the actual problem as well.

B. Second Stage: Spot Instances Allocation

In the second stage, the SaaS providers compete for the unused IaaS provider resources made available via a bidding mechanism. The idea is that the SaaS providers can increase their revenues by accessing additional resources, while the IaaS provider can make profit from the otherwise unsold resources. We assume that, nevertheless, the Iaas provider sets a minimum price σ^L for spot VMs.

The goal of each SaaS provider is to determine the number of spot instances to allocate which maximizes his profit given the number of flat and on demand instances bought in the first stage. In the second stage, we assume that each SaaS provider u specifies $\sigma^U_{u,k}$, the maximum time unit cost for spot VMs for application k he is willing to pay (see the IaaS problem), and solves the following optimization problem:

Problem SaaS$_{OPT}$

$$\textbf{max } \Theta_u = \textbf{max} \sum_{k \in \mathcal{A}_u} \frac{m_{u,k}\Lambda_{u,k}(\bar{f}_{u,k} + \bar{d}_{u,k} + s_{u,k})}{\mu_{u,k}(\bar{f}_{u,k} + \bar{d}_{u,k} + s_{u,k}) - \Lambda_{u,k}} - s_{u,k}\sigma_{u,k}$$

$$\textbf{subject to: } \sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{A}_u} s_{u,k} \leq s^U \tag{7}$$

$$s_{u,k} \geq 0, \quad \forall k \in \mathcal{A}_u \tag{8}$$

where $\bar{f}_{u,k}$ and $\bar{d}_{u,k}$ represents the number of flat and on demand instances already bought and $s^U = \mathcal{S} - \sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{A}_u} (\bar{f}_{u,k} + \bar{d}_{u,k})$ the amount of unused IaaS capacity, being $\mathcal{S}$ the total amount of VMs the IaaS provider manages and $\sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{A}_u} (\bar{f}_{u,k} + \bar{d}_{u,k})$ the amount of VMs instantiated after the first stage. In this optimization problem, constraint (7) ensures that the total number of spot VMs allocated to the SaaS providers are less than or equal to the ones available at the IaaS provider. Note that, differently from the flat and on demand provisioning

[1] In case of flat instances, this number represents the number of allocated flat instances, among the *already* reserved ones, which will be used to offer the applications.

problem the SaaS providers solve in the first stage, we now have a constraint which involves the decision variables of all the SaaS providers and which is parametrized by the spot instances price $\sigma_{u,k}$, which is a IaaS decision variable.

The goal of the IaaS provider is to determine the cost $\sigma_{u,k}$ of the spot VMs for each application k of every SaaS provider u in order to maximize his profit. The IaaS provider optimization problem is:

$$\begin{aligned} &\textbf{Problem IaaS}_{OPT} \\ &\textbf{max } \Theta_I = \textbf{max} \sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{A}_u} s_{u,k}\sigma_{u,k} \\ &\textbf{subject to: } \sigma^L \leq \sigma_{u,k} \leq \sigma^U_{u,k}, \quad \forall u \in \mathcal{U},\ \forall k \in \mathcal{A}_u \end{aligned} \tag{9}$$

In this setting, the decisions of the SaaS providers and the IaaS provider depend mutually from each other. Indeed, the objective function of the IaaS provider depends on $s_{u,k}$, the decision variables of the SaaS providers, while the objective function of each SaaS provider depends on $\sigma_{u,k}$, the prices of the spot instances, which are the decision variables of the IaaS provider. Moreover, the decision of each SaaS provider depends also on what the others providers do, since constraint (7) couples the variables of all the SaaS providers.

We model such a conflicting situation as a Stackelberg game [6]. Stackelberg games are a particular type of non-cooperative game whereby one player (the leader) takes its decision before the other players (the followers). Given the leader decision, the followers then simultaneously take their own decision. The leader can thus take advantage of the fact that the followers react to its decisions, which leads to a follower *subgame* equilibrium (if any exists), and drive the system to its own optimum. In our model, the IaaS provider acts as a leader by deciding the spot instances prices $\sigma_{u,k}$. The SaaS providers act as followers which must decide the number $s_{u,k}$ of spot instance to buy. Given the spot prices, the SaaS providers thus compete (the SaaS subgame) for the shared pool of available instanced s^U.

IV. Solution Method

The first stage of our provisioning strategy involves the solution of a set of independent convex optimization problems which can be addressed by means of standard techniques. The second stage requires the computation of the *equilibria* of the SaaS/IaaS spot instance Stackelberg game. This is a challenging problem for which no general solution exists. In this paper, we take advantage of the structure and properties of the SaaS provider subgame, and we compute the Stackelberg equilibria by solving a suitable *Mathematical Programs with Equilibrium Constraint* (MPEC). In this section, we first study the SaaS providers subgame and establish some important game properties; then, we present an algorithm to compute an equilibrium of the Stackelberg game.

We denote with $s_u = (s_{u,k})_{k \in A_u}$ the strategy of a SaaS provider $u \in \mathcal{U}$ and with $\sigma = (\sigma_{u,k})_{u=1}^N$ the strategy of the IaaS provider, where $N = |\mathcal{U}|$. Furthermore, we indicate with $s = (s_u)_{u=1}^N$ the set of strategies of all the SaaS providers and with s^{-u} the set of the strategies of all the SaaS providers except the SaaS provider u. For the sake of simplicity, we also rewrite the SaaS providers optimization problem $\mathbf{SaaS}_{OPT}$ in compact form as follows:

$$\begin{aligned} &\textbf{max } \Theta_u(s;\sigma) \\ &\textbf{subject to: } g_u(s) \leq 0 \end{aligned}$$

where $g_u(s)$ is a vector function that represents the problem constraints. We define with $K_u = \{s \ | g_u(s) \leq 0\}$ the SaaS provider u feasible strategies set and we further define $\Omega = \left\{s \ \middle| \sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{A}_u} s_{u,k} \leq s^U\right\}$. Hence, we can define the vector function $g(s) = (g_u(s))_{u=1}^N$ that represents the set $K = (K_1 \times K_2 \dots K_N) \cap \Omega$ of the feasible strategies of all the players. Note that the set K is compact, since each variable $s_{u,k}$ is bounded by s^U.

A. SaaS Providers Subgame

We first consider the followers subgame, i.e., the SaaS providers competition that arises once the IaaS provider fixes his strategy (the spot prices σ). Since the SaaS providers act simultaneously, this subgame can be modeled as a Generalized Nash game [8]. Generalized Nash Equilibrium Problems (GNEPs) differ from Nash Equilibrium Problems (NEPs), in that, while in a NEP only the players objective functions depend on the other players strategies, in a GNEP both the objective functions and the strategy sets depend on the other players strategies. In our problem, the dependence of each player strategy set on the other players strategies is represented by constraint (7) which includes all SaaS providers decision variables $\sigma_{u,k}$. Specifically, our problem is a *Jointly Convex* GNEP [8]. Convexity follows from concavity of the objective function of each SaaS provider in his own decision variable and the convexity of the strategy set; furthermore, it is jointly convex because the constraint involving all players variables is the same for all players.

The solution of jointly convex GNEP problems can be computed by solving a proper *variational inequality* (VI)[2]. In particular, under the condition that the objective function of each player is continuously differentiable, every solution of the $VI(K, F(s;\sigma))$, where $F(s;\sigma) = -[(\nabla_{s_u}\Theta_u(s;\sigma)_{u=1}^N)$, is also an equilibrium of the GNEP [8]. Such equilibrium is known as *variational equilibrium*. In general, a GNEP has multiple or even infinite equilibria, and not all of them are also a solution of the VI. However, the variational equilibrium is more "socially stable" than the other equilibrium of a GNEP and therefore it represents a valuable target for an algorithm [8]. We now establish two key properties of the $VI(K, F(x;\sigma))$, namely that the

[2]Given a subset K of $\Re^n$ and a function $F : K \to \Re^n$, the VI problem, denoted by $VI(K, F)$, consists in finding a point $s^* \in K$ such that $(s - s^*)^T F(x^*) \geq 0 \quad \forall s \in F$.

function F is strongly monotone[3] and the existence of the generalized Nash equilibrium of the followers subgame.

Theorem 1: Function $F(s;\sigma) = -[(\nabla_{s_u}\Theta_u(s;\sigma)_{u=1}^N)$ is strongly monotone.

The theorem proof can be found in [9].

Theorem 2: There exists at least one generalized Nash equilibrium of the followers subgame.

Proof: Such existence directly follows from the strong monotonicity of $F(s;\sigma)$ [10]. ■

B. Stackelberg Game as MPEC

We now turn our attention to the IaaS (leader) problem of determining the optimal pricing strategy. We solve the Stackelberg game using a *Mathematical Programs with Equilibrium Constraint* (MPEC) [11]. An MPEC is an optimization problem whose constraints include variational inequalities. The MPEC arising from our IaaS optimization takes the following form:

$$\textbf{max} \sum_{u\in\mathcal{U}}\sum_{k\in\mathcal{A}_u} s_{u,k}\sigma_{u,k}$$

$$\textbf{subject to: } \sigma^L \leq \sigma_{u,k} \leq \sigma^U_{u,k} \quad \forall u\ \forall k \tag{10}$$

$$F(\sigma) - \nabla_s g(s)\lambda = 0 \tag{11}$$

$$g(s) \leq 0,\ \lambda \geq 0,\ \lambda^T g(s) = 0 \tag{12}$$

where the constraints (11)-(12) are Karush Kuhn Tucker (KKT) conditions [12] associated to the SaaS subgame variational inequality ($\lambda \in \Re^l$ is the vector of Lagrangian multiplier, with l the number of constraints that define K). Observe that this basically the IaaS optimization problem $\textbf{IaaS}_{OPT}$ with the additional constraint that s must be a Generalized Nash equilibrium of the SaaS subgame.

The MPEC cannot be directly solved because the constraints do not satisfy any standard constraints qualification and the complementary-type constraints in (12) are very complicated and difficult to handle. Following [11], we consider a sequence of smooth and regular problems, obtained by perturbing the original problem, the solutions of which converge to a solution of the original problem, under the assumption that function $F(s;\sigma)$ is strongly monotone. Specifically, we consider the pertubated problem $\mathbf{P}(\mu)$ with parameter μ:

$$\textbf{Problem } \mathbf{P}(\mu)$$

$$\textbf{max} \sum_{u\in\mathcal{U}}\sum_{k\in\mathcal{A}_u} s_{u,k}\sigma_{u,k}$$

$$\textbf{subject to: } \sigma^L \leq \sigma_{u,k} \leq \sigma^U_{u,k},\ \forall u\ \forall k \tag{13}$$

$$F(\sigma) - \nabla_s g(s)\lambda = 0 \tag{14}$$

$$g(s) + z = 0 \tag{15}$$

$$\sqrt{(z-\lambda)^2 + 4\mu^2} - (z+\lambda) = 0 \tag{16}$$

[3] F is strongly monotone on K if there exists a constant $c > 0$ such that for all pairs $s, y \in K\ (s-y)^T(F(s)-F(y)) > c\| s-y\|^2$

where $z \in \Re^l$ is an auxiliary variable. In $\mathbf{P}(\mu)$, constraint (15) and (16) replace constraints (12). It is easy to realize that $\mathbf{P}(\mu)$ corresponds to the original problem when $\mu = 0$[4]. We refer the reader to [11] for further details. Problem $\mathbf{P}(\mu)$, $\mu \neq 0$, is a smooth regular problem which

Algorithm 1 Algorithm S [11]

Let $\{\mu^k\}$, $\mu_k \neq 0$, be a sequence with $\lim_{k\to\infty}\mu^k = 0$.
Choose $w^0 = (\sigma^0, x^0, z^0, \lambda^0) \in \Re^{3N+l+l}$, and set $k = 1$
while $||e|| > \epsilon$ **do**
  Find a stationary point w^k of $P(\mu^k)$
  $e = w^k - w^{k-1}$
  $k = k + 1$
end while

can be solved using standard optimization tools. Let $\sigma^*(\mu)$ denote a stationary point of $\mathbf{P}(\mu)$. From [11], we have that $\sigma^*(\mu)$ converges to a stationary point of the $\textbf{IaaS}_{OPT}$ as $\mu \to 0$. To compute a solution we use Algorithm S in [11] (see Algorithm 1), which solves a sequence of problems $P(\mu)$. The algorithm stops when the Euclidean distance between two successive iterations is lower than a suitable threshold ϵ. We verified that in practice the algorithm converges very quickly. In our experiments, the algorithm converged in no more than 3 iterations using $\epsilon = 10^{-4}$.

The proposed algorithm can be executed by the IaaS provider under the assumption that each SaaS provider supplies also the incoming workload prediction $\Lambda_{u,k}$ and the function slopes $m_{u,k}$. Function slopes, in particular, are publicly available because they are advertised by the SaaS providers to their end-users.

V. Experimental Results

We now investigate through numerical experiments the behavior of the proposed provisioning and pricing strategy. We first compute the system equilibria in different scenarios and study the VMs allocation and the associated spot instances prices under different workload and bidding configurations. Then, we compare our strategy with that in [3].

For the analysis, we implemented the algorithms in Section IV as well as those in [3] in MATLAB. For the solution of the MPEC problem via Algorithm S, the parameter μ is initially set to 0.0001 and reduced by a factor of 100 at each iteration and the stopping parameter ϵ is set to 10^{-4}.

A. Provisioning and Pricing Strategies Analysis

We consider one IaaS provider which sells his resources to ten SaaS providers and assume that each SaaS provider offers only one Web service. If not differently stated, we set $\mathcal{S} = 160$, $\varphi = 0.24\$$, $\delta = 1.24\$$, $f_u = 4$, $\mu_{u,1} = 10$ req/s,

[4] Observe that for $\mu = 0$, if $s \in SOL(\sigma)$ is a solution of the original problem then either $g(s) < 0$ and $\lambda = 0$, in which case constraint (16) reduces to $\sqrt{(z)^2} - z = 0$ or $g(s) = 0$, which implies that $z = 0$, and (16) reduces to $\sqrt{(-\lambda)^2} - \lambda = 0$.

$\Lambda_{u,1}$ (req/s)	20	40	60	80
flat	4	4	4	4
on demand	0	3.59	7.38	11.18
spot	2	3.59	4.61	0.81
spot price (\$)	0.25	0.31	0.36	0.5
spot available	120	80	46.1	8.1
spot unsold	100	44.1	0	0

Table I
VMs ALLOCATION AND SPOT PRICE WITH RESPECT TO SAAS PROVIDERS PREDICTED LOAD IN HOMOGENEOUS SCENARIO

	$\Lambda_{u,1}$ (req/s)		$\sigma_{u,1}$ (\$)	
	20	80	0.5	0.3
flat	4	4	4	4
on demand	0	11.18	7.38	7.38
spot	2	7.18	3.65	5.57
spot price (\$)	0.25	0.31	0.43	0.3
spot available	64		46.1	
spot unsold	18.1		0	

Table II
VMs ALLOCATION AND SPOT PRICES FOR SAAS PROVIDERS IN HETEROGENEOUS SCENARIOS

$m_{u,1} = -1$, $U_u^{max} = 0.9$, $\sigma^L = 0.15\$$ and $\sigma_{u,1}^U = 0.5\$$, for all $u \in \{1, \ldots, 10\}$. These parameters correspond to those adopted in [3] for the sake of comparison.

We first consider a homogeneous scenario where the SaaS providers parameters are set as described above. In this scenario, by symmetry, all the SaaS providers obtain the same number of VMs and the same price for spot instances. The results are summarized in Table I for different values of the SaaS providers predicted load $\Lambda_{u,1}$. In the first stage, the SaaS providers determine the amount of flat and on demand instances: in this example, flat instances are always used up to the maximum value $f_u = 4$, independently from the load $\Lambda_{u,1}$; on the other hand, the number of bought on demand instances grows from 0 to 11.18, because more instances are needed to satisfy the QoS constraints when the predicted load increases. This is reflected in the amount of unsold resources after the first stage, which decreases from 120 VMs when $\Lambda_{u,1} = 20$ down to 8.1 VMs when $\Lambda_{u,1} = 80$. In the second stage, the SaaS providers compete for the unsold IaaS capacity. When the demand is high ($\Lambda_{u,1} = 80$), the IaaS provider is able to sell the small fraction of unused resources at the maximum price $\sigma_{u,1}^U = 5\$$. As the demand decreases, the IaaS provider decreases the spot price, to sell more VMs and to increase his revenue. Observe that, when the demand is low ($\Lambda_{u,1} \leq 40$) some capacity remains unsold even after the second stage, because the SaaS providers reach equilibrium between the cost charged on the users (which is a function of the response time) and the cost of additional VMs. At the same time, the IaaS provider has no incentive to further reduce the price. Only when resources are scarce, the IaaS provider maximizes his revenue by charging the maximum price $\sigma_{u,k}^U$; otherwise, it is more profitable for the IaaS provider to lower the spot prices thus selling additional VMs.

We now turn our attention to the provisioning and pricing solutions in heterogeneous scenarios. For the sake of simplicity, we consider only two classes of users. We first consider the case where the SaaS providers have different predicted load. In the second column of Table II we show the results when half of the SaaS providers have a predicted load $\Lambda_{u,1} = 20$ req/s and the other half a predicted load $\Lambda_{u,1} = 80$ req/s. Observe that the number of purchased flat and on demand instances is the same as in the previous homogeneous example. This can be explained by observing that in the first stage the SaaS providers independently determine the amount of flat and on demand instances to buy (under the implicit assumption that the IaaS provider has enough resources to allocate all the requested flat and on demand instances) and the final allocation only depends on the provider parameters. Looking at the spot allocation and price, instead, we observe that while SaaS providers with the lower load are charged a relatively low price and buy only a small number of spot VMs, SaaS providers with the higher load are charged a higher price and buy more spot VMs. We observe that also in this case, at equilibrium, the IaaS provider maximizes his profit by charging less than the maximum price so that overall, the lower per VM profit is compensated by the higher volume of sold instances.

The third column of Table II shows the results when we consider two classes of SaaS providers with different maximum price: five providers have $\sigma_{u,1}^U$ = 0.5\$ while the rest have $\sigma_{u,1}^U$ = 0.3\$. We assume that the SaaS providers have the same predicted load $\Lambda_{u,1}$ = 60 req/s. Because all the providers have the same parameters, except for the bid which impacts only on the second stage, they buy the same number of flat and on demand instances. However, as expected, given the different maximum price, the price and and number of purchased spot instances differs for the two classes of users. It is interesting to compare this to the first scenario we considered, where all the SaaS providers had the same maximum price $\sigma_{u,1}^U$ = 0.5\$; in that case, when the load was $\Lambda_{u,1}$ = 60 req/s the optimal strategy for the IaaS provider was to set $\sigma_{u,1}$=0.36\$ for all the users. In this scenario, however, the maximum price for half of the SaaS providers is only 0.3\$. So, it is no surprise that the optimal pricing strategy for the IaaS provider is to set $\sigma_{u,1}$=0.3\$ for these providers, and a higher price, $\sigma_{u,1}$=0.43\$, for the others. In other words, the revenue that is lost by the IaaS provider by selling spot instances at 0.3\$ is recovered by increasing the spot price for those who bid 0.5\$.

B. Comparison with the Provisioning Scheme in [3]

We now compare the proposed provisioning and pricing scheme with the one presented in [3]. Ardagna et al. study

a provisioning problem very similar to that in this paper. Differently from our two stage allocation strategy, they consider a one stage provisioning problem, where, at the same time: the SaaS providers determine the number of flat, on demand and spot instances to buy as to maximize their revenue given the service SLA; the IaaS provider determines the spot instances price $\sigma_{u,k}$ as to maximize his profit, taking into account that each SaaS provider is characterized by a maximum cost $\sigma^U_{u,k}$ for spot instance per hour. The conflicting situation is modeled as a GNEP and the service provisioning and pricing policy are derived from the game equilibrium. In particular, given the specific problem structure, they show that the dominant IaaS provider strategy consists in setting $\sigma_{u,k} = \sigma^U_{u,k}$, *i.e.*, in charging each SaaS user always the maximum price. They present a general solution method and evaluate the proposed scheme under different scenarios.

Comparing the two approaches, we expect that in our scheme the SaaS providers are more likely to buy a higher number of flat and especially of on demand instances, which are more expensive (but also more reliable as the IaaS provider cannot terminate them) since these type of instances are allocated first. This should result in higher cost for the SaaS provider and higher profit for the IaaS provider. Moreover, in the second stage, since competition for the spot instances is modeled as a Stackelberg game, we expect the IaaS provider to experience higher profits from the spot instances auction characterized, as observed in the previous examples, by lower than the maximum allowed on spot prices, larger volumes and higher overall profit. For the sake of comparison, we simulated a dynamic

SaaS	$\mu_{u,1}$	f^U_u	$\sigma^U_{u,1}$	SaaS	$\mu_{u,1}$	f^U_u	$\sigma^U_{u,1}$
1	11	5	0.38	6	12	3	0.23
2	5	5	0.49	7	12	3	0.44
3	13	3	0.16	8	8	4	0.3
4	14	5	0.83	9	11	5	0.54
5	11	4	0.28	10	6	5	0.42

Table III
SAAS PROVIDERS PARAMETERS

scenario using the two different policies. We considered 10 SaaS providers, each offering a single service. Every hour, each SaaS provider, given the forecasted load for the next hour determined the number and type of VMs to allocate while the IaaS provider determined the price of the spot instances. The predicted load $\Lambda_{u,1}$ of each SaaS provider is each time randomly generated uniformly in the interval $[20, 80]$ req/s. We set for all the providers, $R^{max}_{u,1} = 2s$, $C_{u,1} = 2\$$ (corresponding to $m_{u,1} = -1$) and $U^{max}_u = 0.9$. The other parameters are shown in Table III and were kept constant during the simulation. Since in [3] the variable s^U is fixed a priori and is independent from the amount of flat and on demand instances sold, we fixed $s^U = 30$ for the whole simulation for both policies. We run a simulation corresponding to a period of one week (168 hours). Table IV

	flat	on demand	spot	total
Stackelberg	6961.75	8706.31	5040	20708.06
GNEP	6961.75	3910.19	5040	15911.94
Stackelberg	1670.82$	10795.82$	1734.65$	14201.29$
GNEP	1670.82$	4848.63$	1575.48$	8094.93

Table IV
TOTAL NUMBER OF VMS SOLD AND RELATIVE REVENUE

shows the breakdown of the number of allocated VMs and IaaS revenue per type of instance. As expected, the revenue obtained using our service provisioning and pricing policy is greater than the revenue obtained using the policy in [3]. In particular, the average increment obtained using our policy is 36.3474$ against an average revenue of the IaaS provider obtained using the policy in [3] equal to 48.1842$. As anticipated, our scheme results in a higher number of on demand instances (more than twice as much). The number of spot instances is the same but this is actually a consequence of having a fixed σ^U[5]. Nevertheless, our scheme yields higher spot VMs revenues to the IaaS provider. As shown above, this is a consequence of the fact that in our policy the IaaS provider, by setting a price lower than the SaaS provider maximum bid, incentivates the SaaS providers to buy more spot instances.

SaaS	Stackelberg	GNEP	SaaS	Stackelberg	GNEP
1	73.96	75.55	6	75.44	78.5
2	56.1	55.76	7	79.96	80.31
3	78.04	80.79	8	73.4	78.75
4	81.79	81.54	9	74.86	74.49
5	71.36	73.84	10	62.2	64.86

Table V
AVERAGE SAAS PROVIDERS PROFIT ($).

Table V shows the profit for the different SaaS providers. We can observe that our approach results in a higher number of VMs bought by the SaaS providers and a corresponding higher cost, which justifies the significant larger IaaS provider profits (+64%). Interestingly, the SaaS profits decrease only by a small fraction and in some cases (SaaS 2, 4 and 9) even increase. This is not completely unexpected since, as the number of VMs per service increases, the service response time decreases which in turn, given the SaaS revenue function, yields higher revenues.

VI. RELATED WORK

Game theory is a useful tool to deal with those situations where the interaction across players has to be taken into

[5]For a more detailed comparison we should have modified the model in [3] to reflect our scheme where the number of available spot instances depends on the number of allocated flat and on demand instances.

account and thus can be successfully applied to typical ICT problems, like resource allocation, QoS, pricing, and load balancing. For example, it has been largely applied in networking research, as surveyed in [2].

Game theoretical approaches in Cloud computing have been proposed mainly to deal with resource allocation and pricing issues.A QoS-constrained resource allocation, where Cloud users submit intensive computation tasks is in [13]; however, only a single type of VM instances is considered. The perspective of a IaaS provider is pursued in [5] to determine the optimal suggested prices by the provider and the optimal user demands. In their model the provider suggests differentiated prices according to demand and users update their requests and the problem is formulated as a Stackelberg game. However, the model does not consider spot instances and QoS constraints of SaaS providers. The works in [3], [4] are most closely related to our proposal and we have compared their strategy to ours in Section V-B.

Pricing and performance issues related to spot instances have been recently addressed by works that apply a variety of methodologies different from game theory, such as checkpointing and work migration strategies to minimize the cost and volatility of resource provisioning [14]. Those studies focus on helping the users to better use spot instances, while we aim to maximize the IaaS provider revenue while satisfying the QoS constraints of the SaaS providers.

Some studies have investigated the prices of Amazon EC2 spot instances. A reverse engineering analysis in [15] found that Amazon sets their prices at random from within a tight price interval via a dynamic hidden reserve price. A statistical analysis and modeling of Amazon spot price dynamics is in [16]. Differently from our assumption, EC2 spot instances are priced the same to all its users.

Although still only one IaaS provider offers spot VMs, many argue that market economies will be increasingly prevalent in order to achieve high utilization in data centers and the first public marketplaces for unused capacity such as SpotCloud are already on the scene.

VII. Conclusions

In this paper we presented a service provisioning and pricing strategy for a Cloud system based on a game theoretical approach. We considered several SaaS providers that offer a set of applications with QoS constraints using the Cloud facilities provided by an IaaS provider. We proposed a two stage service provisioning policy: in the first stage, the SaaS providers buy VMs at a fixed price, while in the second stage they bid and compete to buy VMs instantiated on the unused IaaS provider capacity. The spot instances price is dynamically determined by the IaaS provider, which aims to maximize his revenue given the bids. We modeled the second stage conflicting situation as a Stackelberg game and computed its equilibrium price and allocation strategy by solving an MPEC problem. Our numerical results revealed the ability of the IaaS provider to set a price lower than the bids to incentivize the SaaS providers to buy more instances. We also compared our strategy with that in [3] and found that using our strategy the IaaS provider revenue increases, at the expense of a lower profit for the SaaS providers. However, the latter can offer more reliable services and better performance to their users.

As a future work we intend to study a distributed version of the proposed strategy and to investigate a different scenario, where the IaaS provider offers the same spot instance price to the SaaS providers.

Acknowledgments

We would like to thank Prof. F. Facchinei for very fruitful discussions on Stackelberg games and MPEC.

References

[1] Amazon Web Services LLC, "Amazon Elastic Compute Cloud (Amazon EC2)," 2012, http://aws.amazon.com/ec2/.

[2] E. Altman, T. Boulogne, R. El-Azouzi, T. Jiménez, and L. Wynter, "A survey on networking games in telecommunications," *Comput. Oper. Res.*, vol. 33, no. 2, 2006.

[3] D. Ardagna, B. Panicucci, and M. Passacantando, "A game theoretic formulation of the service provisioning problem in cloud systems," in *Proc. of WWW '11*, 2011, pp. 177–186.

[4] ——, "Generalized Nash equilibria for the service provisioning problem in cloud systems," *IEEE Trans. Serv. Comput.*, 2013, to appear.

[5] M. Hadji, W. Louati, and D. Zeghlache, "Constrained pricing for cloud resource allocation," in *Proc. of IEEE NCA '11*, Aug. 2011, pp. 359–365.

[6] D. Fudenberg and J. Tirole, *Game Theory*. MIT Press, 1991.

[7] Y. A. Korilis, A. A. Lazar, and A. Orda, "Achieving network optima using stackelberg routing strategies," *IEEE/ACM Trans. Netw.*, vol. 5, no. 1, pp. 161–173, Feb. 1997.

[8] F. Facchinei and C. Kanzow, "Generalized Nash equilibrium problems," *4OR: A Quarterly Journal of Operations Research*, vol. 5, pp. 173–210, 2007.

[9] V. Di Valerio, V. Cardellini, and F. Lo Presti, "Optimal pricing and service provisioning strategies in cloud systems: a Stackelberg game approach," Univ. Roma Tor Vergata, Tech. Rep. DICII RR-13.01, Feb. 2013.

[10] F. Facchinei and J.-S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems*. Springer, 2003.

[11] F. Facchinei, H. Jiang, and L. Qi, "A smoothing method for mathematical programs with equilibrium constraints," *Mathematical Programming*, vol. 85, pp. 107–134, 1999.

[12] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*. Springer, 2008.

[13] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, "A game-theoretic method of fair resource allocation for cloud computing services," *J. Supercomput.*, vol. 54, no. 2, 2010.

[14] S. Yi, A. Andrzejak, and D. Kondo, "Monetary cost-aware checkpointing and migration on amazon cloud spot instances," *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, 2012.

[15] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing Amazon EC2 spot instance pricing," in *Proc. of IEEE CloudCom'11*, Dec. 2011, pp. 304–311.

[16] B. Javadi, R. K. Thulasiram, and R. Buyya, "Characterizing spot price dynamics in public cloud environments," *Future Gener. Comput. Syst.*, vol. 29, no. 4, pp. 988–999, Jun. 2013.