An Economic Agent Maximizing Cloud Provider Revenues under a Pay-as-you-Book Pricing Model

Felipe Díaz Sánchez, Elias A. Doumith, Sawsan Al Zahr, and Maurice Gagnaire

Institut Mines-Telecom - Telecom ParisTech - LTCI CNRS
Networks and Computer Sciences Department
46, rue Barrault F 75634 Paris Cedex 13 - France
{felipe.diaz,elias.doumith,sawsan.alzahr,
maurice.gagnaire}@telecom-paristech.fr

Abstract. The Cloud computing paradigm offers the illusion of infinite resources accessible to end-users anywhere at anytime. In such dynamic environment, managing distributed heterogeneous resources is challenging. A Cloud workload is typically decomposed into advance reservation and on-demand requests. Under advance reservation, end-users have the opportunity to reserve in advance the estimated required resources for the completion of their jobs without any further commitment. Thus, Cloud service providers can make a better use of their infrastructure while provisioning the proposed services under determined policies and/or time constraints. However, estimating end-users resource requirements is often error prone. Such uncertainties associated with job execution time and/or SLA satisfaction significantly increase the complexity of the resource management. Therefore, an appropriate resource management by Cloud service providers is crucial for harnessing the power of the underlying distributed infrastructure and achieving high system performance. In this paper, we investigate the resource provisioning problem for advance reservation under a *Pay-as-you-Book* pricing model. Our model offers to handle the extra-time required by some jobs at a higher price on a best-effort basis. However, satisfying these extra-times may lead to several advance reservations competing for the same resources. We propose a novel economic agent responsible for managing such conflicts. This agent aims at maximizing Cloud service provider revenues while complying with SLA terms. We show that our agent achieves higher return on investment compared to intuitive approaches that systematically prioritize reserved jobs or currently running jobs.

Keywords: Cloud computing, Resource provisioning, Advance reservation, Pricing models, Pay-as-you-Book, Economic agents, SLA.

1 Introduction

Cloud computing is a large-scale distributed computing paradigm wherein IT (Information Technology) resources are delivered to end-users as a service. Using virtualization technologies, physical IT resources (*e.g.,* processing power,

K. Vanmechelen, J. Altmann, and O.F. Rana (Eds.): GECON 2012, LNCS 7714, pp. 29–45, 2012.

data storage, network bandwidth, etc.) can be packaged along with an operating system and a set of software into a flexible and scalable virtual machine (VM). End-users can dynamically customize, lease, and release VMs through the Internet according to their needs. Moreover, Cloud computing promises to provide IT resources to end-users as metered services. In analogy to traditional utilities such as water, gas, electricity, etc., Cloud service providers (CSPs) seek to meet fluctuating end-users needs and charge them for resources based on usage rather than on a flat-rate basis.

In Cloud computing, resource provisioning can be performed under on-demand or/and advance reservation plans (*e.g.,* Amazon EC2 and GoGrid). Under on-demand plan, CSPs charge end-users proportional to their resource consumption on a *Pay-as-you-Go* basis (*e.g.,* Amazon EC2 On-Demand Instances). In such a pricing model, resource consumption is measured in fine-grained measurement unit, *e.g.,* data storage consumption is typically measured in gigabytes. Furthermore, Cloud resource provisioning must be elastic, allowing end-users to dynamically lease/release resources to cope with their fluctuating and unpredictable needs. Large scale providers with virtually unlimited resources (*e.g.,* Amazon EC2) can guarantee such elasticity. However, small- and medium-sized providers with relatively limited resources may not be able to instantaneously satisfy all requests.

Another classical resource management strategy is to employ an advance reservation (AR) mechanism. Under AR plan, end-users submit their requests to the CSP beforehand and commit to use the requested service during a given time period by paying a reservation fee. In return, the CSP offers its services at a lower price compared to the on-demand plan. In doing so, the CSP is able to lock resources and thus guarantee that end-users can access the required resources during the reserved time period [1]. Moreover, AR allows the CSP to maximize its resource utilization and yield optimal profits. However, end-users requests are often subject to uncertainties (*e.g.,* job execution time) which may result in under-/over-provisioning problems. In such cases, the CSP has to decide whether or not to satisfy additional requests taking into account available resources and SLAs (Service Level Agreement) agreed with its end-users. To this end, an appropriate resource management by the CSP is crucial for harnessing the power of the underlying distributed infrastructure and achieving high system performance.

In our previous work [2], we studied the problem of Cloud resource provisioning in an on-demand fashion. Indeed, we considered job requests with time-variable capacity requirements whereas the CSP only relies on the capacity requirement upon the request arrival. We investigated different algorithms to solve this resource provisioning problem and compared them in terms of resource utilization as well as VMs dropping and rejection ratios. In this paper, we investigate the problem of Cloud resource provisioning for AR under a *Pay-as-you-Book* pricing model. Our model offers to handle the extra-time required by some jobs at a higher price, on a best-effort basis. Indeed, ARs running for a longer period than expected may lead to resource conflicts with other ARs. In order to resolve such resource conflicts, we propose in this work an economic agent responsible for

managing the under-provisioning problem. Our economic agent aims at maximizing the CSP revenues while complying with the SLA terms. Through numerical simulation, we show that our agent achieves higher revenues compared to intuitive approaches that systematically prioritize reserved jobs or currently running jobs.

The remainder of this paper is organized as follows. In Section 2, we provide a detailed state-of-the-art focusing on AR-based resource provisioning approaches. In Section 3, we present our formulation of the resource provisioning problem under a *Pay-as-you-Book* pricing model and emphasize our contribution with regard to related works. We also introduce the economic agent responsible for managing resource conflicts caused by under-estimated jobs. Numerical results and performance evaluation are given in Section 4. We then conclude our paper in Section 5 with some directions for future work.

2 Related Work

Advance reservation has been introduced in Grid and Cloud environments as an efficient way to guarantee the availability of IT resources for use at a specific time in the future. In order to handle AR, the CSP needs some information specifying the quantity of resources required by the job, the ready-time when this job can start its execution, the expected job execution time, and its deadline. Based on the capacity and time requirements of the jobs, we can classify existing studies on AR into:

2.1 Advance Reservation Specified by Cloud Service Providers

This type of AR is tightly related to the subscription-based pricing model, widely proposed by CSPs. Under this pricing model, end-users must commit to use the service for a given time period by paying a one-time fee; in exchange, the CSP guarantees the availability of the required resources at reduced hourly rates. This type of reservation operates on a time-interval basis. At the beginning of each time-interval, the end-user may adjust the amount of resources to be reserved by the CSP for the next time-interval. Conducted research studies can be classified into short-term reservation plans [3, 4] (*e.g.,* fine granularity of 10-minute/1-hour time-intervals) and long-term reservation plans (*e.g.,* several years time-intervals) [5, 6].

In [3], the authors investigated pricing policies for guaranteed bandwidth reservation in the Cloud on a short-term basis such as hours or tens of minutes. Requests are characterized by an estimated average bandwidth requirement, its variability, and the percentage of the traffic flow to be satisfied with guaranteed bandwidth. As for the CSP, it computes the current bandwidth reservation in order to guarantee the required performance in a probabilistic way. It also decides on the reservation fee taking into account the burstiness and the time correlation of the various requests. The authors in [4] investigated a similar problem where a broker is introduced between the CSPs and the end-users. While the broker sells guarantees to end-users individually, it jointly reserves bandwidth

from multiple CSPs for the mixed demand, exploiting statistical multiplexing to save reservation cost. The problem was solved using a game theory approach where the equilibrium bandwidth price depends on the demand expectation, its burstiness as well as its correlation to the market.

The long-term reservation plan has been first studied in [5]. The authors considered a single CSP and proposed an algorithm that selects the number of VMs to be reserved by an end-user while deploying a service in the Cloud. In order to cope with request fluctuations and unpredictability, additional resources may be dynamically provisioned with an on-demand plan. The proposed algorithm minimizes the global cost of using a mixture of reserved and on-demand instances by taking advantage of the different pricing models within the same provider. The authors of [6] generalized the problem to the context of multiple CSPs taking into account the uncertainty on end-users future requests and providers resource prices. They formulated the problem as an integer stochastic program and solved it numerically using various approaches.

2.2 Advance Reservation Specified by End-Users

In this type of AR, end-users have a higher flexibility as they can specify, in addition to their capacity requirements, various time constraints associated with the execution of their jobs. Time constraints can be expressed in terms of various parameters such as ready-time, job duration, and job deadline. Thus, end-users have the opportunity to reserve in advance the estimated required resources for the completion of their jobs without any further commitment. In the sequel, we define the AR window as the time-interval delimited by the ready-time and the deadline of a given job request. AR specified by end-users can be classified into three main categories as follows:

a) Strict Start and Completion Time: This type of job is characterized by a job duration equal to its AR window. In other words, end-users require the resources at a specified exact time in the future and for a specified duration. This type of jobs does not leave any flexibility to the CSP to reschedule the job at a different time period. Several studies have shown that jobs with strict start and completion time lead to high fragmentation of the resources availability by increasing the number of time intervals that are left unused [7, 8]. These time intervals can be used by other types of requests such as spot and on-demand instances.

The authors of [9] investigated the provisioning of computing, storage, and networking resources in order to satisfy AR requests. They considered several basic services and highlighted how distributed data storage and multicast data transfer can satisfy a larger number of end-users and improve resource utilization of CSPs. The business model of the aforementioned problem has been investigated in [10]. The authors proposed and compared three pricing strategies assessing the expectations of both end-users and CSPs.

b) Flexible Start But Strict Completion Time: This type of jobs is characterized by a higher flexibility than the former as the AR window is larger

than its execution time. However, these jobs are time-critical and, if accepted, the CSP must ensure that they will complete prior to their firm deadline. Thus, CSPs may use various mechanisms to efficiently arrange, manage, and monitor their resources. For instance, the authors in [1] introduced a model based on computational geometry that allows CSPs to record and efficiently verify the availability of their resources during the SLA negotiation and planning phase. According to this model, when the CSP lacks resources, a flexible alternative solution, referred to as counter-offer, can be generated in order to satisfy the end-user. Hence, the CSP's reputation can be enhanced by improving its ability to satisfy as many end-users as possible leading to higher resource utilization and consequently higher profits. The authors in [11] investigated a negotiation mechanism that allows either parties (CSPs and end-users) to modify the SLA or to make counter proposals in order to converge to a mutually acceptable agreement. In the investigated scenarios, once the SLA has been agreed upon, the CSP has to execute the job at the specified time. Numerical simulations have been carried out to highlight the benefit brought by time-flexible job requests. The authors in [12] investigated the impact of the AR window size on the blocking probability and the resource utilization for various models of inter-arrival and service times under the first-come-first-served scheduling policy.

In [13], the authors investigated the resource provisioning problem in a market-oriented Cloud considering ARs with flexible window size that is a function of the requirements and the budgets of end-users. The aim of this study is to propose a fair management algorithm that guarantees the QoS (Quality of Service) requirements of end-users while increasing the expected benefit of CSPs. For this purpose, the authors introduced a weighted cost function that enables service differentiation relying on time constraints disparity of the requests. An exact linear formulation [13] as well as a heuristic approach [14] have been considered for the numerical performance evaluation. Instead of charging fixed prices, the authors in [15] propose to automatically adjust the price for accessing the resources, whenever necessary, in order to increase the CSP revenues. By charging variable prices, CSPs can give incentives to end-users with less urgent requirements to shift to using the service during off-peak periods and benefit from lower prices. As the prices are adjusted based on the expected workload and the resource availability, ARs submitted a long time in advance are privileged with cheaper prices compared to late ARs.

Similar investigations have been carried out in a slightly different environment. The new environment allows the CSP to modify the execution schedule of already accepted ARs in order to accommodate new requests right up until each execution starts [16]. Such rescheduling of existing jobs is carried out while respecting the deadline constraints specified in the SLA. The authors have shown that this mechanism can mitigate the negative effects of AR and improve the performance of reservation-based schedulers as it tends to reduce the amount of time intervals where resources remain free. Another solution to improve resource utilization is to make use of comprehensive overbooking which is particularly efficient in scenarios with no-show policy, job cancelation [17], and over-estimated

execution time of jobs [18]. In this context, rescheduling existing jobs may allow overbooked jobs to get access to the resources during their full execution period if previous jobs do not show up or finish earlier. The Earliest Deadline First scheduler have been investigated to provide probabilistic real-time guarantees for AR over time-shared machines [19]. With this scheduling strategy, an admission control policy is developed where new job requests are accepted if they do not break the QoS constraints of previously accepted reservations. This can be achieved for instance by changing the priority of the running jobs to ensure that the execution completes prior to its deadline.

c) Flexible Start and Completion Time: This type of jobs is also characterized by a high flexibility. However, the AR window is not clearly defined. Instead of defining a ready-time and a firm deadline for the execution of each job, the end-user provides a set of time-intervals along with its preferences represented by a utility function. The utility function represents the level of satisfaction that the end-user will experience as a result of the negotiation outcome. This satisfaction may depend on several parameters such as the time of execution, the price of the resources, the delays, the QoS requirements, etc. Usually, not being able to reach an agreement is the worst possible outcome and the end-user receives a null utility as its request is rejected. Dynamic pricing based on resource utilization and end-users classification was introduced in [20]. Such dynamic pricing strategy allows the CSP to adapt the price to set incentives for using the resources during off-peak periods. Two different approaches, which are already well established in other areas, are compared in [21] namely, reservation realized by derivative markets in a perfect competition CSPs environment and yield management techniques assuming an imperfect competition environment. The authors analyze the different requirements in order to apply the proposed approaches in the Cloud and provide models to derive the suitable reservation price. The authors in [22] introduced a bilateral negotiation mechanism for Cloud service reservation that simultaneously considers price and execution time. Numerical simulations have been used to compare the proposed mechanism to traditional pricing models used by current CSPs namely, fixed-prices for on-demand and reserved instances, and variable prices for spot instances. The Time-of-Use pricing policy has been investigated in [23]. According to this policy, the price of accessing resources is totaly independent from the utilization ratio of the requested resources but varies within a day. The optimal pricing strategy that maximizes the end-user satisfaction is derived.

3 Problem Formulation

In this paper, we focus our investigations on VM provisioning and usage for compute-intensive and/or processing-intensive scientific applications. Under this assumption, all VMs are already configured with a considerable amount of CPU resources and dedicated memory space. Once a job is running on a given VM, the underlying resources associated with this VM (*e.g.,* CPU power, memory space, network bandwidth, etc.) are intensively used and cannot benefit from

statistical multiplexing. Therefore, an incoming job request only has to express its requirements in terms of VMs without explicitly specifying their configurations. Although the end-users have the illusion of infinite resources within the Cloud, the CSPs are always constrained with limited resource availability. For this purpose, we consider in this study a large data-center, owned by a single CSP, that can host up to $\mathcal{N}$ VMs.

Many scientific applications such as telemedicine, multimedia, or air traffic flow management require the combination and orchestration of several services to meet their requirements. As the resources are being shared by multiple applications which are completely unaware of each other, the use of AR has been proposed as a means to provide time-guarantees on the successful completion of the submitted jobs. The AR mechanism allows end-users to reserve enough resources across independently administrated domains prior to their job's execution. In order to efficiently handle ARs, the CSP needs information regarding the required quantity of resources, the ready-time as well as the execution time of the jobs. As the execution time of the applications may vary from one run to another, it is a tedious task for end-users to provide these values. This is especially true for distributed applications since their execution time highly depends on the interaction between the various implied services.

Due to demand uncertainty, job requests can be classified into under-estimated and over-estimated jobs. Over-estimated jobs will run for a shorter period in comparison to their stipulated execution time. Conversely, under-estimated jobs will run for a longer period than expected. Such inaccuracy in estimating job execution time can result in lower resource utilization and higher rejection rates. However, performance degradation is less severe for job requests characterized by flexible start but strict completion time, or flexible start and completion time. Indeed, these types of jobs can benefit from the backfilling mechanism where the CSP reschedules all the accepted jobs in order to adapt to the changing conditions. For instance, when an over-estimated job leaves the system, the CSP invokes the scheduler in order to achieve larger contiguous idle time periods. These idle periods can facilitate the accommodation of future requests as well as the provisioning of additional time for requests that have exceeded their specified execution time. Thus, instead of aborting the execution of under-estimated jobs, the CSP investigates the feasibility of providing them with extra-time without missing the deadlines of other accepted jobs.

Nevertheless, to the best of our knowledge, demand uncertainty has never been investigated in the context of ARs characterized by strict start and completion time. Previous investigations in this matter assume that the jobs are perfectly known [7–10] or propose to terminate any under-estimated application that is still executing once its reservation period expires. In our study, we offer to handle the extra-time required by some jobs at a higher price on a best-effort basis. Moreover, we propose to manage any resource conflict that may arise between an under-estimated job and another already reserved job while complying with the SLA terms and maximizing the CSP revenues.

3.1 Job Characterization

Scientific applications are typically modeled as workflows consisting of tasks, data elements, control sequences, and data dependencies. A workflow describes the order in which several jobs must be performed by different entities in order to achieve a given outcome. A workflow management engine is responsible for managing and controlling the execution of these jobs. It also allows end-users to specify their requirements using the workflow specification. Thus, the workflow Ω^i of a given end-user $\mathcal{U}^i$ can be modeled by a sequence of jobs ω_j^i, where j denotes the index of the job ω_j^i in the workflow Ω^i $(j = 1 \cdots \mathcal{J}^i)$. Each job ω_j^i, characterized by *"strict start and completion time"* (*cf.* Section 2.2a), is represented by a tuple $(n_j^i, \alpha_j^i, \beta_j^i, \gamma_j^i)$, where n_j^i denotes the number of required VMs, α_j^i the ready-time of the job, β_j^i its completion time estimated by the end-user, and γ_j^i its real completion time obtained once executed on the given cluster. A workflow completes when all its jobs are completed. In our study, we have considered a set of $\mathcal{M}$ workflows $(i = 1 \cdots \mathcal{M})$ composed of a sequence of jobs to be executed within a given time interval $[0, \Delta]$.

3.2 Initial Scheduling of Job Requests

Since ARs are made prior to job execution, the CSP can use various scheduling approaches in order to optimize the resource utilization of its infrastructure, and consequently increases its revenues. At this stage, the CSP has only the knowledge of the execution time estimated by end-users. Even though these estimations are often imprecise, the CSP has to decide whether to accept $(\varpi^i = 1)$ or reject $(\varpi^i = 0)$ each workflow Ω^i depending on its resources availability. As stated previously, a workflow Ω^i is accepted if all its jobs ω_j^i $(j = 1 \cdots \mathcal{J}^i)$ can be satisfied.

The initial scheduling problem can be formulated as follows. Given the number $\mathcal{N}$ of VMs and the set of $\mathcal{M}$ workflows, the CSP has to determine, for each accepted workflow, the physical machine that will host it. This should be carried out while respecting the limited resources of the CSP and the fixed ready-time and completion time of end-users jobs. The main objective of the CSP at this stage is to maximize the utilization $\mathcal{G}$ of its resources which can be expressed mathematically as:

$$\mathcal{G} = \frac{1}{\mathcal{N} \times \Delta} \sum_{i=1}^{\mathcal{M}} \sum_{j=1}^{\mathcal{J}^i} \varpi^i \times n_j^i \times \left(\beta_j^i - \alpha_j^i\right). \tag{1}$$

This problem turns out to be similar to the 2-dimensional bin packing problem with rejection. In order to solve this problem, we will use a very straightforward sequential algorithm commonly known as "Decreasing First Fit" (FFD) algorithm. This a simple offline heuristic algorithm that achieves a near-optimal solution for the classical 1-dimensional bin packing problem [24]. The FFD strategy operates in two phases. First, it sorts the workflows in decreasing order of their

cumulated reservation time $\left(\sum_{j=1}^{\mathcal{J}^i}(\beta_j^i - \alpha_j^i)\right)$. Then, it processes the workflows according to the previous order, and schedules the jobs of the selected workflow in the first VM with sufficient remaining capacity during their respective reservation intervals. If none of the VMs can (partially or fully) accommodate the incoming workflow, the workflow will be rejected.

3.3 Pay-as-you-Book Pricing Model

Numerous economic models, including microeconomic and macroeconomic principles, have been investigated in the literature for setting the appropriate price for accessing a service. A pricing policy can be derived from various parameters such as the supply-and-demand and their value to the end-users. The commodity market, posted price, tender, bargaining, and auction models are among the commonly used economic models for managing the resources in the Cloud [25]. In this paper, we focus on the *Pay-as-you-Book* pricing model. It is a flat price commodity market model where the CSP specifies its service price and charges end-users for the amount of resources they reserve. Let Γ^R be the hourly rate of a reserved instance. Γ^R is independent of the service quality and the number of jobs. Upon, the acceptance of a workflow Ω^i, the CSP expects the payment of a reward or fee for the successful completion of this reservation. This reservation fee $\mathcal{C}^i$ can be expressed as follows:

$$\mathcal{C}^i = \sum_{j=1}^{\mathcal{J}^i} n_j^i \times (\beta_j^i - \alpha_j^i) \times \Gamma^R. \tag{2}$$

If all the jobs of a given workflow Ω^i finish within their respective reservation period ($\gamma_j^i \leq \beta_j^i$), the end-user does not have to pay any additional fee. However, it may happen that a job takes more time to execute than initially estimated ($\gamma_j^i > \beta_j^i$). In this case, the CSP can allocate the required resources for a longer period for a higher hourly rate Γ^O on a best-effort basis ($\Gamma^O > \Gamma^R$). In other words, the CSP cannot guarantee that the job will continue running until its real completion time γ_j^i. Let θ_j^i be the time when an under-estimated job ω_j^i successfully ends ($\theta_j^i = \gamma_j^i$) or is forced to terminate by the CSP if the resources are reserved for executing another job ($\theta_j^i < \gamma_j^i$). In this case, the end-user is requested to pay, for each under-estimated job ω_j^i, an additional fee $\mathcal{F}_j^i$ equal to:

$$\mathcal{F}_j^i = n_j^i \times (\theta_j^i - \beta_j^i) \times \Gamma^O. \tag{3}$$

When the CSP accepts an AR, the end-user expects to be able to access the agreed resources at the specified ready-time. However, changes may occur between the time when the end-user submits the reservation and this specified ready-time. This can happen for various reasons such as end-users canceling or modifying requests, resource failures, and errors in the estimation of the execution time. Since an AR is considered as a commitment by the CSP, failing to meet this commitment may result in the provider paying the end-user a penalty

$\mathcal{P}_j^i$ larger than the reservation fee. For each rejected job at its ready-time, the CSP is requested to reimburse the end-user an amount $\mathcal{P}_j^i$ equal to:

$$\mathcal{P}_j^i = n_j^i \times (\beta_j^i - \alpha_j^i) \times (\Gamma^R + \Gamma^P), \tag{4}$$

where Γ^P represents the credit that the CSP has to return to the end-user if it is unable to start the job.

3.4 An Economic Agent for Maximizing CSP Revenues

According to the previous description, we can distinguish three scenarios namely, over-estimated jobs (*cf.* Figure1a), under-estimated jobs without any conflict (*cf.* Figure1b), and under-estimated jobs resulting in a conflict (*cf.* Figure1c) with other ARs. The first two scenarios are trivial since the CSP does not have to intervene and the AR will end normally. For these scenarios, the CSP can keep the reservation fee and will obtain an additional fee for executing any under-estimated job ($\theta_j^i = \max(\beta_j^i, \gamma_j^i)$). However, in the third scenario, a conflict arises as an under-estimated job ω_j^i is competing for the same resources as an incoming AR $\omega_{j'}^{i'}$. Thus, the CSP has to decide at the ready-time $\alpha_{j'}^{i'}$ of the new AR $\omega_{j'}^{i'}$ whether to keep running the under-estimated job ω_j^i or abort it.

Targeting higher revenues, the CSP first has to estimate the average extra-time δ required by such jobs. This can be easily obtained by analyzing the past history of all compute-intensive job executions and hence adjusting δ accordingly. Based on this estimation, the CSP can evaluate the different choices for resolving any conflict. On one hand, if the under-estimated job ω_j^i is kept running, the CSP estimates getting from end-user $\mathcal{U}^i$ an additional fee equal to $\digamma_1 = n_j^i \times (\delta + \beta_j^i - \alpha_{j'}^{i'}) \times \Gamma^O$. However, the CSP has to pay the end-user $\mathcal{U}^{i'}$ a penalty $\mathcal{P}_{j'}^{i'}$ equal to $\digamma_2 = n_{j'}^{i'} \times (\beta_{j'}^{i'} - \alpha_{j'}^{i'}) \times (\Gamma^R + \Gamma^P)$. On the other hand, if the under-estimated job ω_j^i is aborted and the new AR $\omega_{j'}^{i'}$ is executed, the CSP can keep the reservation fee but will not obtain any additional benefit. By comparing the values of $\digamma_1$ and $\digamma_2$, the CSP will decide on the best way to resolve this conflict. If the CSP decides to keep the under-estimated job, it should negotiate with the owner of the incoming AR if it accepts to delay its current execution and gets in exchange the penalty specified in the SLA and a new time slot for executing its job. We assume that the end-user can accept such a proposal with a probability ρ.

4 Numerical Results

4.1 Experimental Setup

In our simulations, we consider a single CSP with limited resources that can host up to $\mathcal{N} = 10$ VMs simultaneously. We consider a simulation period of 4 days (or equivalently $\Delta = 96$ hours). In our investigations, we only consider workflows Ω^i

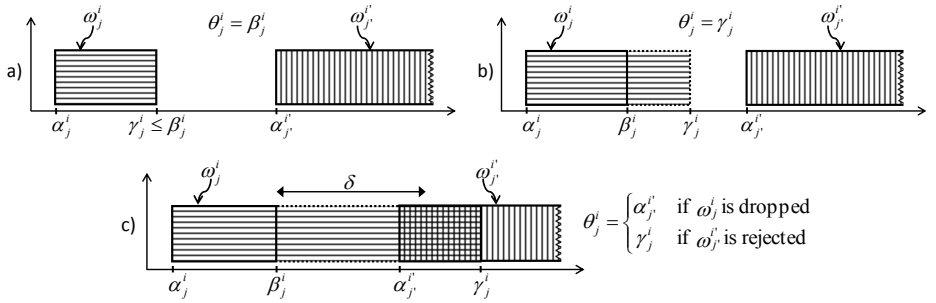

Fig. 1. Possible scenarios of running jobs

with one single job ($\mathcal{J}^i = 1$) requiring a single VM (n_j^i=1). The ready-time α_j^i of a job is chosen uniformly in the interval $[0, \Delta = 96[$ while its estimated execution time μ_j^i follows a negative exponential law of mean $\hat{\mu} = 5$ hours bounded by a maximum duration of 8 hours ($\beta_j^i = \alpha_j^i + \mu_j^i$). The percentage ψ of under-estimated jobs varies in the set $\{20\%, 30\%, 40\%\}$ and the extra-time λ_j^i required by these reservations also follows a negative exponential law of mean $\hat{\lambda}$ equal to 1 or 2 hours ($\gamma_j^i = \beta_j^i + \lambda_j^i$). Without loss of generality, we have fixed the value of Γ^R to 1 and assumed that $\Gamma^O = 3$ and $\Gamma^P = 1$. Finally, the probability ρ of a successful negotiation between the CSP and the end-users has been fixed to 100%.

Under the aforementioned parameters, we have chosen to consider two different AR loads: light loads with $\mathcal{M} = 100$ and heavy loads with $\mathcal{M} = 200$. These AR loads have been inspired from the traces found in [26]. These traces provide, among other information, the submit time, the requested time, the execution time, the identifier, and the status of the jobs.

All experiments have been repeated 1000 times. For each scenario, we report the average values computed over these different runs of:

- The percentage $\mathcal{R}_i$ of ARs that are rejected at the end of the offline initial scheduling.
- The percentage $\mathcal{R}_d$ of ARs that are accepted at the end of the offline initial scheduling but are dropped during their execution because they under-estimated their execution time.
- The percentage $\mathcal{R}_r$ of ARs that are accepted at the end of the offline initial scheduling but are rejected at their ready-time because the CSP decided to keep running an under-estimated job.
- The percentage $\mathcal{R}_a$ of ARs that are accepted and executed during their complete activity period. It is obvious that the following equation holds:

$$\mathcal{R}_i + \mathcal{R}_d + \mathcal{R}_r + \mathcal{R}_a = 1. \tag{5}$$

- The revenues Ξ of the CSP computed as a function of Γ^R, Γ^O, and Γ^P as follows:

$$\Xi = \sum_i \mathcal{C}^i + \sum_{\{\text{extended } \omega_j^i\}} \mathcal{F}_j^i - \sum_{\{\text{rejected } \omega_j^i\}} \mathcal{P}_j^i. \quad (6)$$

  It is worth noting that in our investigations, we assume that the CSP expenses (*i.e.,* CapEx and OpEx) do not change with the number of running jobs. Hence, the CSP profits have the same trend as the CSP revenues and could be deduced accordingly.

- The average utilization ratio χ of the CSP resources during the simulation period Δ. χ is computed in the same way as $\mathcal{G}$ (Eq. (1)) taking into account the aforementioned percentages $\mathcal{R}_d$ and $\mathcal{R}_r$.

As stated previously, the AR mechanism allows end-users to reserve enough resources across independently administrated domains prior to job execution. The end-users may also have QoS requirements that can be expressed in terms of various parameters such as deadlines, security, trust, and budget associated with service invocation. The QoS parameters along with the time and quantity of resources requested by the end-user are encoded in the SLA. Thus, the SLA can be viewed as a formal agreement between the CSP and the end-users specifying the services, priorities, responsibilities, guarantees, etc. of both parties. In this study, the SLA is defined, among other parameters, by:

- The set of workflows $\left\{\Omega^i = \left\{\omega_j^i(n_j^i, \alpha_j^i, \beta_j^i), \forall j = 1 \cdots \mathcal{J}^i\right\}, \forall i = 1 \cdots \mathcal{M}\right\}$ to be executed. For each job ω_j^i, the number of required VMs as well as their ready and estimated completion times are also specified.
- The hourly rate Γ^R for running an AR and the hourly rate Γ^O for extending it beyond its estimated completion time.
- The credit Γ^P due by the CSP to the end-user in the case of non-compliance with the SLA terms.

4.2 Reference Scenarios

The goal of this study is to investigate an economic agent responsible for resource management under a *Pay-as-you-Book* pricing model. This economic agent has to achieve *"end-user satisfaction"* by providing QoS guarantees for ARs, *"cost effectiveness"* by efficiently maximizing the CSP revenues, and *"robustness"* by intelligently handling uncertainties such as those in user-estimated execution times. In order to assess the performance of the proposed agent, we will introduce three intuitive strategies that can be adopted by the CSP.

On-Demand Approach: No ARs are made at all and the resource allocation is performed online. Upon the arrival of a new job request ω_j^i, the CSP evaluates its instantaneous resource utilization. If enough free resources are available, the new request is accepted; otherwise, it is rejected. In return, the end-user is expected to pay a higher price Γ^O for accessing the resources as they are not reserved in

advance. This approach does not ensure end-user satisfaction with a workflow composed of multiple jobs as there is no guarantee that all its job instances will be accepted. However, as we have considered in our experiments' workflows with a single job, the on-demand approach can be considered as a good reference scenario. Moreover, for workflows with a single job, this strategy is characterized by a null percentage of dropped ARs during their execution ($\mathcal{R}_d = 0$) and a null percentage of rejected ARs prior to their execution ($\mathcal{R}_r = 0$).

Highest Priority to Running Jobs (HPRU): Under this strategy, the CSP will never abort a running job and always try to postpone the incoming AR that causes the conflict to a later period through negotiations. The only incentive for the end-user to accurately estimate its job execution time is motivated by the lower price of reserved instances ($\Gamma^O > \Gamma^R$). This strategy is characterized by a null percentage of dropped ARs during their execution ($\mathcal{R}_d = 0$).

Highest Priority to Reserved Jobs (HPRE): Under this strategy, under-estimated jobs are penalized as they are aborted whenever a conflict arises after they have executed for their estimated execution times. In order to protect their application from forced termination, end-users with critical applications must ensure that the estimated execution times are sufficient for their applications to be completed. This strategy is characterized by a null percentage of rejected ARs prior to their execution ($\mathcal{R}_r = 0$).

4.3 Performance Evaluation

Impact of the Number of Submitted Jobs: In this first scenario, we have fixed the percentage ψ of under-estimated jobs to 20% and the average extra-time required by these jobs to $\hat{\lambda} = 1$.

Table 1. Impact of the number of submitted jobs

	$\mathcal{M} = 100$ jobs					$\mathcal{M} = 200$ jobs				
	$\mathcal{R}_i$	$\mathcal{R}_d$	$\mathcal{R}_r$	χ	Ξ	$\mathcal{R}_i$	$\mathcal{R}_d$	$\mathcal{R}_r$	χ	Ξ
Initial Scheduling	0.10%	0%	0%	35.45%	30500	7.64%	0%	0%	67.51%	58000
On-Demand	0.20%	0%	0%	38.54%	99750	9.68%	0%	0%	67.01%	173250
HPRU	0.10%	0%	8.17%	35.09%	33750	7.64%	0%	10.01%	64.77%	61000
HPRE	0.10%	8.97%	0%	37.15%	32750	7.64%	11.29%	0%	69.64%	61000
Economic Agent	0.10%	6.42%	2.32%	37.03%	35250	7.64%	8.00%	2.91%	69.34%	64750

As expected, the on-demand approach ensures the highest CSP revenues as the end-users are paying a higher price during all the execution of their jobs ($\Gamma^O = 3 \times \Gamma^R$). It also achieves a high overall acceptance ratio $\mathcal{R}_a$ as it does not have to deal with estimation uncertainties. This latter behavior is expected to change in the case of workflows with multiple jobs. We notice that both the HPRU and the HPRE strategies achieve similar revenues Ξ for the CSP.

However, the HPRU strategy achieves the highest acceptance ratio $\mathcal{R}_a$ for AR, while the HPRE has a better performance in terms of resource utilization χ. Our proposed economic agent achieves slightly lower resources utilization compared to the HPRE strategy and keeps the percentage of rejected AR prior to their execution $\mathcal{R}_r$ at an acceptable value. In summary, our proposed economic agent is a trade-off in terms of resource utilization and acceptance ratio between the intuitive HPRU and HPRE strategies, but outperforms both of them in terms of CSP revenues. Indeed, our proposed economic agent achieves an average increase of almost 6% in the CSP revenues. These conclusions still hold independently of the number of submitted jobs.

Impact of the Percentage of under-Estimated Jobs and Their Execution Extra-Time: In this scenario, we vary the percentage ψ of under-estimated jobs in $\{20\%, 30\%, 40\%\}$ and the average extra-time $\hat{\lambda}$ required by these jobs in $\{1, 2\}$.

Table 2. Impact of the percentage of under-estimated jobs and their execution extra-time

		$\hat{\lambda}=1$					$\hat{\lambda}=2$				
		$\mathcal{R}_i$	$\mathcal{R}_d$	$\mathcal{R}_r$	χ	Ξ	$\mathcal{R}_i$	$\mathcal{R}_d$	$\mathcal{R}_r$	χ	Ξ
$\psi=20\%$	Initial Scheduling	7.64%	0%	0%	67.51%	58000	7.43%	0%	0%	67.58%	58000
	On-Demand	9.68%	0%	0%	67.01%	173250	11.12%	0%	0%	68.31%	176250
	HPRU	7.64%	0%	10.01%	64.77%	61000	7.43%	0%	11.21%	65.89%	65000
	HPRE	7.64%	11.29%	0%	69.64%	61000	7.43%	12.81%	0%	70.15%	61500
	Economic Agent	7.64%	8.00%	2.91%	69.34%	64750	7.43%	5.53%	6.51%	69.40%	67500
$\psi=30\%$	Initial Scheduling	7.44%	0%	0%	67.45%	58000	7.45%	0%	0%	67.49%	58000
	On-Demand	11.10%	0%	0%	68.24%	176250	13.48%	0%	0%	70.66%	180750
	HPRU	7.44%	0%	14.34%	63.60%	62000	7.45%	0%	15.91%	65.22%	67750
	HPRE	7.44%	17.07%	0%	70.61%	62250	7.45%	19.34%	0%	71.38%	63250
	Economic Agent	7.44%	11.89%	4.33%	70.15%	67750	7.45%	9.52%	8.09%	69.98%	71750
$\psi=40\%$	Initial Scheduling	7.51%	0%	0%	67.54%	58000	7.48%	0%	0%	67.48%	58000
	On-Demand	12.68%	0%	0%	69.52%	179250	15.97%	0%	0%	71.27%	183750
	HPRU	7.51%	0%	18.10%	62.66%	62750	7.48%	0%	19.89%	64.71%	69750
	HPRE	7.51%	22.71%	0%	71.78%	63750	7.48%	25.50%	0%	72.63%	62500
	Economic Agent	7.51%	15.67%	5.59%	71.11%	71000	7.48%	12.36%	10.27%	70.72%	75250

As the initial scheduling does not have any knowledge about the error in estimating the execution time, it achieves the same performance independently of the values of ψ and $\hat{\lambda}$. We notice that the percentage of ARs that are rejected prior to their execution $\mathcal{R}_r$ in the HPRU strategy increases with the percentage of under-estimated jobs. However, this increase is less pronounced than the increase observed in the HPRE strategy for the percentage of dropped ARs during their execution $\mathcal{R}_d$. Finally, our proposed economic agent keeps its superiority and still achieves a trade-off in terms of resource utilization and acceptance ratio between

the HPRU and HPRE strategies, but it outperforms both of them in terms of CSP revenues. Indeed, as the percentage of under-estimated jobs increases, the additional gain in the CSP revenues increases from almost 6% for $\psi = 20\%$ to around 12.25% for $\psi = 40\%$. Moreover, as the average extra-time required by the jobs increases, the difference between the HPRU and the HPRE strategies becomes more pronounced as the HPRU strategy achieves higher revenues.

5 Conclusions

Reflecting the recent trend of augmenting Cloud computing with AR provisioning plans, we investigate in this paper the problem of resource provisioning under a *Pay-as-you-Book* pricing model considering ARs characterized by under-estimated execution times. Our model offers to handle the extra-time required by jobs at a higher price, on a best-effort basis. Indeed, the extra-time required by an AR plan may lead to resource conflicts with other AR plans. In order to resolve such resource conflicts, we propose in this work an economic agent responsible for managing the under-provisioning problem. Our economic agent aims to achieve the end-user satisfaction by complying with the SLA terms on one hand as well as the CSP satisfaction by maximizing its revenues through intelligent resource management on the other hand. In this paper, we limited our investigations to compute-intensive applications requesting Virtual Machines. However, this work can be easily generalized to any IaaS resources.

In order to assess the performance of our proposed agent, we have compared our proposed economic agent with two intuitive approaches that systematically prioritize reserved jobs or currently running jobs. Our economic agent achieves a trade-off between the two intuitive strategies in terms of resource utilization and acceptance ratio, but outperforms both of them in terms of CSP revenues. These conclusions still hold independently of the number of submitted jobs, the percentage of under-estimated jobs, and the average duration of the extra-time required. Future studies will extend the results presented in this paper to the case of workflows with multiple jobs. In addition, we intend to enhance the initial scheduling algorithm in order to introduce higher flexibility in the decisions of the economic agent. This economic agent could be be augmented with additional features that implement overbooking techniques. In doing so, the CSP can overcome performance degradation in case of job cancellations.

Acknowledgments. This work is granted by the CompatibleOne project funded by French institutions.

References

1. Lu, K., Roblitz, T., Yahyapour, R., Yaqub, E., Kotsokalis, C.: QoS-aware SLA-based Advanced Reservation of Infrastructure as a Service. In: IEEE CloudCom Conference (November-December 2011)

2. Díaz Sánchez, F., Doumith, E.A., Gagnaire, M.: Impact of Resource over-Reservation (ROR) and Dropping Policies on Cloud Resource Allocation. In: IEEE CloudCom Conference (November-December 2011)
3. Niu, D., Feng, C., Li, B.: Pricing Cloud Bandwidth Reservations Under Demand Uncertainty. In: ACM Sigmetrics/Performance Conference (2012)
4. Niu, D., Feng, C., Li, B.: A Theory of Cloud Bandwidth Pricing for Video-on-Demand Providers. In: IEEE INFOCOM Conference (March 2012)
5. San-Aniceto, I., Moreno-Vozmediano, R., Montero, R., Llorente, I.: Cloud Capacity Reservation for Optimal Service Deployment. In: IARIA Cloud Computing Conference (September 2011)
6. Chaisiri, S., Lee, B.S., Niyato, D.: Optimization of Resource Provisioning Cost in Cloud Computing. IEEE Transactions on Services Computing 5(2) (April-June 2012)
7. Smith, W., Foster, I., Taylor, V.: Scheduling with Advanced Reservations. In: International IPDPS Symposium (2000)
8. Topcuoglu, H., Hariri, S., Wu, M.Y.: Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. IEEE Transactions on Parallel and Distributed Systems 13(3) (March 2002)
9. Aoun, R., Doumith, E.A., Gagnaire, M.: Resource Provisioning for Enriched Services in Cloud Environment. In: IEEE CloudCom Conference (November-December 2010)
10. Aoun, R., Gagnaire, M.: Towards a Fairer Benefit Distribution in Grid Environments. In: IEEE/ACS AICCSA Conference (May 2009)
11. Venugopal, S., Chu, X., Buyya, R.: A Negotiation Mechanism for Advance Resource Reservations Using the Alternate Offers Protocol. In: International IWQoS Workshop (June 2008)
12. Kaushik, N., Figueira, S., Chiappari, S.: Flexible Time-Windows for Advance Reservation Scheduling. In: IEEE MASCOTS Symposium (September 2006)
13. Aoun, R., Gagnaire, M.: An Exact Optimization Tool for Market-Oriented Grid Middleware. In: IEEE CQR Workshop (May 2009)
14. Aoun, R., Gagnaire, M.: Service Differentiation Based on Flexible Time Constraints in Market-Oriented Grids. In: IEEE GLOBECOM Conference (November-December 2009)
15. Yeo, C.S., Venugopal, S., Chu, X., Buyya, R.: Autonomic Metered Pricing for a Utility Computing Service. Future Generation Computer Systems 26(8) (October 2010)
16. Netto, M.A., Bubendorfer, K., Buyya, R.: SLA-Based Advance Reservations with Flexible and Adaptive Time QoS Parameters. In: International ICSOC Conference (2007)
17. Sulistio, A., Kim, K.H., Buyya, R.: Managing Cancellations and No-Shows of Reservations with Overbooking to Increase Resource Revenue. In: IEEE CCGRID Conference (May 2008)
18. Birkenheuer, G., Brinkmann, A.: Reservation-Based Overbooking for HPC Clusters. In: IEEE CLUSTER Conference (September 2011)
19. Konstanteli, K., Kyriazis, D., Varvarigou, T., Cucinotta, T., Anastasi, G.: Real-Time Guarantees in Flexible Advance Reservations. In: IEEE COMPSAC Conference, vol. 2 (July 2009)
20. Püschel, T., Neumann, D.: Management of Cloud Infrastructures: Policy-Based Revenue Optimization. In: International ICIS Conference (December 2009)

21. Meinl, T., Anandasivam, A., Tatsubori, M.: Enabling Cloud Service Reservation with Derivatives and Yield Management. In: IEEE CEC Conference (November 2010)
22. Son, S., Sim, K.M.: A Price-and-Time-Slot-Negotiation Mechanism for Cloud Service Reservations. IEEE Transactions on Systems, Man, and Cybernetics 42(3) (June 2012)
23. Saure, D., Sheopuri, A., Qu, H., Jamjoom, H., Zeevi, A.: Time-of-Use Pricing Policies for Offering Cloud Computing as a Service. In: IEEE SOLI Conference (July 2010)
24. Yue, M.: A Simple Proof of the Inequality FFD(L) $\leqslant$ 11/9 OPT(L) + 1, $\forall L$ for the FFD bin-packing algorithm. Acta Mathematicae Applicatae Sinica (English Series) 7(4) (1991)
25. Buyya, R., Abramson, D., Giddy, J., Stockinger, H.: Economic Models for Resource Management and Scheduling in Grid Computing. Concurrency and Computation: Practice and Experience 7(13-15) (2002)
26. Parallel Workloads Archive: Logs of Real Parallel Workloads from Production Systems, `http://www.cs.huji.ac.il/labs/parallel/workload/logs.html`