Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Selling Reserved Instances in Cloud Computing

Changjun Wang[1], **Weidong Ma**[2], **Tao Qin**[2], **Xujin Chen**[3], **Xiaodong Hu**[3], **Tie-Yan Liu**[2]
[1]Beijing University of Technology, BJC-SEC, Beijing, China
[2]Microsoft Research, Beijing, China
[3]Chinese Academy of Sciences, Beijing, China
[1]wcj@amss.ac.cn, [2]{weima, taoqin, tyliu}@microsoft.com, [3]{xchen, xdhu}@amss.ac.cn

Abstract

In this paper, we study the problem of designing new mechanisms for selling reserved *instances* (also referred to as *virtual machines*) in cloud computing. Unlike the practice in today's clouds in which users only have a few predefined options to reserve instances (i.e., either 1-year reservation or 3-year reservation), we allow users to reserve resources for any length and from any time point in the future. Our goal is to maximize the social welfare. We propose two mechanisms, one for the case where all the jobs are tight (their lengths are exactly their reservation time intervals), and the other for the more general case where jobs are delayable and have some flexibility on their reservations. Both of the mechanisms are prompt in the sense that the acceptance and the payment for a job is determined at the very moment of its arrival. We use competitive analysis to evaluate the performance of our mechanisms, and show that both of the mechanisms have a competitive ratio of $O(\ln(kT))$ under some mild assumption, where k (res. T) is the maximum ratio between per-instance-hour valuation (res. length) of any two jobs. We then prove that no algorithm can achieve a competitive ratio better than $\ln(2kT)$ under the same assumption. Therefore, our mechanisms are optimal within a constant factor.

1 Introduction

Cloud computing becomes more and more popular nowadays because it can provide scalable and elastic access to IT resources and applications via the Internet [AWS, 2015; Azure, 2015; Google Cloud Platform, 2015]. Infrastructure as a service (IaaS), in which computing resources are mainly sold through *instances* (also referred to as *virtual machines*), is one of the several fundamental models of cloud computing. There are two main pricing models in today's IaaS market: the pay-as-you-go model (for on-demand instances), and the subscription model (for reserved instances).[1]

The pay-as-you-go model is the most commonly used pricing mechanism, in which the cloud provider sets a fixed per-instance-hour price, and users pay for their utilization of instance hours. In the subscription model, a user should first pay some upfront fee[2] for the instances he/she is going to reserve for a pre-defined period of future time. Then the user can use the reserved resource whenever he/she wants during the subscription period, under a significantly discounted per-instance-hour usage price.

The pay-as-you-go pricing mechanism is easy to implement, since users do not need to specify the length of their jobs at the beginning. However, the pay-as-you-go price is usually higher than the subscription price for the same type of instances. Therefore, many users prefer to reserve some resources in the near future, if their demand is predictable. However, the options for buying reserved instances through the subscription model are very limited in today's cloud industry. For example, in Amazon EC2 users can only make a 1-year or 3-year reservation, and they cannot adjust their reservation according to their real demand.

To address the limitation of the current pricing model for reserved instances, in this work we design new pricing mechanisms which allow users reserve instances depending on their own need and offer them more flexibility. In particular, we allow a user i to specify the number of instances n_i he/she needs, the start time r_i, the end time d_i, and the time length l_i of his/her reservation. Note that the reservation can be tight (i.e., $d_i - r_i = l_i$) or delayable (i.e., $d_i - r_i > l_i$). For example, a user can submit a delayable reservation saying that he/she wants to reserve 5 instances for one week during next month. When the user submits his/her reservation to the cloud, the system will post the total price for the reservation immediately. If he/she is willing to pay the price, his/her reservation will be accepted. Otherwise, his/her reservation will be rejected. Once a reservation is accepted it will never be interrupted.

We focus on posted pricing mechanisms for selling reserved instances because posted pricing mechanisms have certain advantages. For instance, a user can adjust his/her reservation interval to minimize his/her cost. He/she may also have enough time to switch to other cloud providers if his/her value is lower than the price offered by a cloud provider, since

[1]There is a third one, the auction based pricing model for spot instances. However, it is supported by only one of the mainstream cloud provider.

[2]The upfront fee could be zero depending on his/her choice.

posted pricing mechanisms provide immediate price response to his/her request. Another advantage of posted price is that users do not need to reveal their exact value to the mechanism, which is their private information.

Our goal is to maximize the social welfare. We propose two posted pricing mechanisms based on the utilization rate of the cloud. The main idea of the mechanisms is that the price of a reservation is positively correlated with the utilization rate of the cloud at its reservation period: When a reservation request arrives, if the cloud is heavily loaded (i.e. the available capacity is relatively small) during its reservation period, it needs to pay more to make it accepted; if its value is not large enough, it will be rejected, and the available capacity will be left to more valuable requests. This kind of pricing principle has already been used in many businesses such as air tickets selling, hotel reservation and so on. Different from those businesses, the cloud reservation faces more challenges. For example, the reservation may be quite flexible, starts at any time and the demands are hard to predict.

Our work can be summarized as follows:

- We first study the case in which all the reservations are tight. For this case, we propose a pricing mechanism which determines the price of a reservation based on the utilization rate of the cloud. The mechanism has a competitive ratio of $1 + 12\ln(3kT + 1)$ under a mild assumption, where k (res. T) is the maximum ratio between per-instance-hour valuation (res. length) of any two jobs.
- We then extend our result to a more general case where reservations can be delayable and have some flexibility. By slightly modifying the previous mechanism, we derive a new pricing mechanism that has the same competitive ratio.
- We also prove that no deterministic algorithm can obtain a competitive ratio less than $\ln(2kT)$ under the same assumption, which shows the optimality (within a constant factor) of our mechanisms.

2 Related Work

Because of its popularity and great impact to IT industry, cloud computing has attracted a lot of attention from research community [Abhishek *et al.*, 2012; Fox *et al.*, 2009; Funke *et al.*, 2012; Jain *et al.*, 2014; Vinothina *et al.*, 2012]. We only review those papers closely related to our work.

Our work is part of the large body of literature on designing truthful online mechanisms [Friedman and Parkes, 2003; Parkes *et al.*, 2004; Hajiaghayi, 2005; Parkes, 2007]. However, most mechanisms in these papers are auction-based mechanisms, in which the payment of each agent is determined at the (reported) deadline of his/her job, while in our mechanisms, the payment is determined right upon the arrival of the agent (i.e., posted price). In this sense, our setting is more suitable for cloud computing because if an agent immediately gets notified upon his arrival that his job cannot be completed, he can switch to other cloud computing platforms with minimum time cost.

The online resource allocation problem for cloud computing has been investigated in many papers [Mashayekhy *et al.*, 2014; Zaman and Grosu, 2012; Zhang *et al.*, 2013]. A closely related work is [Zhang *et al.*, 2013], in which the authors propose an online mechanism framework for cloud resource allocation. Different from our work, agents' valuation functions are continuous and concave in their model: An agent will get a positive value even if his/her demand is only partially satisfied, and the more resources he/she is allocated, the more value he/she will get. In our model, the valuation function of an agent is binary: He/she gets a positive value if and only if his/her reservation is accepted and fully satisfied.

Azar and Khaitsin [2011] consider an online ad placement problem which is similar to our problem. Each ad has an arrival time, a deadline, a length and a value. Similar to our work, they are interested in truthful "prompt" mechanism where the payment is determined for an agent at the very moment of his/her arrival. The difference is that the jobs (ads) in their setting are of the same length, while jobs (reservation) in our model are of different length. That is, their problem can be regarded as a special case of our problem.

Chakrabarty et al. [2013] consider the online version of knapsack problem. Both their work and our work belong to a class of online decision problem: whether to accept an item (reservation) at its arrival. The difference is that their problem is much simpler than ours. Once an item is accepted in the online knapsack problem, it will be in the knapsack forever and one does not need to concern whether to pack the item inside the knapsack. In contrast, in our model, once a reservation is accepted, we need to find a time interval for the reservation (and determine its payment), and the resources can be re-used by other jobs when the time interval of the reservation has passed.

The reservation problem of perishable products originates from the airline industry [Williamson, 1992], and then draws a lot of attention in hotel industry [Baker and Collier, 1999; Bitran and Gilbert, 1996; Bitran and Mondschein, 1995; Goldman *et al.*, 2002; McMahon-Beattie, 2002; Schütze, 2008]. Comparing to the airline reservation, the hotel reservation problem is more closely related to ours since the hotel booking requests can occur for different lengths and can therefore overlap. Bitran and Mondschein [1995] consider the room allocation problem, and formulate this problem as a stochastic and dynamic programming model. Baker and Collier [1999] compare the performances of five booking control policies. Our work has obvious differences with these work on hotel reservation. First, all these work focuses on the revenue maximization, while our goal is to maximize the social welfare. Second, they empirically evaluate the performance of their algorithms by simulation, while we theoretically study the performance of our pricing mechanisms by competitive analysis.

3 Problem Formulation

In this section, we formally setup the problem of designing pricing mechanisms for instance reservation in cloud.

Let C denote the capacity of a cloud provider, i.e., the cloud provider has C instances (virtual machines) for reservation. Customers/agents arrive sequentially and submit their reservation requests. For simplicity, we also call a reservation

j a job, which is characterized by $(r_j, d_j, l_j, n_j, v_j)$, where r_j is the earliest start time of the reservation, d_j is the latest end time of the reservation, $l_j (\leq d_j - r_j)$ is the time length of the reservation, n_j ($\ll C$) is number of instances to reserve, and v_j is the value of the reservation. We call a job *tight* (resp. *delayable*) if $l_j = d_j - r_j$ (resp. $l_j < d_j - r_j$). For any job set S, we use $v(S)$ to denote $\sum_{j \in S} v_j$.

We consider an online setting, in which jobs arrive online and the provider has no knowledge about future jobs. The agents are *self-interested* and may misreport the attributes of their reservations to be better off. Once a job j arrives, the agent (the owner of the job) submits its attributes $\hat{r}_j, \hat{d}_j, \hat{l}_j$ and $\hat{n}_j$ to the cloud system (note that $\hat{r}_j$ is not necessarily equal r_j, so are $\hat{d}_j$ etc.), the system immediately calculates the price p_j of the reservation, and then the agent decides whether to accept the price. The *utility* of j is $v_j - p_j$ if he/she accepts the price, and 0 otherwise. We assume each agent is *rational*: he/she will accept the price and buy the reserved instances if $v_j \geq p_j$, and reject the price and give up the reservation otherwise. Note that the agent does not need to reveal his/her value v_j to the system. Similar to [Hajiaghayi, 2005], we restrict the misreports of agents, i.e., $\hat{r}_j \geq r_j$, $\hat{d}_j \leq d_j$, $\hat{l}_j \geq l_j$ and $\hat{n}_j \geq n_j$.

In consideration of the potential misreport of agents, we are concerned with incentive compatible (also *truthful*) mechanisms. A mechanism is *incentive compatible* (IC) if, for any agent j, regardless of the behaviors of other agents, truthful reporting his/her job maximizes his/her utility.

The goal of the system is to design mechanisms that can maximize the social welfare (the total value of the accepted jobs). Once a job is accepted, it cannot be rejected or interrupted in the future. We use the competitive ratio [Parkes, 2007] to evaluate the performance of a mechanism, i.e., to compare the mechanism against the optimal offline solution.

Definition 1. *A mechanism $\mathcal{M}$ is β-*competitive *in terms of social welfare if for any job sequence θ, we have $SW(\mathcal{M}, \theta) \geq 1/\beta OPT(\theta)$, where $SW(\mathcal{M}, \theta)$ denotes the social welfare achieved by $\mathcal{M}$ over θ and $OPT(\theta)$ denotes the optimal social welfare over θ.*

It is easy to verify that no mechanism can achieve a constant competitive ratio if there is no restriction on lengths or value density of jobs. Therefore, we make the following assumptions.

- We assume the *per-instance-hour valuation* $\rho_j = \frac{v_j}{n_j l_j}$ of every job is supported in $[\rho_{\min}, \rho_{\max}]$.
- The length of every job falls in a known interval $[l_{\min}, l_{\max}]$.

Denote $k := \frac{\rho_{\max}}{\rho_{\min}}$ and $T := \frac{l_{\max}}{l_{\min}}$. Without loss of generality, we further assume $\rho_{\min} = 1$ and $l_{\min} = 1$.

4 The Mechanisms

In this section, we design mechanisms for selling reserved instances. To warm up, we first consider a special case where all the jobs are tight. This case reflects the way that current clouds provide reserve services. Therefore it is helpful to study it separately to provide theoretical guideline to real applications. In addition, this case is much easier and can help to demonstrate the idea of our proposal.

We then extend our result to the general case in which jobs may be delayable.

In our mechanisms, when a job j arrives, we will set a p_j for its request $(\hat{r}_j, \hat{d}_j, \hat{l}_j, \hat{n}_j)$. If the job (agent) accepts the price and pays for it for its reservation, then we say our mechanism *accepts* the job.

4.1 A Simple Case with Tight Jobs

In this subsection, we consider a simple case that all jobs are tight, i.e. $l_j = d_j - r_j, \forall j$. In other words, $[r_j, d_j]$ is the exact reservation time period for request j. In the following, we first present our post-price mechanism and then prove that the competitive ratio of the mechanism is $O(\ln(kT))$.

Let $\gamma_j(t)$ denote the *utilization rate* of the cloud at time t when job j arrives, i.e.

$$\gamma_j(t) := \sum_{i \in S_j \text{ and } \lfloor r_i \rfloor \leq t < \lceil d_i \rceil} \frac{n_i}{C},$$

where S_j denotes the set of jobs that come earlier than j and are accepted by the cloud.

Our proposed mechanism is shown in Mechanism 1, in which $f(x)$ is an auxiliary function defined as below.

$$f(x) := (3kT + 1)^x - 1$$

The intuitive requirement on the auxiliary function is that the price monotonically depends on the cloud utilization rate x: the higher the utilization rate, the higher the price. With this in mind, one can actually choose different forms of auxiliary functions, and will correspondingly obtain different competitive ratios. We show that this specific form can lead to a good competitive ratio.

Mechanism 1

Pricing rule:
When job j arrives,

- If j does not overfill the system (i.e. $\forall t \in \left[\lfloor r_j \rfloor, \lceil d_j \rceil\right)$, $\gamma_j(t) + \frac{n_j}{C} \leq 1$), set price
$$p_j = \int_{\lfloor r_j \rfloor}^{\lceil d_j \rceil} \int_{\gamma_j(t)}^{\gamma_j(t) + \frac{n_j}{C}} \frac{f(x) \cdot \rho_{\min} \cdot C}{3} dx dt;$$
- Otherwise, set price $p_j = +\infty$.

Allocation rule:
If j accepts the price p_j and pays for it, allocate n_j instances to j from r_j to d_j.

Note that when calculating the payment in the mechanism, we round the start time downwardly and the end time upwardly to integers. For example, for a job with $r_j = 0.5$ and $d_j = 1.5$, we will treat it as $r_j = 0$ and $d_j = 2$; if it is accepted, we will assume it occupies the allocated instances for the time interval $[0, 2)$ while computing the utilization rate of the cloud, even if its actual reservation interval is $[0.5, 1.5)$.

The purpose of such an operation is to ease the theoretical analysis.

Since Mechanism 1 gives a take-it-or-leave-it price, it is obviously truthful about user's private value. Recall that we restrict the misreports of each agent j to $\hat{r}_j \geq r_j$, $\hat{d}_j \leq d_j$, $\hat{l}_j \geq l_j$ and $\hat{n}_j \geq n_j$. It is easy to check that Mechanism 1 is also truthful about other attributes.

By the definition of $\gamma_j(t)$ we have the following observation.

Observation 1. *For any $t^z \in \mathbb{Z}^+$ and any j, the utilization rate $\gamma_j(t)$ during $[t^z, t^z+1)$ is a constant.*

Theorem 1. *Mechanism 1 is IC, and its competitive ratio is at most $1 + 12\ln(3kT+1)$ if all the jobs are tight and $\frac{n_j}{C} \leq \frac{\ln(3/2)}{\ln(3kT+1)}, \forall j$.*

For any job sequence, denote S as the set of jobs accepted by Mechanism 1 and S^* as the set of jobs accepted by the optimal offline algorithm. We need to prove

$$\frac{v(S^*)}{v(S)} = \frac{v(S \cap S^*) + v(S^* \setminus S)}{v(S)} < 1 + 12\ln(3kT+1).$$

Since $v(S \cap S^*) \leq v(S)$, we only need to prove

$$\frac{v(S^* \setminus S)}{v(S)} < 12\ln(3kT+1). \tag{1}$$

Let $\gamma(t)$ be the utilization rate of time t after all the jobs in S have been accepted. We will prove the following inequalities separately (Lemmas 1-2), and then Theorem 1 follows directly.

$$v(S^* \setminus S) < 2\int_0^{+\infty} f(\gamma(t)) \cdot \rho_{\min} \cdot C dt$$

$$v(S) \geq \frac{\int_0^{+\infty} f(\gamma(t)) \cdot \rho_{\min} \cdot C dt}{6\ln(3kT+1)}$$

Lemma 1. $v(S^* \setminus S) < 2\int_0^{+\infty} f(\gamma(t)) \cdot \rho_{\min} \cdot C dt$.

Proof. Note that we assume all the jobs are rational. For any job $j \in S^* \setminus S$, its rejection by the mechanism due to one of the two reasons: $v_j < p_j$ or the system will over fill if accepting j. We show that in both cases,

$$v_j \leq \int_{\lfloor r_j \rfloor}^{\lceil d_j \rceil} \frac{f(\gamma_j(t) + \frac{n_j}{C}) \cdot \rho_{\min} \cdot n_j}{3} dt. \tag{2}$$

If $v_j < p_j$, then by the definition of p_j and the monotonicity of $f(x)$, we have

$$\begin{aligned} v_j \quad &< \quad p_j = \int_{\lfloor r_j \rfloor}^{\lceil d_j \rceil} \int_{\gamma_j(t)}^{\gamma_j(t)+\frac{n_j}{C}} \frac{f(x) \cdot \rho_{\min} \cdot C}{3} dx dt \\ &\leq \quad \int_{\lfloor r_j \rfloor}^{\lceil d_j \rceil} \frac{f(\gamma_j(t) + \frac{n_j}{C}) \cdot \rho_{\min} \cdot n_j}{3} dt, \end{aligned}$$

If it is the over-fill case, by Observation 1, there must exist some time interval $[t_1, t_2) \subseteq \big[\lfloor r_j \rfloor, \lceil d_j \rceil\big)$ such that $t_2 - t_1 \geq 1$ and $\gamma_j(t) + \frac{n_j}{C} > 1$ for any $t \in [t_1, t_2)$. According to the assumptions on job's attributes, we have

$$v_j \leq kT\rho_{\min} \cdot n_j = \frac{f(1)\rho_{\min} \cdot n_j}{3},$$

which combines with $\gamma_j(t) + \frac{n_j}{C} > 1$ for any $t \in [t_1, t_2)$ implying

$$\begin{aligned} v_j \quad &\leq \quad \int_{t_1}^{t_2} \frac{f(1) \cdot \rho_{\min} \cdot n_j}{3} dt \\ &< \quad \int_{t_1}^{t_2} \frac{f(\gamma_j(t) + \frac{n_j}{C}) \cdot \rho_{\min} \cdot n_j}{3} dt \\ &\leq \quad \int_{\lfloor r_j \rfloor}^{\lceil d_j \rceil} \frac{f(\gamma_j(t) + \frac{n_j}{C}) \cdot \rho_{\min} \cdot n_j}{3} dt. \end{aligned}$$

With all the above facts, we have

$$\begin{aligned} v(S^* \setminus S) &< \sum_{j \in S^* \setminus S} \int_{\lfloor r_j \rfloor}^{\lceil d_j \rceil} \frac{f(\gamma_j(t) + \frac{n_j}{C}) \cdot \rho_{\min} \cdot n_j}{3} dt \\ &\leq \sum_{j \in S^* \setminus S} \int_{\lfloor r_j \rfloor}^{\lceil d_j \rceil} \frac{f(\gamma(t) + \frac{n_j}{C}) \cdot \rho_{\min} \cdot n_j}{3} dt \\ &= \sum_{j \in S^* \setminus S} \int_{\lfloor r_j \rfloor}^{\lceil d_j \rceil} \frac{\rho_{\min} \cdot n_j}{3} \cdot [(3kT+1)^{\frac{n_j}{C}} \cdot f(\gamma(t)) + \\ &\quad (3kT+1)^{\frac{n_j}{C}} - 1] dt. \end{aligned}$$

Recall that $\frac{n_j}{C} \leq \frac{\ln(3/2)}{\ln(3kT+1)}$. Thus we get $(3kT+1)^{\frac{n_j}{C}} \leq \frac{3}{2}$. Furthermore,

$$\begin{aligned} v(S^* \setminus S) &< \sum_{j \in S^* \setminus S} \int_{\lfloor r_j \rfloor}^{\lceil d_j \rceil} \frac{f(\gamma(t)) \cdot \rho_{\min} n_j}{2} dt \\ &\quad + \sum_{j \in S^* \setminus S} \int_{\lfloor r_j \rfloor}^{\lceil d_j \rceil} \frac{\rho_{\min} n_j}{6} dt \\ &\leq \frac{1}{2} \int_0^{+\infty} f(\gamma(t)) \cdot \rho_{\min} \cdot \Big(\sum_{j \in S^* \setminus S \text{ and } \lfloor r_j \rfloor \leq t < \lceil d_j \rceil} n_j \Big) dt \\ &\quad + \sum_{j \in S^* \setminus S} \frac{(l_j + 2)\rho_{\min} n_j}{6} \\ &\leq \int_0^{+\infty} f(\gamma(t)) \cdot \rho_{\min} \cdot C dt + \frac{v(S^* \setminus S)}{2}. \end{aligned}$$

The last inequality is from the fact that $l_j \geq 1$ and $v_j \geq l_j n_j \rho_{\min}$.

Therefore,

$$v(S^* \setminus S) < 2\int_0^{+\infty} f(\gamma(t)) \cdot \rho_{\min} \cdot C dt.$$

This completes the proof. $\square$

Lemma 2. $v(S) \geq \frac{\int_0^{+\infty} f(\gamma(t)) \cdot \rho_{\min} \cdot C dt}{6\ln(3kT+1)}$.

Proof. Since all the jobs in S are accepted by the mechanism and all the jobs are rational, we obtain

$$\begin{aligned} v(S) &\geq \sum_{j \in S} \int_{\lfloor r_j \rfloor}^{\lceil d_j \rceil} \int_{\gamma_j(t)}^{\gamma_j(t)+\frac{n_j}{C}} \frac{f(x) \cdot \rho_{\min} \cdot C}{3} dx dt \\ &= \sum_{j \in S} \int_{\lfloor r_j \rfloor}^{\lceil d_j \rceil} \int_{\gamma_j(t)}^{\gamma_j(t)+\frac{n_j}{C}} \frac{\big((3kT+1)^x - 1\big)\rho_{\min} \cdot C}{3} dx dt \end{aligned} \tag{3}$$

Since for any job j, $l_j = d_j - r_j \geq 1$ and its per-instance-hour value is at least $\rho_{\min}$, we have

$$\begin{aligned} & \sum_{j \in S} \int_{\lfloor r_j \rfloor}^{\lceil d_j \rceil} \int_{\gamma_j(t)}^{\gamma_j(t)+\frac{n_j}{C}} \frac{\rho_{\min} \cdot C}{3} dx dt \\ \leq & \sum_{j \in S} \frac{(l_j+2) \cdot \rho_{\min} \cdot n_j}{3} \leq v(S). \end{aligned} \tag{4}$$

Using (3) and (4), we have

$$\begin{aligned} 2v(S) & \geq \sum_{j \in S} \int_{\lfloor r_j \rfloor}^{\lceil d_j \rceil} \int_{\gamma_j(t)}^{\gamma_j(t)+\frac{n_j}{C}} \frac{(3kT+1)^x \cdot \rho_{\min} C}{3} dx dt \\ & = \frac{1}{3} \int_{t:\gamma(t) \neq 0} \int_0^{\gamma(t)} (3kT+1)^x \cdot \rho_{\min} C dx dt \\ & = \frac{1}{3 \ln(3kT+1)} \int_0^{+\infty} f(\gamma(t)) \cdot \rho_{\min} \cdot C dt. \end{aligned}$$

Thus we derive

$$v(S) \geq \frac{\int_0^{+\infty} f(\gamma(t)) \cdot \rho_{\min} \cdot C dt}{6 \ln(3kT+1)}$$

This completes the proof. □

4.2 The General Case with Delayable Jobs

In this subsection, we consider the general case that $l_j \leq d_j - r_j$ for any job j. For this general case, we show that with a slight modification on Mechanism 1, we obtain a new mechanism with the same competitive ratio for the general case.

In our mechanism, for every job j, we will only consider allocating it to time interval $[r_j, r_j + l_j)$ or $[t^z, t^z + l_j)$, where $t^z \in \mathbb{Z}$ and $\lceil r_j \rceil \leq t^z \leq \lfloor d_j - l_j \rfloor$. Let a_j^* denote the start time of the interval allocated to j if j is accepted by our mechanism. Apparently,

$$a_j^* \in A_j := \{r_j, \lceil r_j \rceil, \lceil r_j \rceil + 1, \ldots, \lfloor d_j - l_j \rfloor\}.$$

We still use $\gamma_j(t)$ to denote the utilization rate of every time t according to the allocation of the mechanism when j arrives, i.e.

$$\gamma_j(t) := \sum_{i \in S_j \text{ and } a_i^* \leq t < \lceil a_i^* + l_i \rceil} \frac{n_i}{C},$$

where S_j denotes the set of jobs that come earlier than j and are accepted by the cloud. Define $A_j^F \subseteq A_j$ to be the set of feasible start time that if allocate j from this time, the system will not overfill, i.e.,

$$A_j^F := \{a_j \in A_j : \forall t \in [\lfloor a_j \rfloor, \lceil a_j + l_j \rceil),\ \gamma_j(t) + \frac{n_j}{C} \leq 1\}.$$

Besides, the auxiliary function $f(x)$ is the same as that in the previous subsection.

Mechanism 2

Pricing rule:
When job j arrives,

- If $A_j^F \neq \emptyset$, set price
$$p_j = \min_{a_j \in A_j^F} \int_{\lfloor a_j \rfloor}^{\lceil a_j + l_j \rceil} \int_{\gamma_j(t)}^{\gamma_j(t)+\frac{n_j}{C}} \frac{f(x) \cdot \rho_{\min} \cdot C}{3} dx dt;$$
- Otherwise, set price $p_j = +\infty$.

Allocation rule:
If j accepts the price p_j, allocate j to time interval $[a_j^*, a_j^* + l_j)$, where

$$a_j^* = \arg \min_{a_j \in A_j^F} \int_{\lfloor a_j \rfloor}^{\lceil a_j + l_j \rceil} \int_{\gamma_j(t)}^{\gamma_j(t)+\frac{n_j}{C}} \frac{f(x) \cdot \rho_{\min} \cdot C}{3} dx dt.$$

Note in Mechanism 2, an agent still does not need to reveal its private value. Now we show the competitive ratio of Mechanism 2 is also $O(\ln(kT))$ for the case with delayable jobs.

Theorem 2. *Mechanism 2 is IC, and its competitive ratio is at most $1 + 12\ln(3kT+1)$ if $\frac{n_j}{C} \leq \frac{\ln(3/2)}{\ln(3kT+1)}, \forall j$.*

Proof. We still use S and S^* to denote the sets of jobs that are accepted by Mechanism 2 and the optimal offline algorithm respectively. For any job $j \in S^* \setminus S$, denote its allocated time interval by the optimal offline algorithm as $[t^j, t^j + l_j) \subseteq [r_j, d_j)$. Since j is rational and not accepted by our mechanism, it must be one of the following cases:

- The following inequality holds.
$$\begin{aligned} & \int_{\lfloor t^j \rfloor}^{\lceil t^j + l_j \rceil} \int_{\gamma_j(t)}^{\gamma_j(t)+\frac{n_j}{C}} \frac{f(x) \cdot \rho_{\min} \cdot C}{3} dx dt \\ \geq & \min_{a_j \in A_j^F} \int_{\lfloor s_j \rfloor}^{\lceil s_j + l_j \rceil} \int_{\gamma_j(t)}^{\gamma_j(t)+\frac{n_j}{C}} \frac{f(x) \cdot \rho_{\min} \cdot C}{3} dx dt \\ > & v_j \end{aligned}$$
- There exists some time period $[t_1, t_2) \subseteq [\lfloor t^j \rfloor, \lceil s_j + l_j \rceil)$ such that $\gamma_j(t) + \frac{n_j}{C} > 1, \forall t \in [t_1, t_2)$ and $t_2 - t_1 \geq 1$.

Then similar to the proof of Theorem 1, we can still prove

$$v(S^* \setminus S) < 2 \int_0^{+\infty} f(\gamma(t)) \cdot \rho_{\min} \cdot C dt$$

and

$$v(S) \geq \frac{\int_0^{+\infty} f(\gamma(t)) \cdot \rho_{\min} \cdot C dt}{6 \ln(3kT+1)},$$

where $\gamma(t)$ is the final utilization rate after all jobs in S are accepted by Mechanism 2. Thus

$$\frac{v(S^*)}{v(S)} = \frac{v(S \cap S^*) + v(S^* \setminus S)}{v(S)} < 1 + 12 \ln(3kT+1).$$

This completes the proof. □

4.3 Optimality of Our Mechanisms

One may wonder whether the competitive ratio bound of our mechanisms depends on the assumption that $\frac{n_j}{C} \le \frac{\ln(3/2)}{\ln(3kT+1)}$, $\forall j$. In fact, without any restriction on n_j and C, no deterministic algorithm can obtain a competitive ration better than $C \cdot kT$. This can be seen from the following example.

Example 3. *Consider two jobs. The first job has unit length, unit demand, unit value, and its reservation interval is $[0, 1]$. For the second job, its length is T, its number of machines is C, its value is $C \cdot kT$ and its reservation interval is $[0, T]$, i.e., their types are $(0, 1, 1, 1, 1)$ and $(0, T, T, C, C \cdot kT)$, respectively. The first job comes earlier than the second one. If an algorithm accepts the first job, the second job will not be accepted, and then the competitive ratio is at least $C \cdot kT$. If the mechanism does not accept the first job, the second job will not show up, and then the competitive ratio is ∞.*

Besides, in today's cloud industry, the capacity C is very large, e.g., a provider usually hosts tens of thousands of servers which can serve even more virtual instances. Therefore, our assumption holds in practice.

In the following, we give a lower bound $\Omega(\ln(kT))$ on the competitive ratio of any deterministic algorithm for the case $n_j = 1, \forall j$. This bound shows that our mechanisms are optimal within a constant factor.

Theorem 3. *The competitive ratio of any deterministic algorithm for this problem is at least $\ln(2kT)$.*

Proof. We prove the theorem by constructing an example and show that no deterministic algorithm performs well for this example.

In this example we assume $n_j = 1$ for any job j. Therefore the maximum value of any job j is at most kT.

Given any parameter $\eta > 0$. Let

$$M = \log_{1+\eta}(2kT) = \frac{\ln(2kT)}{\ln(1+\eta)} \text{ and } M' = \ln_{1+\eta}(kT).$$

Suppose there are $M + 1$ bundles of jobs. For $i = 0, 1, \ldots, M'$, the i-th bundle consists of C tight jobs with value $(1 + \eta)^i$, and all the C jobs' reservation periods include a same time interval $[t, t + 1]$. For $i = M' + 1, \ldots, M$, the i-th bundle consists of $2C$ tight jobs with value $\frac{1}{2}(1+\eta)^i$, half of which end at $t+0.5$, and the others start at $t+0.5$. Besides, the i-th bundle comes earlier than the $(i + 1)$-th bundle for $i = 1, 2, \ldots, M - 1$.

Given any online algorithm. For $i = 1, 2, \ldots, M'$, let x_i be the number of jobs in the i-th bundle accepted by the algorithm. For $i = M' + 1, \ldots, M$, let y_i (resp. z_i) be the number of accepted jobs in i-th bundle that end at $t + 0.5$ (resp. start at $t + 0.5$), and x_i be $\frac{y_i+z_i}{2}$. Since we can only accept at most C jobs that reserving a same time interval, thus we have

$$\sum_{i=0}^{M'} x_i + \sum_{i=M'+1}^{M} y_i \le C$$

$$\sum_{i=0}^{M'} x_i + \sum_{i=M'+1}^{M} z_i \le C$$

Adding the above two inequalities, we have

$$\sum_{i=0}^{M} x_i \le C.$$

We first prove that there always exists some h ($1 \le h \le M$) such that

$$\sum_{i=0}^{h} \frac{x_i \cdot (1+\eta)^i}{(1+\eta)^h} \le \frac{C(\eta+1)}{\eta M}.$$

Denote $S_j = \sum_{i=0}^{j} \frac{x_i \cdot (1+\eta)^i}{(1+\eta)^j}$. Then we have

$$\sum_{j=1}^{M} S_j = \sum_{j=1}^{M}\sum_{i=0}^{j} \frac{x_i \cdot (1+\eta)^i}{(1+\eta)^j} \le \sum_{i=0}^{M} x_i \cdot (\sum_{j=i}^{M} (1+\eta)^{i-j})$$
$$\le \sum_{i=0}^{M} x_i \frac{\eta+1}{\eta} \le \frac{C(\eta+1)}{\eta}.$$

Since all $S_j \ge 0$, thus there must exist some $h \in [1, M]$ such that $S_h \le \frac{C(\eta+1)}{\eta M}$.

If we only use the first $h + 1$ bundles as input, then the jobs that are accepted by the algorithm must be the same as the previous one ($M + 1$ bundles), and the total value of the accepted jobs is

$$\sum_{i=0}^{h} x_i \cdot (1+\eta)^i.$$

However, apparently the optimal offline algorithm is to accept all the jobs in the h-th bundle whose social welfare is $C \cdot (1 + \eta)^h$. Then the competitive ratio of the given online algorithm is at least

$$\frac{\eta M}{1+\eta} = \frac{\ln(2kT)}{1+\eta} \cdot \frac{\eta}{\ln(1+\eta)}.$$

Since $\lim_{\eta \to 0} \frac{\eta}{\ln(1+\eta)} = 1$, we derive the conclusion by setting $\eta \to 0$. $\square$

5 Future Work

In this work, we have designed two mechanisms with worst-case performance guarantee for selling reserved instances in cloud. There are multiple aspects to investigate to make the mechanisms practically useful.

First, we have assumed that the agents know the lengths of their reservations and required them to report their job lengths. It is possible that they do not have this knowledge. The cloud provider may need to take this responsibility. How to estimate the length of a job is an independent research topic.

Second, we have considered only one cloud provider. In today's cloud market, there are usually multiple providers offering the same or similar service. It is interesting to consider the competition between cloud providers.

Third, we have focused on social welfare maximization. Another research topic is revenue maximization for the cloud provider.

References

[Abhishek *et al.*, 2012] Vineet Abhishek, Ian A Kash, and Peter Key. Fixed and market pricing for cloud services. *arXiv preprint arXiv:1201.5621*, 2012.

[AWS, 2015] Amazon web services, 2015. http://aws.amazon.com/.

[Azar and Khaitsin, 2011] Yossi Azar and Ety Khaitsin. Prompt mechanism for ad placement over time. In *Algorithmic Game Theory*, pages 19–30. Springer, 2011.

[Azure, 2015] Microsoft azure, 2015. http://azure.microsoft.com/en-us/.

[Baker and Collier, 1999] Timothy Kevin Baker and David A Collier. A comparative revenue analysis of hotel yield management heuristics. *Decision Sciences*, 30(1):239–263, 1999.

[Bitran and Gilbert, 1996] Gabriel R Bitran and Stephen M Gilbert. Managing hotel reservations with uncertain arrivals. *Operations Research*, 44(1):35–49, 1996.

[Bitran and Mondschein, 1995] Gabriel R Bitran and Susana V Mondschein. An application of yield management to the hotel industry considering multiple day stays. *Operations Research*, 43(3):427–443, 1995.

[Chakrabarty *et al.*, 2013] Deeparnab Chakrabarty, Yunhong Zhou, and Rajan Lukose. Online knapsack problems. 2013.

[Fox *et al.*, 2009] Armando Fox, Rean Griffith, A Joseph, R Katz, A Konwinski, G Lee, D Patterson, A Rabkin, and I Stoica. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28:13, 2009.

[Friedman and Parkes, 2003] Eric J Friedman and David C Parkes. Pricing wifi at starbucks: issues in online mechanism design. In *Proceedings of the 4th ACM conference on Electronic commerce*, pages 240–241. ACM, 2003.

[Funke *et al.*, 2012] Daniel Funke, Fabian Brosig, and Michael Faber. Towards truthful resource reservation in cloud computing. In *Performance Evaluation Methodologies and Tools (VALUETOOLS), 2012 6th International Conference on*, pages 253–262. IEEE, 2012.

[Goldman *et al.*, 2002] Paul Goldman, Richard Freling, Kevin Pak, and Nanda Piersma. Models and techniques for hotel revenue management using a rolling horizon. *Journal of Revenue and Pricing Management*, 1(3):207–219, 2002.

[Google Cloud Platform, 2015] Google cloud platform, 2015. https://cloud.google.com/.

[Hajiaghayi, 2005] Mohammad T Hajiaghayi. Online auctions with re-usable goods. In *Proceedings of the 6th ACM conference on Electronic commerce*, pages 165–174. ACM, 2005.

[Jain *et al.*, 2014] Navendu Jain, Ishai Menache, Joseph Seffi Naor, and Jonathan Yaniv. A truthful mechanism for value-based scheduling in cloud computing. *Theory of Computing Systems*, 54(3):388–406, 2014.

[Mashayekhy *et al.*, 2014] Lena Mashayekhy, Mahyar Movahed Nejad, Daniel Grosu, and Athanasios V Vasilakos. Incentive-compatible online mechanisms for resource provisioning and allocation in clouds. *Memory (GB)*, 1(3.75):7–5, 2014.

[McMahon-Beattie, 2002] Una McMahon-Beattie. The strategy and tactics of pricing: A guide to profitable decision making. *Journal of Revenue and Pricing Management*, 1(3):286–287, 2002.

[Parkes *et al.*, 2004] David C. Parkes, Satinder P. Singh, and Dimah Yanovsky. Approximately efficient online mechanism design. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pages 1049–1056, 2004.

[Parkes, 2007] David C Parkes. Online mechanisms. 2007.

[Schütze, 2008] Jörg Schütze. Pricing strategies for perishable products: the case of vienna and the hotel reservation system hrs. com. *Central European Journal of Operations Research*, 16(1):43–66, 2008.

[Vinothina *et al.*, 2012] V Vinothina, R Sridaran, and Padmavathi Ganapathi. A survey on resource allocation strategies in cloud computing. *International Journal of Advanced Computer Science & Applications*, 3(6), 2012.

[Williamson, 1992] Elizabeth Louise Williamson. Airline network seat inventory control: Methodologies and revenue impacts. Technical report, [Cambridge, Mass.: Massachusetts Institute of Technology, Dept. of Aeronautics & Astronautics], Flight Transportation Laboratory,[1992], 1992.

[Zaman and Grosu, 2012] Sharrukh Zaman and Daniel Grosu. An online mechanism for dynamic vm provisioning and allocation in clouds. In *5th International Conference on Cloud Computing (CLOUD)*, pages 253–260. IEEE, 2012.

[Zhang *et al.*, 2013] Hong Zhang, Bo Li, Hongbo Jiang, Fangming Liu, Athanasios V Vasilakos, and Jiangchuan Liu. A framework for truthful online auctions in cloud computing with heterogeneous user demands. In *Proceedings of INFOCOM*, pages 1510–1518. IEEE, 2013.