# A Double Auction Mechanism to Bridge Users' Task Requirements and Providers' Resources in Two-Sided Cloud Markets

Li Lu, Jiadi Yu*, *Member, IEEE,* Yanmin Zhu, *Member, IEEE,* and Minglu Li

**Abstract**—Double auction-based pricing model is an efficient pricing model to balance users' and providers' benefits. Existing double auction mechanisms usually require both users and providers to bid with the unit price and the number of VMs. However, in practice users seldom know the exact number of VMs that meets their task requirements, which leads to users' task requirements inconsistent with providers' resource. In this paper, we propose a truthful double auction mechanism, including a matching process as well as a pricing and VM allocation scheme, to bridge users' task requirements and providers' resources in two-sided cloud markets. In the matching process, we design a cost-aware resource algorithm based on Lyapunov optimization techniques to precisely obtain the number of VMs that meets users' task requirements. In the pricing and VM allocation scheme, we apply the idea of second-price auction to determine the final price and the number of provisioned VMs in the double auction. We theoretically prove our proposed mechanism is individual-rational, truthful and budget-balanced, and analyze the optimality of proposed algorithm. Through simulation experiments, the results show that the individual profits achieved by our algorithm are 12.35% and 11.02% larger than that of scale-out and greedy scale-up algorithms respectively for 90% of users, and the social welfare of our mechanism is only 7.01% smaller than that of the optimum mechanism in the worst case.

**Index Terms**—Double auction mechanism, two-sided cloud markets, VM trading, Lyapunov optimization.

## 1 Introduction

In cloud trading markets, providers sell resources, such as VMs, while users purchase resources for tasks. Every participant in such a market aims to maximize their individual profits. Pricing is a critical factor to balance benefits of both users and providers. Typically, an efficient pricing model can maximize individual profits of each participant in the trading. However, although providers offer multiple pricing models, e.g., Amazon EC2 [1] employs on-demand, reserve and spot pricing models, and Microsoft Azure [2] employs pay-as-you-go, resellers pricing models etc., users still cannot easily make an optimum resource provisioning decision to meet their requirements. This is because the prior knowledge of users in cloud trading is quite different from that of providers, i.e., users can only assess the computing requirement of their tasks, and cannot match these requirements to the capability of VMs that providers offer.

Although IaaS providers usually present the capacity of their VMs, such as the number of vCPU, memory, etc, the actual capability, i.e., the computational or storage capability, of VMs are not clear for users. For example, a user intends to provision VMs for a website, whose daily requests are about 100,000. However, IaaS providers like AWS only provide the capacity of VMs. Thus, it is necessary for the user to assess the actual capability of every VM types and select appropriate one for the website. To help users select VMs and deploy their applications on IaaS cloud, many software vendors even provide guides [3], [4], [5], [6]. For example, Scylla provides some performance evaluation results on deploying NoSQL on AWS EC2, which help users to choose appropriate VMs for NoSQL [3]. However, these guides are only suggestions, with which users still need to select VMs manually, which obstruct users to adopt cloud as the main computing paradigm. This problem would be more emerging in two-sided cloud markets.

A two-sided cloud market [7] means a user can purchase cloud resources from multiple providers, meanwhile a provider can sell cloud resources to multiple users, which is illustrated in Fig. 1. Since different providers offer different pricing models, it is more difficult and confusing for users to make an optimum provisioning decision. Double auction is an efficient trading mechanism for two-sided markets, which enables users and providers bid with their own needs. Li et al. [8] design a double auction mechanism to enable a federation of providers trade with each other. Also, Samimi et al. [9] propose a combinatorial double auction to enable users bid with combinations of resources. However, there is an underlying assumption in these existing mechanisms which is not realistic, i,e., users and providers in the trading need to know the capability of each VM, and further check whether the capability of provisioned resources can meet the requirement of their tasks. In practice, users usually do not precisely know the capability of each VM offered by IaaS providers. Therefore, it is necessary to design a double auction mechanism which bridges users' task requirements and providers' cloud resources.

To design a double auction mechanism for bridging users' task requirements and providers' resources in IaaS

- *L. Lu, J. Yu, Y. Zhu, and M. Li are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, P.R.China, 200240.*
  *E-mail: {luli_jtu, jiadiyu, yzhu, mlli}@sjtu.edu.cn.*
- *(Corresponding Author: Jiadi Yu.)*

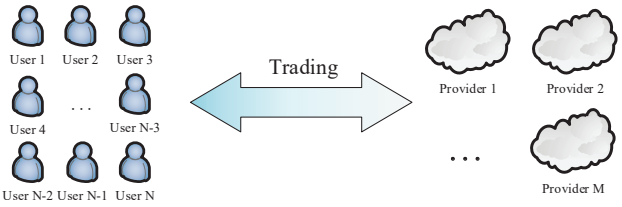



Fig. 1. Illustration of a two-sided cloud markets.

cloud, we face several challenges as follows. First, since users' task requirements are different from providers' resources, i.e., users' task requirements are presented in a computing capability form while providers' resources are shown in VM form, it is hard to bridge the bids between users and providers in the double auction mechanism. Next, a double auction mechanism should be individual-rational, truthful and budget-balanced, but respectively applying truthful auction mechanisms for users and for providers cannot lead to a truthful double auction in two-sided cloud markets. Finally, since the workload of users' tasks fluctuates over time, users' actual task requirements for resources cannot be accurately determined.

In this paper, we present a truthful double auction mechanism for VM trading to bridge users' task requirements and providers' resources in two-sided cloud markets. To implement the double auction mechanism, we first model the VM trading with double auction, and then propose a matching process as well as a pricing and VM allocation scheme for the double auction mechanism. In the matching process, to precisely obtain the number of VMs that needs to be provisioned for users' task requirements under workload fluctuation, we formulate the VM provisioning problem as an optimization problem, and solve it through Lyapunov optimization techniques. In the pricing and VM allocation scheme, we apply the idea of second-price auction to determine the final price and the number of allocated VMs of the double auction based on the matching results. Meanwhile, we prove the proposed double auction mechanism is individual-rational, truthful and budget-balanced, and analyze the optimality of the proposed algorithm. Finally, based on TPC-W workload and OpenStack performance data, we conduct simulations to evaluate the performance of our mechanism as well as algorithm, and results show that our mechanism and algorithm are feasible and efficient.

We highlight our main contributions as follows:

- We model the VM trading with double auction, and design a double auction mechanism to bridge users' task requirements and providers' resources in two-sided markets, from which both cloud providers and users are able to benefit.
- We formulate the VM provisioning problem as an optimization problem to minimize users' cost, and propose a cost-aware resource provisioning algorithm to solve the problem using Lyapunov optimization techniques.
- We theoretically prove that the proposed double auction mechanism is individual-rational, truthful as well as budget-balanced, and analyze the optimality of the relative cost-aware resource provisioning algorithm.

In the rest of this paper, the related works are reviewed in Section 2. Then we model the VM trading in two-sided markets with double auction, and formulate the VM provisioning problem as an optimization problem in Section 3. Section 4 and Section 5 present design details of a double auction mechanism for two-sided markets and a coat-aware resource provisioning algorithm respectively. In Section 6, we prove the proposed double auction mechanism satisfying individual rationality, truthfulness as well as budget balance, and analyze the optimality of the proposed algorithm. Section 7 shows our performance evaluation results. Finally, we make a conclusion in Section 8.

## 2 Related Work

There are some pricing models in commercial cloud platforms, such as on-demand, reserve [1] and spot [10] pricing models. Google Cloud Platform [11] even provides a slight amount of permanent free resources. Cloud providers employ these pricing models to incentivize different kinds of users to employ their cloud resources. Under these pricing models, many researches [12], [13], [14], [15], [16] focus on determining provisioning decisions to optimize users' profits. [12], [13], [14] analyze the spot pricing model for AWS EC2, and derive optimal bidding strategies for users based on provider pricing model. [15], [16] design provisioning algorithms based on users' needs under on-demand and reserve pricing models. Recently, some works [17], [18], [19] introduce the concept of group buying into cloud resources trading and enable users be more confident to adopt cloud computing as their main computing paradigm. However, these works only focus on determining VM provisioning under fixed pricing models, which do not consider the relationship between demand and supply in cloud markets.

To take the relationship between demand and supply into consideration, there are some works [20], [21], [22], [23], [24] focusing on designing auction-based pricing models. [20], [21] propose online combinatorial auction mechanisms to optimize long-term efficiency and model heterogeneous VMs in practice, which enable users bid with a VM provisioning decision instead of the number of each type VMs. These approaches release users from selecting specific type VMs, and improve the profits of users in the trading. [22], [23], [24] take users' heterogeneous demands into consideration and propose online algorithms based on auction framework. More recently, some works [25], [26], [27], [28] employ game theory for dynamic pricing of cloud resources and further improve users' profits in cloud resource trading. However, since there is only one provider in the auction, users do not have the choice of purchasing resource from multiple providers to achieve higher cost-efficiency.

As the development of secure data collaboration techniques [29], [30], cloud integration services become popular [31], [32], [33], which leads to a trend that multiple users and providers participate in cloud trading, which depicts a two-sided cloud market. To enable multiple users and multiple providers participate in the trading, double auction-based pricing models in two-sided markets attract attentions of researchers. Li et al. [8] propose a double auction mechanism in a cloud federation market, in which a cloud has the roles of both buyer and seller. The buy-bids and sell-bids

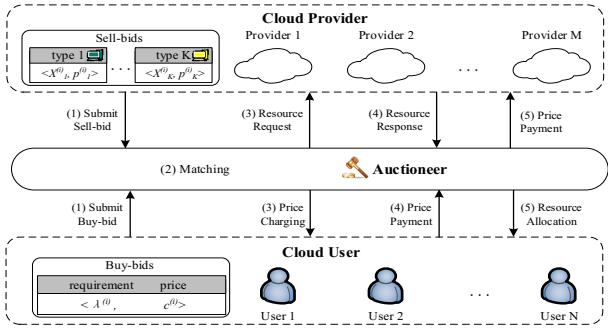



Fig. 2. Workflow in a two-sided market with double auction.

are all multi-unit bids in this mechanism. They also derive true values of bids for providers in the cloud federation market. Samimi et al. [9] design a combinatorial double auction mechanism for resources allocation. This mechanism enables participants submit combinatorial bids. Zheng et al. [34] design a double auction for network resources instead of typical computing resources. Some other works [35], [36] consider the fairness in cloud resource trading, which reduce the probability of user drop problem and ensure the long-term developments of cloud markets. All these works assume users have the prior knowledge of providers' VM capabilities. However, in real cloud trading markets, users' task requirements and providers' resources are presented in different forms, i.e., users only know the computing requirements of their tasks instead of providers' VM capabilities. Although cloud providers usually offer detailed VM capacity information [1], such as the number of vCPU, memory, etc, it is difficult for users to match their computing requirements with the various VM capacities, and further incapable to select appropriate VMs for the tasks [3], [4], [5]. Such a problem is more emerging when different cloud providers are introduced. For example, Scylla provides some performance evaluation results on deploying NoSQL on AWS EC2, which help users to choose appropriate VMs for NoSQL on AWS [3], while TechRepublic provides a guide to help users choose appropriate VMs in Windows Azure [6]. Thus, it is necessary to automatically bridge users' task requirements and providers' resources.

Our work focus on designing a double auction mechanism for two-sided markets to bridge users' task requirements and providers' cloud resources in their bids.

## 3 System Model

In this section, we first model the VM trading in two-sided markets with double auction, and then formulate a VM provisioning problem.

### 3.1 Modeling VM Trading in Two-Sided Markets with Double Auction

In a VM trading two-sided market, multiple cloud providers sell their VMs, while multiple users purchase VMs for their tasks. We assume that there are $N$ users and $M$ providers in the market. Since predefined VM types are beneficial to improve the utilization of resources, and popular cloud platforms [1], [2] adopt this resource encapsulation manner, we also treat VMs as resource units. Each cloud provider has $K$ types of VMs. Users are encouraged to choose multiple types of VMs to finish their tasks under different pricing models because of higher cost-efficiency [15]. To incentivize users and providers participating in VM trading of the two-sided market, a double auction-based pricing model can be adopted.

Fig. 2 shows the workflow of the VM trading in a two-sided market with double auction. First, both users and providers submit their buy-bids and sell-bids to an auctioneer respectively. Then the auctioneer matches winners of users and providers based on these bids through a matching process. Next, the auctioneer informs users and providers of matching results, and requests payments from users as well as resources from providers. After that, users and providers submit payment and resources to the auctioneer respectively. Finally, the auctioneer allocates resources to users and completes payment to providers.

Under the double auction framework, We first describe bids of users and providers, and the actual price and actual amount of resources determined by the auctioneer.

*Users (Buyers)*: A buy-bid for a user $i$ is denoted as $b^{(i)} =< \lambda^{(i)}, c^{(i)} >$, where $\lambda^{(i)}$ is the task requirement of user $i$, while $c^{(i)}$ is the payment for the VM provisioning decision in a unit time. Users buy VMs from cloud to finish one task, whose revenue denotes as $R^{(i)}$. Moreover, $\widetilde{\lambda}^{(i)}$ and $\widetilde{c}^{(i)}$ are true values of the task requirement $\lambda^{(i)}$ and the payment $c^{(i)}$ respectively, i.e., the task requirements of tasks that users really intend to handle and the maximum price that users are willing to pay. Users manipulate their bids strategically to optimize their own profits.

*Providers (Sellers)*: A sell-bid for a provider $j$ is denoted as $< X_m^{(j)}, p_m^{(j)} >, \forall m \in \{1, \ldots . K\}$, where $X_m^{(j)}$ is the maximum number of type-$m$ VMs that the provider $j$ is able to sell, and $p_m^{(j)}$ is the unit price of type-$m$ VMs. Running a type-$m$ VM will incur an operational cost $C_m^{(j)}$. Moreover, $\widetilde{X}_m^{(j)}$ and $\widetilde{p}_m^{(j)}$ are true values of the maximum number of VMs $X_m^{(j)}$ and the unit price $p_m^{(j)}$ respectively, i.e., the maximum amount of type-$m$ VMs that providers really intend to sell, and the unit price that providers are willing to charge. Similar to users, providers manipulate their bids strategically to optimize their own profits.

*Auctioneer*: To avoid the risk of unfair double auction, we assume the auctioneer is a third-party platform. The auctioneer executes the double auction mechanism for users and providers, after it collects all buy-bids and sell-bids. In the double auction, the auctioneer determines the actual paying price $\widehat{c}^{(i)}$ and actual task requirement $\widehat{\lambda}^{(i)}$ of a user $i$, as well as the actual charging price $\widehat{p}_m^{(j)}$ and actual number $\widehat{X}_m^{(j)}$ of a provider $j$ for type-$m$ VMs. In real world, there are many third-party cloud integration service providers [31], [32], [33], which integrate the cloud resources from different cloud providers, and provide users a unified API to utilize the resources, whose functions are similar to that of the auctioneer. Therefore, the cloud integration service providers can be seen as the auctioneer in the double auction.

After the definitions above, the utilities of buyers and sellers under the double auction are defined as follows.

**Definition 1.** *(Utility of Buyer): For a user $i$, the utility of the user is the difference between the task revenue and payment for*

*the VM provisioning decision, i.e.,*

$$w_b^{(i)} = \begin{cases} R^{(i)} - \widehat{c}^{(i)} & \textit{user } i \textit{ wins the buy-bid} \\ 0 & \textit{otherwise.} \end{cases} \quad (1)$$

**Definition 2.** *(Utility of Seller): For a provider $j$, the utility of the provider is the difference between the VM charging price and operational cost of all VM types, i.e.,*

$$w_s^{(j)} = \sum_{m=1}^{K} w_m^{(j)}, \quad (2)$$

*where*

$$w_m^{(j)} = \begin{cases} \widehat{X}_m^{(j)} \widehat{p}_m^{(j)} - C_m^{(j)} & \textit{provider } j \textit{ wins the sell-bid} \\ 0 & \textit{otherwise.} \end{cases}$$

Based on the definition of utilities for buyers and sellers, the social welfare can be defined.

**Definition 3.** *(Social Welfare): The social welfare is the sum of the utilities of both buyers and sellers in the auction, i.e.,*

$$w_{sw} = \sum_{i=1}^{N} w_b^{(i)} + \sum_{j=1}^{M} w_s^{(j)}. \quad (3)$$

Through the description of basic notations in the double auction above, we model the VM trading scenario in two-sided markets, in which bid-forms of users and providers, as well as utilities of users, providers and the social welfare are defined.

### 3.2 Formulating VM Provisioning Problem

From the definition above, we can see that users' task requirements are in the form of service rate $\lambda^{(i)}$, while providers' resources are in the form of different VM types $X_m^{(j)}, \forall m \in \{1, \cdots, K\}$. It is hard for the auctioneer to determine the winner from users and providers based on these bids. To bridge the difference between users' task requirement and providers' resources, we formulate a VM provisioning problem which transforming the task requirement in users' bids to VM types.

For higher utilization of cloud resources, providers usually adopt non-linear pricing models to incentivize users to purchase heterogeneous VMs [16]. Utilizing heterogeneous VMs to finish users' tasks, VM migrations cannot be avoidable [37]. Specifically, assume the old provisioning decision is $x^{old}$ and new provisioning decision is $x^{new}$, whose $m^{th}$ entry means the number of type-$m$ VMs. If some types of VMs need to be provisioned while other types of VMs need to be removed in the new provisioning decision, i.e., $x^{new} - x^{old} \neq 0$, the data from removed VMs should be transferred to the new VMs, which leads to VM migrations. Thus, it is necessary to balance the cost and migration delay in the VM provisioning decision making.

We first investigate the migration mechanism in the cloud. Usually, if a user needs to start a new VM, the old VM first shutdowns and uploads the image to image server, and then the new VM downloads the image from image server and starts. Thus, the transferring delay is doubled and the migration delay $\alpha(t)$ is

$$\alpha(t) = 2\frac{\mathcal{D}(t)}{\Theta} + \Phi, \quad (4)$$

where $\Phi$ is the VM start time, $\Theta$ is the bandwidth and $\mathcal{D}$ is the image size. $\Phi$ is a constant. Since the image transferring can be distinguished into inner-cloud and inter-cloud transferring, for the bandwidth $\Theta$, there are two different bandwidths accordingly, i.e., the inner-cloud bandwidth $\Theta_1$ and inter-cloud bandwidth $\Theta_2$. $\Theta$ is subject to a Bernoulli distribution $B(\Theta_1, \Theta_2, p)$, where $p$ is the occurrence possibilities of the inner-cloud transferring. Thus, $\Theta$ can be set as the expected value, i.e.,

$$E(\Theta) = \Theta_1 p + \Theta_2(1-p). \quad (5)$$

For the image size $\mathcal{D}(t)$, it is related to VM provisioning decisions between two constitute time slots, i.e.,

$$\mathcal{D}(t) = d(t) \sum_{x_m(t-1) > x_m(t)} |x_m(t-1) - x_m(t)|, \quad (6)$$

where $d(t)$ is the average image size of one VM. Since all images should be uploaded to the image server before starting a new VM, the image size $\mathcal{D}(t)$ equals the volume of old disk images. To ensure users can achieve stable performance with our VM provisioning decision, the inter-cloud bandwidth $\Theta_2$ is set as a constant. In real world, many cloud providers offer minimum network bandwidth guarantees in Service Level Agreement (SLA) [38], [39]. We utilize the expectation of all providers' minimum bandwidths in the double auction as the inter-cloud bandwidth $\Theta_2$, to ensure users can achieve good performance under any situations.

Except for the delay in migrations of VMs, there may be migration costs while a VM is migrated from one cloud provider to another one. But in real world, many cloud providers offer free migration services, such as Amazon provides the server migration service[6], which is free for users to migrate the workload to AWS [40]. Thus, we regard the migration cost as zero in the formulation of VM provisioning problem.

In two-sided markets, users' goal is to minimize their provisioning costs, while constraints are to ensure that provisioned VMs meet performance needs, and to control the cumulative delay in a reasonable range. Since this VM provisioning problem is under the double auction, there would be an additional constraint, i.e., the provisioning cost should not be larger than the cost in users' bids. The migration delay is considered as the main source of the cumulative delay constraint due to the fact that the migration delay is far larger than other kinds of delay, such as the data transferring delay. Moreover, since the provisioning cost and cumulative delay monotonically increase as time goes on, we minimize time-averaged provisioning costs under the time-averaged cumulative delay constraint. Therefore, the problem is formulated as

$$\begin{aligned} \min \quad & \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{m=1}^{K} x_m^{(i)}(t)\, \rho_m \\ s.t \quad & \sum_{m=1}^{K} x_m^{(i)}(t)\, \mu_m \geq \lambda^{(i)} \quad \forall t \\ & \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \alpha(t) \leq \mathcal{T} \\ & \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{m=1}^{K} x_m^{(i)}(t)\, \rho_m \leq c^{(i)} \\ & x_m^{(i)}(t) \in N \quad \forall t, m, \end{aligned} \quad (7)$$

where $\rho_m$ is the expected unit price of type-$m$ VMs, $\mu_m$ is the maximum service rate of type-$m$ VMs, $\lambda^{(i)}$ is the user $i$'s task requirement, $c^{(i)}$ is the costs in user $i$'s bid, $\mathcal{T}$ is the maximum tolerable delay, and $x_m(t)$ is the number of type-$m$ VMs. The number of type-$m$ VMs $x_m(t)$ is determined by solving optimization problem (7) in the time slot $t$.

## 4 Double Auction Mechanism

In this section, we design a double auction mechanism for two-sided markets to bridge users' task requirements and providers' resources.

In a two-sided cloud market, the task requirement in a buy-bid of a user $i$ is $\lambda^{(i)}$, and the number of type-$m$ VMs in a sell-bid of a provider $j$ is $X_m^{(j)}$. We assume that the resources from providers are always larger than that requested by users, i.e., providers have infinite resources in the view of users, which is consistent with reality. So, in our work, users do not need to know the maximum resources amount of providers. Since users need sufficient VMs to finish their tasks while they are unwilling to pay more for extra VMs, they have no reason to submit a task requirement which is not their true value. Hence, in our mechanism, we consider $\lambda^{(i)}$ in buy-bids and $X_m^{(j)}$ in sell-bids are always true values, while the payment $c^{(i)}$ in buy-bids and unit price $p_m^{(j)}$ in sell-bids are strategically determined by users and providers respectively.

In the mechanism, actions conducted by the auctioneer can be divided into two steps. The first step is that the auctioneer determines winners from users and providers at the beginning of each time slot, i.e., winners matching process. The second step is that the auctioneer determines the price and amount of resources. In the following, we elaborate the details of these two steps accordingly.

1) **Matching**: Since the paying price in buy-bids and charging price in sell-bids are different, the auctioneer cannot find the critical price to match winners from users and providers. Hence, the auctioneer first calculates the bid density of each buy-bid, which is defined as

$$ppr^{(i)} = \frac{c^{(i)}}{x^{(i)T}\mu}, \tag{8}$$

where $\mu = (\mu_1, \ldots, \mu_K)^T$ is the maximum service rate for each type of VMs, and $x^{(i)} = (x_1^{(i)}, \ldots, x_K^{(i)})^T$ is the provisioning decision of user $i$. The provisioning decision $x^{(i)}$ can be determined with $\lambda^{(i)}$ and $c^{(i)}$ in user' $i$'s bid through solving the optimization problem (7), which is described in Section 5. The bid density can be expressed as price-performance ratio, i.e., the price per service capability. Higher bid density means the user is willing to pay higher price for resources. Hence, based on the bid density of buy-bids and charging price in sell-bids, the auctioneer can find an approximate critical price in the double auction.

For buy-bids, the auctioneer sorts all buy-bids based on bid densities in a descending order, i.e.,

$$\phi^{(1)} \geq \cdots \geq \phi^{(i)} \geq \cdots \geq \phi^{(N)}. \tag{9}$$

For sell-bids, the auctioneer sorts all sell-bids for type-$m$ VMs based on unit price in an ascending order, i.e.,

$$\theta_m^{(1)} \leq \cdots \leq \theta_m^{(j)} \leq \cdots \leq \theta_m^{(M)}, \forall m \in \{1, \ldots, K\}. \tag{10}$$

**Algorithm 1** Double Auction Mechanism

**Input:** $K$: the number of VM types
$N$: the number of users
$M$: the number of providers
$b^{(i)} =< \lambda^{(i)}, c^{(i)} >, \forall i \in [1, N]$: buy-bids of $N$ users
$s_m^{(j)} =< X_m^{(j)}, p_m^{(j)} >, \forall j \in [1, M], \forall m \in [1, K]$: sell-bids of $M$ providers and $K$ VM types
**Output:** $Buyers$: the winning buyers list
$Sellers$: the winning sellers list
$Buy_p$: the cost of the winning buyers list
$Sell_p$: the unit cost of the winning sellers list
$Buy_r$: the allocated computing capabilities to the winning buyers in $Buyers$
$Sell_r$: the amount of allocated resources from the winning sellers in $Sellers$

```
while Eq. (11) is satisfies do
    obtain the VM provisioning decision x^(i), ∀i ∈ [1, ··· , N] through solving the optimization problem (7), i.e., x^(i) = CA-RP(λ^(i), c^(i), V).
    traverse all buy-bids and sell-bids, then find the one winning buyer and multiple winning sellers based on Eq. (11).
    add the winning buyer to the list Buyer, and add the winning sellers to the list Sellers.
    calculate the payment of the winning buyer and the unit price of the winning sellers based on Eq. (12) and (13) respectively, as well as the amount of allocated resources based on Eq. (14) and (15).
    add the payment of winning buyer and the unit price of winning sellers to the lists Buy_p and Sell_p respectively, as well as add the amount of allocated resources to the lists Buy_r and Sell_r respectively.
    remove the winning buyer and the allocated resources of the winning sellers from the auction.
end while
return Buyers, Sellers, Buy_p, Sell_p, Buy_r, Sell_r
```

After both buy-bids and sell-bids are sorted, the auctioneer judges whether there is any user or provider wins the auction through the inequality as follows,

$$\phi^{(2)} \cdot x^{(i')T}\mu \geq \sum_{m=1}^{K} x_m^{(i')}\theta_m^{(j^*)}, \tag{11}$$

where $i'$ is the index of the buy-bid whose bid density is the largest. If there exists a $j = j^*$ satisfying the Eq. (11), the $i'$-th buy-bid (i.e., $ppr^{(i')} = \phi^{(1)}$) wins the auction as well as all sell-bids with $p_m^{(j)} \geq \theta_m^{(j^*)}$ (not include the $j^*$) win the auction, i.e., sell-bids with $\theta_m^{(1)}, \ldots, \theta_m^{(j^*-1)}$ win the sell-bid. If there are several $j$ satisfying the Eq. (11), the largest $j$ is chosen as $j^*$. Otherwise, no one wins the auction. Here, the second largest bid density $\phi^{(2)}$, and the $j^*$th smallest charging price $\theta^{(j^*)}$ are critical prices in buy-bids and sell-bids respectively.

2) **Pricing and VMs Allocation**: In double auction markets, it is an NP-hard problem to determine the price [41]. Thus, we apply another pricing scheme which is similar to second-price auction [42] based on our matching results:

- the price paid by the winning buyer is

$$\widehat{c}^{(win)} = \phi^{(2)} \cdot x^{(win)T}\mu; \tag{12}$$

- the unit price of type-$m$ VMs charged by the winning seller $win_j$ is

$$\widehat{p}_m^{(win_j)} = \theta_m^{(j^*)}. \tag{13}$$

The winning buyer is the user with the largest bid density. To ensure that buyers bid with their true values, the winning buyer cannot pay what it bids. Since Eq. (11) enables the second largest bid density to satisfy the budget-balance property, the winning buyer pays $\phi^{(2)} \cdot x^{(win)T}\mu$. Similarly, winning sellers cannot pay what they bid. Winning sellers charge with $\theta^{(j^*)}$ to maintain the budget-balance property.

Moreover, the number of VMs is:

- the computing capability purchased by the winning buyer to meet the task requirement is

$$\widehat{\lambda}^{(win)} = \sum_{m=1}^{K} \mu_m \cdot \sum_{\substack{j \in \{i | p_m^{(i)} = \theta_m^{(k)}, \\ k=1,\ldots,j^*-1\}}} \widehat{X}_m^{(j)}; \tag{14}$$

- the number of type-$m$ VMs sold by the winning seller $win_j$ is

$$\widehat{X}_m^{(win_j)} = \begin{cases} \lceil \dfrac{x_m^{(win)}}{(j^*-1)} \rceil & \dfrac{x_m^{(win)}}{(j^*-1)} \leq X_m^{(win_j)} \\ X_m^{(win_j)} & otherwise. \end{cases} \tag{15}$$

Each winning seller allocates the same amount of resources to the winning buyer. Since $j^*$ is determined in the matching process, the time complexity of pricing and VMs allocation scheme is $O(1)$.

The matching process above chooses only one winning buyer in the double auction, which is far from efficient. Thus, to achieve efficiency while ensuring truthfulness, we further perform the mechanism as Algorithm 1 shows. Obviously, Eq. (11) cannot be satisfied when the round of the double auction tends to infinity. We assume in the worst case, it consumes $k$ rounds to finish the double auction. Thus, the time complexity of the mechanism is $O(k \cdot (M \cdot K + 1))$.

Overall, we present the double auction mechanism above, which consist of winners matching process, as well as pricing and VMs allocation scheme. Our proposed mechanism satisfies three economic properties of double auction. 1) **Individual Rationality**: both buyers and sellers obtain a nonnegative profit by participating in the double auction. 2) **Truthfulness**: both buyers and sellers cannot achieve higher profits bidding without their true values. 3) **Budget Balance**: the auctioneer would not pay for the extra surplus, i.e, the total payment collected from buyers should be larger than the charging price of sellers. The detailed proofs of these properties are described in Section 6.

## 5 Cost-Aware Resource Provisioning Algorithm

As mentioned in Section 4, the matching process of the designed double auction mechanism needs to get the provisioning decision $x^{(i)}, \forall i \in [1, N]$ for determining the bid density of each buyer. And the provisioning decision $x^{(i)}$ can be determined through solving the optimization problem (7). In this section, we propose a cost-aware resource provisioning algorithm based on Lyapunov optimization techniques [43] to get the provisioning decision for the double auction mechanism.

Since there are one time-averaged objective and two time-averaged constraints in the optimization problem (7), it is difficult to utilize convex optimization solving techniques to solve the problem. We first simplify the problem (7). In the problem (7), we formulate the provisioning cost as the product of provisioning decision and the expected unit price of VMs. Since users cannot predict the sell prices of providers, the expected unit price is usually different from the paying price in the double auction, i.e., $\rho_m \neq p_m$, which further leading to the inequality between the expected cost and actual cost in the double auction, i.e.,

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{m=1}^{K} x_m^{(i)}(t)\rho_m \neq \widehat{c}^{(i)}. \tag{16}$$

Thus, the third constraint of the problem (7) depends on the property of the double auction mechanism. Since the double auction mechanism satisfies individual rationality(which is described in Section 6, it is not necessary to guarantee the third constraint in the problem (7), i.e., this constraint can be eliminated.

After the simplification, there are only one time-averaged objective and one time-averaged constraint in the problem (7). Thus, we can employ Lyapunov optimization techniques to solve it.

During a time slot $t$, the cumulative delay increases or decreases, which seems like a queue with arrival and departure. Thus, we first introduce a virtual queue which is defined as:

$$Q(t) = \begin{cases} max\{Q(t-1) + \alpha(t-1) - \mathcal{T}, 0\} & t > 0 \\ 0 & t = 0, \end{cases} \tag{17}$$

where $\alpha(t)$ is the cumulative delay, and $\mathcal{T}$ is the maximum tolerable cumulative delay.

**Theorem 1.** *(Queue Stability): the cumulative delay constraint of the optimization problem (7), i.e.,*

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \alpha(t) \leq \mathcal{T}, \tag{18}$$

*is satisfied if and only if the virtual queue is stable, i.e.,*

$$\lim_{T\to\infty} \frac{Q(t)}{T} = 0. \tag{19}$$

*Proof.* According to Eq. (17), we have:

$$Q\left(t+1\right) \geq Q\left(t\right) + \alpha(t) - \mathcal{T}. \tag{20}$$

Applying Eq. (4) to Eq. (20), we sum up both sides of the equality above over time slots $t \in \{0, T-1\}$, and divide it with $T$. Then Letting $T \to \infty$, and applying the initial status of virtual queue, i.e., $Q\left(0\right) = 0$, we have

$$\lim_{T\to\infty} \frac{Q\left(T\right)}{T} \geq \frac{2}{\Theta} \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathcal{D}\left(t\right) + \Phi - \mathcal{T}.$$

Then applying the cumulative delay constraint, the necessity is proved.

When the length of virtual queue is large, it means that the cumulative delay has far exceeded the time that user can tolerant. Since the virtual queue is stable, i.e., $\lim_{T\to\infty}\frac{Q(t)}{T}=0$, we could get:

$$\frac{2}{\Theta}\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathcal{D}\left(t\right)+\Phi-\mathcal{T}\leq\lim_{T\to\infty}\frac{Q(T)}{T}=0.$$

Thus, the sufficiency is proved.

Therefore, the cumulative delay constraint is equivalent to the stability of the virtual queue, i.e., Eq. (18) is equivalent to Eq. (19). □

According to Theorem 1, the cumulative delay constraint in the optimization problem (7) is transformed to the stability of the virtual queue $Q(t)$.

Then, we formulate the stability of the virtual queue based on the Lyapunov optimization framework. For each time slot, the quadratic Lyapunov function is defined as

$$L(t)=\frac{1}{2}Q(t)^2. \tag{21}$$

Based on the Lyapunov function, the Lyapunov drift is defined as the difference between backlogs of the queue in two constitute time slots, i.e.,

$$\Delta L(t)=E\{L(t+1)-L(t)|Q(t)\}. \tag{22}$$

The Lyapunov drift $\Delta L(t)$ is defined as a tool to measure the stability of the virtual queue.

Under the framework of Lyapunov optimization techniques, there is an upper bound of the Lyapunov drift.

**Theorem 2.** *(Upper Bound of Lyapunov Drift): For any time slot t, given any possible VM provisioning decision, the Lyapunov drift $\Delta L\left(t\right)$ could be bounded as follows*

$$\Delta L\left(t\right)\leq M+Q\left(t\right)E\{2\frac{\mathcal{D}\left(t\right)}{\Theta}+B|Q\left(t\right)\}, \tag{23}$$

*where* $M=\frac{1}{2}\left(2\frac{\mathcal{D}_{max}}{\Theta}+\Phi-\mathcal{T}\right)^2$, *and* $B=\Phi-\mathcal{T}$.

*Proof.* First, applying Lyapunov function to Lyapunov drift, the Lyapunov drift is derived as:

$$\Delta L\left(t\right)=\frac{1}{2}E\{Q^2\left(t+1\right)-Q^2\left(t\right)|Q\left(t\right)\}. \tag{24}$$

Combined with Eq. (17), $\max^2\{a,0\}\leq a^2$ and $B=\Phi-\mathcal{T}$, we have

$$\begin{aligned}Q^2\left(t+1\right)\leq Q^2\left(t\right)+\left(2\frac{\mathcal{D}_{max}}{\Theta}+B\right)^2\\+2Q\left(t\right)\left(2\frac{\mathcal{D}\left(t\right)}{\Theta}+B\right),\end{aligned} \tag{25}$$

where $D_{max}$ denotes the maximum disk image size.

Applying Eq. (25) to Eq. (24), the Lyapunov drift is further transformed to Eq. (23). □

Based on the Lyapunov optimization framework, we take the Lyapunov drift $\Delta L(t)$ combined with a penalty term $V\mathcal{C}^{(i)}(t)$ as the objective, i.e.,

$$\Delta L(t)+V\mathcal{C}^{(i)}(t),$$

**Algorithm 2** Cost-Aware Resource Provisioning Algorithm (CA-RP)

**Input:** $\lambda^{(i)}$: the task requirement in user $i$'s bid
$c^{(i)}$: the provisioning cost in user $i$'s bid
$V$: the weight under Lyapunov optimization
**Output:** $x^{(i)}$: the VM provisioning decision

```
initialize several parameters, i.e.,
  Θ: the network bandwidth,
  Φ: the startup time of a VM,
  T: the maximum tolerable cumulative delay,
  ρ_m: the expected cost of VM type m,
  μ_m: the maximum service rate of VM type m.
for t in [0, T] do
  if t=0 then
    Q(t) = 0
    D(t) = d(t) Σ |x_m(t)|
  else
    if Q(t-1) + α(t-1) - T > 0 then
      Q(t) = Q(t-1) + α(t-1) - T
    else
      Q(t) = 0
    end if
    D(t) = d(t) Σ_{x_m(t-1)>x_m(t)} |x_m(t-1) - x_m(t)|
  end if
  C^(i)(t) = Σ_{m=1}^{K} x_m^(i)(t) ρ_m
  B = Φ - T
  construct the one-slot Lyapunov optimization problem as Eq. (27).
  solve the one-slot Lyapunov optimization problem using gradient descent method, and get the VM provisioning decision x^(i).
end for
return x^(i)
```

where $V$ is the weight under the Lyapunov optimization framework, $\mathcal{C}^{(i)}\left(t\right)$ denotes the provisioning cost $\sum_{m=1}^{K}x_m^{(i)}\left(t\right)\rho_m$ for simplicity. According to Theorem 2, we have

$$\begin{aligned}\Delta L(t)+V\mathcal{C}^{(i)}(t)\leq&M+V\mathcal{C}^{(i)}(t)\\&+Q(t)E\{2\frac{\mathcal{D}(t)}{\Theta}+B|Q(t)\}.\end{aligned} \tag{26}$$

Since $M$ is a constant, it can be eliminated from the objective. Hence, the optimization problem (7) is transformed to

$$\begin{aligned}\min\quad&V\mathcal{C}^{(i)}(t)+Q(t)\left(2\frac{\mathcal{D}(t)}{\Theta}+B\right)\\s.t\quad&\sum_{m=1}^{K}x_m^{(i)}(t)\mu_m\geq\lambda^{(i)}(t)\quad\forall t\\&x_m^{(i)}(t)\in N\qquad\forall t,m.\end{aligned} \tag{27}$$

Through the analysis above, the time-averaged optimization problem (7) is transformed to the one-slot optimization problem (27). Users can determine their bids by solving problem (27) to achieve their maximum profits. There is a weight $V$ representing a tradeoff between the cumulative delay and provisioning cost in problem (27). By tuning $V$, users can either minimize the provisioning cost without consideration of the delay, or ensure the cumulative delay does

not exceed the maximum tolerable range without pursuing provisioning cost minimization. The cost-aware resource provisioning algorithm is shown in Algorithm 2. Through cost-aware resource provisioning algorithm, the matching process of the double auction mechanism can obtain the provisioning decision $x^{(i)}$ to determine the winning buyers and winning sellers.

## 6 Theoretical Analysis

In this section, we first prove that our double auction mechanism satisfies three economic properties, and then analyze the optimality of the cost-aware resource provisioning algorithm.

### 6.1 Properties of Double Auction Mechanism

As mentioned in Section 4, the designed double auction mechanism should satisfy three economic properties: 1) Individual Rationality; 2) Truthfulness; 3) Budget Balance. In the mechanism, since the matching as well as pricing and VMs allocation processes in different rounds are independent with each other, we only need to prove each property in one round of the double auction.

**Theorem 3.** *(Individual Rationality): No winning buyer pays more than its buy-bid price, and no winning seller is paid less than its sell-bid price, i.e.,*

$$\widehat{c}^{(i)} \leq c^{(i)} \quad \& \quad \widehat{p}_m^{(j)} \geq p_m^{(j)}, \quad \forall i \in \{1,\ldots,N\}, \; j \in \{1,\ldots,M\}, \; m \in \{1,\ldots,K\}. \tag{28}$$

*Proof.* For a winner of the buy-bid: if a user $i$ wins the buy-bid, the bid density $ppr^{(i)}$ is the largest bid density among all buy-bids. And the payment in the bid is

$$c^{(i)} = ppr^{(i)} \cdot x^{(i)T}\mu. \tag{29}$$

Assume $\widehat{ppr}^{(i)}$ is the actual bid density of the payment. According to the matching process, we have $\widehat{ppr}^{(i)} = \phi^{(2)}$. Further, the actual payment is

$$\widehat{c}^{(i)} = \widehat{ppr}^{(i)} \cdot x^{(i)T}\mu = \phi^{(2)} \cdot x^{(i)T}\mu. \tag{30}$$

Since $ppr^{(i)} \geq \phi^{(2)}$, we have $c^{(i)} \geq \widehat{c}^{(i)}$ based on Eq. (29) and (30).

For a winner of the sell-bid: if a provider $j$ wins the sell-bid of type $m$, $p_m^{(j)}$ is among the lowest $(j^*-1)$ unit prices of all sell-bids. According to the matching process, the actual unit price from the provider $j$ is

$$\widehat{p}_m^{(j)} = \theta_m^{(j^*)}.$$

Since $p_m^{(j)} \leq \theta_m^{(j^*)}$, we have $p_m^{(j)} \leq \widehat{p}_m^{(j)}$. □

Before we prove the second property of the mechanism, i.e., truthfulness, we first present two related lemmas.

**Lemma 1.** *(Monotonic winner determination): Given buy-bids $\{b^{(1)},\ldots,b^{(N)}\}$ and sell-bids $\{s_m^{(1)}\ldots,s_m^{(M)}\}(\forall m \in \{1,\ldots,K\})$, we have*

*1) If a user $i$ wins the buy-bid by bidding with $b^{(i)}$, then the user $i$ would also win by bidding with $b^{'}$, in which the bid density $ppr^{'} > ppr^{(i)}$.*

*2) If a provider $j$ wins the sell-bid by bidding with $s_m^{(j)}$, then the provider $j$ would also win the sell-bid by bidding with $s_m^{'}$, in which the unit price $p_m^{'} < p_m^{(j)}$.*

*Proof.* We prove the above cases respectively.

1) Since a user $i$ wins the buy-bid with $b^{(i)} =< \lambda^{(i)}, c^{(i)} >$, we know the corresponding bid density $ppr^{(i)}$ is the largest one, i.e.,

$$ppr^{(i)} \geq ppr^{(k)}, \forall k \in [1,N], k \neq i.$$

With $ppr^{'} > ppr^{(i)}$, we have

$$ppr^{'} > ppr^{(k)}, \forall k \in [1,N], k \neq i.$$

Thus, if the user $i$ proposes a buy-bid with $b^{'}$, it can still win the buy-bid according to our matching decision, because the winner has the largest bid density.

2) Since a provider $j$ wins the sell-bid with $p_m^{(j)}$ for type-$m$ VM, we know that

$$p_m^{(j)} \leq \theta_m^{(j^*)},$$

where $j^*$ is the critical index described above. With $p_m^{'} < p_m^{(j)}$, we also ensure that

$$p_m^{'} < \theta_m^{(j^*)},$$

which means the $j^{th}$ provider can also win the sell-bid.

Thus, we prove the matching process satisfies the monotonic property. □

**Lemma 2.** *(Bid-independent pricing): Given buy-bids $\{b^{(1)},\ldots,b^{(N)}\}$ and sell-bids $\{s_m^{(1)},\ldots,s_m^{(M)}\}(\forall m \in \{1,\ldots,K\})$, we have:*

*1) If a user $i$ wins the buy-bid by bidding with $b^{(i)}$ and $b^{'}$, the paying price $\widehat{c}^{(i)}$ is the same for both.*

*2) If a provider $j$ wins the sell-bid by bidding with $s_m^{(j)}$ and $s^{'}$, the unit price $\widehat{p}_m^{(j)}$ paid to provider $j$ is the same for both.*

*Proof.* We prove the above cases respectively.

1) Since a user $i$ wins the buy-bid, it is charged with $\phi^{(2)} \cdot x^{(j)T}\mu$. Also, we have $ppr^{(i)} \geq \phi^{(2)}$ and $ppr^{'} \geq \phi^{(2)}$. And as long as user i wins the buy-bid, the value of the second largest bid density $\phi^{(2)}$ would not change. Thus, the charged price of the user $i$ is independent with $ppr^{(i)}$ and $ppr^{'}$. Since the bid $b^{(i)} =< \lambda^{(i)}, c^{(i)} >$, and Eq. (8), the charged price is independent with $b^{(i)}$ and $b^{'}$.

2) Since a provider $j$ wins the sell-bid, the price paid to the provider $j$ is $\theta_m^{(j^*)}$, which is the $j^*$th lowest sell-bid price. According to our winner determination, we have $p_m^{(j)} \leq \theta_m^{(j^*)}$ and $p^{'} \leq \theta_m^{(j^*)}$. As long as provider $j$ wins the sell-bid, $p_m^{(j)}$ and $p^{'}$ must be ones among the $1 \sim j^*-1$ sell-bids. Thus, the price paid to the provider $j$ is independent with $p_m^{(j)}$ and $p^{'}$. Since the bid $s_m^{(j)} =< X_m^{(j)}, p_m^{(j)} >$, the price paid to the provider $j$ is independent with $s_m^{(j)}$ and $s^{'}$.

Thus, we prove the pricing process satisfies bid-independent property. □

**Theorem 4.** *(Truthfulness): The truthfulness of an auction mechanism means no buyers (users) or sellers (providers) can achieve higher profits by bidding with values other than its true values of the buy or sell bids, i.e., if a buyer bids with $c^{(i)} \neq \widetilde{c}^{(i)}$, the utility*

*it achieves $w_b^{(i)} \leq \widetilde{w}_b^{(i)}$; or if a seller bids with $p_m^{(j)} \neq \widetilde{p}_m^{(j)}$, the utility it achieves $w_s^{(j)} \leq \widetilde{w}_s^{(j)}$.*

*Proof.* We prove truthfulness from two sides.

**For User**: there are two cases as follows.

**Case 1**: if a user $i$ wins the buy-bid with truthful bidding, the paying price is $\phi^{(2)} \cdot x^{(i)T}\mu$. If the user $i$ bids untruthfully with $c^{(i)} > \widetilde{c}^{(i)}$, we know that the user $i$ still wins the bid without any utility increase according to Lemma 1; if the user $i$ bids untruthfully with $c^{(i)} < \widetilde{c}^{(i)}$, we know that the user $i$ may either win the bid with no change in utility according to Lemma 2, or lose the bid with reduction in utility.

**Case 2**: if a user $i$ loses the buy-bid with truthful bidding, the paying price would be 0. If the user $i$ bids untruthfully with $c^{(i)} > \widetilde{c}^{(i)}$, we know that the user $i$ may either win the bid with a non-positive utility gain, or lose the bid with no change in its utility; if the user $i$ bids untruthfully with $c^{(i)} < \widetilde{c}^{(i)}$, the user $i$ still loses the bid and the paying price is zero.

Thus, we know that users cannot achieve higher utility by bidding untruthfully.

**For Provider**: there are also two cases as follows.

**Case 1**: if a provider $j$ wins the sell-bid with truthful bidding for type-$m$ VMs, the charging price is $\theta_m^{(j^*)}$. If the provider $j$ bid untruthfully with $p_m^{(j)} < \widetilde{p}_m^{(j)}$, according to Lemma 1, we know that the provider $j$ still wins the bid without any utility increase; if the provider $j$ bid untruthfully with $p_m^{(j)} > \widetilde{p}_m^{(j)}$, we know that the provider $j$ may either win the bid with no change in its utility according to Lemma 2, or lose the bid with a reduction in its utility.

**Case 2**: if a provider $j$ loses the sell-bid with truthful bidding for type-$m$ VMs, the charging price is 0. If the provider $j$ bids untruthfully with $p_m^{(j)} < \widetilde{p}_m^{(j)}$, the provider $j$ may either win the bid with a non-positive utility gain or lose the bid leading to no change in the utility; if the provider $j$ bids untruthfully with $p_m^{(j)} > \widetilde{p}_m^{(j)}$, since the true value $\widetilde{p}_m^{(j)}$ cannot win the sell-bid, a higher price still lose the bid, leading to no change in the charging price and provider's utility.

Thus, we know that providers cannot achieve higher utility by bidding untruthfully. □

**Theorem 5.** *(Budget Balance): For the auctioneer, the total payment collected from users is no less than the overall price charged by providers, i.e.,*

$$\sum_{i=1}^{N}(\widehat{c}^{(i)} - \sum_{j=1}^{M}\sum_{m=1}^{K}\widehat{p}_m^{(j)}x_m^{(i)}) \geq 0.$$

*Proof.* We define the paying price for buy-bids and charging price for sell-bids based on the proposed double auction mechanism. Then we prove the auctioneer does not pay extra surplus for the auction.

**Payment of buyers**: According to the winners matching process, only the buyer with the largest bid density wins the buy-bid. We assume that user $i$ wins the bid, then the payment is

$$payment = \phi^{(2)} \cdot x^{(i)T}\mu. \tag{31}$$

**Overall price of sellers**: According to the winners matching process, we determine the critical index $j^*$ based on Eq. (11). Thus the overall price charged by providers is

$$overall\_price = \sum_{m=1}^{K} x_m^{(i)}\theta_m^{(j^*)}. \tag{32}$$

From Eq. (31), (32) and (11), we have

$$\sum_{i=1}^{N}(\widehat{c}^{(i)} - \sum_{j=1}^{M}\sum_{m=1}^{K}\widehat{p}_m^{(j)}x_m^{(i)})$$
$$=\phi^{(2)} \cdot x^{(i)T}\mu - \sum_{m=1}^{K} x_m^{(i)}\theta_m^{(j^*)} \geq 0$$

Therefore, the budget balance property is guaranteed. □

According to Theorem 3, 4 and 5, these three economic properties are satisfied by the proposed double auction mechanism to bridge users' task requirements and providers' resources.

### 6.2 Optimality of Cost-Aware Resource Provisioning Algorithm

In problem (27), there is a weight $V$ which represents a tradeoff between the provisioning cost and cumulative delay. We analyze the optimality of the cost-aware resource provisioning algorithm based on the weight $V$.

We assume that a user $i$ sets a fixed $V$ ($V > 0$), which is independent with the virtual queue status, and applies the algorithm to determine the provisioning decision. The optimal provisioning cost in theory is defined as

$$\mathcal{C}^* = \frac{1}{T}\sum_{t=0}^{T-1}\mathcal{C}^{(i)}(t). \tag{33}$$

Based on the cumulative delay constraint in problem (7), we have $E\{\alpha(t)\} \leq \mathcal{T}$. Thus, there exists an $\epsilon$ leading to

$$E\{\alpha(t)\} \leq \mathcal{T} - \epsilon, \tag{34}$$

where $\epsilon > 0$. Taking advantages of the definition of $\Delta L(t)$, we sum up both sides of Eq. (26) over time slot $t \in \{0, \ldots, T-1\}$, in which all inner terms are eliminated. Also, we apply $P^*$ to it and have

$$\frac{V}{T}\sum_{t=0}^{T-1}\mathcal{C}^{(i)}(t) + \frac{L(T) - L(0)}{T} \leq M + V\mathcal{C}^* - \frac{\epsilon}{T}\sum_{t=0}^{T-1}Q(t). \tag{35}$$

Letting $T \to \infty$, since $L(T)$ would not be infinity and $L(0)$ is a constant, we further have

$$\frac{L(T) - L(0)}{T} = 0.$$

On the one hand, since $\frac{V}{T}\sum_{t=0}^{T-1}\mathcal{C}^{(i)}(t) > 0$ for any $T$, we let $T \to \infty$ in Eq. (35), and have

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}Q(t) \leq \frac{M + V\mathcal{C}^*}{\epsilon}. \tag{36}$$

The left side of Eq. (36) is the time-averaged delay. According to Eq. (36), the time-averaged cumulative delay

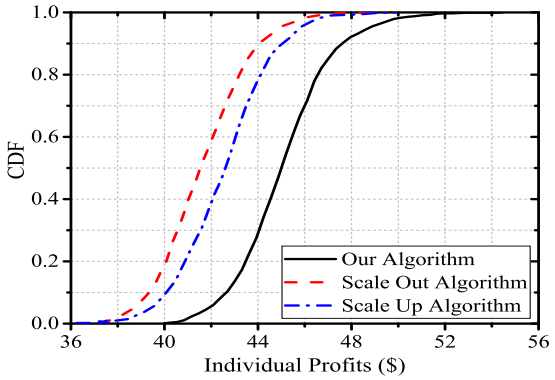



Fig. 3. CDF of individual profits under three different algorithms.

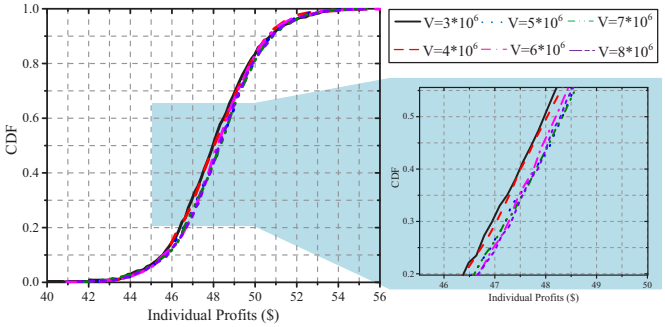



Fig. 4. CDF of individual profits under six different weight $V$s.

has a upper bound, i.e., $\frac{M+VC^*}{\epsilon}$, which is related to the weight $V$. If $V \to 0$, the upper bound $\frac{M+VC^*}{\epsilon} \to \frac{M}{\epsilon}$ which is a constant. Thus, the cumulative delay cannot exceed the value. If $V \to \infty$, the upper bound $\frac{M+VC^*}{\epsilon} \to \infty$, which implies the cumulative delay tends to infinity.

On the other hand, since $\frac{\epsilon}{T}\sum_{t=0}^{T-1} Q(t) \geq 0$ for any $T$, from Eq. (35), we have

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathcal{C}^{(i)}(t) \leq \frac{M}{V} + \mathcal{C}^*. \tag{37}$$

Similar to Eq. (36), we know there is an upper bound $\frac{M}{V} + \mathcal{C}^*$ of the time-averaged provisioning cost. If $V \to 0$, the upper bound $\frac{M}{V} + \mathcal{C}^* \to \infty$, i.e., the provisioning cost tends to infinity. If $V \to \infty$, the upper bound $\frac{M}{V} + \mathcal{C}^* \to \mathcal{C}^*$, which is the optimal value of the provisioning cost in theory. Thus, the provisioning cost tends to the optimal value $\mathcal{C}^*$.

We know that $V$ is a weight which represents a trade-off between the provisioning cost and cumulative delay. When $V \to 0$, the cumulative delay is controlled while the provisioning cost tends to infinity. When $V \to \infty$, the provisioning cost tends to the optimal value while the cumulative delay is out of control. Thus, users can tune weight $V$ to meet their actual needs, and achieve optimality by using the online algorithm.

## 7 Experimental Results

In this section, we conduct simulations to evaluate the performance of the proposed double auction mechanism and cost-aware resource provisioning algorithm.

### 7.1 Simulation Setup

In the simulations, we consider 5 types of VMs shown in Table 1, whose information is from AWS EC2 [1]. The price in Table 1 is the unit price under on-demand pricing model. We configure the OpenStack [44] platform with 5 VM types described in Table 1, and then get the maximum service rate of these 5 VM types, i.e., $\mu$, by running TPC-W [45] in each type of VMs. We also simulate the activities of a business-oriented transactional web server based on TPC-W as the task requirement $\lambda$.

There are 20 cloud providers in the simulations. The number of servers for each provider is in the range of $[500, 750]$. Each server can provide $[40, 20, 10, 5, 2]$ VMs of each type respectively. The unit price and maximum number for an arbitrary type of VMs are different among different cloud providers. To calculate the operational cost for providers, we consider the cooling power consumption and server power consumption. Since the cooling power consumption and server power consumption are both at $1kW/h$, each server consumes power at $2kW/h$ in total. Combined with unit electricity prices for four geographic locations [46], we get the operational cost for each cloud provider.

In the simulations, there are 1,000 users participating in the double auction. The task requirement of their tasks is generated by TPC-W, which is very close to the task requirement of e-commercial websites in real world [45]. The unit price that users would like to pay in their bids is lower than unit price in Table 1. Since the unit time in commercial cloud platforms is 1 hour, we set one time slot the same as it.

### 7.2 Evaluation of Individual Profits

We compare the performance of our algorithm with two baseline algorithms, i.e., the scale-out and greedy scale-up algorithms. The *scale-out algorithm* [47] is the most popular algorithm in practice, in which a user first determines the type and number of VMs as needed, and then decides the payment in bids based on the type and number of VMs. For *greedy scale-up algorithm* [48], a user chooses to provision a more powerful VM firstly instead of increase the number of VMs. If the most powerful VM still cannot meet the user's needs, the user increases the number of VMs. In these two algorithms, the expected unit cost $\rho_m$ of an arbitrary type-$m$ VMs is set the same as that in our algorithm. We also implement an offline *optimum mechanism* for the double auction, which aims to maximize the social welfare and is not truthful.

TABLE 1
VM Configurations and Prices in AWS

| VM type | Configurations | Price/h |
|---|---|---|
| m4.large | 2 vCPUs, 6.5 ECU, 8G RAM | $0.979 |
| m4.xlarge | 4 vCPUs, 13 ECU, 16G RAM | $1.226 |
| m4.2xlarge | 8 vCPUs, 26 ECU, 32G RAM | $2.553 |
| m4.4xlarge | 16 vCPUs, 53.5 ECU, 64G RAM | $5.057 |
| m4.10xlarge | 40 vCPUs, 124.5 ECU, 160G RAM | $12.838 |
| Selected Region | Asia Pacific (Singapore) | |
| OS | Windows with SQL Standard | |

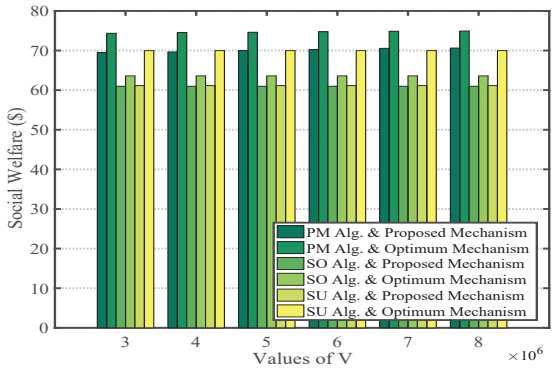



Fig. 5. Social welfare with different weight $V$ under different mechanism. PM is the proposed profit maximization algorithm, SO is the scale out algorithm, and SU is the greedy scale up algorithm.

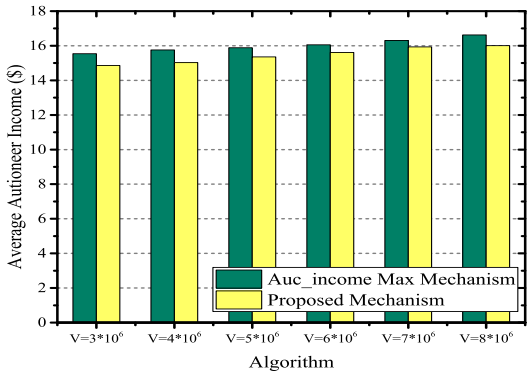



Fig. 6. Income of the auctioneer under different algorithms with optimum and proposed mechanisms.

Fig. 3 shows the CDF of individual profits under our algorithm, scale-out and scale-up algorithms. We can see that the individual profits under our algorithm are significantly larger than that under two other algorithms. For 90% of users, the individual profits under our algorithm are 12.35% and 11.02% larger than that under scale-out and greedy scale-up algorithms respectively. Fig. 4 shows the CDF of individual profits under our algorithm with 6 different weight $V$s. We can find that the individual profits increases as the value of weight $V$ increases, which is consistent with our theoretical analysis before.

### 7.3 Evaluation of Social Welfare

We also compare the time-averaged social welfare of the proposed algorithm with that of two other algorithms with different weight $V$ under the proposed and optimum mechanisms, as shown in Fig. 5. We can find that the proposed algorithm achieves larger social welfare than two other algorithms with an arbitrary weight $V$, under arbitrary mechanism. In the worst case, the social welfare in proposed algorithm is 14.92% and 14.73% larger than that in scale-out and greedy scale-up algorithms under proposed mechanism respectively. Also, we find that although the social welfare under the proposed mechanism is smaller than that under the optimum mechanism, the differences between them are all smaller than 8%. In the worst case, the social welfare under the proposed mechanism is only 7.01% smaller than that under the optimum mechanism. Since the optimum mechanism does not consider truthfulness in the double auction, the proposed mechanism can achieve both efficiency and truthfulness.

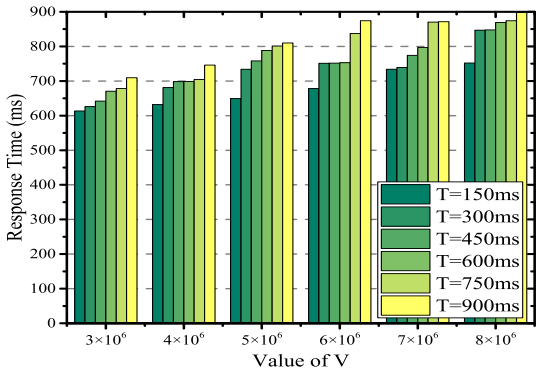



Fig. 7. Response time under different maximum tolerable cumulative delay $\mathcal{T}$.

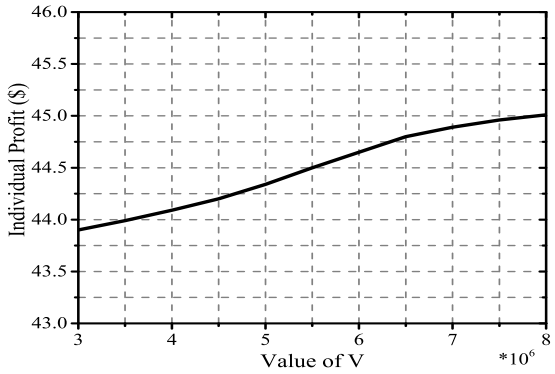



Fig. 8. Individual Profits under different weight $V$s

### 7.4 Evaluation of Auctioneer Income

Except for the evaluations of individual profits and social welfare, we also evaluate the auctioneer income under the proposed mechanism and algorithm. The auctioneer income $auc\_income$ is defined as the difference between the payments of buyers and overall prices of sellers, i.e., $auc\_income = \sum_{i=1}^{N}(\widehat{c}^{(i)} - \sum_{j=1}^{M}\sum_{m=1}^{K}\widehat{p}_m^{(j)}x_m^{(i)})$. We also implement an *auc_income max mechanism*, which aims to maximize the income of auctioneer without considering the truthfulness. Fig. 6 shows the time-averaged auctioneer income of CA-RP algorithm under the proposed mechanism and auc_income max mechanism respectively. It can be observed that under the proposed mechanism, the time-averaged auctioneer incomes are all larger than 14 dollars, which indicates that the proposed mechanism can help the auctioneer achieve stable incomes. Also, the income of auctioneer under the proposed mechanism is only 3.5% smaller than that under the auc_income max mechanism. In Section 6, we prove that the proposed mechanism is budget-balanced, i.e., the auctioneer income would not be negative. This result further demonstrates that the auctioneer can achieve a stable income from the double auction, which motivates the double auction to run for a long term.

### 7.5 Impact of Maximum Tolerable Delay $\mathcal{T}$

We study the relationship between the response time and maximum tolerable cumulative delay. Maximum tolerable

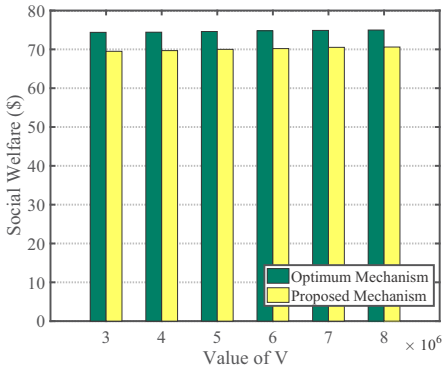



Fig. 9. Social Welfares under different weight $V$s.

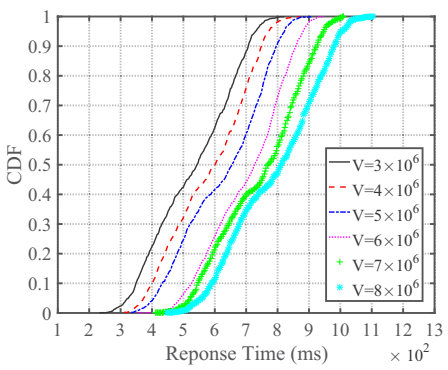



Fig. 10. CDF of response time under different weight $V$s.

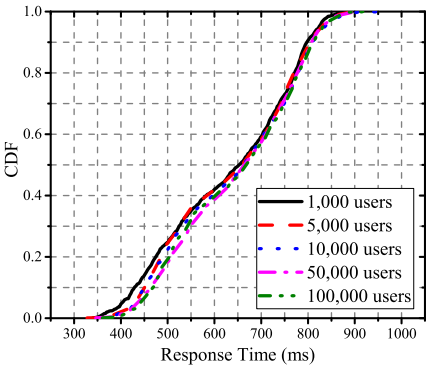



Fig. 11. CDF of the response time with different scales of users.

cumulative delay $\mathcal{T}$ constraints the time-averaged cumulative delay in a reasonable range, which is mentioned in optimization problem (7). Fig. 7 shows the impact of the maximum tolerable cumulative delay on the response time. From Fig. 7, we can see that for an arbitrary weight $V$, the response time increases as maximum tolerable cumulative delay $\mathcal{T}$ increases. The response time under $\mathcal{T} = 900ms$ is at least 28.65% larger than that under $\mathcal{T} = 150ms$. This is because the cumulative delay constraint of the optimization problem (7) relaxes as the maximum tolerable cumulative delay $\mathcal{T}$ increases.

From simulation results in Fig. 7, we can find that response time is significantly influenced by the maximum tolerable cumulative delay $\mathcal{T}$. Since the relationship between the response time and maximum tolerable cumulative delay $\mathcal{T}$ is monotonic, $\mathcal{T}$ should be set as small as possible.

### 7.6 Impact of Weight $V$

We further evaluate the impact of the weight $V$ on the individual profits, social welfare and response time. Fig. 8 shows the relationship between the individual profits and weight $V$ based on Fig. 4, and Fig. 9 shows the relationship between the social welfare and weight $V$ based on Fig. 5. As the value of weight $V$ increases, the individual profits and social welfare both increase. The individual profits under the largest $V = 8 \times 10^6$ are 2.12% larger than that under the smallest $V = 3 \times 10^6$. The social welfare under the largest $V = 8 \times 10^6$ is 1.63% larger than that under the smallest $V = 3 \times 10^6$. Whether under the proposed mechanism or the optimum mechanism, the social welfares both increase as the weight $V$ increases. This is because the provisioning cost decreases as the weight $V$ increases, leading to increase of individual profits and social welfare, according to our theoretical analysis in Section 6.

Fig. 10 shows the relationship between the response time and weight $V$. We can see that the response time increases as the value of the weight $V$ increases. For 80% of users, the response time under the largest $V = 8 \times 10^6$ is 30.78% larger than that under the smallest $V = 3 \times 10^6$ on average. According to the theoretical analysis, the cumulative delay increases as the weight $V$ increases. Since the increase of the cumulative delay leads to increase of the response time, the response time increases as the weight $V$ increases.

From simulation results in Fig. 8 and Fig. 9, we can see that the weight $V$ does not have significant impact on the individual profits and social welfare. Increasing the weight $V$ from $3 \times 10^6$ to $8 \times 10^6$ only increases 2.12% and 1.63% on individual profits and social welfare respectively. However, in Fig. 10, we find that as the weight $V$ increases, the response time increases significantly. For 80% of users, increasing the weight $V$ from $3 \times 10^6$ to $8 \times 10^6$ leads to 30.78% increasing on response time. Therefore, the weight $V$ should be set with a smaller value to constraint the response time in a reasonable range.

### 7.7 Impact of User Scale

We conduct a further simulation to evaluate the impact of user scale, i.e., the number of users in the double auction, on the proposed mechanism and algorithm. In the simulation, the task requirements are generated by TPC-W under 1,000 users, 5,000 users, 10,000 users, 50,000 users and 100,000 users respectively, and other settings are all the same as the description in Section 7.1. Fig. 11 shows CDF of the response time under the proposed mechanism and algorithm with different scales of users. We can see that the response times under different scales of users present little differences between each other. This is because the time complexity of the proposed mechanism is $O(k \cdot (M \cdot K + 1))$, which is irrelevant to the user scale, i.e., the number of users $N$. Therefore, different number of users would not bring significant extra computational complexity to the mechanism.

## 8 Conclusion

In this paper, we propose a truthful double auction mechanism to bridge users' task requirements and providers' resources in two-sided cloud markets. To precisely obtain the provisioning decision for winner determination in matching process of the mechanism, we further design a cost-aware resource provisioning algorithm based on Lyapunov optimization techniques. With theoretical analysis, we prove our proposed mechanism is individual-rational, truthful and budget-balanced, as well as analyze the optimality of our algorithm. By conducting simulations, we demonstrate that our mechanism is efficient and feasible.

## Acknowledgments

This research was sponsored by National Key R&D Program of China (No. 2017YFC0803700).

## REFERENCES


[1] Amazon, "Ec2 instance pricing - amazon web service (aws)." [Online]. Available: https://aws.amazon.com/ec2/pricing/?nc1=h_ls, 2017.

[2] Microsoft, "Pricing overview - how azure pricing works — microsoft azure." [Online]. Available: https://azure.microsoft.com/en-us/pricing/, 2017.

[3] Scylla, "Choosing ec2 instances for nosql." [Online]. Available: http://www.scylladb.com/2016/02/26/best-amazon-ec2-instance-nosql/, 2016.

[4] ParkMyCloud, "How to choose between aws instance types for cost savings." [Online]. Available: http://www.parkmycloud.com/blog/how-to-choose-between-aws-instance-types-for-cost-savings/, 2015.

[5] S. Blue, "Choosing the right aws reserved instances." [Online]. Available: http://www.strategic-blue.com/chooseris/, 2014.

[6] TechRepublic, "How to choose the right windows azure subscription." [Online]. Available: http://www.techrepublic.com/blog/the-enterprise-cloud/how-to-choose-the-right-windows-azure-subscription/, 2011.

[7] A. Hagiu and J. Wright, "Multi-sided platforms," *International Journal of Industrial Organization*, vol. 43, pp. 162–174, 2015.

[8] H. Li, C. Wu, Z. Li, and F. C. Lau, "Virtual machine trading in a federation of clouds: Individual profit and social welfare maximization," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1827–1840, 2013.

[9] P. Samimi, Y. Teimouri, and M. Mukhtar, "A combinatorial double auction resource allocation model in cloud computing," *Information Sciences*, vol. 357, pp. 201–216, 2014.

[10] Amazon, "Amazon ec2 spot iinstance pricing." [Online]. Available: https://aws.amazon.com/ec2/spot/pricing/, 2017.

[11] Google, "Google computing engine pricing — compute engine documentation — google cloud platform." [Online]. Available: https://cloud.google.com/compute/pricing, 2017.

[12] L. Zheng, C. Joe-Wong, C. W. Tan, M. Chiang, and X. Wang, "How to bid the cloud," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 71–84, 2015.

[13] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing amazon ec2 spot instance pricing," *ACM Trans. Econ. Comput.*, vol. 1, no. 3, pp. 16:1–16:20, 2013.

[14] B. Javadi, R. K. Thulasiramy, and R. Buyya, "Statistical modeling of spot instance prices in public cloud environments," in *Proc. IEEE/ACM UCC'11*, 2011.

[15] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud," in *Proc. IEEE ICDCS'11*, 2011.

[16] R. N. Calheiros, R. Ranjan, and R. Buyya, "Virtual machine provisioning based on analytical performance and qos in cloud computing environments," in *Proc. IEEE ICPP'11*, pp. 559–570, 2011.

[17] X. Yi, F. Liu, D. Niu, H. Jin, and J. C. S. Lui, "Cocoa: Dynamic container-based group buying strategies for cloud computing," *ACM Transactions on Modeling & Performance Evaluation of Computing Systems*, vol. 2, no. 2, p. 8, 2017.

[18] C. Lee, P. Wang, and D. Niyato, "A real-time group auction system for efficient allocation of cloud internet applications," *IEEE Transactions on Services Computing*, vol. 8, pp. 251–268, March 2015.

[19] J. Wang, X. Xiao, J. Wang, K. Lu, X. Deng, and A. A. Gumaste, "When group-buying meets cloud computing," in *Proc. IEEE INFOCOM'16*, pp. 1–9, April 2016.

[20] S. Zaman and D. Grosu, "A combinatorial auction-based mechanism for dynamic vm provisioning and allocation in clouds," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 129–141, 2013.

[21] W. Shi, L. Zhang, C. Wu, Z. Li, and F. Lau, "An online auction framework for dynamic resource provisioning in cloud computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 1, pp. 71–83, 2014.

[22] L. Zhang, Z. Li, and C. Wu, "Dynamic resource provisioning in cloud computing: A randomized auction approach," in *Proc. IEEE INFOCOM'14*, 2014.

[23] H. Zhang, H. Jiang, B. Li, F. Liu, A. V. Vasilakos, and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands," *IEEE Transactions on Computers*, vol. 65, no. 3, pp. 805–818, 2016.

[24] J. Li, Y. Zhu, J. Yu, C. Long, G. Xue, and S. Qian, "Online auction for iaas clouds: towards elastic user demands and weighted heterogeneous vms," in *Proc. IEEE INFOCOM'17*, 2017.

[25] X. Yi, F. Liu, Z. Li, and H. Jin, "Flexible instance: Meeting deadlines of delay tolerant jobs in the cloud with dynamic pricing," in *Proc. IEEE ICDCS'16*, pp. 415–424, 2016.

[26] W. Wei, X. Fan, H. Song, X. Fan, and J. Yang, "Imperfect information dynamic stackelberg game based resource allocation using hidden markov for cloud computing," *IEEE Transactions on Services Computing*, 2016. doi: 10.1109/TSC.2016.2528246.

[27] C. Esposito, M. Ficco, F. Palmieri, and A. Castiglione, "Smart cloud storage service selection based on fuzzy logic, theory of evidence and game theory," *IEEE Transactions on Computers*, vol. 65, pp. 2348–2362, Aug 2016.

[28] K. Li, C. Liu, K. Li, and A. Y. Zomaya, "A framework of price bidding configurations for resource usage in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, pp. 2168–2181, Aug 2016.

[29] X. Dong, J. Yu, Y. Luo, Y. Chen, G. Xue, and M. Li, "Achieving secure and efficient data collaboration in cloud computing," in *Proc. IEEE/ACM IWQoS'13*, 2013.

[30] X. Dong, J. Yu, Y. Zhu, Y. Chen, Y. Luo, and M. Li, "Seco: Secure and scalable data collaboration services in cloud computing," *Computers & Security*, vol. 50, no. Supplement C, pp. 91 – 105, 2015.

[31] Codit, "Integration cloud." [Online]. Available: http://www.integrationcloud.eu/, 2017.

[32] Dell, "Dell boomi." [Online]. Available: https://boomi.com/, 2017.

[33] Talend, "Talend real-time open source big data integration software." [Online]. Available: https://www.talend.com/, 2017.

[34] Z. Zheng, Y. Gui, F. Wu, and G. Chen, "Star: Strategy-proof double auctions for multi-cloud, multi-tenant bandwidth reservation," *IEEE Transactions on Computers*, vol. 64, pp. 2071–2083, July 2015.

[35] J. Murillo, B. Lpez, V. Muoz, and D. Busquets, "Fairness in recurrent auctions with computing markets and supply fluctuations," *Computational Intelligence*, vol. 28, no. 1, pp. 24–50, 2012.

[36] G. Baranwal and D. P. Vidyarthi, "A fair multi-attribute combinatorial double auction model for resource allocation in cloud computing," *Journal of Systems and Software*, vol. 108, pp. 60 – 76, 2015.

[37] L. Lu, J. Yu, Y. Zhu, S. Qian, G. Xue, and M. Li, "Cost-efficient vm configuration algorithm in the cloud using mix scaling strategy," in *Proc. IEEE ICC'17*, 2017.

[38] Ctera, "Aws s3 infrequent access vs azure cool blob storage comparison." [Online]. Available: http://www.ctera.com/company/blog/aws-s3-ia-vs-azure-cool-blob-comparison/, 2016.

[39] Microsoft, "My personal azure faq on azure networking slas, bandwidth, latency, performance, slb, dns, dmz, vnet, ipv6 and much more." [Online]. Available: https://blogs.msdn.microsoft.com/igorpag/2014/09/28/my-personal-azure-faq-on-azure-networking-slas-bandwidth-latency-performance-slb-dns-dmz-vnet-ipv6-and-much-more/, 2015.

[40] Amazon, "Aws server migration service — pricing." [Online]. Available: https://aws.ama-zon.com/server-migration-service/pricing/?nc1=h_ls, 2017.

[41] T. Sandholm and S. Suri, "Market clearability," in *International Joint Conference on Artificial Intelligence*, pp. 1145–1151, 2001.

[42] B. Edelman, M. Ostrovsky, and M. Schwarz, "Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords," *American Economic Review*, vol. 97, no. 1, pp. 242–259, 2007.

[43] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[44] Rackspace Cloud Computing, "OpenStack Open Source Cloud Computing Software." [Online]. Available: http://www.openstack.org/, 2017.

[45] TPC Organization, "TPC-W - Homepage." [Online]. Available: http://www.tpc.org/tpcw/, 2017.

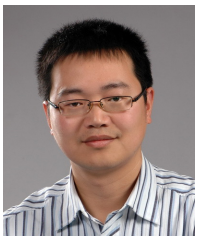
[46] FERC, "Federal energy regulatory commission." [Online]. Available: http://www.ferc.gov, 2017.

[47] S. Imai, T. Chestna, and C. A. Varela, "Elastic scalable cloud computing using application-level migration," in *Proc. IEEE/ACM UCC'12*, 2012.

[48] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: Elastic resource scaling for multi-tenant cloud systems," in *Proc. ACM SOCC'11*, 2011.

**Li Lu** received the bachelor degree in Computer Science and Technology from Xi'an Jiaotong University, Xi'an, China, in 2015. He is currently a Ph.D. student in Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include cloud computing, mobile and ubiquitous computing, cyber security and privacy.

**Jiadi Yu** received the Ph.D. degree in Computer Science from Shanghai Jiao Tong University, Shanghai, China, in 2007. He is currently an Associate Professor in Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. Prior to joining Shanghai Jiao Tong University, he was a postdoctoral fellow in the Data Analysis and Information Security (DAISY) Laboratory at Stevens Institute of Technology from 2009 to 2011. His research interests include cyber security and privacy, mobile and pervasive computing, cloud computing and wireless sensor networks. He is a member of the IEEE and the IEEE Communication Society.



**Yanmin Zhu** received the Ph.D. from the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology in 2007. He is currently a Professor with the Department of Computer Science and Engineering at Shanghai Jiao Tong University. His research interests include wireless sensor networks and mobile computing. Before that, he was a Research Associate with the Department of Computing at Imperial College London. He is a member of the IEEE and the IEEE Communication Society.

**Minglu Li** is graduated from the School of Electronic Technology, University of Information Engineering, in 1985 and received the Ph.D. degree in computer software from Shanghai Jiao Tong University (SJTU) in 1996. He is a full professor and the vice chair of the Department of Computer Science and Engineering and the director of Grid Computing Center of SJTU. Currently, his research interests include grid computing, services computing, and sensor networks.