Designing a Strategy-Proof Online Auction Mechanism for Deadline Aware Cloud Resource Allocation

A RENCI Technical Report
TR-15-02

renci

Tianrong Zhang
Department of Computer Science
University of North Carolina at Chapel Hill

Yufeng Xin
(Renaissance Computing Institute (RENCI
University of North Carolina at Chapel Hill

Designing a Strategy-Proof Online Auction Mechanism for Deadline-aware Cloud Resource Allocation

Tianrong Zhang[†] Yufeng Xin[⋆]
[†]Department of Computer Science, University of North Carolina at Chapel Hill
[⋆]RENCI, University of North Carolina at Chapel Hill
[†]trzhang@cs.unc.edu [⋆]yxin@renci.org

***Abstract*—Cloud computing facilitates resource sharing and provides flexibility and affordability. Resource allocation and pricing is critical to the system performance, and auction has been believed as a promising approach for fairness and efficiency. To design a practical cloud resource auction, we need to consider five major challenges, including strategy-proofness, online framework, heterogeneous request, deadline-awareness, and social welfare maximization. However, none of existing works have fully addressed these five challenges. In this paper, we present the first strategy-proof online combinatorial auction for deadline-aware cloud resource allocation which jointly considers the five design challenges. Our analyses show that our auction achieves both strategy-proofness and approximate efficiency on social welfare. Our evaluation results verify the efficacy of our design and show that our auction achieves good social welfare without the loss on request completion.**

I. Introduction

Cloud computing emerges as a transformative paradigm to accommodate various services and reform computing environments. Leveraging resource virtualization technologies, cloud computing offers users the opportunity of sharing cloud resources and the flexibility and affordability for processing diverse jobs with minimal concern on infrastructure reliability and management overhead. Indisputably, resource allocation and pricing plays a critical role in cloud system performance. Fixed pricing policies fail to react to market supply and demand fluctuation, leading to either overpricing or underpricing both of which jeopardize overall system social welfare.

Among the best-known market-based allocation mechanisms, auctions stand out as a promising approach to effectively managing cloud resource supply and demand for fairness and allocation efficiency [4]. Recently, Amazon EC2 has launched a bidding-style model, named Spot Instances [1], which offers a variety of pre-configured virtual machines (VMs) with much lower prices than on-demand resources. Nevertheless, as a first-step attempt on market-based cloud resource allocation scheme design, Spot Instances inevitably suffers from design limitations such as unpredictable job termination, vulnerability to market manipulation, and inflexible bidding support.

This work was supported by US National Science Foundation grant CNS-1329745.

Designing a practical auction mechanism for cloud resource allocation is challenging, because not only difficulties in general auction design have to be addressed, but also inherent characteristics of cloud resource request should be taken into account. We list four major challenges as follows:

Strategy-proofness: Cloud resource users are normally rational and selfish, and thus are usually interested in maximizing their utilities. In cloud resource auctions, selfish users can misreport their information intrinsic valuations, resource demands, etc. Such market manipulation will not only hurt the benefit of truthful users but also jeopardize overall system performance.

Online framework: In real world, cloud users arrive on and depart from the cloud platform on the fly. The online nature makes cloud resource auctions fundamentally different from offline scenarios as decisions will be made without complete future information. The temporal aspect of user behavior, *i.e.*, reporting arrival and departure time, can make the auctions unfortunately more vulnerable to strategical manipulation, which further complicates the design.

Heterogeneous requests: A practical cloud resource auction mechanism should be combinatorial to accommodate heterogeneous requests from cloud users. The combinatorial characteristic comes in two dimensions, namely, VM bundling and time elasticity. On one hand, cloud computing applications often consist of multiple layers that usually need different VM instances. On the other hand, different jobs naturally have different required running time. Therefore, a combinatorial auction will be essential to give users flexibility on resource request expression.

Deadline-awareness: In practice, cloud resource requests are usually associated with different deadline requirements. For example, a resource request for running data analytic jobs is often time sensitive; while one for running general batch jobs usually has a loose deadline. A practical cloud resource auction is desired to be able to recognize job deadlines, which will be beneficial to both cloud systems and users. For cloud systems, deadline-awareness can improve allocation capacity and social welfare by enhancing request scheduling flexibility.

For users, deadline-awareness reduces competition intensity, which increases users' winning chances and thus lowers cost, leading economies of scale.

Social welfare: The last but not least challenge is to maximize social welfare, *i.e.*, the sum of all winning users' valuations on their requests. However, finding the allocation of the optimal social welfare in a combinatorial auction is normally computationally intractable. Hence, designing an approximately efficient allocation mechanism will be of great interest.

In this paper, we conduct an in-depth study on the problem of auction mechanism design for cloud resource allocation, jointly considering all above challenges. Although a number of designs, *e.g.*, [12]–[14], [16], [17], were recently proposed, unfortunately, none of these works has fully addressed all the four design challenges (as shown in Table I). We propose a strategy-proof online auctIon framework for deadline-aware cloud resource allocation. In the cloud resource auction, all cloud resource users arrive in the system on the fly, and submit their sealed resource requests to the cloud resource system. The cloud resource system runs the auction to decide the auction winners, resource allocations, and the charges for winners.

We make the following contributions in this paper:

- To the best of our knowledge, we present the first strategy-proof online combinatorial auction mechanism for the problem of deadline-aware cloud resource allocation.
- Our analysis shows that our proposed auction mechanism is strategy-proof and achieves a nontrivial competitive ratio.
- We implement our auction design and evaluate its performance. Our simulation results verify the efficacy of our design.

The rest of this paper is organized as follows. In Section II, we present technical preliminaries. In Section III, we demonstrate main design principles. In Section IV, we describe our auction design in detail. In Section V, we present the analysis on our design. In Section VI, we present evaluation results. In Section VII, we review related works. Finally, we conclude our paper in Section VIII.

II. Preliminaries and Problem Formulation

In this section, we first describe our auction model for deadline-aware cloud resource allocation. Next, we review several important solution concepts used in this paper. Lastly, we present the problem formulation.

A. Cloud Resource Auction Model

We model the process of deadline-aware cloud resource allocation as an online combinatorial auction. A cloud system, called seller, would like to lease resources to cloud resource users. The cloud system contains multiple types of resources (*e.g.*, CPU, RAM, and Disk), and we denote the resource set by $C \triangleq \{c_i | i \in [C]\}$,[1] meaning that there are c_i units of type-i resource in system. The cloud system operates in a discrete time-slotted fashion spanning time $T \triangleq \{t | t \in [T]\}$, where the duration of each time slot is τ. It offers multiple VM types $M \triangleq \{m_i | i \in [M]\}$, of which each type m_i refers to a vector $m_i \triangleq (m_i^1, m_i^2, \ldots, m_i^{|C|})$, meaning that a type-$i$ VM instance contains m_i^j units of type-j resource. During cloud system operation, VM instances can be dynamically assembled as long as required resources are available.

There is a set of cloud resource users N, called buyers,[2] who would like to bid for desired cloud resources by submitting requests to the cloud system. Each buyer $i \in N$ owns her request $\theta_i \in \Theta_i$ from the space Θ_i, and θ_i is buyer i's private information, called type, which is characterized by a tuple

$$\theta_i \triangleq (a_i, h_i, d_i, r_i, v_i), \tag{1}$$

where a_i is i's type release time; h_i is the number of contiguous time slots she requests; d_i is the deadline by which her job needs to be completed; r_i denotes her resource demand vector $r_i \triangleq (r_i^1, r_i^2, \ldots, r_i^{|M|})$, indicating that r_i^j instances of type-j VM are requested within θ_i; and v_i is her valuation on the request θ_i. We assume that each user is allowed to request no more than r_{max} VMs and no more than h_{max} time slots. In our online auction model, a_i is interpreted as buyer i's arrival time, and d_i is interpreted as her departure time. As is widely assumed (*e.g.*, [3], [10]) as well as is backed by the heart-beat scheme [7], we hold the assumption that every user cannot declare an earlier arrival time or a later departure time than her truthful one. We use $\theta = (\theta_1 \ldots, \theta_{|N|})$ to denote the vector of all buyers' types, and by the notation convention use $\theta_{-i} = (\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_{|N|})$ to denote the same vector without buyer i's type. Thus, we have $\theta = (\theta_i, \theta_{-i})$.

In runtime, each buyer $i \in N$ arrives in the cloud system in time slot $\hat{a_i}$ and declares to the seller her resource request $\hat{\theta_i} \triangleq (\hat{a_i}, \hat{h_i}, \hat{d_i}, \hat{r_i}, \hat{v_i})$.[3] The cloud resource auction determines the winners and schedules their requests on the fly. Formally, our online auction mechanism for deadline-aware resource allocation is defined as $\Gamma = (\Theta_1, \ldots, \Theta_{|N|}, \mathcal{F}(\cdot))$, where the function $\mathcal{F}(\cdot) : \Theta_1 \times \ldots \times \Theta_{|N|} \rightarrow \mathcal{O}$ maps the declared types to a allocation outcome $o \in \mathcal{O}$. An outcome $o = (W, S, P)$ consists of a winner set W, a scheduling decision vector $S = (s_1, \ldots, s_{|N|})$, and a payment vector $P = (p_1, \ldots, p_{|N|})$. If a buyer i is selected as a winner, *i.e.*, $i \in W$, she will be granted to start running her job in time slot s_i and will need to pay p_i for her resource usage.

We consider that all users are rational and selfish, and their objectives are to maximize their own utilities. Each buyer i has a quasilinear utility u_i defined as

$$u_i(\mathcal{F}(\hat{\theta_i}, \hat{\theta_{-i}}), \theta_i) \triangleq \begin{cases} v_i - p_i, & \text{if } i \in W; \\ 0, & \text{o.w.} \end{cases} \tag{2}$$

[1]For the convenience of description, given a set X, we use $|X|$ to denote its size, and use $[X]$ to denote the set $\{1, 2, \ldots, |X|\}$. Unless otherwise specified, this definition shall apply to all notations throughout this paper.

[2]We use buyer and user interchangeably in this paper.

[3]$\hat{v_i}$ is buyer i's declared valuation, and thus represents her bid in the auction indicating the maximum price she is willing to pay for $\hat{\theta_i}$.

TABLE I
COMPARISON WITH EXISTING WORKS

Auction designs	Strategy-proofness	Online framework	Heterogeneous requests	Deadline-awareness	Approximate social welfare
[13]	✔	✗	✔	✗	✔
[17]	✔	✗	✔	✗	✔
[16]	✔	✔	✗	✔	✔
[14]	✔	✔	✗	✗	✔
[12]	✔	✔	✔	✗	✔
Our auction	✔	✔	✔	✔	✔

We assume that a buyer has no preference over different outcomes, if the utility is the same to herself. In contrast to users, the overall objective of the auction mechanism is to maximize social welfare, which is defined as follows.

Definition 1 (Social Welfare): The social welfare in an online auction is the sum of all winners' valuations on the allocated items, *i.e.*, on resource requests in our cloud resource auction, within the lifespan.

$$SW \triangleq \sum_{t\in T}\sum_{i\in W_t} v_i, \tag{3}$$

where W_t is the set of winners determined within time slot t, so $W = \cup_{t\in T} W_t$.

B. Economic Properties

We briefly review several solution concepts from game theory and mechanism design. These solution concepts build the theoretical foundation of our cloud resource auction design.

Definition 2 (Dominant Strategy [8]): A strategy st_i is player i's dominant strategy, if for any $st'_i \neq st$ and any strategy profile of the other players st_{-i}, we have

$$u_i(st_i, st_{-i}) \geq u_i(st'_i, st_{-i}). \tag{4}$$

Definition 3 (Strategy-Proof Mechanism [6]): A direct-revelation mechanism is a mechanism, in which the only strategy available to players is to make claims about their preferences to the mechanism. A direct-revelation mechanism is strategy-proof if it satisfies both *incentive-compatibility* and *individual-rationality*. Incentive-compatibility means revealing truthful information is a dominant strategy for each player. Individual-rationality means each player can always achieve at least as much expected utility from faithful participation as staying outside.

In our online combinatorial cloud resource auction, the strategy-proofness means that no buyer $i \in N$ can increase her utility by reporting $\theta'_i \neq \theta_i$. *i.e.*, every buyer's best strategy to simply reveal her truthful request tuple, *i.e.*, $\hat{\theta_i} = \theta_i$.

C. Resource Allocation Problem

We formulate the online cloud resource allocation problem as a multidimensional multiple-choice knapsack problem (MMKP). The MMKP here can be described as follows. There are $|N|$ groups of items, where each item represents a specific resource assignment θ_{ij} associated with a fixed starting time s_{ij}, and each group l_i represents the set of i's all possible allocations θ_{ij} in which $s_{ij} \in [a_i, d_i - h_i + 1] \cup 0$, where $s_{ij} = 0$ means not winning. Let r_{ijt} be the resource demand vector of θ_{ij} in time slot t, and thus $r_{ijt} = r_i$ if $t \in [s_{ij}, s_{ij} + h_i)$ and $s_{ij} \neq 0$; otherwise $r_{ijt} = (0)^{|M|}$. The objective of the MMKP is to pick exactly one item from each group for maximum total value of the collected items, subject to resource constraints. Consequently, our resource allocation problem can be formulated as an integer programming.

Problem: *Online Combinatorial Cloud Resource Allocation*
Objective:

$$Maximize \quad \sum_{i\in N}\sum_{j\in[l_i]} x_{ij}\hat{v}_i \tag{5}$$

Subject to:

$$\sum_{i\in N}\sum_{j\in[l_i]}\sum_{k\in[M]} x_{ij} m_k^c r_{ijt}^k \leq c_j, \qquad \forall c \in [C], \forall t \in [T] \tag{6}$$

$$\sum_{j\in l_i} x_{ij} = 1, \qquad \forall i \in N \tag{7}$$

$$x_{ij} = \{0,1\}, \qquad \forall i \in N, \forall j \in [l_i] \tag{8}$$

Here, x_{ij} are the picking variables, and $x_{ij} = 1$ indicates that user i is allocated requested resource and granted to starting in time slot s_{ij}. We use $\hat{v}_i$ rather than v_i in the objective, because the cloud system only knows users' bids instead of their private valuations. However, if the cloud resource auction is strategy-proof, all users will have incentives to faithfully submit their intrinsic valuations as bids.

Due to users dynamic arrival and departure, decisions in an online cloud resource auction should be made on the fly. However, deriving the optimal allocation needs complete knowledge over the entire lifespan, which is apparently not practical. Even if delay computation of payments could be tolerable, solving the above integer programming is still NP-hard, which makes the celebrated VCG mechanism [8] inapplicable. Considering the computational intractability of this problem, we propose an alternative design with an online greedy allocation algorithm that will achieve approximately efficient social welfare.

III. DESIGN PRINCIPLES AND FRAMEWORK

In this section, we present main design principles that shape the online framework of our deadline-aware cloud resource auction.

A. Temporal Bidding Competition

Achieving strategy-proofness is challenging in an online cloud resource auction because of the lack of future information. Supply curve [5] based allocations, *e.g.*, [16], could be an effective method, as they can carry out allocation by considering each request sequentially and individually. However, this advantage potentially turns into a shortcoming in that the characteristic of loose competition produces performance uncertainty in terms of social welfare. For example, supposing there is a single unit of CPU for lease, and multiple users bid for it on the fly. Supply curve based schemes will allocate the CPU to the first user whose bid exceeds the marginal price without considering the following users at all.

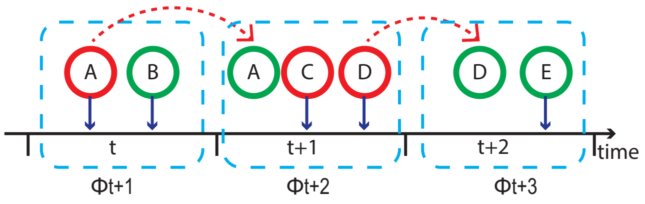

Fig. 1. An illustrative example of temporal bidding competition. Buyers $A - E$ arrive on the go to bid for resources. We use red circles to represent losing users, and green circles to represent winning users. Dashed-border rectangles indicate active user sets.

To obtain reasonable performance on social welfare, our auction is instead constructed by a series of single round auctions. Figure 1 illustrates the framework of temporal bidding competition. Specifically, our online auction maintains a group of active users Φ. When a user joins the system (*e.g.*, user A in time slot t), she will be included in Φ. By the beginning of each time slot t, our auction allows all users in group Φ (*e.g.*, A and B in time slot t) to compete resources by bidding, and then determines the winners W_t (*e.g.*, user B in time slot t). Winning users will be removed from Φ (*e.g.*, user B), so will the users whose requests can no longer be satisfied due to passed deadlines (*e.g.*, user C).

B. Adaptive Auction Window

Heterogeneous requests should be supported to offer bidding flexibility. However, due to the two-dimensional combinatorial characteristic, conventional online auction designs tailored to the canonical expiring items environment [7] turn to be not feasible to solve our cloud resource allocation problem. These online auctions are generally designed by connecting single round auctions serially. Hence, all these single round auction modulars are temporally separated, *i.e.*, none of allocations temporally spans more than one round. Despite the simplicity, it is difficult to choose an appropriate lifespan for all single round auctions. A shorter lifespan can cause the incapability of processing longer jobs; while a longer lifespan can make users suffer from waiting, leading to delayed response or even worse allocation failure.

Different from the conventional design, our auction exploits a novel framework using adaptive auction window. Cloud resource usage could be difficult to be precisely predicted, however, clear diurnal usage pattern and periodic demand variation exist [2], [11], providing the insight that request arrival and resource usage exhibit certain variation trends over a relatively long duration. Taking advantage of this, our auction

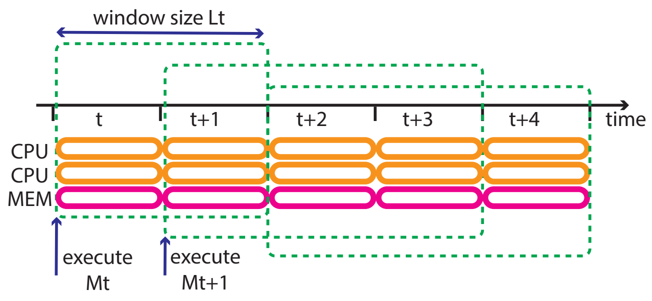

Fig. 2. The illustration of adaptive auction window. Cloud resources in different colors represent different types of resources. Dashed-border rectangles represent auction windows.

dynamically adjusts the auction window size, *i.e.*, the number of time slots within which resources will be auctioned off in each round. Specifically, as shown in figure 2, the auction conducts a single round auction $\mathcal{M}_t$ by each time slot $t \in T$. Right after determining the allocation and clearing the market for $\mathcal{M}_t$, the auction will update and release the window size L_{t+1} for $\mathcal{M}_{t+1}$ (please refer to Section IV-C) before accepting requests within time slot t.

C. Deadline-awareness

Cloud resource requests are usually associated with deadlines, indicating users can have multiple interested allocations. The simplest method to accommodate multi-minded users is to accept recursive requests submission, *i.e.*, each user can repeatedly submit her request until it is accepted or its deadline passes the requirement. The other approach exploited in this paper is to allow the system to recognize deadlines associated with requests. We use a simple example shown in Figure 3 to illustrate the benefit of deadline-awareness.

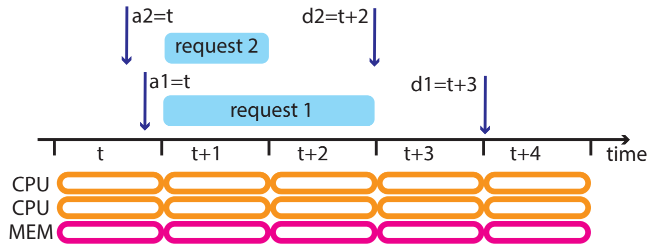

Fig. 3. An illustrative example to show the efficacy of deadline-awareness.

TABLE II
ALLOCATION RESULTS OF THE ILLUSTRATIVE EXAMPLE

Schemes	$\mathcal{M}_{t+1}$	$\mathcal{M}_{t+2}$
Recursive submission:	$s_1 = t+1,\ s_2 = 0$	$s_2 = 0$
Deadline-awareness Case 1:	$s_1 = t+1,\ s_2 = 0$	$s_2 = 0$
Deadline-awareness Case 2:	$s_1 = t+2,\ s_2 = t+1$	—

Suppose request 1 and request 2 are both submitted in time slot t, and both need 2 CPUs and 1 MEM. Request 1 needs 2 time slots and has the deadline of $t+3$; while request 2 needs 1 time slot and has the deadline of $t+2$. We assume request 1 has higher priority. Table II shows the allocation results. By recursive submission, request 1 will be assigned to start in time slot $t+1$; and request 2 will not be accepted due to conflict. However, a deadline-aware allocation auction can result in two possible results. One is the same as that by recursive submission, but the other one will successfully accept both these two requests. Therefore, deadline-awareness could increase the opportunity of scheduling more requests, leading to better system performance.

IV. Auction Design

In this section, we present our auction mechanism in details. Our auction consists of four major components: winner determination, load balancing scheduling, dynamic window adaptation , and clearing price calculation. Considering that computing the optimal allocation is not practical, our auction exploits a greedy algorithm to determine winners. Furthermore, our auction uses a load balancing scheduling scheme to decide resource assignments for winning users. In addition, our auction incorporates a novel scheme of dynamic window adaptation into the online framework for improving allocation capacity and optimizing social welfare. Finally, our auction uses a critical bid based scheme to determine the clearing prices for the winners.

A. Winner Determination

We first introduce average per-unit bid, which will be used as the ranking metric for our greedy allocation algorithm.

Definition 4 (Average Per-unit Bid): Given each user i's declared resource request $\hat{\theta_i} = (\hat{a_i}, \hat{h_i}, \hat{d_i}, \hat{r_i}, \hat{v_i})$, her corresponding average per-unit bid π_i is defined as

$$\pi_i = \frac{\hat{v_i}}{\hat{h_i}\mathcal{A}(\hat{r_i})}, \tag{9}$$

where

$$\mathcal{A}(\hat{r_i}) = \sqrt{\frac{\sum_{j\in[C]}\left(\mathcal{X}(\hat{r_i},j)\right)^2}{|C|}}, \tag{10}$$

and

$$\mathcal{X}(\hat{r_i},j) = \frac{\sum_{k\in[M]} \hat{r_i^k} m_k^j}{c_j}. \tag{11}$$

Here, $\mathcal{X}(\hat{r_i},j)$ calculates the aggregate proportion of resource j requested by user i per time slot; and $\mathcal{A}(\hat{r_i})$ quantifies user i's overall resource usage per time slot by calculating the quadratic mean of $\mathcal{X}(\hat{r_i},j), \forall j \in [C]$. The rationale of calculating the quadratic mean of usage proportions instead of their sum, although it has been widely used (*e.g.*, [13]), is that the quadratic mean can reflect the intensity of resource usage more effectively in a multi-type resource environment. Supposing the system contains two types of resources, and user A and B submit the same request except that user A requests 50% of each resource, while user B requests 100% and 0, respectively. The expected result should be that user A will be considered preferentially, as B's request will exhaust one resource, leading to severe scheduling unavailability. The average per-unit bid defined above can differentiate these two requests, however, it would not if we calculated the sum. Similar to [10], we assume $\pi_i \in [\pi_{min}, \pi_{max}]$.

The auction ranks all users in Φ_t according to average per-unit bids $\pi_i, i \in \Phi_t$ in non-ascending order. In $\mathcal{M}_{t+1}$, following the sorted user list, the auction greedily selects each user $i \in \Phi_t$ as a candidate, and calls the algorithm $\mathcal{LBS}(\cdot)$ (please refer to Section IV-B)to determine her resource assignment. If $\mathcal{LBS}(\cdot)$ returns a successful assignment, *i.e.*, $s_i \neq 0$, i will be added into winner set W_{t+1}. Algorithm 1 $\mathcal{WD}(\cdot)$ shows the pseudo code of the process of determining winners.

Algorithm 1: Winner Determination $\mathcal{WD}(\cdot)$

```
Input: Request set $\{\hat{\theta_i} | i \in \Phi_t\}$
Output: Winner set $W_t$, and scheduling decision $D_t$
1 Calculate $\pi_i, \forall i \in \Phi_t$;
2 Sort $\pi_i$: $\pi_1 \geq \pi_2 \geq \ldots \geq \pi_{\Phi_t}$;
3 for $j = 1$ to $\Phi_t$ do
4     $s_j \leftarrow \mathcal{LBS}(\hat{\theta_j})$;
5     if $s_j \neq 0$ then
6         $W \leftarrow W \cup j$;
7         $D \leftarrow D \cup (j, s_j)$;
8 return $W$ and $D$;
```

B. Load Balancing Scheduling

Since each user may have multiple equally preferred allocations, *i.e.*, multiple assignments with different starting times, an auction needs to schedule requests for winning users. A desired request scheduling scheme should be able to balance resource occupation, so that resources in different time slots can be allocated smoothly, avoiding allocation fragmentation and meanwhile reserving allocation availability for the future rounds. To this end, we design a load balancing scheduling scheme $\mathcal{LBS}(\cdot)$ based on a measure called allocation occupation proportion.

Definition 5 (Allocation Occupation Proportion): Suppose in $\mathcal{M}_k$, x_i^t units of type-i, $i \in [C]$ resource in time slot t, $t \in [k, k+L_k)$ have been allocated. The immediate allocation occupation proportion λ of $\mathcal{M}_k$ is defined as

$$\lambda_k = \sqrt{\frac{\sum_{t=k}^{k+L_k-1}\sum_{i\in[C]}\left(\Upsilon(t)(x_i^j/c_i)^2\right)}{L_k|C|}}, \tag{12}$$

where

$$\Upsilon(t) = \begin{cases} 0, & \text{if } t = k; \\ \gamma^{t-k}, & \text{if } t \in (k, k+L_k). \end{cases} \tag{13}$$

Our auction schedules the request of each winning user by choosing the resource assignment that will result in the minimum allocation occupation proportion. Specifically, when a buyer $i \in \Phi_k$ is selected as a winner in $\mathcal{M}_k$, the auction will attempt to schedule it on all feasible assignments in terms of current resource availability and the deadline requirement, and calculate each corresponding λ_k. Then, the auction will select the one which corresponds to the minimum λ_k. Algorithm 2 summarizes this process.

Algorithm 2: Load Balancing Scheduling $\mathcal{LBS}(\cdot)$

```
Input: Buyer $i$'s resource request $\hat{\theta}_i$
Output: Buyer $i$'s scheduling decision $s_i$
1  /* Suppose currently in $\mathcal{M}_k$, $y_c^t$ units of
      type-$c$ resource are available in time
      slot $t$, $c \in [C]$, $t \in [k, k+L_k)$.          */
2  $\lambda_k \leftarrow \infty,\ \lambda'_k \leftarrow \infty,\ s_i \leftarrow 0,\ s'_i \leftarrow 0$;
3  for $j = k$ to $k + L_k - 1$ do
4      if $j + h_i - 1 > d_i$ then
5          break;
6      if $\exists x \in [C] \wedge \exists y \in [j, j+h_i) \Rightarrow x_c^t < \sum_{z\in[M]} \hat{r_i^z} m_z^c$
       then
7          continue;
8      schedule $\hat{\theta}_i$ with $s'_i = j$, and calculate $\lambda_k$;
9      if $\lambda_k < \lambda'_k$ then
10         $\lambda'_k \leftarrow \lambda_k, s_i \leftarrow j$;
11     removed scheduled $\hat{\theta}_i$, $s'_i \leftarrow 0$;
12 schedule $\hat{\theta}_i$ in $s_i$;
13 return $s_i$;
```

C. Dynamic Window Adaptation

Resource allocation for two-dimensionally combinatorial heterogeneous requests with service guarantee is faced with a challenging design dilemma. On one hand, accepting more current requests increases the risk of having to reject more future requests which might be of higher valuation; on the other hand, delaying decisions has to face the risk of few requests being submitted in the future and pending requests passing deadlines.

To alleviate the dilemma, our design exploits a novel online framework, namely dynamic window adaptation, which dynamically and adaptively updates the auction window size to manage the resource supply. Specifically, our auction updates auction window size according to the average allocation occupation proportion λ_H calculated as

$$\lambda_H \leftarrow (1-\omega)\lambda_H + \omega\lambda_t. \tag{14}$$

Here, λ_t is the ultimate allocation occupation proportion of $\mathcal{M}_t$ that is calculated right after the auction finishes $\mathcal{M}_t$, *i.e.*, by time slot t, and ω is the proportion averaging constant. Upon obtaining λ_t, the window size for the next auction $\mathcal{M}_{t+1}$ will be updated as

$$L_{t+1} = \begin{cases} max\{L_{min}, L_t - 1\}, & \text{if } \lambda_H < \alpha; \\ min\{L_{max}, L_t + 1\}, & \text{if } \lambda_H > \beta, \end{cases} \tag{15}$$

where α and β are moving thresholds.

D. Clearing Price Calculation

In the auction, the clearing price is calculated based on critical average per-unit bid.

Definition 6 (Critical Average Per-unit Bid): The critical average per-unit bid $\pi_i^\star$ for buyer i is defined as the minimum value that her average per-unit bid π_i must exceed to win her request in the cloud resource auction.

According to the definition, we note that: (1) For each buyer i, if $\pi_i > \pi_i^\star$, i will be ultimately selected as a winner; otherwise, she will lose; (2) $\pi_i^\star$ is determined by $\hat{\theta}_{-i}$, and thus independent from i's declared request $\hat{\theta}_i$.

In the auction, each buyer i's critical average per-unit bid $\pi_i^\star$ is determined when she is about to depart from the cloud system, *i.e.*, either she finishes running her job, or she is excluded from Φ because of passed deadline. The auction calculates $\pi_i^\star$ for each $i \in N$ by the following procedures:

1) Construct a request set of $\hat{\theta}_i^t = (\hat{a_i^t}, \hat{h}_i, \hat{d_i^t}, \hat{r_i}, \hat{v_i})$, $\forall t \in [\hat{a}_i, \hat{d}_i - \hat{h}_i]$, where $d_i^t = a_i^t + h_i$.
2) Suppose buyer i declares her request as $\hat{\theta}_i^t$, which means her arrival time is a_i^t, and her request should be accepted or refused immediately due to $d_i^t = a_i^t + h_i$. Since we have already had all historical information by time slot $\hat{d}_i + 1$, we can recover the entire allocation process and replay $\mathcal{M}'_t$.
3) We calculate $\pi_i^{t\star}$ that is buyer i's critical average per-unit bid, supposing $\hat{\theta}_i^t$ is her declared request. $\pi_i^{t\star}$ can be calculated by rerunning Algorithm 1 in $\mathcal{M}'_t$, until $\hat{\theta}_i^t$ cannot be satisfied. The threshold critical average per-unit bid is then determined as $\pi_i^{t\star}$.
4) We finally calculate buyer i's critical average per-unit bid $\pi_i^\star$ as $\pi_i^\star = min\{\pi_i^{t\star} | t \in [\hat{a}_i, \hat{d}_i - \hat{h}_i]\}$.

Since the size of set $\hat{\theta}_i^t$ is $\hat{d}_i - \hat{h}_i - \hat{a}_i + 1$, the entire process can be completed in polynomial time.

For any winning buyer $i \in W$, given her critical average per-unit bid $\pi_i^\star$, her clearing prices p_i is calculated as

$$p_i = \pi_i^\star \times \hat{h}_i \times \mathcal{A}(\hat{r_i}) \tag{16}$$

V. Analysis

In this section, we prove that our cloud resource auction is strategy-proof and achieves a nontrivial competitive ratio.

A. Strategy-proofness

Individual rationality

Theorem 1: The auction achieves individual rationality.

Proof: Suppose each truthful winning buyer $i \in W$ is determined in $\mathcal{M}_t$, *i.e.*, $i \in W_t$. According to the procedures of clearing price calculation, we have $\pi_i^{t\star} \le \pi_i$, because otherwise, i cannot win in $\mathcal{M}_t$. Since $\pi_i^\star = min\{\pi_i^{t\star} | t \in [\hat{a}_i, \hat{d}_i - \hat{h}_i]\}$, we have $\pi_i^\star \le \pi_i$. Hence, we can obtain that

$$\begin{aligned} u_i(\mathcal{F}(\theta_i, \hat{\theta_{-i}}), \theta_i) &= v_i - p_i \\ &= \pi_i h_i \mathcal{A}(r_i) - \pi^\star h_i \mathcal{A}(r_i) \\ &= (\pi_i - \pi^\star) h_i \mathcal{A}(r_i) \ge 0 \end{aligned}$$

We can see that for any winning user, her utility will be no less that 0. By not participating in the cloud resource auction, a buyer cannot win any resources, and her utility is 0. So participating is always not worse than staying outside. Therefore, the auction satisfies individual rationality. ■

Incentive compatibility

Lemma 1: If $\forall i \in N$ is selected as a winner determined in $\mathcal{M}_t, t \in T$, *i.e.*, $i \in W_t$, the window size L_t is calculated independently from her request $\hat{\theta}_i$.

Proof: Assume by contradiction that $\exists i \in N$ and $\exists t \in T$, such that $i \in W_t$ and $L'_k \neq L_k$ is caused by that buyer i declares an untruthful request $\hat{\theta}_i = \hat{\theta}'_i \neq \hat{\theta}_i$. Then, according to the window size calculation by equations (14) and (15), buyer i must be selected as a winner in $\mathcal{M}_k$, $k < t$, *i.e.*, $i \in W_k$, contradicting the assumption of $i \in W_t$. ■

Lemma 2: The following inequality holds for all $i \in N$, θ_i, $\hat{\theta_{-i}}$: $\forall \hat{a}_i = a'_i > a_i$, $\hat{d}_i = d'_i < d_i$,

$$\begin{aligned} & u_i(\mathcal{F}((a_i, h_i, d_i, r_i, v_i), \hat{\theta_{-i}}), \theta_i) \quad \geq \\ & u_i(\mathcal{F}((\hat{a}_i, h_i, \hat{d}_i, r_i, v_i), \hat{\theta_{-i}}), \theta_i). \end{aligned}$$

Proof: According to the procedures of clearing price calculation, we have $\pi_i^{'\star} = min\{\pi_i^{t\star} | t \in [a'_i, d'_i - h_i]\}$ and $\pi_i^{\star} = min\{\pi_i^{t\star} | t \in [a_i, d_i - h_i]\}$. Since $a'_i > a_i$ and $d'_i < d_i$, we have $\{\pi_i^{t\star} | t \in [a'_i, d'_i - h_i]\} \subset \{\pi_i^{t\star} | t \in [a_i, d_i - h_i]\}$, and thus $\pi_i^{'\star} \geq \pi_i^{\star}$. Now we distinguish three cases:

Case 1: Buyer i will not be selected as a winner by declaring whether $\hat{\theta}_i = \theta_i = (a_i, h_i, d_i, r_i, v_i)$ or $\hat{\theta}_i = \theta'_i = (a'_i, h_i, d'_i, r_i, v_i)$. In this case, the inequality trivially holds, because her utilities are both 0.

Case 2: Buyer i will be selected as a winner by declaring $\hat{\theta}_i = \theta_i = (a_i, h_i, d_i, r_i, v_i)$, but will not by $\hat{\theta}_i = \theta'_i = (a'_i, h_i, d'_i, r_i, v_i)$. Since we have proved that the auction satisfies individual rationality, in this case, the inequality also trivially holds.

Case 3: Buyer i will be selected as a winner by declaring whether $\hat{\theta}_i = \theta_i = (a_i, h_i, d_i, r_i, v_i)$ or $\hat{\theta}_i = \theta'_i = (a'_i, h_i, d'_i, r_i, v_i)$. In this case, we can obtain that

$$\begin{aligned} & u_i(\mathcal{F}(\theta_i, \hat{\theta_{-i}}), \theta_i) - u_i(\mathcal{F}(\theta'_i, \hat{\theta_{-i}}), \theta_i) \\ = \quad & \pi_i^{'\star} h_i \mathcal{A}(r_i) - \pi_i^{\star} h_i \mathcal{A}(r_i) \geq 0 \end{aligned}$$

Therefore, we complete this proof. ■

Lemma 3: The following inequality holds for all $i \in N$, θ_i, $\hat{\theta_{-i}}$: $\forall \hat{a}_i = a'_i > a_i$, $\hat{d}_i = d'_i < d_i$, $\hat{h}_i = h'_i \neq h_i$, $\hat{r}_i = r'_i \neq r_i$,

$$\begin{aligned} & u_i(\mathcal{F}((\hat{a}_i, h_i, \hat{d}_i, r_i, v_i), \hat{\theta_{-i}}), \theta_i) \quad \geq \\ & u_i(\mathcal{F}((\hat{a}_i, \hat{h}_i, \hat{d}_i, \hat{r}_i, v_i), \hat{\theta_{-i}}), \theta_i). \end{aligned}$$

Proof: According to the procedures of clearing price calculation, we have $u_i(\mathcal{F}((\hat{a}_i, h_i, \hat{d}_i, r_i, v_i), \hat{\theta_{-i}}), \theta_i) \geq 0$. If buyer i declares $\hat{h}_i$ and $\hat{r}_i$, such that the resource demand and running duration cannot be satisfied, then $u_i(\mathcal{F}((\hat{a}_i, \hat{h}_i, \hat{d}_i, \hat{r}_i, v_i), \hat{\theta_{-i}}), \theta_i) = 0$, because her job cannot complete at all. Therefore, we only consider that buyer i declares h'_i and r'_i, which results in higher demand than actual need. We denote $\pi_i^{'\star} = min\{\pi_i^{'t\star} | t \in [a'_i, d'_i - h_i]\}$ as buyer i's critical average per-unit bid when declaring $\hat{\theta}_i = \theta'_i = (a'_i, h_i, d'_i, r_i, v_i)$, and $\pi_i^{''\star} = min\{\pi_i^{''t\star} | t \in [a'_i, d'_i - h'_i]\}$ as when declaring $\hat{\theta}_i = \theta''_i = (a'_i, h'_i, d'_i, r'_i, v_i)$. Since $h'_i > h_i$ and r'_i requires more resources that r_i, we have $\pi_i^{'t\star} \leq \pi_i^{''t\star}$, $t \in [a'_i, d'_i - h'_i]$. Hence, we have $\pi_i^{'\star} \leq \pi_i^{''\star}$. Now we we distinguish three cases:

Case 1: Buyer i will not be selected as a winner by declaring whether $\hat{\theta}_i = \theta'_i$ or $\hat{\theta}_i = \theta''_i$. In this case, the inequality trivially holds, because her utilities are both 0.

Case 2: Buyer i will be selected as a winner by declaring $\hat{\theta}_i = \theta'_i$, but will not by $\hat{\theta}_i = \theta''_i$. Since $u_i(\mathcal{F}(\theta'_i, \hat{\theta_{-i}}), \theta_i) \geq 0$, in this case, the inequality also trivially holds.

Case 3: Buyer i will be selected as a winner by declaring whether $\hat{\theta}_i = \theta'_i$ or $\hat{\theta}_i = \theta''_i$. In this case, we can obtain that

$$\begin{aligned} & u_i(\mathcal{F}(\theta'_i, \hat{\theta_{-i}}), \theta_i) - u_i(\mathcal{F}(\theta''_i, \hat{\theta_{-i}}), \theta_i) \\ = \quad & \pi_i^{''\star} h_i \mathcal{A}(r_i) - \pi_i^{'\star} h_i \mathcal{A}(r_i) \geq 0 \end{aligned}$$

Therefore, we complete this proof. ■

Lemma 4: The following inequality holds for all $i \in N$, θ_i, $\hat{\theta_{-i}}$: $\forall \hat{a}_i = a'_i > a_i$, $\hat{d}_i = d'_i < d_i$, $\hat{h}_i = h'_i \neq h_i$, $\hat{r}_i = r'_i \neq r_i$, $\hat{v}_i = v'_i \neq v_i$,

$$\begin{aligned} & u_i(\mathcal{F}((\hat{a}_i, \hat{h}_i, \hat{d}_i, \hat{r}_i, v_i), \hat{\theta_{-i}}), \theta_i) \quad \geq \\ & u_i(\mathcal{F}((\hat{a}_i, \hat{h}_i, \hat{d}_i, \hat{r}_i, \hat{v}_i), \hat{\theta_{-i}}), \theta_i). \end{aligned}$$

Proof: We denote $\pi_i^{''\star} = min\{\pi_i^{''t\star} | t \in [a'_i, d'_i - h'_i]\}$ as buyer i's critical average per-unit bid when declaring $\hat{\theta}_i = \theta''_i = (a'_i, h'_i, d'_i, r'_i, v_i)$, and $\pi_i^{'''\star} = min\{\pi_i^{'''t\star} | t \in [a'_i, d'_i - h'_i]\}$ as when declaring $\hat{\theta}_i = \theta'''_i = (a'_i, h'_i, d'_i, r'_i, v'_i)$. Now we we distinguish four cases:

Case 1: Buyer i will not win by declaring whether $\hat{\theta}_i = \theta''_i$ or $\hat{\theta}_i = \theta'''_i$. In this case, the inequality trivially holds.

Case 2: Buyer i will not win by declaring $\hat{\theta}_i = \theta''_i$, but will win by $\hat{\theta}_i = \theta'''_i$. Suppose by declaring $\hat{\theta}_i = \theta'''_i$, buyer i will win in $\mathcal{M}_t$, $\exists t \in [a'_i, d'_i - h'_i]$. Then, we must have $\hat{v}_i > v_i$ and $\pi'''_i \geq \pi_i^{'''t\star} = \pi_i^{''t\star} \geq \pi''_i$. Hence, we have

$$\begin{aligned} u_i(\mathcal{F}(\theta'''_i, \hat{\theta_{-i}}), \theta_i) &= \pi_i h_i \mathcal{A}(r_i) - \pi_i^{'''t\star} h'_i \mathcal{A}(r'_i) \\ &\leq \pi_i h_i \mathcal{A}(r_i) - \pi_i^{''} h'_i \mathcal{A}(r'_i) \\ &\leq \pi_i h'_i \mathcal{A}(r'_i) - \pi_i h'_i \mathcal{A}(r'_i) = 0 \end{aligned}$$

Case 3: Buyer i will win by declaring $\hat{\theta}_i = \theta''_i$, but will not win by $\hat{\theta}_i = \theta'''_i$. Suppose by declaring $\hat{\theta}_i = \theta''_i$, buyer i will win in $\mathcal{M}_t$, $\exists t \in [a'_i, d'_i - h'_i]$.

$$\begin{aligned} u_i(\mathcal{F}(\theta''_i, \hat{\theta_{-i}}), \theta_i) &= v_i - \pi_i^{''t\star} h'_i \mathcal{A}(r'_i) \\ &= (h'_i \mathcal{A}(r'_i))(\pi_i^{''} - \pi_i^{''t\star}) \geq 0 \end{aligned}$$

Case 4: Buyer i will win by declaring $\hat{\theta}_i = \theta''_i$, and will also win by $\hat{\theta}_i = \theta'''_i$. In this case, we must have $\pi_i^{''\star} = \pi_i^{'''\star}$. Hence, we have

$$\begin{aligned} u_i(\mathcal{F}(\theta''_i, \hat{\theta_{-i}}), \theta_i) &= v_i - \pi_i^{''t\star} h'_i \mathcal{A}(r'_i) \\ &= v_i - \pi_i^{'''t\star} h'_i \mathcal{A}(r'_i) \\ &= u_i(\mathcal{F}(\theta'''_i, \hat{\theta_{-i}}), \theta_i) \end{aligned}$$

Therefore, we complete this proof. ■

Theorem 2: The auction achieves Incentive compatibility.

Proof: Combining Lemma 2, 3, and 4, we have that the following inequality holds for all $i \in N$, θ_i, $\hat{\theta_{-i}}$: $\forall \hat{a_i} \geq a_i$, $\hat{d_i} \leq d_i$, $\hat{h_i}$, $\hat{r_i}$, $\hat{v_i}$,

$$\begin{aligned} u_i(\mathcal{F}((a_i, h_i, d_i, r_i, v_i), \hat{\theta_{-i}}), \theta_i) \quad &\geq \\ u_i(\mathcal{F}((\hat{a_i}, \hat{h_i}, \hat{d_i}, \hat{r_i}, \hat{v_i}), \hat{\theta_{-i}}), \theta_i). \end{aligned}$$

Therefore, for any buyer $i \in N$, her best strategy is to simply declare her truthful requests, *i.e.*, $\hat{\theta}_i = \theta_i$. ■

Theorem 3: Combining theorem 1 and 2, the auction achieves both individual rationality and incentive compatibility, and therefore achieves strategy-proofness.

B. Competitive Ratio

Theorem 4: The auction achieves a competitive ratio of $\frac{\pi_{max}}{\pi_{min}}\eta h_{max}\rho$, where $\eta = r_{max}h_{max}\max_{x\in[M]}\sum_{k\in[C]} m_x^k$ and $\rho = \max_{\forall i,j\in[M]} \mathcal{A}(m_i)/\mathcal{A}(m_j)$.

Proof: We assign each unit of resources with a different label, and use $c_{x,y}$, $x \in [C], y \in [1, c_x]$ to denote the y^{th} unit of type-i resource. Since the cloud system is operated in a time-slotted fashion, we further use c_{xy}^t, $x \in [C], y \in [1, c_x], t \in T$ to denote the resource slot of the y^{th} unit of type-x resource in time slot t. Hence, for each winning buyer $i \in W$, her resource assignment can be denoted as a set of resource slots $\Omega_i = \{c_{xy}^t | \sum c_{xy}^t = \sum_{k\in[M]} r_i^k m_k^x, \forall x \in [C], \forall t \in [s_i, s_i + h_i)\}$. We use W_{OPT} to denote the set of winning user in the optimal solution, and similarly use W_{OUR} to denote the set of winning users in our auction. Let $\Omega_{OPT} = \{\Omega_i | i \in W_{OPT}\}$ represent the set of winning users' assignments in the optimal solution, and similarly let $\Omega_{OUR} = \{\Omega_i | i \in W_{OUR}\}$ represent the set of winning users' assignments in the optimal solution.

We claim that $\forall \Omega_i \in (\Omega_{OPT} - \Omega_{OUR})$, $\exists \Omega_j \in (\Omega_{OUR} - \Omega_{OPT})$ denoted by Ω_j^i s.t. $\exists c_{xy}^t \in (\Omega_i \cap \Omega_j)$. Assume by contradiction that $\exists \Omega_i \in (\Omega_{OPT} - \Omega_{OUR})$, $\forall \Omega_j \in (\Omega_{OUR} - \Omega_{OPT})$, s.t. $(\Omega_i \cap \Omega_j) = \emptyset$. Since Ω_i does not conflict with each scheduled request in Ω_{OUR}, Ω_i must be a feasible resource assignment for i. Consequently, i must be a winner in OUR, contradicting the assumption. Now we can see that for each $\Omega_i \in \Omega_{OPT}$, there must exist $\Omega_j^i \in \Omega_{OUR}$. Since $|\Omega_i| \leq \eta = r_{max}h_{max}\max_{x\in[M]}\sum_{k\in[C]} m_x^k$, there are at most η number of $\Omega_i \in \Omega_{OPT}$ mapped to each $\Omega_j \in \Omega_{OUR}$. Hence, for each v_j and all corresponding v_i, we have

$$\eta \frac{v_j}{h_j \mathcal{A}(r_j)} \cdot \frac{\pi_{max}}{\pi_{min}} \geq \sum \frac{v_i}{h_i \mathcal{A}(r_i)}.$$

We use ρ to denote $\max\limits_{\forall i,j\in[M]} \mathcal{A}(m_i)/\mathcal{A}(m_j)$, then we have

$$\eta \frac{\pi_{max}}{\pi_{min}} h_{max} \rho v_j \geq \sum v_i.$$

Since this holds for each $v_j, \Omega_j \in \Omega_{OUR}$, our proof completes. ■

VI. Performance

A. Methodology

We implement our auction design and evaluate its performance. In our simulations, we consider a the cloud system which contains 200 units of CPU, 800 units of RAM, and 1600 units of Disk. As shown in table III, the cloud system provides 5 types of VMs for various processing requirements. Each time slot is 10min, and the cloud system runs 1 week, *i.e.*, spanning $6 * 24 * 7$ time slots.

TABLE III
VM TYPES

VM		CPU	RAM	Disk
v.s	small	1	4	8
v.c	computation-intensive	2	4	8
v.r	memory-intensive	1	8	8
v.d	storage-intensive	1	4	16
v.l	large	2	8	16

We simulate the daily usage pattern [11] by varying the number of users arriving in the system in different time. Specifically, the number of users arriving within in each hour t is determined by $Q(t) = (x + xysin(t \cdot \frac{\pi}{12}))\varphi(t)$. x controls the average number of users, and we choose various user densities in our simulation; y contols the temporal variance, reflecting the daily usage pattern. According to [11], the peak-mean-ratio is about 1.3, so we set $y = 0.3$; to reflect the dynamics, we set $\varphi(t) \in [0.95, 1.05]$. Each user can submit her request of no more than 3 VM instances, and each resource requests is randomly selected from 10 minutes to 2 hours. The other parameters are set as: $L_{min} = 18$, $L_{max} = 36$, $\gamma = 1.05$, $\alpha = 0.2$, and $\beta = 0.5$.

Since most existing designs, *e.g.*, [12]–[14], [17], do not consider request deadline, we select first-arrive-first-serve (fafs) as the benchmark, which is similar to COCA in [16]. We compare the performance of our proposed auction with the benchmark in terms of social welfare and request completion ratio. Each simulation is run 100 times, the result is averaged.

B. Performance on social welfare

In this set of our simulations, we compare the social welfare of our auction with fafs under various average number of users arriving per hour.

Figure 4 shows the ratio of social welfare of our auction to fafs. We see that when the average number of users arriving per hour is small, our auction can achieve nearly the same social welfare as fafs. When more users arrives, our auction can achieve better social welfare than fafs, and the ratio of social welfare of our auction to fafs increases as user number increases. Under the highest user density in our simulation, *i.e.*, 800 users arriving in the system on average, our auction can improve social welfare by more than 35%.

C. Performance on completion ratio

In this set of our simulations, we compare the request completion ratio of our auction with fafs under various average number of users arriving per hour.

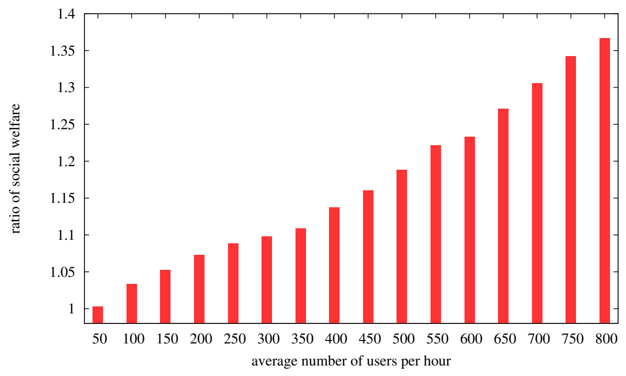

Fig. 4. An illustrative example to show the efficacy of deadline-awareness.

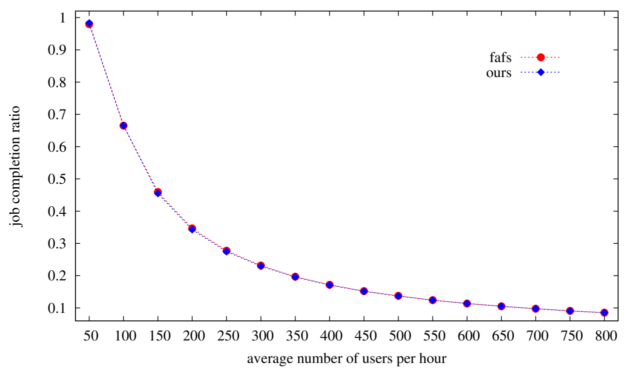

Fig. 5. An illustrative example to show the efficacy of deadline-awareness.

Figure 5 shows the comparison of request completion ratio between our auction and fafs. We see that as the average number of users increases, the ratio decreases due to the limited resources. It is also clear that our auction and fafs can always satisfy almost the same number of requests under various user density.

Overall, we can see that with various average number of users arriving in the system per hour, our auction outperforms fafs on social welfare, and they both satisfy almost the same number of requests. Therefore, we conclude that compared with fafs, our cloud resource auction can improve social welfare without any loss on request completion.

VII. Related Work

Recently, cloud resource auction design has been attracting substantial research interests. A number of elegant mechanisms and strategies have been proposed to study this problem from various perspectives. Some of these works are focused on single round cloud resource auction design. For example, Wang *et al.* [13] proposed a cloud computing resource auction that is computationally efficient and collusion resistant. In [17], Zhang *et al.* designed a truthful auction using the LP decomposition for dynamic VM provisioning. Meanwhile, several online cloud resource auction mechanisms were proposed. In [16], Zhang *et al.* designed a bidding language to capture user demand characteristics and applied to a cloud resource auction design. Wang *et al.* [14] proposed an online auction for cloud markets, achieving near-optimal allocation capacity. Shi *et al.* [12] stepped forward and proposed an online auction for cloud resource provisioning by using a primal-dual algorithm to decompose the long-term optimization into independent one-shot optimization problems.

Another category of related work is online auction mechanism design. Lavi and Nisan [5] initiated the study of online auction in the domain of computer science. Parkes *et al.* [9] analyzed VCG-based online mechanism with Markov decision process. Porter [10] studied mechanism design for online real-time jobs scheduling. Wu *et al.* [15] studied the problem of online auction design with time discounting values. However, none of these designs fully supports deadline-aware heterogeneous request, and thus can be directly applied to solving the problem considered herein.

VIII. Conclusion

In this paper, we have modeled the problem of deadline-aware cloud resource allocation as an online combinatorial auction, and proposed a strategy-proof and approximately efficient cloud resource auction. We have implemented our auction and performed extensive evaluations. Our evaluation results verify the efficacy of our design, and show that compared with the benchmark of the first-arrival-first-serve scheme, our auction can achieve better performance on social welfare without the loss on request completion.

References

[1] *Amazon EC2 Spot Instances*, http://aws.amazon.com/ec2/purchasing-options/spot-instances/.
[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM (CACM)*, vol. 53, no. 4, 2010.
[3] M. T. Hajiaghayi, R. D. Kleinberg, M. Mahdian, and D. C. Parkes, "Online auctions with re-usable goods," in *Proceedings of the 6th ACM Conference on Economics and Computation (EC'05)*, Vancouver, Canada, Jun 2005.
[4] V. Krishna, *Auction Theory*. Academic Press, 2002.
[5] R. Lavi and N. Nisan, "Competitive analysis of incentive compatible online auctions," in *Proceedings of the 2nd ACM Conference on Economics and Computation (EC'00)*, Minneapolis, MN, USA, Oct 2000.
[6] A. Mas-Colell, M. D. Whinston, and J. R. Green, *Microeconomic Theory*. Oxford University Press, 1995.
[7] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*. Cambridge University Press, 2007.
[8] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. The MIT Press, 1994.
[9] D. C. Parkes, S. Singh, and D. Yanovsky, "Approximately efficient online mechanism design," in *Proceedings of the 18th Annual Conference on Neural Information Processing Systems (NIPS'04)*, Vancouver, Canada, Dec 2004.
[10] R. Porter, "Mechanism design for online real-time scheduling," in *Proceedings of the 5th ACM Conference on Economics and Computation (EC'04)*, New York, New York, USA, May 2004.
[11] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the 2012 ACM Symposium on Cloud Computing (SOCC'12)*, San Jose, CA, USA, Oct 2012.
[12] W. Shi, L. Zhang, C. Wu, Z. Li, and F. C. Lau, "An online auction framework for dynamic resource provisioning in cloud computing," in *Proceedings of the 2014 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'14)*, Austin, TX, USA, Jun 2014.

[13] Q. Wang, K. Ren, and X. Meng, "When cloud meets ebay: Towards effective pricing for cloud computing," in *Proceedings of the 31st IEEE International Conference on Computer Communications (INFOCOM'12)*, Orlando, Florida, USA, Mar 2012.

[14] W. Wang, B. Liang, and B. Li, "Revenue maximization with dynamic auctions in iaas cloud markets," in *Proceedings of the 33rd IEEE International Conference on Distributed Computing Systems (ICDCS'13)*, Philadelphia, PA, USA, Jul 2013.

[15] F. Wu, J. Liu, Z. Zheng, and G. Chen, "A strategy-proof online auction with time discounting values," in *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14)*, Quebec, Canada, Jul 2014.

[16] H. Zhang, B. Li, H. Jiang, F. Liu, A. V. Vasilakos, and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands," in *Proceedings of the 32nd IEEE International Conference on Computer Communications (INFOCOM'13)*, Turin, Italy, Apr 2013.

[17] L. Zhang, Z. Li, and C. Wu, "Dynamic resource provisioning in cloud computing: A randomized auction approach," in *Proceedings of the 33rd IEEE International Conference on Computer Communications (INFOCOM'14)*, Toronto, Canada, Apr 2014.

How to reference this report:
Zhang, T. and Xin, Y. (2015): Designing a Strategy-Proof Online Auction Mechanism for Deadline Aware Cloud Resource Allocation. RENCI, University of North Carolina at Chapel Hill. Text. http://dx.doi.org/10.7921/G0H12ZX1