Resource Pricing and Demand Allocation for Revenue Maximization in IaaS Clouds: A Market-Oriented Approach

Songyuan Li, Jiwei Huang, *Member, IEEE,* and Bo Cheng, *Member, IEEE*

***Abstract*—With more users outsourcing their applications to the cloud, resource pricing becomes an important issue for IaaS cloud management. Jointly considering her own bidding budget and the price of cloud resources, each user is self-motivated to purchase cloud resources according to her resource demand which maximizes her own utility. Meanwhile, the cloud service provider (CSP) regulates the price of cloud resources with a certain profitability objective achieved. With an elaborate resource pricing strategy, the goals from users and the CSP are balanced and respectively satisfied to some extent. This article provides an insight into the market-oriented cloud pricing strategy. In specific, we propose an auction market in the IaaS cloud, where multiple users with heterogeneous bidding budgets and QoS requirements subscribe cloud resources according to their resource demands. The resource pricing and demand allocation scheme targeting revenue maximization also satisfies essential properties including budget feasibility, incentive compatibility and envy-freeness. To attack the NP-hardness and non-convexity of revenue maximization problem, we design a price-incentive resource auction mechanism namely RARM, which preserves an $(1+\alpha)$ approximation ratio on revenue maximization. Finally, we evaluate our RARM mechanism based on the real-world dataset to certify the efficacy of our proposed approach.**

***Index Terms*—IaaS cloud, market-oriented pricing strategy, resource auction, revenue maximization**

I. Introduction

Regarding the user-friendly advantages of offering high-performance but cost-saving service, cloud computing gains more momentum, and attracts more users to execute their applications at cloud [1]. With the increasing scale of cloud users, resource management has become a hot topic in the IaaS cloud community. In order to constantly provide more users with high Quality of Service (QoS), a cloud service provider (CSP) will deploy more cloud resources for the user use. For the CSP itself, excessive resource expansion implies an increase in cost, further reducing the CSP's profit earning. Given this, it necessitates an efficient resource management methodology considering the finite resource capacity rather than the pure resource expansion.

Through an efficient resource management strategy, the conflicting requirements from both sides of users and the CSP should be satisfied [2]. Specifically, cloud users prefer to spend less in purchasing more cloud resources with a higher QoS gained, whereas the CSP expects to gain more service revenue from users. To some extent, the above objectives respectively from users and the CSP collide with each other because of the finite cloud resources. On the one side, users submit requests to the CSP with high QoS demands, resulting in possessing more cloud resources. The more resources allocated to cloud users means a higher QoS offering. On the other side, however, it is impossible for the CSP to supply users with excessive cloud resources regardless of its own profit-rate.

A market-oriented resource pricing strategy can be an effective methodology that balances the conflicting demands between users and the CSP [3]. The CSP holds the power of resource pricing, which is aimed to achieve its profitability goal. As the role of price takers, users are self-motivated to take advantage of their purchasing capacities (i.e., bidding budgets) to conduct resource purchase, and thereby obtain their most desired QoS levels. The user's resource demand varies with the fluctuation of cloud resource price. With a lower price configured by the CSP, the resource demands of all users will be boosted. Oppositely, resource demands of users will be tightened. It follows that an optimal resource price should be worked out to establish an equilibrium between the users' and CSP's objectives, which is the research emphasis of this article.

The majority of CSPs (e.g., Microsoft Azure [4], Google Cloud Platform [5], and Amazon EC2 [6]) generally provide cloud users with various pricing options, where diverse pre-configured VMs are offered to satisfy heterogeneous user demands. Nevertheless, the current pricing schemes adopted in the cloud marketplace are mostly based on the fixed price, both for the *subscription-based* approach [7] and for the *pay-as-you-go* model [8]. The cloud user selecting the *subscription-based* option leases the IaaS infrastructure for a pre-specified subscription period. Meanwhile, the cloud user choosing the *per-as-you-go* model can discretionarily purchase on-demand instances, with the service payment charged according to the actual usage time. These two pricing methods are both based on the fixed pricing, regardless of the market dynamics including time-varied user demands and operational costs. In contrast, the market-oriented resource pricing strategy demon-

Manuscript received October 14, 2020; revised January 1, 2021. This work was supported by National Natural Science Foundation of China (No. 61972414), Beijing Nova Program of Science and Technology (No.Z201100006820082), Beijing Natural Science Foundation (No. 4202066), National Key Research and Development Program of China (No. 2018YFB1003800), and Fundamental Research Funds for Central Universities (No. 2462018YJRC040).

Songyuan Li and Bo Cheng are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: lisy@bupt.edu.cn, chengbo@bupt.edu.cn).

Jiwei Huang is with the Beijing Key Laboratory of Petroleum Data Mining, China University of Petroleum - Beijing, Beijing 102249, China (corresponding author, e-mail: huangjw@cup.edu.cn).

strates great market elasticity to identify market fluctuations in the cloud marketplace, which contributes to earning the maximum revenue for the CSP.

In this article, we study the market-oriented resource pricing strategy, and design a resource auction mechanism for multiuser IaaS clouds. With service bids dynamically proposed by users over time, the CSP will accordingly modulate the cloud resource price to regulate the resource demands of users. The user is self-motivated to purchase a certain amount of cloud resources based on her resource demand [9]. As adopted by some emerging cloud vendors [10] [11], the requested cloud resources can be bundled for each user, respectively as a custom-built VM. Meanwhile, the CSP earns the maximum revenue from users with a minimum profit rate ensured. The minimum profit-rate guarantee makes for recouping energy costs and gaining sufficient profits from users.

Revenue maximization is a classical problem of the resource auction design in cloud environments. Many existent literatures [12]–[14] are always based on the winner determination process, where cloud resources are discriminatorily priced for different users. Such a second-price auction, however, does not make sense that identical items should be valued at the same price. To fill this research gap, our resource auction mechanism employs a single-price auction in which cloud resources are marketed at the same unit price. Furthermore, our proposed resource auction mechanism also meets several essential properties, including budget feasibility, incentive compatibility, and envy-freeness. These three auction properties are simultaneously taken into accounts by very few literatures, but can significantly strengthen the sustainability of our resource auction mechanism. Details are as below:

- ***Budget feasibility*** alleges that a user's bidding budget should sufficiently cover her service payment [15], which is generally regarded as a basic requirement for auction mechanism design.
- ***Incentive compatibility*** eliminates the opportunism of misreporting the bidding budget, no matter what bidding strategies are employed by other users [16]. This property ensures that the user has the incentive to place her true bidding budget.
- ***Envy-freeness*** indicates a fairness criterion in the economic domain. Under a cloud price setting, each user can be allocated the amount of resources that maximize her utilities [17]. In this way, each user can always prefer her own allocated amount of cloud resources to other possible allocation results (including other users' allocated resources).

To summarize, our main contributions are listed as follows.

1) We model an auction market for IaaS clouds that adopts the market-oriented resource pricing strategy, based on which the revenue maximization problem is defined for the CSP. The NP-hardness and computational intractability of the revenue maximization problem are identified.
2) We develop a price-incentive resource auction mechanism namely `RARM` for multiuser IaaS clouds, where a computational-efficient resource pricing and demand allocation algorithm called `Revenue-Max` is proposed. The `Revenue-Max` algorithm can gain a near-optimal revenue for the CSP, with an $(1+\alpha)$ approximation ratio preserved.
3) Extensive simulations based on real-world datasets are conducted to manifest the efficacy of our proposed approach, with our auction properties empirically verified.

The remainder of our article is organized as follows. Section II reviews the background and related work. In Section III, we introduce and formulate our system model. Then, in Section IV, we formulate the user's utility and rational cloud purchase strategy, and then define the revenue maximization problem for the CSP. In Section V, we devise a resource auction mechanism called `RARM`. The resource pricing and demand allocation algorithm termed `Revenue-Max` solves the revenue maximization problem in an effective but efficient manner. Section VI evaluates the effectiveness and efficiency of our proposed approach. In Section VII, we conclude the research work and look into our future directions.

II. Background and Related Work

A. Pricing Methods in Cloud Marketplace

The increasing momentum of cloud computing services leads to various cloud pricing methods within the cloud marketplace. A variety of cloud pricing methods range from the incipient *subscription-based* approach [7], to the popular *pay-as-you-go* model [8], then to the recent *market-oriented* pricing strategy [18].

The nascent cloud pricing method can be approximately dated back to the subscription-based SaaS pricing model provided by Salesforce.com [7]. The subscription-based pricing approach is technically consolidated by software multitenancy. Each cloud tenant jointly shares the public operational cost, but still has private space to host her own cloud application. Thus, the subscription-based method presents a comparatively low price for infrastructure rental. A cloud user can freely subscribe the cloud instance with a fixed price differentiated in various CPU types or memory/disk sizes, for a certain period of time (e.g., in months/years).

To further stimulate the potential users conducting cloud computing transformation, a more attractive cloud pricing method emerges as a pay-as-you-go pricing model [8]. The cloud user selecting the pay-as-you-go model can purchase on-demand instances, and is only charged for the actual usage time of instances (e.g., in minutes/hours). The pay-as-you-go pricing method brings economic benefits to the cloud user, preventing from the over-subscribed instances. Thanks to its appeal to cloud users, the majority of CSPs, including Amazon EC2 [6], Google Cloud Platform [5], and Microsoft Azure [4], present the option of pay-as-you-go pricing model to cloud users. However, it is noted that on-demand instances are generally set at a fixed price as well, thereby incapable of reflecting the time-varied user demands [19] and the dynamic operational costs [20]. Hence, the CSP who adopts the pay-as-you-go pricing approach is less proficient in accurately recouping the dynamic operational costs and earning the maximum revenue from cloud users.

In the face of the above-mentioned limitations, a more flexible cloud pricing method then comes up as a market-oriented pricing strategy [18]. It can sufficiently reflect the

economic behavior in the cloud marketplace, through regulating the cloud price dynamically with supply and demand. One of the notorious is the Amazon EC2 Spot Instance, which takes up the dynamic/auction-based pricing mechanism [21]. The Amazon EC2 Spot Instance targets to make full use of the idle EC2 instances. In comparison with on-demand instances, the price of Spot Instances is normally with a sizable discount (up to 90 %). Technically, the price of Spot Instances is dynamically modulated based on the long-term trend in supply and demand for Spot Instance capacity. By means of this flexible cloud pricing strategy, both sides of the CSP and cloud users are incentivized in terms of economic benefits. Cloud users receive a better economic stimulus, and then voluntarily adjust their purchasing behavior in response to the volatile cloud marketplace; also, the CSP can earn much more service revenue [22] under the market-oriented cloud pricing strategy.

B. Market-Oriented Cloud Pricing Strategy

As stated in Section II-A, the market-oriented cloud pricing strategy demonstrates great advantages over the two other fixed-pricing methods (i.e., subscription-based, pay-as-you-go). Given this, it has drawn significant attention from academia and industry, to dynamically determine the pricing and allocation of cloud resources.

Regarding many indirect/hidden factors inhabited in the cloud marketplace, a black-box data-driven method termed online learning [23] can shed light on the market characteristics. The revealed market characteristics are utilized to design or study the market-oriented pricing strategy. Zhang et al. [24] exploit the past market information to predict future sales with the multi-armed-bandit-based online learning approach, based on which the online resource pricing decision is made. Prasad et al. [25] propose an online-learning-based Fisher market that enables online resource pricing and allocation, where both marketing randomness and resource integrality gap are fully considered. On the side of the cloud user with limited budgets, Wu et al. [26] take advantage of the online learning technique to infer her optimal purchasing bundle of on-demand and spot instances. The online learning approach assuredly brings some insight into the market-oriented cloud pricing strategy, but it still meets with crucial challenges. The success of the online learning approach originates from comprehensive marketing data, but an extensive data collection in the cloud marketplace is impracticable in general cases. Since the marketing data of each sector may belong to different market entities, the exploitation of cross-section marketing data matters for data security and privacy [27]. Yet, the online learning approach is commonly implemented in a centralized manner, with centralized data collection generally required.

In contrast, some domain-knowledge-based white-box methods can be intrinsically decentralized, including the game-theoretic approach and the auction-based approach. These approaches are driven by domain knowledge, thereby gaining better explainability than the online learning method. Regarding the game-theoretic approach, Cardellini et al. [28] formulate the IaaS provider's hybrid instance selling process as a Stackelberg game, with the objective of earning as much service revenue as possible. Ghosh et al. [29] investigate the interactions between IoT/wireless/cloud service providers in IoT scenario as a combination of sequential and parallel non-cooperative games, where an equilibrium pricing strategy is acquired through the game process. Li et al. [9] develop a price-incentive resource auction mechanism in cloud environments, whose objective is to stimulate the maximum cloud users served at the cloud. Siew et al. [30] introduce the game theory to study the sharing economy in the mobile cloud environment, based on which dynamic pricing mechanisms are designed for welfare/profit maximization. As for the auction-based approach, Jin et al. [12] put forward an incentive-compatible double auction mechanism for mobile cloud computing, where mobile devices' and cloudlets' contributions are dynamically priced. Hosseinalipour et al. [13] demonstrate the market-oriented interactions between different entities as a two-phase auction model, where the two stages of auctions are respectively formulated as an option-based sequential auction and an auction/flat-mixed market. Lu at al. [14] concern about a two-sided cloud market environment with multiple CSPs, on the basis of which a double auction mechanism is presented to match the users' and CSPs' requirements.

This article intends to make contributions towards the market-oriented cloud pricing strategy, with several unique features. *Firstly*, we conduct an elaborate formulation of user utility. With a voluntary basis, the cloud user purchases a specified amount of cloud resources according to her resource demand, where her own maximum utility is obtained. Unlike conceiving demand allocation as an assignment problem [28], [29] regardless of user utility, our scenario setting stands a lot closer to the realistic cloud marketplace. *Secondly*, we carry out a single-price auction where cloud resources are marketed with the same unit price, which differs from the existent literatures [12]–[14] adopting a second-price auction. It fits more with our intuitive knowledge that identical items should be valued at the same price. The second-price auction, nonetheless, proposes discriminatory prices for different users, which breaks the fairness principle in cloud pricing. *Thirdly*, we conduct an elaborate design on auction mechanism to enable some essential properties (i.e., *budget feasibility*, *incentive compatibility*, and *envy-freeness*) satisfied. Although taken into account by very few literatures, these three auction properties actually play an important part in the sustainability of our resource auction mechanism.

III. System Model

System Overview. Fig. 1 demonstrates our auction market in the multiuser IaaS cloud, which works in a time-slotted manner. The time horizon is discretized into a sequence of time slots at the duration of τ, indexed by t. At each time slot t, multiple users arbitrarily determine to place the service bid, and compete with each other to obtain cloud resources within the IaaS cloud infrastructure. The CSP collects the latest service bids proposed by various users at the beginning of each time slot t, and hereby makes decisions on the spot cloud resource price $p(t)$ as well as resource allocation scheme $A(t)$. The user u_i whose service bid is accepted is allocated

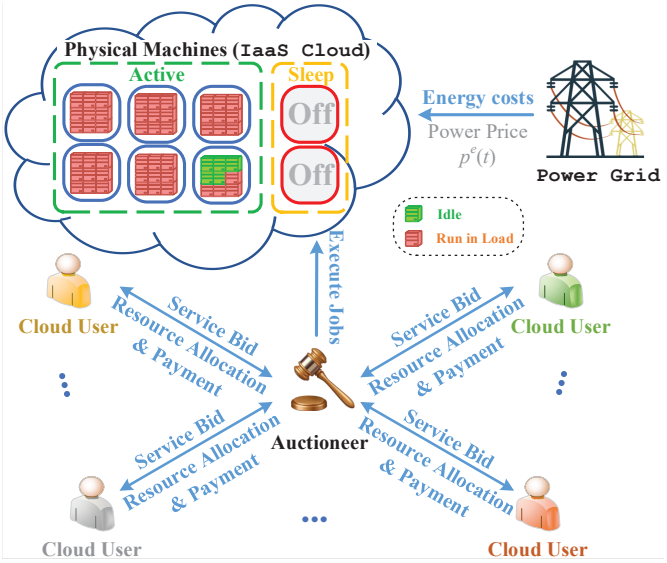

Fig. 1. Auction Market in the Multiuser IaaS Cloud.

$a_i(t)$ units of cloud resources equaling to her resource demand. The $a_i(t)$ units of allocated resources can be bundled as a custom-built VM [31], where the user u_i is served during the time slot t. With $a_i(t)$ units of allocated cloud resources, the user u_i is charged a service payment of $p(t) \cdot a_i(t)$.

Cloud Users. There are totally N users placing service bids over time slots, and we suppose that $N(t) \leq N$ users (denoted by $\mathcal{U}(t)$) place service bids at the time slot t. The set of N users is denoted by $\mathcal{U} = \{u_1, ..., u_N\}$, thus $\mathcal{U}(t) \subset \mathcal{U}$. Without loss of generality, we assume that each user u_i places a single service bid during one time slot. The user who concurrently places multiple service bids can be conceived as a group of users.

At each time slot t, each user $u_i \in \mathcal{U}(t)$ negotiates with the CSP by placing her service bid $\beta_i(t) = (s_i, b_i, \chi_i, t_i^-, t_i^+)$, in which the service type s_i, the bidding budget b_i, and the elastic QoS requirement $[t_i^-, t_i^+]$ are proposed. The service type s_i suggests the type of application requested by the user u_i for execution at the IaaS cloud. The bidding budget b_i implies the maximum monetary expense that the user u_i is willing to be charged per time slot for cloud resource subscription. The size of the executed job for user u_i during the period of cloud subscription is estimated by χ_i. Meanwhile, the user u_i's QoS requirement is elastic as $[t_i^-, t_i^+]$, indicating the desired range of response time.

During the time slot t, each user is allocated a_i units of cloud resources which can take various forms (e.g., CPU units, memory space, disk storage). Inspired by the *degree of parallelism* adopted by the Hadoop/Spark framework [32], the amount of allocated cloud resources $a_i(t)$ can reflect the number of allotted compute slots while each compute slot is compiled by a fix number of varied resources.

Cloud Service Provider. We restrict our research spotlight on a homogeneous set of physical machines administered by the CSP, as in [33] [34]. There are M physical machines administered by the CSP, each of which is equipped with r units of computational resources. To dwindle unnecessary operational costs and raise the profit earning, the CSP simply turns on the physical machines running in load, while the idle physical machines are shut down into the sleep mode. At each time slot t, the CSP collects the service bids from different users, and assigns $a_i(t)$ units of cloud resources to each user $u_i \in \mathcal{U}(t)$. Hence, as in [35], the number of physical machines required to be active during the time slot t is estimated by (1).

$$m(t) = \left\lceil \frac{\sum_{u_i \in \mathcal{U}(t)} a_i(t)}{r} \right\rceil \tag{1}$$

The CSP's operation costs primarily derive from the energy costs of IaaS infrastructure [20]. Suppose the average energy cost to perform a physical machine per time slot is $\widetilde{c}$, the total energy cost used for operating $m(t)$ physical machines during the time slot t is $\widetilde{c} \cdot m(t)$. The power price $p^e(t)$ generally fluctuates over time slots [36]. Thus, the CSP should be charged for the energy bill of $p^e(t) \cdot \widetilde{c} \cdot m(t)$ at the time slot t .

At each time slot t, the CSP also determines the spot unit resource price as $p(t)$, where each user $u_i \in \mathcal{U}(t)$ is charged for a service payment of $p(t) \cdot a_i(t)$. Under the cloud resource price of $p(t)$, the CSP should not only recoup energy costs from users, but also gain a minimum γ profit rate. Thus, the following constraint (2) is implied.

$$p(t) \cdot a_i(t) \geq (1 + \gamma) \cdot p^e(t) \cdot \widetilde{c} \cdot m(t) \tag{2}$$

IV. Problem Statement

A. User Utility Model

We firstly formulate the utility model for the user, representing the cloud user's preference towards different cloud resource allocation results. In this article, for a user u_i, her user utility jointly depends on the QoS gain from the resource allocation $a_i(t)$ and the associated momentary payout $p(t) \cdot a_i(t)$.

For a user u_i, her QoS gain (i.e., response time $t_i(a_i(t))$) reflects her "happiness" towards diverse resource allocation results $a_i(t)$. With more cloud resources $a_i(t)$ allocated, the user u_i can gain a better QoS level. As stated in the experimental evidence in [37] and the regression analysis results in [38], the response time $t_i(a_i(t))$ obtained by user u_i can be estimated as (3), where $\theta_{s_i,0}$, $\theta_{s_i,1}$, $\theta_{s_i,2}$, and $\theta_{s_i,3}$ are regression parameters specialized to the service type s_i.

$$\begin{aligned} t_i(a_i(t)) \; = \theta_{s_i,0} + \theta_{s_i,1} \cdot \frac{\chi_i}{a_i(t)} + \theta_{s_i,2} \cdot a_i(t) \\ + \theta_{3,s_i} \cdot \log(a_i(t)) \end{aligned} \tag{3}$$

In Eq. (3), the first term $\theta_{s_i,0}$ indicates the fixed time cost reflecting the part of serial computation; the second term $\theta_{s_i,1} \cdot \chi_i / a_i(t)$ characterizes the influence of job size χ_i towards the response time, which implies a greater job size χ_i results in a longer response time; the third term $\theta_{s_i,2} \cdot a_i(t)$ represents the all-to-one communication cost over compute slots (i.e., $a_i(t)$) [39], incurred by scheduling/serializing tasks across multiple compute slots; the last term $\theta_{3,s_i} \cdot \log(a_i(t))$ captures the communication pattern across compute slots like aggregation trees [40].

TABLE I
NOTATION AND DESCRIPTION

Notation	Description
τ, t	duration, index of a time slot.
u_i, $\mathcal{U}$, N	index, set, number of users placing the service bid over time slots.
$\mathcal{U}(t)$, $N(t)$	subset, number of users placing the service bid at the time slot t, i.e., $\mathcal{U}(t) \subset \mathcal{U}$, $M(t) \leq N$.
M	number of physical machines administered by the CSP.
$r, \widetilde{c}$	computational capacity, run-time costs (i.e., average energy costs per time slot) of each cloud physical machine.
$m(t)$	number of active physical machines during the time slot t.
$p^e(t)$	power price at the time slot t.
$p(t)$	spot price per unit of cloud resources at the time slot (i.e., billing cycle) t.
$\beta_i(t)$	service bid placed by the user u_i at the time slot t, where $\beta_i(t) = (s_i, b_i, \chi_i, t_i^-, t_i^+)$.
s_i	service type for the service bid $\beta_i(t)$, indicating the type of application requested by the user u_i for execution within the IaaS cloud.
b_i	bidding budget for the service bid $\beta_i(t)$, indicating the maximum monetary expense that the user u_i would like to be charged per time slot.
χ_i	size of the executed job for the user u_i during the period of her cloud subscription.
$[t_i^-, t_i^+]$	QoS requirement for the service bid $\beta_i(t)$, indicating the desired range of response time.
$[a_i^-, a_i^+]$	resource requirement for the service bid $\beta_i(t)$, indicating the desired range of cloud resource allocation.
$a_i(t)$	number of cloud computational units subscribed by the user u_i at the time slot t.
$A(t)$	resource allocation scheme at the time slot t, i.e., $A(t) \leftarrow \langle a_i(t) \rangle_{u_i \in \mathcal{U}(t)}$.
$\rho_i(a_i(t))$	QoS progress rate gained by the user u_i from $a_i(t)$ units of purchased cloud resources.
$g_i(a_i(t))$	degree of "happiness" gained by the user u_i from $a_i(t)$ units of purchased cloud resources, in terms of QoS progress.
$v_i(p(t), a_i(t))$	utility function for the user u_i as a bivariate function of $p(t)$ and $a_i(t)$.
$d_i(p_i(t)))$	user u_i's resource demand determined by her own utility maximization problem.
γ	minimum profit rate required by the CSP, i.e., ratio of profit over cost.
$\pi(t)$	overall service revenue earned by the CSP at the time slot t.

As mentioned in Section III, the user u_i proposes an elastic QoS requirement as the desired range of response time $[t_i^-, t_i^+]$. According to (3), the desired range of cloud resource allocation $[a_i^-, a_i^+]$ can be correspondingly figured out, as in (4). It is pointless for the user u_i to allocate $a_i(t) > a_i^+$ units of cloud resources, gaining a response time shorter than t_i^-. Additionally, it is unsatisfactory to allocate $a_i(t) < a_i^-$ cloud resources, incurring a response time longer than t_i^+.

$$a_i^- = t_i^{-1}(t_i^+), \quad a_i^+ = t_i^{-1}(t_i^-) \tag{4}$$

From the above, the "happiness" of the user u_i gained from $a_i(t)$ units of cloud resources, denoted by $g_i(a_i(t))$, can be formulated in (5) below. When $a_i(t) \in [a_i^-, a_i^+]$, we adopt the QoS progress rate $\rho_i(a_i(t))$ to evaluate the $g_i(a_i(t))$. The QoS progress rate $\rho_i(a_i(t))$ is defined as the response time $t_i(a_i(t))$ relative to t_i^+, formulated by (6). It is noted that, the QoS progress rate $t_i(a_i(t))$ is absolutely not proportionate to the amount of allocated resources $a_i(t)$, but with strong concavity [37].

$$g_i(a_i(t)) = \begin{cases} \rho_i(a_i^+) & \text{if } a_i(t) \in \left(a_i^+, +\infty\right) \\ \rho_i(a_i(t)) & \text{if } a_i(t) \in \left[a_i^-, a_i^+\right] \\ 0 & \text{if } a_i(t) \in \left[0, a_i^-\right) \end{cases} \tag{5}$$

$$\rho_i(a_i(t)) = \frac{t_i^+}{t_i(a_i(t))} \tag{6}$$

When gaining the QoS progress $\rho_i(a_i(t))$, the user u_i also has to be charged the service payment of $p(t) \cdot a_i(t)$. Different from the QoS gain, the service payment $p(t) \cdot a_i(t)$ contributes negatively to the user utility. To summarize, the user utility function $v_i(a_i(t))$ can be formulated as (7). The service payment $p(t) \cdot a_i(t)$ cannot exceed the bidding budget b_i; thus, the service payment $p(t) \cdot a_i(t)$ is normalized by the bidding budget b_i in (7).

$$v_i(p(t), a_i(t)) = \begin{cases} \rho_i(a_i^+) - \frac{p(t) \cdot a_i(t)}{b_i} & \text{if } a_i(t) \in \left(a_i^+, +\infty\right) \\ \rho_i(a_i(t)) - \frac{p(t) \cdot a_i(t)}{b_i} & \text{if } a_i(t) \in \left[a_i^-, a_i^+\right] \\ -\frac{p(t) \cdot a_i(t)}{b_i} & \text{if } a_i(t) \in \left[0, a_i^-\right) \end{cases} \tag{7}$$

B. Rational Resource Demand

After formulating the user utility model, we then investigate the cloud resource demand of each user $u_i \in \mathcal{U}(t)$ under a certain determined spot unit resource price $p(t)$. When the spot price per unit of cloud resources has quoted $p(t)$, each user $u_i \in \mathcal{U}(t)$ rationally determines her resource demand $d_i(p(t))$ through solving a utility maximization problem. In other words,

$$d_i(p(t)) \triangleq \arg\max_{a_i(t)} v_i(p(t), a_i(t)) \tag{8}$$

As indicated in (7), however, the segmentation and non-convexity of the user utility function $v_i(\cdot)$ prevent from solving

out the resource demand $d_i(p(t))$ efficiently. Given this, the following two cases are divided into discussion.

– **Case 1:** *The user u_i can purchase the maximum amount of cloud resources less than a_i^-, even if stretching its bidding budget b_i.* In this case, the user u_i's QoS requirement $[t_i^-, t_i^+]$ cannot be satisfied. Instead, it would arouse a negative utility if the user u_i purchases cloud resources. Hence, the user u_i inclines to forgo cloud resource subscription with $d_i(p(t)) = 0$.

– **Case 2:** *The user u_i has the enough budget b_i to purchase $a_i(t) > a_i^-$ units of cloud resources.* In this case, the user u_i would not like to purchase the amount of cloud resources greater than a_i^+, because her QoS requirement will be overfulfilled beyond T_i^- if $a_i(t) > a_i^+$, barely making for disutility instead. Therefore, the resource demand $d_i(p(t))$ maximizing the user u_i's utility is supposed to be found out within $[a_i^-, a_i^+]$.

From now on, we start by solving the resource demand $d_i(p(t))$ for the Case 2. Here, we formulate the Problem P1 which can determine the optimal $a_i(t)$ maximizing the user u_i's utility over $[a_i^-, a_i^+]$. The constraint (C1.1) is the requirement of ***budget-feasible*** resource allocation scheme [31], suggesting that the service payment $p(t) \cdot a_i(t)$ cannot exceed the bidding budget b_i.

$$\max_{a_i(t)} \quad \rho_i(a_i(t)) - \frac{p(t) \cdot a_i(t)}{b_i} \tag{P1}$$

$$\text{s.t.} \quad p(t) \cdot a_i(t) \leq b_i \tag{C1.1}$$

$$a_i^- \leq a_i(t) \leq a_i^+ \tag{C1.2}$$

It is noticeable that, Problem P1 is a convex optimization problem (The proof is omitted). Thus, we can solve the Problem P1 by seeking out the solution of the corresponding Karush-Kuhn-Tucker (KKT) equations [41]. The computational complexity of solving the KKT equations exponentially increases with the number of inequality constraints in the original convex optimization problem [42]. There are $K = 3$ inequality constraints (i.e., the upper bound of (C1.1), and the upper and lower bound of (C1.2)) in Problem P1, thus we can solve the Problem P1 for the user u_i through KKT equations in the computational complexity of $\mathcal{O}(2^K) = \mathcal{O}(1)$.

Based on the above, we epitomize the Case 1 and Case 2 to put forward the Rational Resource Demand Allocation Algorithm namely RARD, as shown in Algorithm 1. Each user $u_i \in \mathcal{U}(t)$ is allocated $a_i(t)$ units of cloud resources according to their own resource demands $d_i(p(t))$. When the spot unit resource price quotes $p(t)$, each user $u_i \in \mathcal{U}(t)$'s resource demand allocation can be sequentially determined, requiring the overall computational complexity of $\mathcal{O}(N(t))$. Moreover, the Problem P1 of each user u_i is independent with no correlation, thus the resource demand of each user $u_i \in \mathcal{U}(t)$ can also be solved in parallel, where the computational complexity is reduced to $\mathcal{O}(1)$.

C. Optimization Problem

The objective of our optimization problem is to maximize the service revenue from the CSP's standpoint. Therefore, we formulate the resource pricing and demand allocation problem as the Problem P2 below.

Algorithm 1: RARD: Rational Resource Demand Allocation Algorithm

Input: Spot Unit Resource Price $p(t)$; Service Bids of Users $\mathcal{U}(t)$ in the Time Slot t.
Output: Resource Allocation Scheme $A(t)$.

```
for each u_i ∈ U(t) do
    if p(t) · a_i^- > b_i then
        a_i(t) ← 0;   ▷ Case 1
    else
        Obtain the user u_i's resource demand allocation
        a_i(t) = d_i(p(t)) where her own utility v_i(a_i(t))
        is maximized, by solving the KKT equations of
        Problem P1;  ▷ Case 2
return A(t) ← ⟨a_i(t)⟩_{i=1}^{N(t)};
```

$$\max_{a_i(t), p(t)} \quad \pi(t) = p(t) \cdot \sum_{u_i \in \mathcal{U}(t)} a_i(t) \tag{P2}$$

$$\text{s.t.} \quad p(t) \cdot \sum_{u_i \in \mathcal{U}(t)} a_i(t) \leq M \cdot r \tag{C2.1}$$

$$p(t) \cdot \sum_{u_i \in \mathcal{U}(t)} a_i(t) \geq (1+\gamma) \cdot \widetilde{c} \cdot p^e(t) \cdot m(t) \tag{C2.2}$$

$$a_i(t) = d_i(p(t)) \quad \forall u_i \in \mathcal{U}(t) \tag{C2.3}$$

The constraint (C2.1) indicates the resource capacity constraint of IaaS infrastructure, and the constraint (C2.2) ensures the minimum γ profit rate required by the CSP. The constraint (C2.3) embodies the ***envy-free*** resource allocation scheme [43], meaning that each user $u_i \in \mathcal{U}(t)$ is allocated cloud resources by their own resource demand, which maximizes their own utilities.

Solving the Problem P2 is non-trivial because of the following two aspects. *Firstly*, our optimization Problem P2 belongs to the family of bin packing problem that is NP-hard to solve out [44]. Each physical machine in the IaaS cloud is conceived as a bin with finite computational resources. Our objective is to figure out the resource allocation scheme across multiple users with the service revenue maximized at each time slot t. *Secondly*, the user utility function $v_i(\cdot)$ (i.e., Eq.(7)) is in a piecewise and non-convex form. In Problem P2, each user u_i's resource allocation $a_i(t)$ is determined based on her own utility maximization, hence making the problem solving more complicated.

V. Mechanism Design

A. RARM: Resource Auction Mechanism Framework

To attack the aforementioned computation challenges, we develop a Resource Auction Mechanism for Revenue Maximization called RARM. We identify the RARM mechanism at each time slot t into two steps, as demonstrated in Fig. 2.

- **Step I:** After receiving the service bids proposed in the time slot t, the CSP firstly specifies the feasible resource

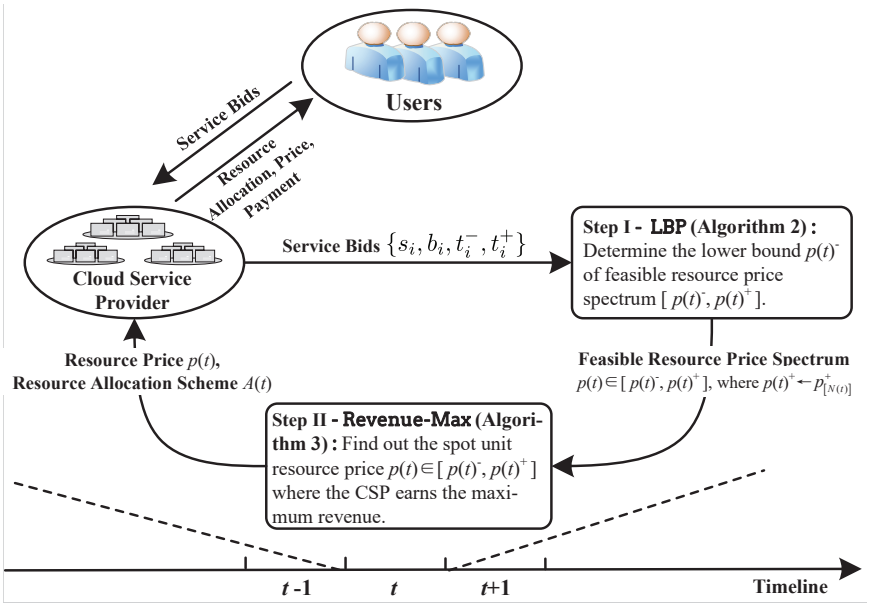

Fig. 2. Overview of our RARM Mechanism.

price spectrum of $p(t)$, denoted as $[p(t)^-, p(t)^+]$. Under the spot unit resource price $p(t) \geq p(t)^-$, not only the resource capacity constraint (C2.1) of IaaS infrastructure is satisfied, but also the minimum γ-profit-rate requirement is most likely to be ensured. The upper bound of price $p(t)^+$ indicates a threshold of resource price, where all the users $u_i \in \mathcal{U}(t)$ will abandon cloud resource subscription with $d_i(p(t)) = 0$ if $p(t) > p(t)^+$, hence the CSP does not gain service revenue from users.

- **Step II:** Within the resource price spectrum $[p(t)^-, p(t)^+]$, the CSP finalizes the revenue-optimal spot unit resource price $p(t)$ together with the associated resource allocation scheme $A(t)$, where the CSP earns the maximum service revenue with a minimum profit rate γ assuredly gained. The CSP manipulates the active/sleep mode of physical machines based on $A(t)$, and collects the service payment from the accepted users whose $a_i(t) > 0$.

B. Step I: Determine the Feasible Resource Price Spectrum

We firstly present the notation of p_i^+, on which our mechanism design depends. According to the bidding budget b_i and the desired range of cloud resource allocation $[a_i^-, a_i^+]$, we acquire each user u_i's maximum acceptable price p_i^+ per unit cloud resource, formulated in (9). If $p(t) > p_i^+$, then the user u_i will abandon cloud resource subscription with $d_i(p(t)) = 0$, hence the CSP cannot earn any service revenue from the user u_i. Here, we sort up the users $u_i \in \mathcal{U}(t)$ according to p_i^+ in an non-decreasing rank $\Theta = \langle u_{[1]}, ..., u_{[N(t)]} \rangle$, where $u_{[i]}$ indicates the user ranked in the i^{th} place of Θ.

$$p_i^+ = \frac{b_i}{a_i^-} \quad \text{for each } u_i \in \mathcal{U}(t) \tag{9}$$

The feasible spectrum of spot unit resource price $p(t)$ is defined as $[p(t)^-, p(t)^+]$. Specifically, we adopt $p^+_{[N(t)]}$ as the upper bound $p(t)^+$ of $[p(t)^-, p(t)^+]$, as in (10). This is because, the CSP cannot gain service revenue with $\pi(t) = 0$ if $p(t)$ is set higher than $p^+_{[N(t)]}$. Additionally, we determine the lower bound $p(t)^-$ of $[p(t)^-, p(t)^+]$, where both the resource capacity constraint (C2.1) and the minimum-γ-profit-rate constraint (C2.2) are taken into accounts.

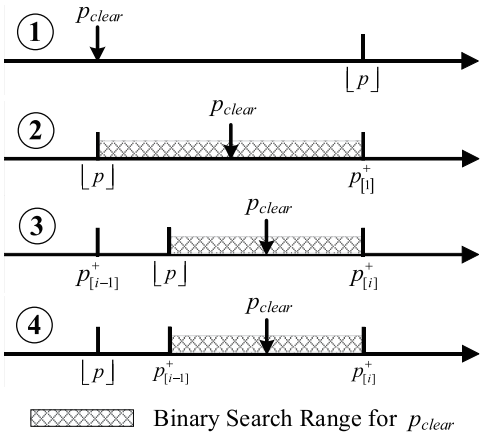

Fig. 3. Binary search range for p_{clear} in four cases.

We specifically determine the lower bound $p(t)^-$ of $[p(t)^-, p(t)^+]$ as follows. On the one hand, $p^+_{[i]}$ $(1 \leq i \leq N(t))$ is leveraged as the binary search boundary for the resource-clearing price p_{clear}. At the resource-clearing price p_{clear}, the total resource supply is equated to the overall demand allocation of users $u_i \in \mathcal{U}(t)$, which is $\sum_{u_i \in \mathcal{U}(t)} d_i(p(t)) = M \cdot r$. Thus, setting $p(t) \geq p_{clear}$ keeps the resource capacity constraint (C2.1) satisfied. On the other hand, we introduce the notation of $\lfloor p \rfloor = \widetilde{c} \cdot p^e(t) \cdot (1+\gamma)/r$, suggesting the unit resource price which equally split a physical machine's bottom price $\widetilde{c} \cdot p^e(t) \cdot (1+\gamma)$. By means of checking if $p(t) > \lfloor p \rfloor$, it can be roughly judged whether the minimum profit rate of γ is gained at the price of $p(t)$, in correspondence to the profit-rate constraint (C2.2). To sum up, the lower bound $p(t)^-$ is

Algorithm 2: LBP: Determining the Lower Bound of Spot Unit Resource Price

Input: Service Bids of Users $\mathcal{U}(t)$ in the Time Slot t.
Output: Lower Bound $p(t)^-$ of Spot Resource Price $p(t)$.

1 Obtain the resource allocation scheme $A(t)$ under $p(t) = \lfloor p \rfloor$ using the RARP algorithm;
2 **if** $\sum_{i=1}^{N(t)} a_i(t) \leq M \cdot r$ **then**
3 **return** $p(t)^- \leftarrow \lfloor p \rfloor$;
4 **else**
5 **for** *each* $i = \{1, 2, ..., N(t)\}$ **do**
6 Obtain the resource allocation scheme $A(t)$ under $p(t) = p^+_{[i]}$ using the RARD algorithm;
7 **if** $\sum_{j=1}^{N(t)} a_j(t) \leq M \cdot r$ **then**
8 $p^r \leftarrow p^+_{[i]}$;
9 **if** $i > 1$ and $p^+_{[i-1]} > \lfloor p \rfloor$ **then** $p^l \leftarrow p^+_{[i-1]}$;
10 **else** $p^l \leftarrow \lfloor p \rfloor$;
11 **break**;
12 **while** $p^r - p^l \geq \xi$ **do**
13 $p^m \leftarrow (p^r + p^l)/2$;
14 Obtain the resource allocation scheme $A(t)$ under $p(t) = p^m$ using the RARD algorithm;
15 **if** $\sum_{j=1}^{N(t)} a_j(t) \leq M \cdot r$ **then** $p^r \leftarrow p^m$;
16 **else** $p^l \leftarrow p^m$;
17 **return** $p(t)^- \leftarrow p^r$;

determined in (10).

$$p(t)^+ = p^+_{[N(t)]}, \quad p(t)^- = \max\{p_{clear}, \lfloor p \rfloor\} \tag{10}$$

The algorithm of determining the lower bound $p(t)^-$ called LBP is shown in Algorithm 2. Note that, the resource-clearing price p_{clear} can be figured out with the binary search method (Line 12-17). The range of binary search is specified in Line 1-11, according to the following four cases, as demonstrated in Fig. 3.

– **Case 1:** $p_{clear} \leq \lfloor p \rfloor$. We need not seek the resource-clearing price p_{clear} with the binary search method, but straightforwardly determine the lower bound $p(t)^-$ as $\lfloor p \rfloor$ (Line 1-3).

– **Case 2:** $\lfloor p \rfloor < p_{clear} \leq p^+_{[1]}$. We traverse $p^+_{[j]}$ $(1 \leq j \leq N(t))$ to find out the first $p^+_{[j]} > \lfloor p \rfloor$ under which the resource capacity constraint (C2.1) is satisfied, which is $p^+_{[1]}$ in this case. Here, the binary search range for p_{clear} is specified as $[\lfloor p \rfloor, p^+_{[1]}]$ (Line 8 and 10).

– **Case 3:** $p^+_{[i-1]} \leq \lfloor p \rfloor < p_{clear} \leq p^+_{[i]}$, where $2 \leq i \leq N(t)$. In this case, the first $p^+_{[j]} > \lfloor p \rfloor$ $(1 \leq j \leq N(t))$ which satisfies the resource capacity constraint (C2.1) is $p^+_{[i]}$ $(i > 1)$. And because $\lfloor p \rfloor \geq p^+_{[i-1]}$, the corresponding binary search range for p_{clear} is determined as $[\lfloor p \rfloor, p^+_{[i]}]$ (Line 8 and 10). Further, since $p_{clear} > \lfloor p \rfloor$, then the lower bound $p(t)^- = p_{clear}$ according to (10).

– **Case 4:** $\lfloor p \rfloor < p^+_{[i-1]} < p_{clear} \leq p^+_{[i]}$, where $2 \leq i \leq N(t)$. In this case, the difference from the Case 3 lies in $\lfloor p \rfloor < p^+_{[i-1]}$. Therefore, the binary search range for p_{clear} is supposed to be $[p^+_{[i-1]}, p^+_{[i]}]$ (Line 8 and 9). Meanwhile, in similar to the Case 3, the lower bound $p(t)^- = p_{clear}$.

In the LBP algorithm, $p^+_{[j]}$ $(1 \leq j \leq N(t))$ is traversed to specify the binary search range for p_{clear} (Line 5-11). Under each $p^+_{[j]}$ where $1 \leq j \leq N(t)$, it takes the computational complexity of $\mathcal{O}(1)$ to obtain $N(t)$ users' resource demands in parallel with the RARD algorithm. After $[p^r, p^l]$ is specified as the binary search range for p_{clear}, the lower bound $p(t)^-$ (i.e., p_{clear}) is acquired in an iterative manner (Line 12-17), with the $\mathcal{O}(\log(\frac{p^r - p^l}{\xi}))$ computational complexity. Here, ξ is a threshold coefficient indicating the binary-search termination condition. Therefore, the computational complexity of LBP algorithm is $\mathcal{O}(N(t) + \log(\frac{p^r - p^l}{\xi}))$.

Algorithm 3: Revenue-Max: Resource Pricing and Demand Allocation Algorithm for Revenue Maximization

Input: Feasible Resource Price Spectrum $[p(t)^-, p(t)^+]$.
Output: Spot Unit Resource Price $p(t)$; Resource Allocation Scheme $A(t)$.

1 Initialize $\hat{\pi}(t) \leftarrow 0$, $p(t) \leftarrow$ null, $A(t) \leftarrow$ null;
2 Calculate $D \leftarrow \lfloor \frac{\log(p(t)^+/p(t)^-)}{\log(1+\alpha)} \rfloor + 1$;
3 **for** *each* $d = \{1, 2, ..., D\}$ **do**
4 $\hat{p}_d \leftarrow p(t)^- \cdot (1+\alpha)^{d-1}$;
5 Obtain the resource allocation scheme $A(t)$ under $p(t) = \hat{p}_d$ using the RARD algorithm;
6 **if** $\frac{\hat{p}_d \cdot \sum_{i=1}^{N(t)} a_i(t)}{(1+\gamma)} \geq \widetilde{c} \cdot p^e(t) \cdot \left\lceil \frac{\sum_{u_i \in \mathcal{U}(t)} a_i(t)}{r} \right\rceil$ **and** $\hat{\pi} p_d \cdot \sum_{i=1}^{N(t)} a_i(t) > \pi(t)$ **then**
7 $\hat{\pi}(\hat{t}) \leftarrow \hat{p}_d \cdot \sum_{u_i \in \mathcal{U}(t)} a_i(t)$;
8 $\hat{p}(t) \leftarrow \hat{p}_d$, $\hat{A}(t) \leftarrow \langle a_i(t) \rangle_{i=1}^{N(t)}$;
9 **return** $p(t) \leftarrow \hat{p}(t)$ and $A(t) \leftarrow \hat{A}(t)$;

C. Step II: Finalize the Resource Price and Allocation Scheme

Given the feasible resource price spectrum $[p(t)^-, p(t)^+]$, we find out the revenue-optimal spot unit resource price $p(t)$ as well as the associated resource allocation scheme $A(t)$. As mentioned in Section IV-C, our revenue maximization Problem P2 suffers from the computational intractability which results from the segmentation and non-convexity of user utility function $v_i(\cdot)$ (i.e., Eq.(7)).

Inspired by the specific structure of our optimization problem, we propose a simple but efficient Resource Pricing and Demand Allocation Algorithm for Revenue Maximization named Revenue-Max which could gain a near-optimal revenue. To be specific, we discretize the continuous resource price spectrum of $[p(t)^-, p(t)^+]$ into D discrete candidate prices, where D is formulated in (11). Note that, $\alpha > 0$ is a constant coefficient that controls the trade-off between computational complexity and approximation ratio (detailed in Proposition 1). Let $\hat{p}_d$ represent the d^{th} candidate price, which is defined in (12).

$$D = \left\lfloor \frac{\log(p(t)^+/p(t)^-)}{\log(1+\alpha)} \right\rfloor + 1 \tag{11}$$

$$\hat{p}_d = p(t)^- \cdot (1+\alpha)^{d-1} \tag{12}$$

TABLE II
APPROXIMATION RATIO V.S. COMPUTATIONAL COMPLEXITY $\left(p(t)^+/p(t)^- = 5\right)$

Approximation Ratio $(1+\alpha)$	1.02	1.03	1.04	1.05	1.10
Number of Discrete Prices D	82	55	42	33	17

The pseudo-code of the `Revenue-Max` algorithm is shown in Algorithm 3. The `Revenue-Max` algorithm calculates the service revenue $\pi(t)$ for each candidate price $\hat{p}_d$. Then, we pick out one of candidate prices $\hat{p}(t) = \hat{p}_d$ amongst the D discrete prices, under which the CSP earns the maximum service revenue with the minimum profit rate of γ definitely gained. Here, the spot unit resource price $p(t)$ is finalized as $\hat{p}(t)$, and the corresponding resource allocation scheme $\hat{A}(t)$ is obtained with the `RARD` algorithm. Finally, the CSP earns the service revenue of $\hat{\pi}(t) = \hat{p}(t) \cdot \hat{A}(t)$.

Proposition 1. The `Revenue-Max` algorithm achieves an approximation ratio of $(1+\alpha)$ on the CSP's revenue maximization, requiring the parallel computational complexity of $\mathcal{O}(D)$.

Proof. For the true revenue-optimal spot unit resource price $p^*(t)$, there must exist an integer $z \in [1, D]$ such that $p(t)^- \cdot (1+\alpha)^{z-1} \leq p^*(t) \leq p(t)^- \cdot (1+\alpha)^z$. The CSP earns the optimal revenue as $\pi^*(t)$ at the price of $p^*(t)$. Here, we can obtain that

$$\begin{aligned}\pi^*(t) &= p^*(t) \times \sum_{u_i \in \mathcal{U}(t)} d_i(p^*(t)) \\ &\leq \left(p(t)^- \cdot (1+\alpha)^z\right) \times \sum_{u_i \in \mathcal{U}(t)} d_i(p^*(t)) \\ &\leq (1+\alpha) \times \left(p(t)^- \cdot (1+\alpha)^{z-1}\right) \times \sum_{u_i \in \mathcal{U}(t)} d_i\left(p(t)^- \cdot (1+\alpha^{z-1})\right) \\ &\leq (1+\alpha) \times \hat{\pi}\end{aligned} \tag{13}$$

where the inequality (13) is derived from the fact that $d_i(p(t))$ is non-increasing with the price $p(t)$. Henceforth, the approximation ratio of the `Revenue-Max` algorithm is proved to be $(1+\alpha)$. Since we need to respectively calculate the corresponding service revenue at all D candidate prices. At each candidate price $\hat{p}_d$, it takes the computational complexity of $\mathcal{O}(1)$ to obtain $N(t)$ users' resource demands in parallel with the `RARD` algorithm. Thus, the parallel computational complexity of the `Revenue-Max` algorithm is $\mathcal{O}(D)$. □

It is suggested in Proposition 1 that, the constant coefficient α balances the trade-off between computational efficiency and approximation ratio, as exemplified in Table II when $p(t)^-/p(t)^+ = 5$. In a general way, we devise the computational-efficient approximation algorithm `Revenue-Max` to obtain a near-optimal result, by switching the continuous price decision range into a discrete decision domain.

Theorem 1 (**Incentive Compatibility**). *A user u_i can always place its true bidding budget b_i with no incentive to misreport, regardless of what bidding strategies are adopted by other users.*

Before proving Theorem 1, we preliminarily present the definition of *IC-Regret* identified in [45] to quantify the incentive compatibility of our resource auction mechanism.

Definition 1 (**IC-Regret**). *For a user $u_i \in \mathcal{U}$, let b_i be her truthful bidding budget, whereas $b_i' \in \mathbb{R}^+$ is the bidding budget that the user u_i actually places to the CSP. It is possible to be $b_i' \neq b_i$, where the user u_i misreports her own bidding budget. Given the above notations, the IC-Regret $rgt_i(b_i)$ is defined for the user u_i as follows.*

$$rgt_i(b_i) = \max_{b_i' \in \mathcal{B}} \left(v_i(b_i') - v_i(b_i)\right) \tag{14}$$

where $\mathcal{B}$ is the feasible domain of bidding budget b_i, e.g., $\mathbb{R}^+$.

According to Definition 1, an incentive-compatible mechanism is supposed to be with $rgt_i(b_i) = 0$ for any user $u_i \in \mathcal{U}$. Instead, a higher IC-Regret $rgt_i(b_i) = 0$ for the user u_i implies a stronger incentive to misreport her own bidding budget.

Proof. In order to prove the property of incentive compatibility, it is required to clarify that a user can only obtain the maximum utility through truthfully placing her own bidding budget. With no loss of generality, we choose an arbitrary user $u_i \in \mathcal{U}$ for the classified discussion, primarily in the following two cases.

- *Case* 1: *The service bid of user u_i should be rejected with her true bidding budget b_i, i.e., $a_i(t) = 0$.*

(a) *The user u_i proposes an untruthful bidding budget such that $b_i' < b_i$*: The user u_i with the bidding budget b_i is rejected, because her unaffordability to purchase a minimum of a_i^- units of cloud resources. According to the ***budget-feasible*** guarantee, the user u_i still cannot afford herself to purchase at least a_i^- units of cloud resources for the minimum QoS requirement, hence resulting in no increase of utility gain $v_i(\cdot)$. Therefore, the IC-regret

$$rgt_i(b_i) = \max_{b_i' \in (0, b_i)} \left(v_i(b_i') - v_i(b_i)\right) = 0$$

holds true in this case.

(b) *The user u_i proposes an untruthful bidding budget such that $b_i' > b_i$*: The user u_i attempts to have the cloud resource allocated by overstating her bidding budget as $b_i' > b_i$. Even though $a_i(t) > a_i^-$ units of cloud resources are allocated through misreporting the bidding budget, the user u_i still cannot complete a transaction with $a_i(t) > a_i^-$ units of cloud resources paid. Instead, she is forced to abandon the cloud migration with $a_i(t) = 0$. After all, an overstated bidding budget $b_i' > b_i$ could not overturn the truth that the user u_i is incapable of affording a minimum of a_i^- units of cloud resources. In this way, it is true that

$$rgt_i(b_i) = \max_{b_i' \in (b_i, +\infty)} \left(v_i(b_i') - v_i(b_i)\right) = 0.$$

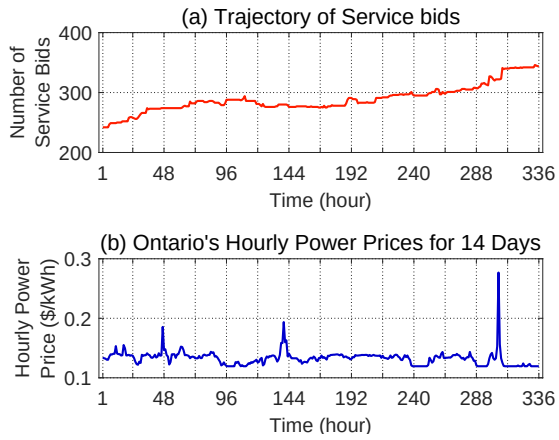

Fig. 4. Overview of the Real-World Trace Data.

- *Case* 2: *The service bid of user* u_i *could be accepted with her true bidding budget* b_i.
  - As stated in the Problem P1, the user u_i should purchase $a_i(t)$ units of cloud resources where her utility $v_i(\cdot)$ over cloud resource allocation is maximized. By Eq. (7), the optimal resource allocation $a_i(t)$ that maximizes the user utility is closely associated with her proposed bidding budget. No matter understating or overstating her bidding budget, the optimal resource allocation $a_i(t)$ in IaaS clouds would accordingly vary. In other words, the "optimal" resource allocation $a_i'(t)$ derived from an understated/overstated bidding budget b_i' cannot be the real-optimal resource allocation $a_i(t)$ relating to the true bidding budget b_i. Thus, it is concluded that

$$rgt_i(b_i) = \max_{b_i' \in \mathbb{R}^+ - \{b_i\}} (v_i(b_i') - v_i(b_i)) < 0,$$

  which implies that the user u_i has no incentive to misreport her own bidding budget.

Based on the above-indicated cases, we can finalize a statement such that for any user $u_i \in \mathcal{U}$,

$$rgt_i(b_i) = \max_{b_i' \in \mathbb{R}^+} (v_i(b_i') - v_i(b_i)) = 0,$$

which indicates the ***incentive compatibility*** of our resource auction mechanism. □

To summarize, our RARM mechanism provides the guarantee of *budget feasibility*, *incentive compatibility*, and *envy-freeness*. It solves the resource pricing and demand allocation problem with (1+α)-approximate revenue maximization achieved. The desired approximation ratio (1+α) can be discretionarily configured by the pre-defined coefficient α.

VI. Performance Evaluation

A. Experimental Setup

We employ simulations in this section to study the efficacy of our RARM mechanism. We consider a public IaaS cloud environment where multiple users are open to place service bids for cloud resources and execute their applications at the cloud. Each user proposing the service bid specifies her service type s_i, which is randomly drawn from four representative data analytics applications including Classification, Naive Bayes, Regression and KMeans. The regression coefficients $\theta_{s_i,0}$, $\theta_{s_i,1}$, $\theta_{s_i,2}$ and $\theta_{s_i,3}$ in Eq. (3) for these four service types are explicitly given in [38].

TABLE III
Parameter Settings

Param.	Value
τ	1 Hour [46]
M	100
$\widetilde{c}$	600 Watts [47]
γ	0.5
α	0.02
r	10 [47]
b_i	$(\frac{a_i^- + a_i^+}{2}) \cdot \mathcal{N}(0.02, 0.015^2)$ USD

In the trace-driven experiments, we excerpt the 14-day trajectory of VM subscription requests from the public Microsoft Azure Cluster during November 16, 2016 to February 16, 2017 [48]. We conceive each VM subscription request as a service bid β_i, where a_i^- is set based on the number of vCPU cores requested by the VM subscription request, and a_i^+ is scaled as $1.75 \times a_i^-$. To simplify the experiment, we randomly select a fraction of 100 Azure subscribers from the entire dataset. Fig. 4(a) demonstrates the number of service bids placed by these 100 Azure subscribers over 14 days, where there are 242 to 346 service bids concurrently placed per hour. Meanwhile, we collect the Ontarios hourly power price from the Independent Electricity System Operator [49]. The Ontario's hourly power prices from March 16 to March 29, 2020 are shown in Fig. 4(b).

Besides, the parameter values in Table III are adopted in our simulations, where the source of some key values is indicated. We set the duration τ of each time slot as one hour, according to the minimum billing cycle of Microsoft Azure [46]. In terms of the bidding budget, we adopt the assumption in [50] to generate the bidding budget b_i (in unit of \$) for each user based on a normal distribution $((a_i^- + a_i^+)/2) \cdot \mathcal{N}(0.02, 0.015^2)$ (unless specified). All the experiments are conducted on a Windows 10 computer where the processor is Intel Core i7-5500U (2 CPUs, 2.4GHz) with the RAM size of 12 GB.

B. Performance Benchmarks

Our RARM mechanism is compared against four typical approaches, which are the heuristic-based *Simulated Annealing* approach, two state-of-the-art approaches (i.e., *Uniform Price Auction*, and *PIRA*), and a *randomized* baseline approach:

- *Simulated Annealing* [51]: This centralized algorithm is a competitive optimization approach widely applied to the nonlinear optimization problem. Given this, it can greatly approximate the optimal price setting for cloud resources, under which the CSP gains the maximum revenue from cloud users.
- *Uniform Price Auction* [52]: This approach is also known as *clearing price auction*, which is affiliated to the single-price auction. The CSP prioritizes the resource demand of cloud users with higher bid density (i.e., a_i^+/b_i). Their requested

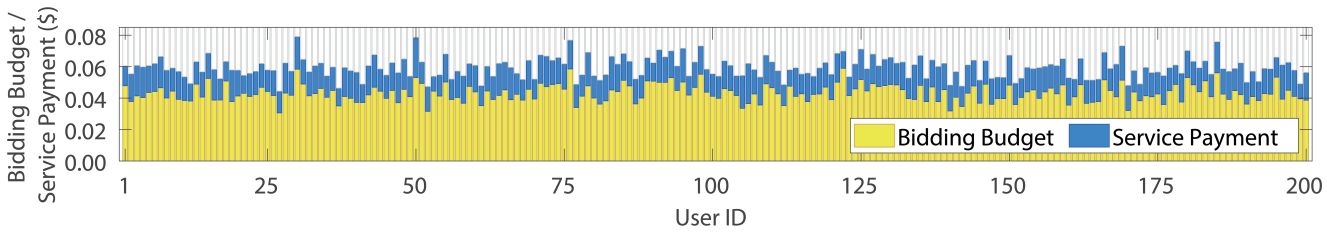

Fig. 5. Bidding Budget v.s. Service Payment across Different Users.

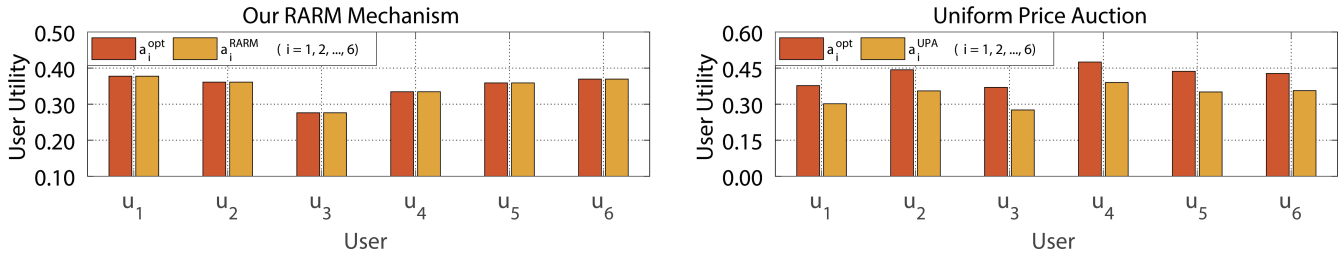

Fig. 6. Comparative Study between our RARM Mechanism and the Uniform Price Auction [52].

cloud resources (i.e., a_i^+) are orderly allocated by the CSP, until the capacity of cloud resources is exhausted or all service bids have been accepted. The unit cloud resource price is single and determined by the lowest winning bid.

- *PIRA* [9]: This approach solves the resource pricing and demand allocation problem in cloud environments, with the aim of incentivizing the maximum users served at cloud on the premise of the CSP's minimum profit rate guaranteed. In comparison with maximizing the CSP's service revenue, it adopts a divergent optimization objective from this article.
- *Random*: This approach firstly configures a randomized resource price $p(t)$ within the feasible resource price spectrum $[p(t)^-, p(t)^+]$. Based on the randomized price setting $p(t)$, each cloud user is hereby allocated the amount of cloud resources according to their price-incentive resource demands.

Note that, the *Uniform Price Auction* approach is utilized to conduct a comparative study with our RARM mechanism, with the property of *envy-freeness* testified. The *Simulated Annealing* algorithm is adopted to compare against our RARM mechanism, further evaluating the optimality and efficiency of our proposed approach. In terms of the *PIRA* and *Random* approaches, they are regarded as performance benchmarks to study the impact of the bidding budget and implement the trace-driven experiments.

In order to reflect the randomness of auction, each experiment is repeated over several times. For our RARM mechanism and the performance benchmarks (other than *Random*), each experiment result is averaged over 10 runs. Regarding the *Random* approach, the experimental result is taken an average over 100 runs.

C. Numerical Experiments

1) Validation on Budget Feasibility: We testify the *budget feasibility* for the resource allocation scheme acquired by our RARM mechanism. To validate the *budget feasible* resource allocation scheme, each user's service payment determined by the RARM mechanism is compared with her own bidding budget, as shown in Fig. 5. Here, we set 200 users who concurrently place the service bid at a time slot, and the presented experimental results are averaged over 100 runs. It can be seen that, all of these 200 user's bidding budget is respectively sufficient to cover the service payment of their own, further verifying the budget feasible resource allocation.

2) Validation on Envy-Freeness: We examine the *envy-freeness* for our resource allocation scheme. To better demonstrate the related experimental results, we streamline our simulating scenario with six users (i.e., $u_1 \sim u_6$) concurrently proposing service bids. We compare our RARM mechanism with the *Uniform Price Auction* approach [52]. In these two approaches, we both define the notation a_i^{opt} for each user u_i $(i = 1, 2, ..., 6)$, which implies the corresponding optimum of cloud resource allocation achieving her own maximum utility v_i.

Fig. 6 demonstrates the comparative results between our RARM mechanism and the *Uniform Price Auction*. Our RARM mechanism always allocates the a_i^{RARM} units of cloud resources to each user u_i, which equals to her own a_i^{opt}. That is to say, each user u_i obtains the amount of cloud resources making her own utility v_i maximized, i.e., $v_i(a_i^{\text{RARM}}) = v_i(a_i^{opt})$. In the *Uniform Price Auction*, nevertheless, the amount of cloud resources a_i^{UPA} allocated for each user u_i is less than her own a_i^{opt}, i.e., $v_i(a_i^{\text{UPA}}) < v_i(a_i^{opt})$. According to the definition of *envy-freeness*, each user can always prefer her own allocated amount of cloud resources to other possible allocation results. In this case, the *Uniform Price Auction* approach does not provide the guarantee of *envy-freeness*. Whereas, the resource allocation scheme obtained by our RARM mechanism is *envy-free* because $v_i(a_i^{\text{RARM}}) = v_i(a_i^{opt})$ for each bidding user u_i.

3) Optimality and Efficiency of our RARM Mechanism: We validate the optimality (i.e., the CSP's service revenue) and

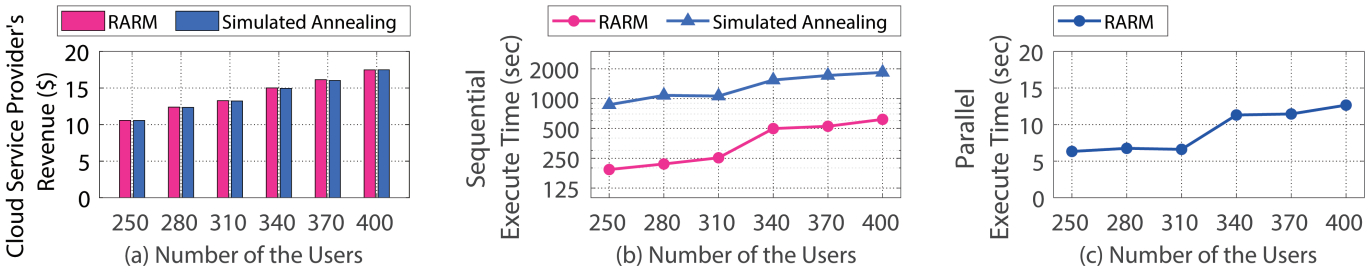

Fig. 7. Comparison of CSP's Service Revenue, and Algorithmic Execute Time.

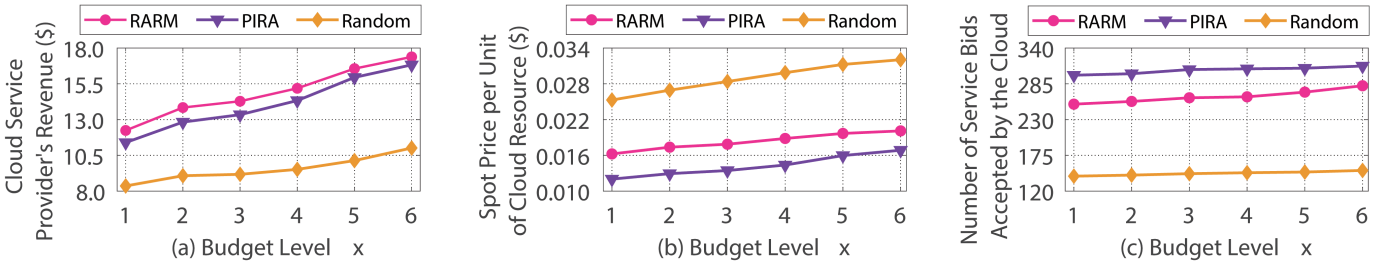

Fig. 8. CSP's Service Revenue, Spot Unit Resource Price, and Number of the Accepted Bids under Different Levels x of Bidding Budget.

the efficiency (i.e., the algorithmic execute time) gained by our RARM mechanism. As shown in Fig. 7, our proposed approach is compared against the *Simulated Annealing* algorithm. Both sequential and parallel execute time are respectively measured. Unlike the centralized *Simulated Annealing* algorithm, our RARM mechanism supports parallel execution where $N(t)$ users concurrently calculate their resource demands. The sequential execute time implies the computational cost, whereas the parallel execute time indicates our RARM mechanism's actual execute time in the cloud marketplace.

It can be seen from Fig. 7 that, the service revenue gained by CSP increases with the number of users who concurrently place service bids. Although the difference in the gained service revenue between RARM and *Simulated Annealing* is minimal, there is a significant quantitative difference on the sequential execute time between these two approaches. In other words, our RARM mechanism acquires a great reduction in the computional cost when earning the almost-the-same amount of service revenue as the *Simulated Annealing* approach. More delightfully, our RARM mechanism is accomplished within a few seconds when conducting parallel execution. It means our proposed approach can shortly complete the decision on resource pricing and demand allocation at the beginning of each time slot. This is what cannot be achieved by the centralized *Simulated Annealing* algorithm. Despite a satisfied optimality acquired, the *Simulated Annealing* algorithm is impractical in the cloud marketplace where the time-variant decision is strictly needed. In a nutshell, our RARM mechanism not only approximates the optimum of service revenue as the competitive *Simulated Annealing* approach, but also has the uniqueness of efficient parallel execution.

4) Impact of Bidding Budget b_i: We evaluate the impact of bidding budget b_i towards the auction results, i.e., the CSP's service revenue, the price setting of cloud resources, the number of accepted service bids. In specific, we set the bidding budget into six levels from $x = 1$ to $x = 6$. At the bidding budget level x, each cloud user u_i places her bidding budget b_i (in unit of \$) based on the normal distribution $((a_i^- + a_i^+)/2) \cdot \mathcal{N}(0.015 + 0.01 \times x, 0.015^2)$. Here, we set 400 users who concurrently place the service bid at a time slot.

The experimental result is given in Fig. 8. When the bidding budget level x grows up, the CSP can earn much more money from bidding users. Thus, the CSP's service revenue increases with the bidding budget level x, as shown in Fig. 8(a). Our RARM mechanism always obtains the highest service revenue amongst these three approaches, which coincides with its objective of maximizing the CSP's service revenue. As indicated by Fig. 8(b), a higher bidding budget level also implies a higher price setting for cloud resources. When the bidding budget is raised, a higher cloud resource price can effectively tighten the outspread resource demand; that is, avoid the short supply of cloud resources. Since the *PIRA* approach targets to stimulate the maximum cloud users, the cloud resource price of *PIRA* approach is lower than our RARM approach. A lower price can incentivize more users served at the cloud, as demonstrated in Fig. 8(c). Our RARM mechanism sacrifices part of cloud users to achieve the maximum revenue from cloud users. In terms of the *Random* approach, it is like a blind price selection that gets the worst performance.

D. Real-World Trace-Driven Experiments

In the trace-driven experiments, we perform our RARM mechanism across 336 time slots (i.e., 14 days) on the basis of the real-world trace in Fig. 4. The empirical results based on the real-world trace are shown in Fig. 9, which compare against two performance benchmarks, i.e., the *PIRA* and the *Random* approaches.

Similar to Fig. 8(a), it can be seen from Fig. 9(a) that our RARM mechanism gains the highest service revenue amongst

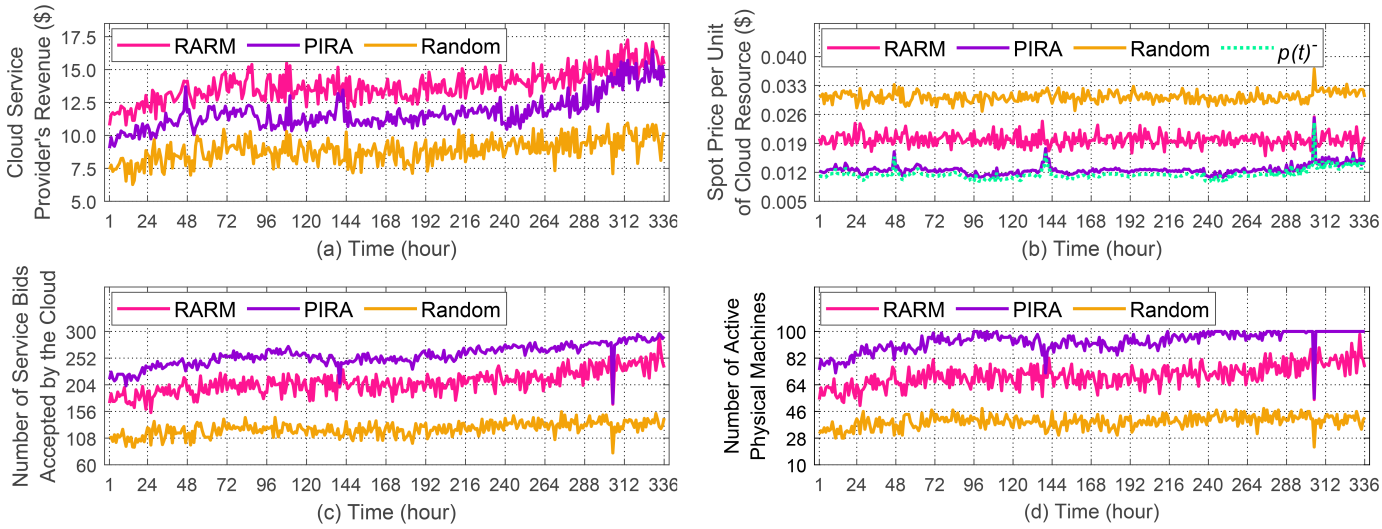

Fig. 9. Empirical Results Based on the Real-World Trace Data.

these three competitive approaches. Since the objective of the *PIRA* approach is to incentivize the maximum cloud users rather than revenue maximization, an acceptable revenue loss is permitted here. Given its aimless behavior, the *Random* approach gains the least service revenue amongst the three methods. In common, the service revenue earned at each time slot jointly oscillates with the fluctuant power price and the trajectory of service bids. On the one hand, the service revenue gained by the CSP needs to sufficiently recoup dynamic energy costs, together with the minimum profit rate of γ acquired. On the other hand, the service revenue presents an overall upward trend when more service bids are places along the time trajectory.

Fig. 9(b) illustrates the cloud pricing results of these three approaches, in accordance with the experimental results revealed in Fig. 8(b) as well. When the cloud resource is overpriced, many users with the finite bidding budget would herby abandon cloud migration with $a_i(t) = 0$, let alone the objective of revenue maximization. Given this, the *PIRA* approach takes as a low price of cloud resources as possible (i.e., slightly higher than $p(t)^-$), under which the maximum cloud users are stimulated as in Fig. 9(c). For the sake of revenue maximization, our RARM mechanism has a higher cloud resource price than *PIRA*. Because of this, our proposed approach incentivizes the smaller amount of cloud users than *PIRA*. As for the *Random* approach, it sets the highest price of cloud resources, resulting from its randomized nature. It can be observed in Fig. 9(c) that, the rising tendency on the number of accepted bids along with the 336 time slots accords with the trajectory of service bids in Fig. 4(a). Meanwhile, the number of active physical machines increases with more service bids accepted by the CSP, as shown in Fig. 9(d).

VII. Conclusion

In this article, we adopt the marker-oriented approach to study the resource pricing and demand allocation problem in multiuser IaaS clouds. In specific, we model the auction market in the IaaS cloud, where each user arbitrarily places her service bid across time slots. Given the spot setting on cloud resource price, each user is allocated cloud resources according to her own resource demand. To gain the maximum revenue from the finite cloud resources, the cloud service provider determines a revenue-optimal resource price setting. To attack the computational challenges of our revenue maximization problem, we put forward a resource auction mechanism namely RARM to make decisions on resource pricing and demand allocation in an effective but efficient manner. Our resource auction mechanism also gains the properties of *budget feasibility*, *incentive compatibility*, and *envy-freeness*. Finally, the efficacy of our RARM mechanism is validated by extensive simulating results based on real-world data. This work is expected to provide an efficient solution for market-oriented resource pricing and demand allocation in the IaaS cloud environment.

Furthermore, we are fully aware of the research limitations with several avenues for our future work. Firstly, we will design a more refined cloud resource pricing scheme which better supports the allocation of multi-dimensional cloud resources (e.g., CPU units, memory space, disk storage). Different types of jobs have various bottleneck resources. The bottleneck resource for computation-intensive jobs is always the CPU unit, whereas the bottleneck for data-intensive jobs shifts into the memory space [35]. Secondly, our RARM mechanism proposes a general framework for cloud resource pricing and allocation from a high-level viewpoint. In order to make it more applicable in the realistic cloud environment, the jobs submitted to the CSP should be further considered as the graph jobs [53] that demonstrate sophisticated topologies. In this case, a fine-grained resource distribution amongst nodes of graph job needs to be addressed [54]. Finally, we intend to evaluate our proposed solution in real-life cloud systems. The experiments in practice can further provide us with a more comprehensive understanding of resource auction design within the realistic cloud environment.

REFERENCES

[1] D. S. Linthicum, "The evolution of cloud service governance," *IEEE Cloud Computing*, vol. 2, no. 6, pp. 86–89, Nov 2015.

[2] B. Zheng, L. Pan, D. Yuan, S. Liu, Y. Shi, and L. Wang, "A truthful mechanism for optimally purchasing IaaS instances and scheduling parallel jobs in service clouds," in *International Conference on Service-Oriented Computing (ICSOC)*, 2018.

[3] C. Wu, R. Buyya, and K. Ramamohanarao, "Cloud pricing models: Taxonomy, survey, and interdisciplinary challenges," *ACM Computing Surveys*, vol. 52, no. 6, 2019.

[4] "Microsoft Azure Cloud Computing Services," https://azure.microsoft.com/en-us.

[5] "Google Cloud Computing Services," https://cloud.google.com.

[6] "Amazon Elastic Compute Cloud (Amazon EC2)," https://aws.amazon.com/ec2/.

[7] M. Benioff and C. Adler, *Behind the Cloud: The Untold Story of How Salesforce.com Went from Idea to Billion-Dollar Company-and Revolutionized an Industry*. Jossey-Bass, San Francisco, CA, pp. 103-105, 2009.

[8] J. Weinman, "The economics of pay-per-use pricing," *IEEE Cloud Computing*, vol. 5, no. 5, pp. 99–107, 2018.

[9] S. Li, J. Huang, and B. Cheng, "A price-incentive resource auction mechanism balancing the interests between users and cloud service provider," *IEEE Transactions on Network and Service Management*, 2020, DOI: 10.1109/TNSM.2020.3036989.

[10] "IONOS Enterprise Cloud," https://www.ionos.com/enterprise-cloud.

[11] "CloudSigma," https://www.cloudsigma.com.

[12] A. Jin, W. Song, and W. Zhuang, "Auction-based resource allocation for sharing cloudlets in mobile cloud computing," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 45–57, 2018.

[13] S. Hosseinalipour and H. Dai, "A two-stage auction mechanism for cloud resource allocation," *IEEE Transactions on Cloud Computing*, 2019, DOI: 10.1109/TCC.2019.2901785.

[14] L. Lu, J. Yu, Y. Zhu, and M. Li, "A double auction mechanism to bridge users' task requirements and providers' resources in two-sided cloud markets," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 4, pp. 720–733, 2018.

[15] D. Zhao, X. Li, and H. Ma, "Budget-feasible online incentive mechanisms for crowdsourcing tasks truthfully," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 647–661, 2016.

[16] S. Dobzinski, N. Nisan, and M. Schapira, "Truthful randomized mechanisms for combinatorial auctions," in *ACM Symposium on Theory of Computing (STOC)*, 2006, p. 644C652.

[17] V. Guruswami, J. D. Hartline, A. R. Karlin, D. Kempe, C. Kenyon, and F. McSherry, "On profit-maximizing envy-free pricing," in *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2005, pp. 1164–1173.

[18] M. Khodak, L. Zheng, A. S. Lan, C. Joe-Wong, and M. Chiang, "Learning cloud dynamics to optimize spot instance bidding strategies," in *IEEE Conference on Computer Communications (INFOCOM)*, 2018, pp. 2762–2770.

[19] L. Zheng, C. Joe-Wong, C. W. Tan, M. Chiang, and X. Wang, "How to bid the cloud," in *ACM Conference on Special Interest Group on Data Communication (SIGCOMM)*, 2015, pp. 71–84.

[20] S. Hou, W. Ni, S. Chen, S. Zhao, B. Cheng, and J. Chen, "Real-time optimization of dynamic speed scaling for distributed data centers," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 2090–2103, 2020.

[21] T. Pham, S. Ristov, and T. Fahringer, "Performance and behavior characterization of Amazon EC2 spot instances," in *IEEE International Conference on Cloud Computing (CLOUD)*, 2018, pp. 73–81.

[22] H. Xu and B. Li, "Dynamic cloud pricing for revenue maximization," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 158–171, 2013.

[23] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations & Trends in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2011.

[24] X. Zhang, C. Wu, Z. Huang, and Z. Li, "Occupation-oblivious pricing of cloud jobs via online learning," in *IEEE Conference on Computer Communications (INFOCOM)*, 2018, pp. 2456–2464.

[25] A. S. Prasad, M. Arumaithurai, D. Koll, Y. Jiang, and X. Fu, "OFM: An online fisher market for cloud computing," in *IEEE Conference on Computer Communications (INFOCOM)*, 2019, pp. 2575–2583.

[26] X. Wu, P. Loiseau, and E. Hyytia, "Toward designing cost-optimal policies to utilize IaaS clouds with online learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 501–514, 2020.

[27] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 1175–1191.

[28] V. Cardellini, V. D. Valerio, and F. L. Presti, "Game-theoretic resource pricing and provisioning strategies in cloud systems," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 86–98, 2020.

[29] A. Ghosh and S. Sarkar, "Pricing for profit in Internet of Things," *IEEE Transactions on Network Science and Engineering*, vol. 6, no. 2, pp. 130–144, 2019.

[30] M. Siew, D. W. H. Cai, L. Li, and T. Q. S. Quek, "Dynamic pricing for resource-quota sharing in multi-access edge computing," *IEEE Transactions on Network Science and Engineering*, 2020, DOI: 10.1109/TNSE.2020.3003051.

[31] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. C. M. Lau, "Online auctions in IaaS clouds: Welfare and profit maximization with server costs," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1034–1047, 2017.

[32] C. Chen, W. Wang, and B. Li, "Speculative slot reservation: Enforcing service isolation for dependent data-parallel computations," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 549–559.

[33] H. Liang, T. Xing, L. X. Cai, D. Huang, D. Peng, and Y. Liu, "Adaptive computing resource allocation for mobile cloud computing," *International Journal of Distributed Sensor Networks*, vol. 9, no. 4, p. 181426, 2013.

[34] C. Qiu, H. Shen, and L. Chen, "Towards green cloud computing: Demand allocation and pricing policies for cloud service brokerage," *IEEE Transactions on Big Data*, vol. 5, no. 2, pp. 238–251, 2019.

[35] C. Wang, B. Urgaonkar, G. Kesidis, A. Gupta, L. Y. Chen, and R. Birke, "Effective capacity modulation as an explicit control knob for public cloud profitability," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 13, no. 1, pp. 2:1–2:25, 2018.

[36] M. Aldossary, K. Djemame, I. Alzamil, A. Kostopoulos, A. Dimakis, and E. Agiatzidou, "Energy-aware cost prediction and pricing of virtual machines in cloud computing environments," *Future Generation Computer Systems*, vol. 93, pp. 442 – 459, April 2019.

[37] C. Chen, W. Wang, and B. Li, "Performance-aware fair scheduling: Exploiting demand elasticity of data analytics jobs," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2018, pp. 504–512.

[38] S. Venkataraman, Z. Yang, M. Franklin, B. Recht, and I. Stoica, "Ernest: Efficient performance prediction for large-scale advanced analytics," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2016, pp. 363–378.

[39] J. Bruck, Ching-Tien Ho, S. Kipnis, E. Upfal, and D. Weathersby, "Efficient algorithms for all-to-all communications in multiport message-passing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 11, pp. 1143–1156, 1997.

[40] B. Moon, I. Fernando Vega Lopez, and V. Immanuel, "Efficient algorithms for large-scale temporal aggregation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 3, pp. 744–759, 2003.

[41] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[42] J. C. Gilbert, *Numerical Optimization: Theoretical and Practical Aspects (Universitext)*, 2006.

[43] Z. Zheng, R. Srikant, and G. Chen, "Pricing for revenue maximization in inter-datacenter networks," in *IEEE Conference on Computer Communications (INFOCOM)*, 2018, pp. 28–36.

[44] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freema & Co., 1979.

[45] Z. Feng, O. Schrijvers, and E. Sodomka, "Online learning for measuring incentive compatibility in ad auctions?" in *The World Wide Web Conference (WWW)*, 2019, pp. 2729 – 2735.

[46] "Pricing calculator - Configure and estimate the costs for Azure products," https://azure.microsoft.com/en-us/pricing/calculator/.

[47] C. Wang, N. Nasiriani, G. Kesidis, B. Urgaonkar, Q. Wang, L. Y. Chen, A. Gupta, and R. Birke, "Recouping energy costs from cloud tenants: Tenant demand response aware pricing design," in *ACM International Conference on Future Energy Systems (e-Energy)*, 2015, pp. 141–150.

[48] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and

R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *ACM Symposium on Operating Systems Principles (SOSP)*, 2017, pp. 153–167.

[49] "Electricity pricing in Ontario," http://www.ieso.ca/power-data.

[50] B. Zheng, L. Pan, S. Liu, and L. Wang, "An online mechanism for purchasing IaaS instances and scheduling pleasingly parallel jobs in cloud computing environments," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 35–45.

[51] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," 1983, vol. 220, no. 4598, pp. 671–680.

[52] V. Krishna, *Auction Theory (Second Edition)*. Academic Press, Elsevier Inc., 2010.

[53] S. Hosseinalipour, A. Nayak, and H. Dai, "Power-aware allocation of graph jobs in geo-distributed cloud networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 4, pp. 749–765, 2020.

[54] M. LiWang, S. Hosseinalipour, Z. Gao, Y. Tang, L. Huang, and H. Dai, "Allocation of computation-intensive graph jobs over vehicular clouds in IoV," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 311–324, 2020.

Songyuan Li received the B.Eng. degree in computer science and technology from the Beijing University of Posts and Telecommunications in 2018. He is currently pursuing the masters degree with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunication, China. He was a recipient of China National Scholarship in 2019. He has published articles in international journals and conference proceedings, including the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, the *Peer-to-Peer Networking and Applications*, the *International Journal of Web and Grid Services*, IEEE ICWS, IEEE SCC, and IEEE ISPA. His current research interests include cloud computing, edge computing, services computing, performance evaluation, and optimization.

Jiwei Huang (Member, IEEE) received the B.Eng. and Ph.D. degrees in computer science and technology from Tsinghua University, in 2009 and 2014, respectively. He was a Visiting Scholar with the Georgia Institute of Technology. He is currently a Professor and the Dean with the Department of Computer Science and Technology, China University of Petroleum, Beijing, China, and the Director with the Beijing Key Laboratory of Petroleum Data Mining. He has published one book and more than 50 articles in international journals and conference proceedings, including the IEEE TRANSACTIONS ON SERVICES COMPUTING, the IEEE TRANSACTIONS ON CLOUD COMPUTING, ACM SIGMETRICS, IEEE ICWS, and IEEE SCC. His research interests include services computing, cloud computing, and performance evaluation. He is a Member of the ACM.

Bo Cheng (Member, IEEE) received the Ph.D. degree in computer science and engineering from University of Electronic Science and Technology of China in 2006. He is currently a Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China. He has published more than 60 articles in international journals and conference proceedings, including the IEEE/ACM TRANSACTIONS ON NETWORKING, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE ICDCS and IEEE ICWS. His current research interests include network services and intelligence, Internet of Things technology, communication software and distributed computing. He serves on the editorial board of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT. He is a Member of the ACM.