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

Journal:	*Transactions on Network Science and Engineering*
Manuscript ID	TNSE-2020-10-0847
Manuscript Types:	Regular Paper
Keywords:	IaaS cloud, market-oriented pricing strategy, resource auction, revenue maximization

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 the user 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. The cloud service provider (CSP), meanwhile, 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 paper 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 allocation scheme targeting to revenue maximization also satisfies essential properties including budget feasibility and envy-freeness. To attack the computational challenges resulted from 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 validate the efficacy of our proposed approach.

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

1 Introduction

Regarding the user-friendly advantages of offering high-performance but cost-saving service, cloud computing gains more and more momentum, and attracts more users to host their applications on cloud [1]. With the increasing scale of cloud users, resource management has become a hot topic in the community of IaaS cloud. 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 on cost, further reducing the CSP's profit earning. Given this, it necessitates an efficient resource management methodology under the limited amount of resources rather than pure resource expansion.

Through an efficient resource management strategy, the conflicting requirements from both sides of users and the CSP should be satisfied [2]. In specific, cloud users prefers 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 user 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.

Market-oriented resource pricing strategy can be an effective methodology which 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, 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 in this paper.

In this paper, 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 the time, the CSP will accordingly modulate the cloud resource price to regulate the resource demands of users. The user is allocated the amount of cloud resources based on her resource demand, while 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. Furthermore, our proposed resource auction mechanism also meets several essential properties, including ***budget feasibility*** [4] and ***envy-freeness*** [5]. Budget feasibility alleges that a user's bidding budget should sufficiently cover her service payment, which is a basic requirement for auction mechanism design. As for envy-

Manuscript received October 15, 2020. 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).

- *S. Li and B. 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).*
- *J. Huang is with the Department of Computer Science and Technology, Beijing Key Laboratory of Petroleum Data Mining, China University of Petroleum - Beijing, Beijing 102249, China (corresponding author, e-mail: huangjw@cup.edu.cn).*

freeness, it indicates a fairness criterion in the economic domain. Under a cloud price setting, each user can be allocated the amount of resources which maximize her utilities. These two auction properties strengthens the sustainability of our resource auction mechanism.

Our main contributions are listed as follows.

• We model a auction market for IaaS clouds which 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.

• We develop a price-incentive resource auction mechanism namely `RARM` for multiuser IaaS clouds, where a computational-efficient resource pricing and allocation algorithm called `Revenue-Max` is proposed. The `Revenue-Max` algorithm can gain a near-optimal revenue for the CSP, with a $(1+\alpha)$ approximation ratio preserved.

• Extensive simulating experiments 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 paper are organized as follows. Section 2 reviews the background and related work. In Section 3, we introduce and formulate our system model. Then, in Section 4, we formulate the user's utility and rational cloud purchase strategy, and then define the revenue maximization problem for the CSP. In Section 5, we devise a resource auction mechanism called `RARM`, where the resource pricing and allocation algorithm termed `Revenue-Max` solves the revenue maximization problem in an effective but efficient manner. Section 6 evaluates the effectiveness and efficiency of our proposed approach. In Section 7, we conclude our research work.

2 Background and Related Work

2.1 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 [6], to the popular *pay-as-you-go* model [7], then to the recent *market-oriented* pricing strategy [8].

The nascent cloud pricing method can be approximately dated back to the subscription-based SaaS pricing model provided by Salesforce.com [6]. The subscription-based pricing approach is technically consolidated by software multi-tenancy. Each cloud tenant jointly shares the public IT operational cost, but still has private space to host her own cloud application. Thus, the subscription-based method presents a comparatively cheap price for IT 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 then emerges as a pay-as-you-go pricing model [7]. 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 [9], Google Cloud Platform [10] and Microsoft Azure [11] 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 [12] and the dynamic IT operational costs [13]. 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 [8]. It can sufficiently reflect the economic behavior of 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 [14]. The Amazon EC2 Spot Instance targets to make full use of the idle EC2 instances. In comparison of 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 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 [15] under the market-oriented cloud pricing strategy.

2.2 Market-oriented Cloud Pricing Strategy

As stated in Section 2.1, 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 cloud marketplace, a black-box data-driven method termed online learning [16] 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. [17] exploited 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 was made. Prasad et al. [18] proposed an online-learning-based Fisher market which enabled online resource pricing and allocation, where both marketing randomness and resource integrality gap were fully considered. On the side of the cloud user with limited budgets, Wu et al. [19] took 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 online learning approach originates from a 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, thus the exploitation of cross-section marketing data matters for data security and privacy [20]. 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 the domain knowledge, thereby gaining better explainability than the online learning method. In regard to the game-theoretic approach, Cardellini et al. [21] formulated 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. [22] investigated 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 was acquired through the game process. Siew et al. [23] introduced the game theory to study the sharing economy in mobile cloud environment, based on which dynamic pricing mechanisms were designed for welfare/profit maximization. As for the auction-based approach, Jin et al. [24] then put forward an incentive-compatible double auction mechanism for mobile cloud computing, where mobile devices' and cloudlets' contribution was dynamically priced. Hosseinalipour et al. [25] demonstrated the market-oriented interactions between different entities as a two-phase auction model, where the two stages of auctions were respectively formulated as a option-based sequential auction and a auction/flat-mixed market. Lu at al. [26] concerned about a two-sided cloud market environment with multiple CSPs, on the basis of which a double auction mechanism was presented to match the users' and CSPs' requirements.

In this paper, we intend to make contributions towards the market-oriented cloud pricing strategy, from an auction-based perspective. Different from existent literatures [24], [25], [26] adopting a second-price auction, we carry out a single-price auction where cloud resources are marketed with the same unit price. It fits more with our intuitive knowledge that identical items should be valued at the same price. Furthermore, we conduct an elaborate design on auction mechanism to enable some essential properties (i.e., ***budget feasibility*** and ***envy-freeness***) satisfied. These two auction properties play an important part in the sustainability of our resource auction mechanism.

3 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 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 make 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 $a_i(t)$ units of cloud resources equaling to her resource demand. 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 bids $\beta_i(t) = (s_i, b_i, t_i^-, t_i^+)$, 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 in 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 user u_i's QoS requirement is elastic as $[t_i^-, t_i^+]$, indicating the desired range of response time.

Cloud Service Provider. We restrict our research spotlight on a homogeneous set of physical machines administered by the CSP, as in [27] [28]. There are M physical machines administered by the CSP, each of which is equipped with r units of computational resources. To dwindle unnecessary energy costs and raise the profit earning, the CSP simply turns on the physical machines running in load, while the idle physical machines are switched off into the sleep mode. At each time slot t, the CSP collects the service bids from different users, and respectively assigns $a_i(t)$ units of cloud resources to each user $u_i \in \mathcal{U}(t)$. Hence, as in [29], 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 mainly derives from the the energy costs of IaaS infrastructure [13]. Suppose the average energy cost to perform a physical machine per time slot is $\widetilde{c}$, then 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 [30]. Thus, the CSP should pay 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 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.

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

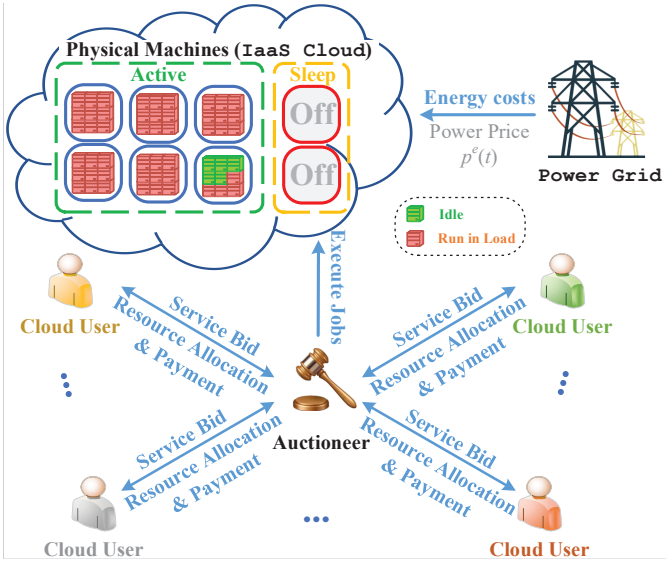

Fig. 1. Auction Market in the Multiuser IaaS Cloud.

4 Problem Statement

4.1 User Utility Model

We firstly formulate the utility model for the user, which represents the cloud user's preference towards different cloud resource allocation results. In this paper, 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 various 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 [31] and the regression analysis results in [32], 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.

$$t_i(a_i(t)) = \theta_{s_i,0} + \theta_{s_i,1} \cdot a_i(t) + \frac{\theta_{s_i,2}}{a_i(t)} + \theta_{3,s_i} \cdot \log(a_i(t)) \quad (3)$$

As mentioned in Section 3, 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^+.

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

From the above, the "happiness" gained 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 worth noting that, the QoS progress rate $t_i(a_i(t))$ is absolutely not proportional to the amount of allocated resources $a_i(t)$, but with strong concavity [31].

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

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

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 $p(t) \cdot a_i(t)$ should not excess 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).

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

4.2 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 unit cloud resource price 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,

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

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 an 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 figured 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 find out 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 [33], suggesting that the service payment $p(t) \cdot a_i(t)$ should

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)};
```

not excess the bidding budget b_i.

$$\max_{a_i(t)} \quad \rho_i(a_i(t)) - \frac{p(t)\cdot a_i(t)}{b_i} \tag{P1}$$
$$\text{s.t.} \quad p(t)\cdot a_i(t) \le b_i \tag{C1.1}$$
$$a_i^- \le a_i(t) \le a_i^+ \tag{C1.2}$$

It is noticeable that, the problem P1 is a convex optimization problem (The proof is omitted). Then, we can solve the problem P1 by seeking out the solution of the corresponding Karush-Kuhn-Tucker (KKT) equations [34]. The computational complexity of solving the KKT equations exponentially increases with the number of inequation constraints in the original convex optimization problem [35]. There are $K = 3$ inequation constraints (i.e., the upper bound of C1.1, and the upper and lower bound of C1.2) in the problem P1, thus we can solve the Problem P1 for the user u_i by means of KKT equations in the computation 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 computation 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 be also solved in parallel, in this case of which the computation complexity is reduced to $\mathcal{O}(1)$.

4.3 Optimization Problem

The objective of our optimization problem is to maximum the service revenue from the CSP's standpoint. Therefore, we formulate the resource pricing and allocation problem as the Problem P1 below.

$$\max_{a_i(t),p(t)} \quad \pi(t) = p(t)\cdot \sum_{u_i\in\mathcal{U}(t)} a_i(t) \tag{P2}$$
$$\text{s.t.} \quad p(t)\cdot \sum_{u_i\in\mathcal{U}(t)} a_i(t) \le M\cdot r \tag{C2.1}$$
$$p(t)\cdot \sum_{u_i\in\mathcal{U}(t)} a_i(t) \ge (1+\gamma)\cdot \widetilde{c}\cdot p^e(t)\cdot m(t) \tag{C2.2}$$
$$a_i(t) = d_i(p(t)) \quad \forall u_i \in \mathcal{U}(t) \tag{C2.3}$$

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 [36], meaning that each user $u_i \in \mathcal{U}(t)$ is allocated cloud resources in accordance with 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 which is NP-hard to solve out [37]. Each physical machine in the IaaS cloud is conceived as a bin with finite computational resources, and 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 the 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.

5 Mechanism Design

5.1 RARM: Resource Auction Mechanism Framework

To attack the aforementioned computation challenges, we design 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 price spectrum of $p(t)$, denoted as $[p(t)^-, p(t)^+]$. Under the spot unit resource price $p(t) \ge p(t)^-$, 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 $a_i(t) > 0$.

5.2 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

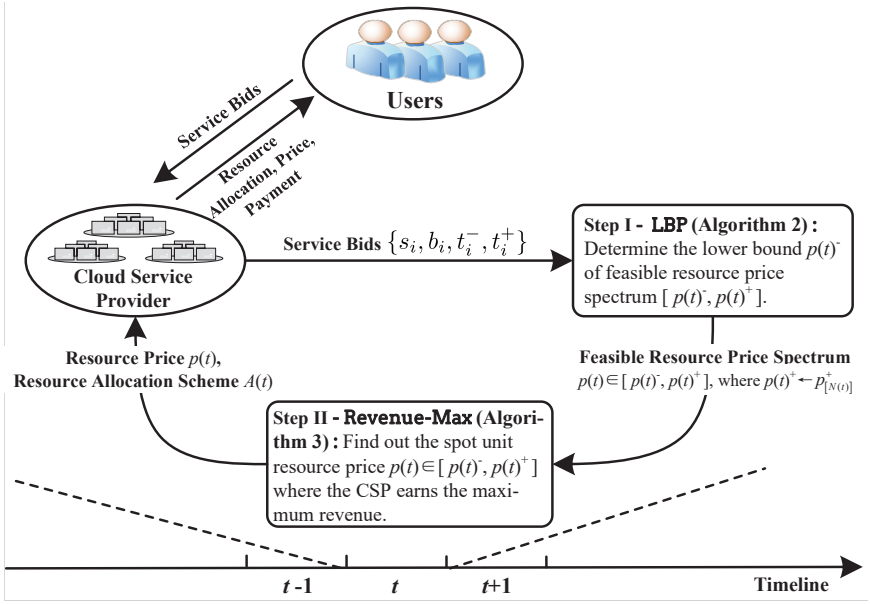

Fig. 2. Overview of Resource Auction Mechanism.

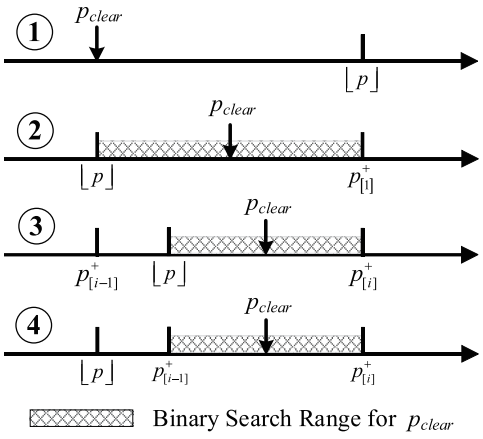

Fig. 3. Binary search range for p_{clear} in four cases.

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 account.

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 machines'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

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 $p(t)$.

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

up, the lower bound $p(t)^-$ is 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 needn't find for 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 computation 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 $\mathcal{O}(\log(\frac{p^r - p^l}{\xi}))$ computation complexity. Here, ξ is a threshold coefficient indicating the binary-search termination condition. Therefore, the computation complexity of LBP algorithm is $\mathcal{O}(N(t) + \log(\frac{p^r - p^l}{\xi}))$.

5.3 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 4.3, 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 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 which controls the trade-off between computational complexity and approximation ratio (detailed in Theorem 1). Let $\hat{p}_d$ represent the d^{th} candidate price, which is defined in (12).

Algorithm 3: Revenue-Max: Resource Pricing and 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 π̂(t) ← 0, p(t) ← null, A(t) ← null;
2  Calculate D ← ⌊log(p(t)^+/p(t)^-) / log(1+α)⌋ + 1;
3  for each d = {1, 2, ..., D} do
4      p̂_d ← p(t)^- · (1+ α)^{d-1};
5      Obtain the resource allocation scheme A(t) under
       p(t) = p̂_d using the RARD algorithm;
6      if p̂_d·Σ_{i=1}^{N(t)} a_i(t) / (1+γ) ≥ c̃ · p^e(t) · ⌈Σ_{u_i∈U(t)} a_i(t) / r⌉ and
       π̂ p_d · Σ_{i=1}^{N(t)} a_i(t) > π(t) then
7          π̂(t) ← p̂_d · Σ_{u_i∈U(t)} a_i(t);
8          p̂(t) ← p̂_d, Â(t) ← ⟨a_i(t)⟩_{i=1}^{N(t)};
9  return p(t) ← p̂(t) and A(t) ← Â(t);
```

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

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

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 obtain with the RARD algorithm. Finally, the CSP earns the service revenue of $\hat{\pi}(t) = \hat{p}(t) \cdot \hat{A}(t)$.

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

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

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

where the inequality (13) is derived from the fact that $d_i(p(t))$ is non-increasing with the price $p(t)$. Henceforth,

TABLE 1
Approximation Ratio v.s. Computational Complexity $(p(t)^+/p(t)^- = 5)$

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

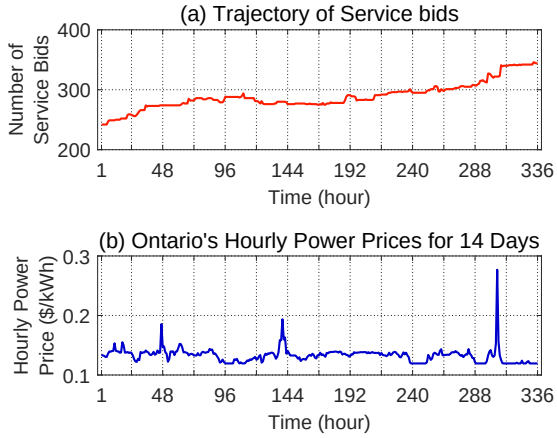

Fig. 4. Overview of Real Trace Data.

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 computation complexity of $\mathcal{O}(1)$ to obtain $N(t)$ users' resource demands in parallel with the `RARD` algorithm. Thus, the parallel computation complexity of the `Revenue-Max` algorithm is $\mathcal{O}(D)$. □

Theorem 1 suggests that, the constant coefficient α balances the trade-off between computational efficiency and approximation ratio, as exemplified in Table 1 when $p(t)^-/p(t)^+ = 5$. To summarize, 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.

6 Performance Evaluation

6.1 Experimental Setup

We employ simulating experiments 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 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, Native.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 [32].

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 [38]. 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

TABLE 2
Parameter Settings

Param.	Value
τ	1 Hour [40]
M	100
$\widetilde{c}$	600 Watts [42]
γ	0.2
α	0.02
r	10 [42]
b_i	$(\frac{a_i^- + a_i^+}{2}) \cdot \mathcal{N}(0.02, 0.015^2)$ USD

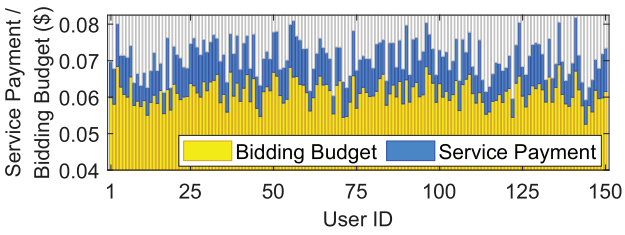

Fig. 5. Bidding Budget v.s. Service Payment across Different Users.

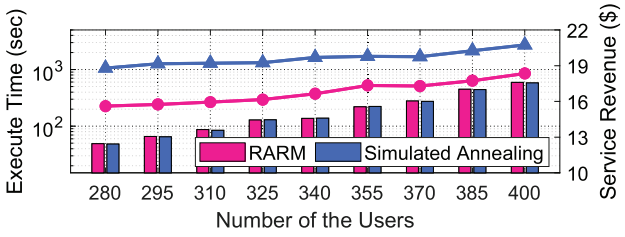

Fig. 6. Comparison of Algorithmic Execute Time, and CSP's Service Revenue.

randomly select a fraction of 100 Azure subscribers from the entire dataset. Fig. 4(a) demontrates 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 [39]. The Ontario's hourly power prices from Mar. 16 to Mar. 29, 2020 are shown in Fig. 4(b).

Besides, the parameter values in Table 2 are adopted in our simulating experiments, where the derivation of some key values are indicated. Note that, the duration τ of each time slot is set as one hour based on the minimum billing cycle of Microsoft Azure [40]. In terms of the bidding budget, we follow the assumption in [41] to generate the bidding budget b_i (in unit of $) for each user based on a normal distribution $((a_i^- + a_i^+)/2) \cdot \mathcal{N}(0.02, 0.015^2)$.

6.2 Experimental Results

Numerical Experiments. We firstly testify the properties of resource allocation scheme acquired by our `RARM` mechanism. To validate the ***budget feasible*** resource alloca-

TABLE 3
Envy-Free Cloud Resource Allocation Results

Net Utility	Allocated Resources of Users $U_1 \sim U_6$					
	$\hat{a}_1$	$\hat{a}_2$	$\hat{a}_3$	$\hat{a}_4$	$\hat{a}_5$	$\hat{a}_6$
User U_1	**0.4126**	-0.1582	-0.1679	-0.1816	-0.1715	-0.2252
User U_2	-0.1280	**0.4118**	-0.2784	-0.2126	-0.2448	-0.1630
User U_3	-0.0184	-0.1748	**0.4188**	-0.1175	-0.1118	-0.1757
User U_4	-0.1783	-0.1733	-0.3690	**0.3649**	-0.2133	-0.3178
User U_5	-0.1002	-0.1496	-0.3284	-0.1831	**0.3492**	-0.3566
User U_6	-0.1125	-0.0406	-0.2676	-0.0501	-0.1718	**0.3957**

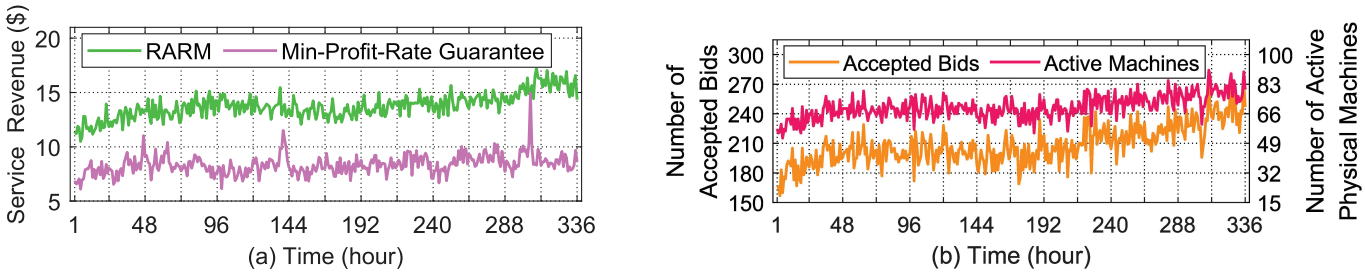

Fig. 7. Empirical Results Based on the Real Trace Data.

tion 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 150 users who concurrently place the service bid at a time slot, and the presented experimental results are averaged over 200 runs. It can be seen that, all of these 150 user's bidding budget is respectively sufficient to cover the service payment of their own, further verifying the budget feasible resource allocation.

Besides, we examine the ***envy-freeness*** of resource allocation scheme. To better demonstrate the related experimental results, we streamline our simulating scenario with 6 users $u_1 \sim u_6$ concurrently proposing service bids. In Table 3, each user u_i's utility $v_i(a_i)$ on her own resource allocation a_i is compared with the utility $v_i(a_j)$ on other users' resource allocation a_j where $j \neq i$. Similarly, the comparative results in Table 3 is also obtained through 200 runs. No matter to the user $u_1 \sim u_6$, the user utility $v_i(a_i)$ on her own allocated resources is always greatest, which indicates the envy-free resource allocation scheme.

In Fig. 6, we also evaluate the execute time and service revenue gained by our RARM mechanism, in comparison with a state-of-art algorithm called *Simulated Annealing* [43]. The *Simulated Annealing* algorithm is a competitive optimization approach widely applied to the nonlinear optimization problem. Here, each experimental result is averaged over 10 runs. The execute time implying the computational cost is measured, where each user's resource demand is sequentially calculated at a certain resource price. With the number of users who concurrently places the service bid, the service revenue gained by CSP increases. The difference on the gained service revenue between RARM and *Simulated Annealing* is minimal, but there is a significant quantitative difference on the execute time between these two approaches. Our RARM mechanism has the minimum execute time of 226.06 seconds at the scale of 280 users, and the maximum of 852.75 seconds at the scale of 400 users.

Real Trace-Driven Experiments. We perform our RARM mechanism across 336 time slots (i.e., 14 days), based on the real-world trace in Fig. 4. The empirical results based on the real trace is shown in Fig. 7. In Fig. 7(a), the revenue baseline in correspondence with the minimum γ profit rate guarantee oscillates with the fluctuant power price, but the service revenue gained by our RARM mechanism is always above the min-profit-rate baseline. It coincides with our expectation that at least γ profit rate should be ensured for the CSP. It can be also seen in Fig. 7(a) 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.

7 Conclusion

In this paper, 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 unit resource price setting, 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 computation challenges of revenue maximization problem, we put forward a resource auction mechanism namely RARM to make decisions on resource pricing and allocation in an effective but efficient manner. Our resource auction mechanism also gains the properties of ***budget feasibility*** 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 IaaS 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] 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.

[5] 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.

[6] 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.

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

[8] 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.

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

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

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

[12] 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.

[13] 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*, 2020, DOI: 10.1109/TNSE.2020.2974250.

[14] 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.

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

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

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

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

[36] 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.

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

[38] 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.

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

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

[41] 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.

[42] 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.

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

Songyuan Li is currently a master candidate in the State Key Laboratory of Networking and Switching Technology at Beijing University of Posts and Telecommunication. He received the B.Eng. degree in computer science and technology from Beijing University of Posts and Telecommunications in 2018. He was the recipient of China National Scholarship in 2019. He has published articles in international journals and conference proceedings, including 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 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 of the Department of Computer Science and Technology, China University of Petroleum, Beijing, China, and the Director of 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 IEEE and the ACM.

Bo Cheng received the Ph.D. degree in computer science and engineering from University of Electronic Science and Technology of China in 2006. He is now a Professor in the State Key Laboratory of Networking and Switching Technology at Beijing University of Posts and Telecommunications. His current research interests include network services and intelligence, Internet of Things technology, communication software and distributed computing, etc. He is a member of the IEEE and the ACM, and serves on the editorial board of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT.