

Revenue Maximization with Dynamic Auctions in IaaS Cloud Markets

Wei Wang, Ben Liang, Baochun Li

Department of Electrical and Computer Engineering
University of Toronto

Abstract—Pricing cloud services plays a pivotal role towards the success of cloud computing. Existing pricing schemes, however, either provide no service guarantees (e.g., Spot Instances in Amazon EC2), or use static on-demand pricing in which the price cannot respond quickly to market dynamics (e.g., On-demand Instances in Amazon EC2). To overcome these problems, in this paper we design *dynamic auctions* where computing instances are periodically auctioned off to accommodate user demands over time. We address the two main challenges of revenue maximization and auction truthfulness. Our design encompasses a capacity allocation scheme, which determines the amount of instances to be auctioned off in each period, as well as the underlying auction mechanisms, based on dynamic payment schemes corresponding to the proposed capacity allocations over time. We show that our design is *two-dimensionally truthful*, and it is *asymptotically optimal* when demand is sufficiently high. Furthermore, by identifying certain optimization structures, we substantially reduce the computational complexity of our solution. Extensive simulations show that our design closely tracks market changes, while generating higher revenues than on-demand pricing.

I. INTRODUCTION

Infrastructure-as-a-Service (IaaS) clouds revolutionize the large part of IT industry by commoditising computing instances so that they can be traded and delivered as another dimension of utility. To ensure business success, a cloud vendor will need to determine how its instances can be optimally priced to cloud users, so that its revenue is maximized. State-of-the-practice pricing includes three types of models, i.e., the *reservation*, *on-demand*, and *bid-based* pricing.

Reservation pricing charges users a one-time payment to reserve an instance for months or years [1], [2]. While reservation pricing provides long-term risk-free income to cloud vendors, its upfront payment makes it less attractive to users with flexible workloads.

On-demand pricing [1], [2], [3], on the other hand, sets a fixed hourly rate and charges users for the incurred instance-hours. Despite its popularity among cloud users, the static hourly rate fails to capture market dynamics, which may result in a revenue loss for a cloud vendor.

To compensate for such deficiency, Amazon has recently introduced Spot Instances, a bid-based pricing model [1]. With Spot Instances, users periodically submit bids to Amazon, who in turn posts a spot price for each period. Those users who bid higher than the spot price get their requests fulfilled in that period. This bid-based pricing offers quick response to market dynamics, yet there is no service guarantee in

Amazon’s design: whenever the spot price goes above the bid, the allocated instances are terminated.

Due to these drawbacks in existing pricing alternatives, we are motivated to design a new pricing strategy that combines the advantages of on-demand and bid-based pricing: it should quickly respond to market dynamics, while still offering service guarantees to cloud users. In this paper, we present our design of *dynamic auctions* in cloud markets, where a sequence of auctions are periodically carried out to accommodate demands. Users bid for instances in these auctions, and every winning user is charged a *constant* usage price produced by the auction mechanism for each instance. Requested instances are then held by winning users at the same price until the users terminate them.

Dynamic auctions are essentially a price discovery process, and are more agile responding to market dynamics. Unlike EC2 Spot Instances, our design offers *guaranteed* services: A winning user enjoys a constant price over time, and its instances will never be terminated against its will. In contrast, spot users bear the risk of price fluctuations and having all running instances terminated whenever the spot price rises above their bids.

Towards an optimal design of dynamic auctions in cloud pricing, there arise two new challenges that have never been addressed. *First*, to maximize the revenue, the underlying auctions need to be jointly designed over different time periods. With guaranteed services, an instance, once auctioned off, is no longer available for future requests until it is terminated by its user. Joint auction designs allow the vendor to optimally reserve some available capacity for future high-bid requests while rejecting current low bidders. *Second*, it is important to ensure that the joint design is *truthful* in *two-dimensional* domains — each user truthfully reports her bid price and the number of requested instances. Being truthful eliminates the user’s incentive for strategic behaviours that may harm the auction integrity, and offers accurate demand information that could be utilized to predict market dynamics in the future.

The proposed auction design addresses the two challenges above. Its original contributions are two-fold. *First*, we present a near-optimal capacity allocation scheme to determine how many instances are auctioned off in each period under optimal auctions. *Second*, we jointly design the underlying auction mechanisms and specify dynamic truthful payment schemes based on the proposed capacity allocations over time. To accomplish these, we first characterize the revenue of truthful auctions and model the capacity allocation problem

as a Markov decision process (MDP). We show its NP-Completeness and present an approximate solution where direct computation requires $O(C^3)$ operations, with C being the vendor’s capacity. We identify special optimization structures in the problem and reduce the computational complexity to $O(C^2)$. We then design auction mechanisms that are specific to the time-dependent capacity allocation scheme. We guarantee two-dimensional truthfulness over time, so that each user truthfully reports her bid price and the number of requested instances even when they possess the same predictive information that the cloud vendor does.

Theoretical analyses show that the proposed dynamic auctions asymptotically approach the optimal revenue as the market demand becomes sufficiently high, which is naturally the case in cloud environments. For low-demand cases, we conduct extensive simulation studies and show that our design still offers a close approximation, with revenue loss less than 2%. Simulation results also indicate that our design outperforms the regular on-demand pricing, generating a 30% revenue gain.

The remainder of this paper is organized as follows. In Sec. II, we briefly survey the related work. Our model and notations are introduced in Sec. III. In Sec. IV, we present our detailed design of dynamic auctions and prove its truthfulness. Our design is evaluated in Sec. V, where both theoretical analysis and extensive simulation studies are given. Sec. VI concludes the paper.

II. RELATED WORK

Many recent works in the literature advocate the use of auction-based mechanisms to allocate virtual resources, primarily in grid computing environments, e.g., [4], [5], [6]. The main focus of these studies is from the user’s perspective, i.e., how to achieve the allocation fairness and efficiency via appropriate mechanisms. For auctions designed from the vendor’s perspective, Zaman and Grosu [7] propose two combinatorial auctions to allocate VMs to users. Zhang *et al.* [8] investigate a VM allocation problem for spot markets by solving a static optimization problem, yet without considering any strategic behaviours among users. Danak and Mannor [9] present two auction-based resource allocation protocols with supply adjustments. Wang *et al.* [10] also propose two combinatorial auctions to price cloud resources, and show that their designs are truthful and collusion-free. All these works focus on one-time resource allocations and restrict their discussions in a single auction period. In contrast, we consider dynamic auctions with emphasis on joint mechanism design over different time periods.

In terms of optimal mechanism designs, classical results are obtained in a single-round auction with unit-demand bidders [11], [12], where the well-known *Revenue Equivalence Theorem* is given. Other single-round designs, e.g., [13], [14], though allowing bidding for multiple units, assume one-dimensional strategies — bidders can only misreport their prices, while truthfully submit quantities of requested units — and are not applicable in cloud environments where the

strategies are essentially two-dimensional (i.e., the quantities can also be misreported).

For multi-round auction designs, Parkes and Singh [15] offer a general framework based on MDP. However, this work is substantially different from ours, as its objective is to maximize the sum utility of bidders (social welfare) instead of the service vendor’s revenue. Furthermore, the MDP state in this general model contains the entire history of the system, leading to a complex formulation with no known solution. Finally, its payment policies are based on computationally intensive VCG designs. In terms of revenue maximization, Vulcano *et al.* [16] optimally design a sequence of periodic auctions to sell a certain amount of products to unit-demand buyers distributed over time. We note that both optimal and truthful results obtained in [16] require the unit-demand assumption, and these results no longer extend to multi-demand bidders, who strategize over both the bid prices and the amount of requests. Besides, discussions in [16] are based on the *sales market*, where products are sold to customers and are never available to sellers after a transaction. Other works that focus on revenue maximizations in the sales market include [17], [18]. In contrast, our discussion allows cloud resources to be *leased*, not sold, to users: after being terminated by previous users, the instances can be hosted again to accommodate new requests. For these reasons, the mechanism design problem discussed in the sales market is a special case of the problem investigated in this paper.

III. MODEL FORMULATION

Suppose a cloud vendor has allocated a fixed capacity C to host a type of virtual instances (e.g., EC2 small instances). That is, at any given time, the allocated computing resources are capable of hosting up to C instances. A sequence of auctions, indexed by $t = 1, 2, \dots$, are periodically carried out (usually hourly, as in EC2 Spot Instances) to accommodate user requests separated in time.

A. User Model

In period t , N_t exogenous users arrive, bidding for virtual instances. We consider impatient cloud users that are interruption-intolerant. That is, users submit requests whenever their demands arise, and cannot be interrupted once their instances are held¹. Each user i , $1 \leq i \leq N_t$, wishes to rent n_i^t virtual instances and has a maximum affordable price v_i^t , known as the *reservation value*, for renting one instance in each period. When the context is clear, we will drop the time index from the notation, e.g., n_i^t is written as n_i . The values of n_i and v_i are private information known only to user i , and are distributed with joint density function $f_{n,v}$ on the support $[1, \bar{n}] \times [0, \bar{v}]$. Denote by $F_{n,v}$ the corresponding CDF of $f_{n,v}$. Both $F_{n,v}$ and $f_{n,v}$ can be learned based on

¹This is a valid assumption for many cloud applications where instantaneous computations are required, e.g., website hosting, online video, and online commerce. For users that are patient and interruption-tolerant, EC2 Spot Instances constitute the best design in terms of revenue maximization, as users do not care about service guarantees in this case.

historical information, provided that the number of bidders is sufficiently large [19], [20], [21], which is naturally true for cloud markets. To join the auction, each user i submits a *two-dimensional* bid (r_i, b_i) , where r_i is the number of requested instances, and b_i is the maximum price that a user wants to pay. Unless explicitly stated, all prices referred to in this paper are calculated per instance per period. Note that a user may misreport by submitting $b_i \neq v_i$ (or $r_i \neq n_i$) if it believes that this is more beneficial.

At the beginning of each period t , the vendor carries out an auction mechanism \mathcal{M}_t and clears the market by deciding which user's request is fulfilled and under what price. No *partial fulfilment* is accepted. A user is either rejected or has all requested r_i instances being held, which is the case in prevalent pricing schemes [1]. Let p_i be the price charged to user i . Once p_i is set, it remains constant for user i until all requested r_i instances are terminated by this user². Specifically, let $l_{i,j}$ be the running time of instance j held for user i . The utility for a winning user i is defined by

$$u_i(r_i, b_i) = \begin{cases} \sum_{j=1}^{n_i} (v_i - p_i) l_{i,j} - \sum_{j=n_i+1}^{r_i} p_i l_{i,j}, & \text{if } r_i \geq n_i; \\ 0, & \text{o.w.} \end{cases} \quad (1)$$

For those rejected users, both the charged price and the utility are zero. Note that in our model, we assume that a user derives no positive utility from the overbooked resources (i.e., instance $j > n_i$ does not generate v_i , as shown in the first case of (1)). However, the overbooking strategy may result in a lower price p_i , and increases the overall utility u_i . Therefore, if a rational user chooses to overbook, all overbooked instances would be released in the next time period to avoid unnecessary costs, i.e., $l_{i,j} = 1$ for all $j = n_i + 1, \dots, r_i$.

The problem of every user i is to find an optimal submission (r_i^*, b_i^*) such that its utility is maximized.

B. The Problem of Dynamic Auction Design

For the cloud vendor, suppose at time t , the available resources are sufficient to host C_t virtual instances. In addition to knowing the exact number of requests in the current time slot t , we assume that the vendor may predict the demand in the near future: it knows the distributions of N_τ , the number of users in τ , for $\tau \leq T = t + w$, with w being some *prediction window*. (Refer to [22] for demand prediction issues.) Note that $w = 0$ if the prediction is unavailable. Denote by $V_\tau^*(C_\tau)$ the maximum expected total revenue collected from τ to T . Let $\Gamma_{\mathcal{M}_t}(Q)$ be the *overall revenue* of auctioning Q instances to accommodate the demand in time t , using mechanism \mathcal{M}_t . To be exact, $\Gamma_{\mathcal{M}_t}(Q) = \sum_{i=1}^{N_t} \sum_{j=1}^{r_i} p_i l_{i,j}$. The cloud vendor's problem is to jointly design a sequence of auction mechanisms $\{\mathcal{M}_t\}$ to maximize its expected revenue collected over the prediction window. We formulate this problem in the

following recursive form:

$$V_t^*(C_t) = \mathbf{E} \left[\max_{\mathcal{M}_t, 0 \leq Q_t \leq C_t} \{ \Gamma_{\mathcal{M}_t}(Q_t) + V_{t+1}^*(C_{t+1}) \} \right], \quad (2)$$

where Q_t is the capacity allocated to accommodate the current demand. The boundary conditions are $V_{T+1}^*(c) = 0$ for all $c = 0, 1, \dots, C$.

We also require the designed mechanism $\{\mathcal{M}_t\}$ to possess some salient economic properties, notably *truthfulness*. That is, for every user i , no matter how the others bid, submitting the true reservation value and the needed number of virtual instances always maximizes its utility, i.e., $u_i(n_i, v_i) \geq u_i(r_i, b_i)$ for all (r_i, b_i) . Being truthful eliminates the incentive for any strategic behaviours that may harm the auction integrity, and it provides accurate market information that could be utilized to predict future demands.

However, without any *a priori* information regarding the usage patterns — the distribution of $l_{i,j}$, it would be almost impossible for a vendor to solve (2). To make the analysis tractable, we assume the instance running time is i.i.d. exponential³. In discrete settings, this implies that $l_{i,j}$ follows the geometric distribution with p.m.f. $P(l_{i,j} = k) = q(1 - q)^{k-1}$, where q is the probability that a currently running instance will be terminated by its user in the next period. Though this is a simple model to allow tractable analysis, it has been shown to give interesting insights into practical systems. We also note that such an exponential usage pattern is widely adopted in the literature [23], [24], [25].

We now analyze the value of C_{t+1} , the available capacity in $t + 1$, in (2). After Q_t instances are allocated at time t , there are $C - C_t + Q_t$ instances being held. Suppose that right before $t + 1$, K of them are terminated by their users. Then at the beginning of $t + 1$, there are $C_{t+1} = C - C_t + K$ instances being available for new requests. Since each hosted instance has a probability q to be terminated in the next period, we know $K \sim \text{Bin}(C - C_t + Q_t, q)$ with p.m.f. $P(K = k) = B(C - C_t + Q_t, k, q)$, where $B(n, k, q) = \binom{n}{k} q^k (1 - q)^{n-k}$. We re-write (2) as

$$V_t^*(C_t) = \mathbf{E} \left[\max_{\mathcal{M}_t, 0 \leq Q_t \leq C_t} \{ \Gamma_{\mathcal{M}_t}(Q_t) + \mathbf{E}_K [V_{t+1}^*(C_t - Q_t + K)] \} \right]. \quad (3)$$

From the above, we see that an optimal design of dynamic auctions decides not only the underlying mechanisms $\{\mathcal{M}_t\}$, but also the number of instances auctioned off in each period (i.e., $\{Q_t\}$). Note that the two decisions are closely coupled with each other. On the one hand, the instance allocation scheme is determined based on the specified underlying mechanisms. On the other hand, to ensure that the allocation scheme is not affected by strategic user behaviours, the underlying mechanisms should also take into account how the instances are allocated over time.

²Here, the price is only effective for held instances. If more instances are required, a user has to rebid for additional ones.

³Without this assumption, the expectation operation in (2) would be taken over the entire historical and future states of the system, and little insight can be drawn from such a model.

IV. DYNAMIC AUCTION DESIGN

This section presents the design of dynamic auctions. We first characterize the revenue of truthful auctions, which allows us to focus on instance allocations first, i.e., to find the optimal Q_t . After that, we present a truthful design for the underlying mechanisms, \mathcal{M}_t , where the payment schemes depend on the specified allocations.

A. Characterizing Revenue for Truthful Mechanisms

By the *Revelation Principle* [11], truthful auctions are revenue maximizing among all auctions. Hence, it suffices to focus only on truthful mechanisms when revenue is of interest. Subsequently, our discussion on capacity allocation focuses on truthful mechanisms. We therefore do not differentiate between the submitted bid (r_i, b_i) and the true request (n_i, v_i) . We derive the auction revenue $\Gamma_{\mathcal{M}_t}$ as follows.

$$\begin{aligned} \mathbf{E}[\Gamma_{\mathcal{M}_t}] &= \mathbf{E}\left[\sum_{i=1}^{N_t} \sum_{j=1}^{n_i} p_i l_{i,j}\right] \\ &= \frac{1}{q} \mathbf{E}\left[\sum_{i=1}^{N_t} \sum_{j=1}^{n_i} p_i\right] = \frac{1}{q} \mathbf{E}[\gamma_{\mathcal{M}_t}], \end{aligned} \quad (4)$$

where the second equality holds because $\mathbf{E}[l_{i,j}] = 1/q$, and $\gamma_{\mathcal{M}_t}$ is the *instantaneous revenue* collected in time t .

It has been shown in [26] that if partial fulfilment is not allowed, then the revenue of a truthful auction is characterized by the following Lemma 1.

Lemma 1: Let $\mathbf{v} = (v_i)$ and $\mathbf{n} = (n_i)$. If partial fulfilment is not accepted, then for any truthful mechanism \mathcal{M}_t , we have

$$\mathbf{E}_{\mathbf{n}, \mathbf{v}}[\gamma_{\mathcal{M}_t}] = \mathbf{E}_{\mathbf{n}, \mathbf{v}}\left[\sum_{i=1}^{N_t} n_i \phi(v_i) x_i(\mathbf{n}, \mathbf{v})\right], \quad (5)$$

where $\phi(v_i) = v_i - \frac{1-F_v(v_i|n_i)}{f_v(v_i|n_i)}$, and $x_i(\mathbf{n}, \mathbf{v})$ takes the value 0 or 1 depending on whether user i loses or wins, respectively.

For mathematical convenience, we take a standard *regularity assumption* [12], that $\phi(\cdot)$ is increasing. This is not a restrictive assumption as it generally holds for most distributions [11].

By Lemma 1, since partial fulfilment is not supported in the cloud market [1], the expected revenue of an auction mechanism only depends on who is to win (i.e., x_i 's), not what they pay (i.e., p_i). Therefore, it is equivalent to write $\gamma_{\mathcal{M}_t} = \sum_{i=1}^{N_t} n_i \phi(v_i) x_i$, and designing an optimal \mathcal{M}_t is equivalent to finding a set of x_i 's to maximize $\gamma_{\mathcal{M}_t}$, under the constraint that the accommodated requests do not exceed the allocated capacity. Formally, an optimal design solves the following 0-1 knapsack problem:

$$\begin{aligned} \max_{x_i \in \{0,1\}} \quad & \sum_{i=1}^{N_t} n_i \phi(v_i) x_i \\ \text{s.t.} \quad & \sum_{i=1}^{N_t} n_i x_i \leq Q_t, \end{aligned} \quad (6)$$

where Q_t is the capacity allocated to period t .

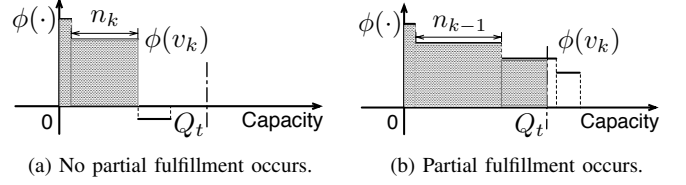


Fig. 1. The value of $\bar{\gamma}_t(Q_t)$ in two cases, shown as the shaded area. (a) The allocated capacity Q_t is sufficient to accommodate all profitable requests. (b) The allocated capacity Q_t cannot accommodate all profitable requests. This illustration shows the case where the last request is partially fulfilled.

B. Dynamic Capacity Allocation

We are now ready to derive the optimal dynamic capacity allocations. Let $\gamma_t^*(Q_t)$ be the instantaneous revenue obtained from the optimal mechanism. That is, $\gamma_t^*(Q_t) = \sum_{i=1}^{N_t} n_i \phi(v_i) x_i^*$ where x_i^* solves the knapsack problem (6). The optimal dynamic auction design problem (3) can now be equivalently written as

$$V_t^*(C_t) = \mathbf{E}\left[\max_{0 \leq Q_t \leq C_t} \left\{ \frac{1}{q} \gamma_t^*(Q_t) + \mu_{t+1}^*(C_t - Q_t) \right\}\right], \quad (7)$$

where $\mu_{t+1}^*(c) = \mathbf{E}_K[V_{t+1}^*(c + K)]$, with $K \sim \text{Bin}(C - c, q)$. The boundary conditions are $V_{T+1}^*(c) = 0$ for all $c = 0, 1, \dots, C$.

Unfortunately, optimally solving (7) is computationally prohibitive, as calculating the optimal revenue $\gamma_t^*(Q_t)$ requires to solve the problem (6), which is NP-Hard. As such, we turn to an approximate solution.

Instead of calculating γ_t^* , we consider its upper bound $\bar{\gamma}_t$ obtained by solving a linear programming relaxation of (6) where fractional x_i 's are allowed, i.e., $\bar{\gamma}_t(Q_t) = \max_{x_i \in [0,1]} \sum_{i=1}^{N_t} n_i \phi(v_i) x_i$, with $\sum_{i=1}^{N_t} n_i x_i \leq Q_t$. The value of $\bar{\gamma}_t$ can be easily calculated as follows. Let bidders be sorted in a decreasing order of their bidding prices, i.e., $v_1 \geq v_2 \geq \dots \geq v_{N_t}$. From the regularity assumption, this implies $\phi(v_1) \geq \dots \geq \phi(v_{N_t})$. To compute $\bar{\gamma}_t(Q_t)$, we sequentially accommodate bidders' requests to accumulate the revenue, from the top valued (bidder 1) to the bottom, until all capacity is allocated. Note that those negatively valued requests (i.e., $\phi(v_i) \leq 0$) will always be rejected. Figs. 1a and 1b illustrate this process, where the shaded areas represent the value of $\bar{\gamma}_t(Q_t)$. One can view the upper bound $\bar{\gamma}_t$ as the revenue obtained *as if* partial fulfilment were accepted, i.e., the last accommodated request might be partially fulfilled.

The vendor now solves an approximate problem by replacing γ_t^* with $\bar{\gamma}_t$ in (7):

$$\bar{V}_t(C_t) = \mathbf{E}\left[\max_{0 \leq Q_t \leq C_t} \left\{ \frac{1}{q} \bar{\gamma}_t(Q_t) + \bar{\mu}_{t+1}(C_t - Q_t) \right\}\right]. \quad (8)$$

where $\bar{\mu}_{t+1}(c) = \mathbf{E}_K[\bar{V}_{t+1}(c + K)]$, $K \sim \text{Bin}(C - c, q)$. The boundary conditions are $\bar{V}_{T+1}(c) = 0$ for all $c = 1, 2, \dots, C$. Let $Q_t^*(C_t)$ be the optimal solution to (8). The vendor then auctions off $Q_t^*(C_t)$ instances to accommodate the current demand.

We choose the above approximation for two reasons. *First*,

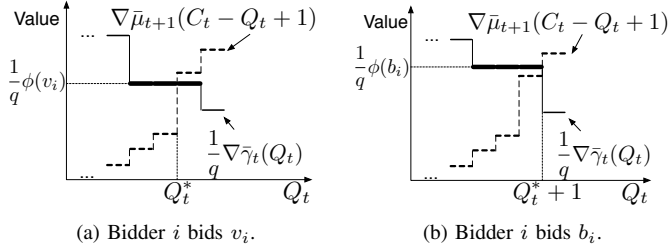


Fig. 2. Bidders can manipulate the capacity allocation via strategic bidding. The dotted curve represents the opportunity cost, while the solid one is the revenue gain. Bidder i 's bid may affect the bold part of the gain curve. (a) Q_t^* capacity is auctioned off if bidder i bids v_i . (b) $Q_t^* + 1$ capacity is auctioned off if bidder i raises its bid.

it closely approaches the optimal revenue. *Second*, it can be efficiently calculated by identifying some optimization structures in (8). We postpone the revenue analysis until later in Sec. V and focus on the structural and computational discussions in the following.

C. Structural Results and Capacity Allocation Rules

Directly computing capacity allocations in (8) is highly inefficient, especially for large vendors with enormous capacities. Since no closed-form solution exists, a standard method is to apply *numerical dynamic programming* [27], with the overall computational complexity of $O(C^3)$. However, in practice, a large vendor is usually capable of holding hundreds of thousands of instances of a certain type simultaneously. In this case, direct computation requires a huge amount of operations, which is prohibitive to complete within a short time.

Instead of direct computation, we identify some optimization structure of (8) and derive a simple *capacity allocation rule* that significantly reduces the complexity to $O(C^2)$.

First, we see that $\bar{\gamma}_t(Q_t)$ is concave w.r.t. Q_t , as stated in Lemma 2. We omit the proof here as it is straightforward by consulting Figs. 1a and 1b.

Lemma 2: Given \mathbf{n} and \mathbf{v} , $\bar{\gamma}_t(\cdot)$ is increasing and concave for all t , i.e., $\nabla \bar{\gamma}_t(c) := \bar{\gamma}_t(c) - \bar{\gamma}_t(c-1)$ is positive and decreasing w.r.t. c .

More importantly, we made the key observation that both \bar{V}_t and $\bar{\mu}_{t+1}$ in (8) are also concave. The proof is given in Appendix A.

Lemma 3: In (8), for every $\tau = t, t+1, \dots, T$, both $\bar{V}_\tau(\cdot)$ and $\bar{\mu}_{\tau+1}(\cdot)$ are increasing and concave.

Such concavities lead to a simple capacity allocation rule. We refer to the recursive revenue equation (8). Suppose instances are allocated one by one at time t , with Q_t starting from 0. Whenever an additional instance is allocated, the revenue marginally increases by $\frac{1}{q} \nabla \bar{\gamma}_t(Q_t)$ (revenue gain). At the same time, we suffer a revenue loss $\nabla \bar{\mu}_{t+1}(C_t - Q_t + 1)$, as the newly allocated instance will be temporarily unavailable to serve future demands (opportunity cost). By the concavity results (Lemma 2 and 3), as instance allocation Q_t increases, the revenue gain $\frac{1}{q} \nabla \bar{\gamma}_t(Q_t)$ decreases while the opportunity cost $\nabla \bar{\mu}_{t+1}(C_t - Q_t + 1)$ increases. To maximize the revenue, we should stop allocating more instances right before the

opportunity cost exceeds the revenue gain. Fig. 2a illustrates an example, where the solid curve represents the revenue gain while the dotted one corresponds to the opportunity cost. We see that the optimal capacity allocation Q_t^* is achieved when the two curves intersect.

The above allocation rule leads to the following optimization structure of recursive equation (8). The proof is given in Appendix A.

Proposition 1: For every realization \mathbf{n} and \mathbf{v} at time $\tau = t, t+1, \dots, T$. The optimal allocation is characterized by

$$Q_\tau^*(c) = \begin{cases} \max_{1 \leq n \leq c} \left\{ n : \frac{1}{q} \nabla \bar{\gamma}_\tau(n) > \nabla \bar{\mu}_{\tau+1}(c - n + 1) \right\}, & \text{if } \frac{1}{q} \nabla \bar{\gamma}_\tau(1) > \nabla \bar{\mu}_{\tau+1}(c); \\ 0, & \text{o.w.} \end{cases} \quad (9)$$

Furthermore, $Q_\tau^*(c+1) - 1 \leq Q_\tau^*(c) \leq Q_\tau^*(c+1)$.

The second statement of Proposition 1 plays a key role in reducing the complexity to $O(C^2)$. It shows that previously calculated results can be leveraged as follows. For each period τ , $\bar{V}_\tau(c)$ is sequentially computed as $C_\tau = C, C-1, \dots, 0$. Though the first computation ($C_\tau = C$) requires searching the entire solution space (i.e., $Q_\tau = 0, 1, \dots, C$), subsequent computations are much more efficient. By Proposition 1, once $Q_\tau^*(C_\tau + 1)$ is calculated, one can quickly determine $Q_\tau^*(C_\tau)$ by choosing between two optimal candidates, $Q_\tau^*(C_\tau) = Q_\tau^*(C_\tau + 1)$ and $Q_\tau^*(C_\tau) = Q_\tau^*(C_\tau + 1) - 1$, and the one resulting in higher revenue $\bar{V}_\tau(C_\tau)$ is the optimal solution. The entire computation only takes $O(C^2)$ operations. We note that the complexity reduction from $O(C^3)$ of applying standard dynamic programming to $O(C^2)$ of the proposed method is significant, especially for large vendors capable of holding hundreds of thousands of instances.

D. Designing the Underlying Auction Mechanisms

With the capacity allocation rule derived above, we are now ready to present the underlying auction mechanisms. We have to ensure that the designed mechanism eliminates the incentive for any strategic behaviours by the cloud users.

One might think that the underlying auction design is trivial once the capacity allocation Q_t^* has been decided: just run existing single-round truthful mechanisms [28] to auction off Q_t^* instances in a period t . However, this turns out to be a significant misunderstanding. In fact, directly applying single-period truthful auctions fails to eliminate strategic behaviours in our problem, as users can manipulate the allocation Q_t^* by submitting different bids. To see this, we refer to Fig. 2. We note that the revenue gain curve is decided by the submitted bids. In Fig. 2, the bold line of the gain curve is decided by bidder i 's submission. By the allocation rule, we see in Fig. 2a that Q_t^* instances are allocated if bidder i bids truthfully. On the other hand, in Fig. 2b, bidder i raises its bid to change the gain curve, leading to an allocation decision of $Q_t^* + 1$ (by the allocation rule). However, this can never happen in single-round auctions [28], [11], [12], [29], [30], in which the number of auctioned items is given and is fixed. For this reason, single-round designs fail to remain truthful in our case.

To overcome the above challenges, we propose Algorithm 1 as the adopted mechanism, with C_t and Q_t^* being the number of available instances and the allocated capacity, respectively. In this algorithm, we artificially insert a *virtual bidder* into the market, who requests an infinite amount of instances at a price $\phi^{-1}(0)$. Introducing a virtual bidder has no effect on the auction result, since its request is never accepted, but it simplifies the revenue expression. Recall that bidders are assumed to be sorted in a decreasing order of prices, $b_1 \geq \dots \geq b_{N_t+1}$. (There are $N_t + 1$ bidders including the virtual one.) For two bidders bidding the same price, the one requesting fewer instances ranks higher than the other.

Algorithm 1 The Truthful Mechanism \mathcal{M}_t with Q_t^* Instances Allocated

1. Let k be the index such that $\sum_{j=1}^k r_j \leq Q_t^* < \sum_{j=1}^{k+1} r_j$
 2. Let $s = \sum_{j=1}^k r_j$
 3. Let $\hat{b}_s = \phi^{-1}(q \nabla \bar{\mu}_{t+1}(C_t - s + 1))$
 4. Top k bidders win, each paying $p = \max\{b_{k+1}, \hat{b}_s\}$
-

Algorithm 1 adopts a similar design as the $(k+1)$ -price auction (i.e., top k bidders win each paying the bid of bidder $k+1$), but with different payment rules. The basic idea is to charge every winner the *least bid* required to win. That is, bidder i wins if (1) it is among the top k bidders ($b_i > b_{k+1}$) and (2) all top k bids are high enough so that the allocated capacity is sufficient to accommodate all their requests. Note that if $b_k > \hat{b}_s$, then by Proposition 1, at least $s = \sum_{j=1}^k r_j$ instances are auctioned off:

$$\begin{aligned} \frac{1}{q} \nabla \bar{\gamma}_t(s) &= \frac{1}{q} \phi(b_k) \\ &> \frac{1}{q} \phi(\hat{b}_s) \\ &= \nabla \bar{\mu}_{t+1}(C_t - s + 1). \end{aligned}$$

It is worth mentioning that the $(k+1)$ -price auction is *untruthful* for multi-demand bidders [28]. However, we show that our design completely eliminates the bidder's incentive for misreporting even in two-dimensional domains (v_i and n_i). To see this, we first show the monotonicity of the produced price, whose proof is given in Appendix B.

Lemma 4 (Price monotonicity): For every bidder i with price b_i , fix all others' submissions. Denote by $p(r_i)$ the price produced by Algorithm 1 when i requests r_i instances. Then $p(r_i)$ is increasing w.r.t. r_i .

By Lemma 4, all bidders' bets on overbooking are off, as requesting more instances results in a higher clearing price, which decreases the bidder's utility (see (1)). Also note that no user has the incentive to request fewer instances than actually needed (otherwise, the utility is 0), we see that users will truthfully report their n_i value.

Corollary 1: For every bidder i , there is no advantage to overbook instances. Furthermore, there is no advantage to misreport n_i , i.e., given b_i , $u_i(r_i, b_i) \leq u_i(n_i, b_i)$ for all r_i .

Now that bidders truthfully report the quantities of requested

instances, they have only one-dimensional strategies on prices b_i . Our truthfulness proof requires the following two lemmas, both of which are proved in Appendix B.

Lemma 5 (Bid monotonicity): Algorithm 1 is *bid-monotonic*. Specifically, if bidder i wins by bidding b_i , then it also wins by bidding $b'_i > b_i$.

Lemma 6 (Critical payment): Algorithm 1 charges the *critical payment* p to winners. Specifically, for every winning bidder $i \leq k$ with price b_i , fix all others' submissions. Bidder i wins again by bidding $b'_i > p$, but loses by bidding $b'_i < p$.

It is well known that if the bidders have one-dimensional strategies, then an auction mechanism is truthful *if and only if* it is bid-monotonic and charges the critical payment to winners [30]. We therefore see the truthfulness of our design.

Proposition 2: Algorithm 1 is two-dimensionally truthful, i.e., $u_i(n_i, v_i) \geq u_i(r_i, b_i)$ for all (r_i, b_i) , $i = 1, 2, \dots, N_t$.

Corollary 2: The designed dynamic auctions $\{\mathcal{M}_t\}$ are truthful in two-dimensional domains.

V. EVALUATIONS

In this section we seek to answer the following question: How well does our design approximate the optimal solution (7)? We evaluate the revenue performance of our design in two scenarios: (1) When market demand is high, we give theoretical analysis and show that the approximation is asymptotically optimal. (2) When market demand is low, we conduct extensive simulations and show that our design still closely approximates the optimal one.

A. Asymptotic Optimality for High-Demand Markets

We analyze the revenue performance of our design in a high-demand market. Let $\gamma_t(\cdot)$ be the instantaneous revenue of Algorithm 1. By Lemma 1, in expectation, it is equivalent to write $\gamma_t(Q_t^*) = \sum_{i=1}^k n_i \phi(i)$. First we present the following lemma, whose proof is given in Appendix C.

Lemma 7: If $Q_t^* \geq \alpha \bar{n}$ for some $\alpha > 1$, Algorithm 1 is $\frac{\alpha}{\alpha-1}$ -competitive:

$$\begin{aligned} \gamma_t(Q_t^*) &\geq \left(1 - \frac{1}{\alpha}\right) \bar{\gamma}_t(Q_t^*) \\ &\geq \left(1 - \frac{1}{\alpha}\right) \gamma_t^*(Q_t^*). \end{aligned}$$

Lemma 7 essentially indicates that in terms of the instantaneous revenue, Algorithm 1 closely approximates the optimal one provided that the allocated capacity is large compared with a single user's request, i.e., $\gamma_t(Q_t^*) \rightarrow \gamma_t^*(Q_t^*)$ if $\alpha \gg 1$. Note that this is usually the case in practice, especially for large vendors.

We next evaluate the revenue performance of our design in the long run. Let $V_t(\cdot)$ be the expected revenue collected within the prediction window by running Algorithm 1, i.e.,

$$V_t(C_t) = \mathbf{E} \left[\frac{1}{q} \gamma_t(Q_t^*) + \mathbf{E}_K [V_{t+1}(C_t - Q_t^* + K)] \right], \quad (10)$$

where Q_t^* is obtained by solving (8) and $K \sim \text{Bin}(C - C_t + Q_t^*, q)$. The boundary conditions are $V_{T+1}(c) = 0$ for all

$c = 0, 1, \dots, C$. Lemma 8 conditionally bounds $V_t(C_t)$. The proof is given in Appendix C.

Lemma 8: If $N_\tau \rightarrow \infty$ for all $\tau = t, \dots, T$, then the approximate solution is $\frac{\alpha}{\alpha-1}$ -competitive for any finite $\alpha > 1$. Specifically, for all C_τ , the following inequalities hold w.p.1:

$$\begin{aligned} V_\tau(C_\tau) &\geq \left(1 - \frac{1}{\alpha}\right) \bar{V}_\tau(C_\tau) \\ &\geq \left(1 - \frac{1}{\alpha}\right) V_\tau^*(C_\tau). \end{aligned}$$

With Lemma 8, we see that our design is asymptotically optimal as the market demand is sufficiently high, which is naturally the case for large vendors.

Proposition 3: The expected revenue $V_t \rightarrow V_t^*$ w.p.1 if the user number $N_\tau \rightarrow \infty$ for all $\tau = t, \dots, T$.

B. Revenue Performance in Low-Demand Markets

Even for a market where the demand is not high, our design still offers a close approximation to the optimal solution. We verify this point via extensive simulations in this section. For convenience, denote by $U_{\mathbb{Z}}(a, b)$ and $U_{\mathbb{R}}(a, b)$ the uniform distributions defined in integer and real domains, respectively, in the range of $[a, b]$.

We adopt a typical scenario for a medium-sized vendor with capacity $C = 10^4$. That is, up to 10^4 virtual instances of a certain type can be held at one time. User demands are separated to 300 time periods. In each period t , there are N_t users bidding for instances, each requesting n_i instances with price v_i . We set $N_t \sim U_{\mathbb{Z}}(1, 300)$, $n_i \sim U_{\mathbb{Z}}(1, 100)$, and $v_i \sim U_{\mathbb{R}}(0.05, 0.1)$. Note that Lemma 3 is no longer effective in this setting, since the market demand is not high ($N_t \leq 300$). We enable short predictions and set the prediction window $w = 5$. (We have tried other values of w and obtained similar results.) It is worth mentioning that though the evaluation presented in this section is based on i.i.d. demands and bid prices, similar results are also observed in non-i.i.d. cases. Each result below has been averaged over 1000 runs.

Revenue vs. Time. We first evaluate the proposed dynamic auction by comparing its revenue against on-demand pricing with a fixed hourly rate (referred to as the fixed-pricing) in a long time span (from months to years). Since it is almost impossible to predict long-term future demands, a widely adopted method is to set a price p^* such that $p^*(1 - F_v(p^*))$ is maximized [21]. Note that F_v is the distribution of a user's reservation value, and $1 - F_v(p)$ is the probability that a user can afford the price p . In this sense, p^* maximizes the expected revenue collected from a single user. We adopt such fixed pricing as a benchmark in our evaluation.

Fig. 3a illustrates the revenue performance of both dynamic auction and fixed pricing, where q is set to 0.5. The revenue is observed to be linearly increasing over time. We see that the designed mechanism accurately captures market dynamics and outperforms fixed pricing, leading to a 30% revenue improvement. Also note that our design closely approaches the theoretical revenue upper bound, with a performance gap less than 2%. This is further verified by Fig. 3b, where a histogram

is presented to show statistics of the ratio of allocated capacity Q_t^* to the maximum individual request \bar{n} . Specifically, over 85% of such ratios exceed 20, and more than 70% exceed 40. These statistics indicate that Lemma 7 is effective in most cases: in general, the allocated capacity is much larger than any single user's request, which leads to a strong performance guarantee for the designed mechanism. Similar results are also observed when other q 's are adopted.

Revenue vs. Capacity. We next verify the structural results obtained in Sec. IV-B by investigating the relationship between the revenue and the capacity. In particular, we conduct simulations in three different market conditions by choosing $q = 0.2, 0.5$, and 0.8 , representing low, medium, and high market dynamics, respectively. For each q , we start from a low capacity setting with $C = 1000$, and increase it until $C = 10000$. For each capacity setting, we calculate the overall revenue collected in all 300 time periods. Fig. 3c illustrates the results, where the solid line represents the theoretical revenue upper bound obtained by solving (8), and the dotted line stands for the actual revenue obtained from dynamic auctions. Again, we see that the approximation design is almost optimal: in all cases, the revenue gap between the upper bound and our design is less than 2%. Also, the gap is found decreasing when more capacity is available in the datacenter. In this case, more capacity is allocated to each period, so that a better performance guarantee is achieved due to Lemma 7. These results suggest that our design is preferred by large vendors.

Another important observation is that all three upper bounds are increasing and concave functions of the capacity, verifying the structural analysis made in Lemma 3. It is interesting to see that the concavity is gradually increased when the market becomes more dynamic (*i.e.*, with larger q). Note that only q is changed in the setting, so that increasing q essentially decreases the entire workloads and revenues, as the expected running time of every instance is shortened. In this case, adding capacity brings less marginal revenue to the vendor, increasing the revenue concavity.

Capacity vs. Price Dynamics. Our final observation is that the price becomes more dynamic as capacity increases. Fig. 3d shows the c.d.f. of the determined prices in each time period in all 1000 runs during the simulation, where q is set to 0.5. We see that with the same demand level, a low capacity ($C = 1000$) intensifies the bid competition among users. In this case, only those who bid very high win and have their requests fulfilled. As a result, over 80% of the prices are higher than 0.09. The situation changes when more capacity is available. As supplies become more abundant, the intensity of bid competition decreases, resulting in more dynamic clearing prices (see $C = 5000$ and $C = 10000$).

VI. CONCLUSIONS

In this paper, we investigate into the problem of efficient pricing for revenue maximization by cloud service vendors. We advocate the use of dynamic auctions to quickly adapt to market changes while offering guaranteed services. To this end, we first analytically characterize the revenue of truthful

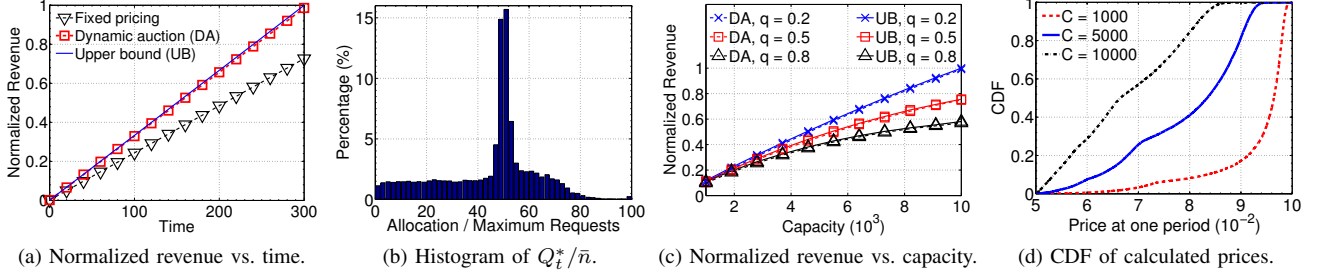


Fig. 3. Performance evaluation of dynamic auctions (DA) and the revenue upper bounds (UB) in 300 simulation periods, with $C = 10000$, $w = 5$, $N_t \sim U_{\mathbb{Z}}(1, 300)$, $r_i \sim U_{\mathbb{Z}}(1, 100)$, and $b_i \sim U_{\mathbb{R}}(0.05, 0.1)$. Unless explicitly stated, q is set to 0.5. All data are averaged over 1000 runs.

auctions and formulate their joint design over time as an MDP. Since the optimal design is NP-Hard, we turn to an approximate solution that can be directly computed within $O(C^3)$. By deriving special optimization structures of the problem, we further reduce the computational complexity to $O(C^2)$, which is significant for cloud vendors with a large capacity. A sequence of two-dimensionally truthful mechanisms are jointly designed with the proposed capacity allocation scheme. The construction of such mechanisms demonstrates the practical validity of the proposed revenue model. Theoretical analysis shows that our design asymptotically approaches the optimal revenue as the market demand becomes sufficiently high, which is naturally the case in reality. Extensive simulation studies further indicate that even for cases where the market demand is not high, our design still offers close approximation to the optimal one.

APPENDIX A ANALYSIS OF OPTIMIZATION STRUCTURES

To analyze the optimization structures, we need the following two technical lemmas.

Lemma 9: Let f be a decreasing and concave function. Define g by $g(n) = \sum_{k=0}^n B(n, k, q)f(k)$. Then g is also decreasing and concave.

Proof sketch: Without loss of generality, we assume $f(0) = 0$. After some algebraic manipulation, we obtain

$$\frac{1}{q} \nabla g(n) = \sum_{k=0}^{n-1} B(n-1, k, q) \nabla f(k+1) \leq 0,$$

and

$$\frac{1}{q^2} \nabla^2 g(n+1) = \sum_{k=0}^{n-1} B(n-1, k, q) \nabla^2 f(k+2) \leq 0$$

for all $n > 1$. Therefore, g is decreasing and concave. ■

Lemma 10: Let f_1 and f_2 be two functions that are increasing and concave. Define f as follows.

$$f(n) = \max_{k=0,1,\dots,n} \{f_1(k) + f_2(n-k)\}. \quad (11)$$

Let k_n^* be an optimal solution to (11), and if there exist multiple solutions, let k_n^* be the smallest one. We have the following four statements.

- (1) f is increasing and concave.
- (2) $k_n^* \geq k$ if and only if $\nabla f_1(k) > \nabla f_2(n-k+1)$.

- (3) $k_n^* = \max\{1 \leq k \leq n : \nabla f_1(k) > \nabla f_2(n-k+1)\}$ if $\nabla f_1(1) > \nabla f_2(n)$; otherwise $k_n^* = 0$.

- (4) $k_{n+1}^* - 1 \leq k_n^* \leq k_{n+1}^*$.

Proof sketch: (1) For any discrete function h , let \bar{h} be its linear interpolation, i.e., $\bar{h}(x) = h(n) + \Delta h(n) \cdot (x-n)$, where $n = \lfloor x \rfloor$. Given n , we define $g(k) = f_1(k) + f_2(n-k)$. It is easy to see that $\bar{g}(x) = \bar{f}_1(x) + \bar{f}_2(n-x)$. Since \bar{g} is a linear interpolation of a discrete function g , we know its maximum value is achieved at the integer point, i.e.,

$$\begin{aligned} f(n) &= \max_{k=0,1,\dots,n} g(k) \\ &= \max_{0 \leq x \leq n} \bar{g}(x) \\ &= \max_{0 \leq x \leq n} \{\bar{f}_1(x) + \bar{f}_2(n-x)\}. \end{aligned}$$

Let $f_0(y)$ be defined by

$$f_0(y) = \max_{0 \leq x \leq y} \{\bar{f}_1(x) + \bar{f}_2(y-x)\}.$$

Since both f_1 and f_2 are increasing and concave, their linear interpolations, \bar{f}_1 and \bar{f}_2 , are also increasing and concave. By [31] (Rule 9 of Theorem 3.1.5), we know f_0 is increasing, concave and piecewise linear. Also note that $f_0(n) = f(n)$ for all integer n . We conclude that f_0 is a linear interpolation of f , i.e., $f_0 = \bar{f}$. In this case, f is increasing and concave.

- (2) Let $f(n) = \max_{0 \leq k \leq n} g(k)$ where

$$g(k) = f_1(k) + f_2(n-k).$$

Since both f_1 and f_2 are concave, g is concave. In this case, $k_n^* \geq k$ if and only if $\nabla g(k) > 0$. We hence conclude the proof by noticing that $\nabla g(k) = \nabla f_1(k) - \nabla f_2(n-k+1)$.

- (3) This statement is a direct application of Statement 2.

(4) First we show $k_n^* < k_{n+1}^* + 1$. By Statement 2, this is equivalent to show $\nabla f_1(k_{n+1}^* + 1) \leq \nabla f_2(n - k_{n+1}^*)$. This is indeed the case as

$$\begin{aligned} \nabla f_1(k_{n+1}^* + 1) &\leq \nabla f_2(n+1 - (k_{n+1}^* + 1) + 1) \\ &\leq \nabla f_2(n - k_{n+1}^*), \end{aligned}$$

where the first inequality is derived by applying Statement 3 to the case of $n+1$, while the second inequality is due to the concavity of f_2 .

We next show that $k_n^* \geq k_{n+1}^* - 1$. By statement 2, this is equivalent to prove $\nabla f_1(k_n^* - 1) > \nabla f_2(n - k_n^* + 2)$, which

is true because

$$\begin{aligned} \nabla f_1(k_n^* - 1) &> \nabla f_2(n + 1 - (k_n^* - 1) + 1) \\ &\geq \nabla f_2(n - k_n^* + 2), \end{aligned}$$

where the first inequality is derived by applying Statement 3 to the case of $n + 1$, while the second inequality is due to the concavity of f_2 . ■

We now prove Lemma 3 by induction.

Proof of Lemma 3: *Basis:* Show that the statement holds for $\tau = T$. In this case, since $\bar{\gamma}(\cdot)$ is increasing, we know that $\bar{V}_T(c) = \frac{1}{q} \mathbf{E}[\bar{\gamma}_T(c)]$, which is concave by Lemma 2. It is easy to see that $\bar{\mu}_{T+1}(c) = 0$ for all $c \geq 0$. We conclude that the statement holds for the basis.

Inductive step: Suppose the statement holds for τ . We now show it also holds for $\tau - 1$. Let $n = C - c$. We have

$$\begin{aligned} \bar{\mu}_\tau(c) &= \sum_{k=0}^{C-c} B(C - c, k, q) \bar{V}_\tau(c + k) \\ &= \sum_{k=0}^n B(n, k, 1 - q) \bar{V}_\tau(C - k). \end{aligned}$$

Applying Lemma 9 to this equation, we see the concavity of $\bar{\mu}_\tau(\cdot)$, since $\bar{V}_\tau(\cdot)$ is concave by the induction assumptions.

For $\bar{V}_{\tau-1}(\cdot)$, we inspect its definition (8). Since both $\bar{\mu}_\tau(\cdot)$ and $\bar{\gamma}_\tau(\cdot)$ are concave, applying Lemma 10 to (8) leads to the concavity of $\bar{V}_{\tau-1}$. ■

With Lemma 2 and 3, we see that Proposition 1 is a direct application of Lemma 10.

Proof of Proposition 1: Since both $\frac{1}{q} \bar{\gamma}_t(\cdot)$ and $\bar{\mu}_{t+1}(\cdot)$ are concave, applying Lemma 10 to (8) leads to the statement. ■

APPENDIX B ANALYSIS OF TRUTHFULNESS

To see the truthfulness of our design, we start by showing that no bidder has the incentive to overbook instances by proving Lemma 4.

Proof of Lemma 4: It suffices to prove $p(r_i) \leq p(r'_i)$ with $r'_i = r_i + 1$. Let Q_t^* be the capacity allocated when i requests r'_i instances. Similar notations also extend to $\bar{\gamma}'_t$, $\bar{\mu}'_{t+1}$, s' , and k' . It is easy to see that $\nabla \bar{\mu}'_{t+1} = \nabla \bar{\mu}_{t+1}$ and

$$\nabla \bar{\gamma}'_t(n) = \begin{cases} \nabla \bar{\gamma}_t(n), & \text{if } n \leq \sum_{j=1}^i r_j; \\ \nabla \bar{\gamma}_t(n - 1), & \text{o.w.} \end{cases} \quad (12)$$

We consider the following two cases.

Case 1: $i > k$. By (12), we see that $\frac{1}{q} \nabla \bar{\gamma}'_t(n) = \frac{1}{q} \nabla \bar{\gamma}_t(n) > \nabla \bar{\mu}_{t+1}(C_t - n + 1) = \nabla \bar{\mu}'_{t+1}(C_t - n + 1)$ for all $n \leq Q_t^*$, while $\frac{1}{q} \nabla \bar{\gamma}'_t(Q_t^* + 1) = \frac{1}{q} \nabla \bar{\gamma}_t(Q_t^* + 1) \leq \nabla \bar{\mu}_{t+1}(C_t - Q_t^*) = \nabla \bar{\mu}'_{t+1}(C_t - Q_t^*)$, where the inequalities are derived from Proposition 1. This implies $Q_t^* = Q_t^*$. In this case, both submissions result in the same winning set — the top k bidders. Therefore, $s' = s + 1$. We conclude that $p(r_i) = \max\{b_{k+1}, \hat{b}_s\} \leq \max\{b_{k'+1}, \hat{b}_{s'}\} = p(r'_i)$.

Case 2: $i \leq k$. In this case, it is easy to show that $\nabla \bar{\gamma}'_t(n + 1) = \nabla \bar{\gamma}_t(n)$ for all $n > Q_t^*$ (see (12)). Three cases are considered: (a) $\frac{1}{q} \nabla \bar{\gamma}_t(Q_t^*) > \nabla \bar{\mu}_{t+1}(C_t - Q_t^*)$. In this case,

$Q_t^* = Q_t^* + 1$ and $k' = k$. (b) $\frac{1}{q} \nabla \bar{\gamma}_t(Q_t^*) \leq \nabla \bar{\mu}_{t+1}(C_t - Q_t^*)$ and $\sum_{j=1}^k r_j < Q_t^*$. In this case, $Q_t^* = Q_t^*$ and $k' = k$. (c) $\frac{1}{q} \nabla \bar{\gamma}_t(Q_t^*) \leq \nabla \bar{\mu}_{t+1}(C_t - Q_t^*)$ and $\sum_{j=1}^k r_j = Q_t^*$. In this case, $Q_t^* = Q_t^*$ and $k' = k - 1$. (All results above can be verified by substituting the corresponding value of Q_t^* to (9).) For cases (a) and (b), since $k = k'$ (submitting r_i or r'_i results in the same winning set), We have $s' = s + 1$, and $p(r_i) = \max\{b_{k+1}, \hat{b}_s\} \leq \max\{b_{k'+1}, \hat{b}_{s'}\} = p(r'_i)$. For case (c), however, the winning sets are different: Bidder k loses if i submits r'_i . As a result, $s' = \sum_{j=1}^{k-1} r_j$ or $s' = \sum_{j=1}^{k-1} r_j + 1$, depending on whether $i = k$. In either case, $s' \leq s = Q_t^*$. We then have $\frac{1}{q} \phi(\hat{b}_{s'}) = \nabla \bar{\mu}_{t+1}(C_t - s' + 1) \leq \nabla \bar{\mu}_{t+1}(C_t - s + 1) = \frac{1}{q} \phi(\hat{b}_s) \leq \nabla \bar{\mu}_{t+1}(C_t - Q_t^* + 1) < \frac{1}{q} \nabla \bar{\gamma}_t(Q_t^*) = \frac{1}{q} \phi(b_k)$. Thus, $p(r'_i) = \max\{\hat{b}_{s'}, b_k\} = b_k$ and $p(r_i) = \max\{\hat{b}_s, b_{k+1}\} = b_{k+1}$. We see the statement holds since $b_{k+1} \leq b_k$. ■

We are now ready to prove Lemma 5 and 6, which lead to the truthfulness of the designed dynamic auctions.

Proof of Lemma 5: Let Q_t^* be the capacity allocated when i bids b'_i . Similar notations also extend to $\bar{\gamma}'_t$ and k' . Since i is in the winning set by submitting b_i , we know $i \leq k$. Now by submitting $b'_i > b_i$, it remains in the top k bidders. In this case, $\nabla \bar{\gamma}'_t(n) \geq \nabla \bar{\gamma}_t(n)$ for all $n \leq \sum_{j=1}^i n_j$ and $\nabla \bar{\gamma}'_t(n) = \nabla \bar{\gamma}_t(n)$ otherwise. Using these inequalities, we see $Q_t^* \geq Q_t^*$ because $\nabla \bar{\gamma}'_t(Q_t^*) \geq \nabla \bar{\gamma}_t(Q_t^*) > \nabla \bar{\mu}_{t+1}(C_t - Q_t^* + 1)$. On the other hand, $Q_t^* < Q_t^* + 1$ because $\nabla \bar{\gamma}'_t(Q_t^* + 1) = \nabla \bar{\gamma}_t(Q_t^* + 1) < \nabla \bar{\mu}_{t+1}(C_t - Q_t^*)$. Therefore, both the capacity allocation and the winning set remain unchanged, i.e., $Q_t^* = Q_t^*$ and $k' = k$, which implies that i wins again. ■

Proof of Lemma 6: Note that the submitted bid is of the form (n_i, b_i) . Let $s_k = \sum_{j=1}^k n_j$. For the first statement, since $b'_i > b_{k+1}$, bidder i remains in the top k bidders. We now consider $\nabla \bar{\gamma}'_t(s_k)$ for the new submission b'_i . Clearly, $\nabla \bar{\gamma}'_t(s_k) = \phi(b'_i)$ if b'_i is the lowest in all top k bids, otherwise $\nabla \bar{\gamma}'_t(s_k) = \phi(b_j)$ for some top bidder j other than i ($j \in \{1, \dots, k\} \setminus \{i\}$). Since j wins when i bids b_i , by Proposition 1, $\frac{1}{q} \phi(b_j) = \frac{1}{q} \nabla \bar{\gamma}_t(s_j) \geq \frac{1}{q} \nabla \bar{\gamma}_t(s_k) > \nabla \bar{\mu}_{t+1}(C_t - s_k + 1)$. Also, since $b'_i > \hat{b}_{s_k}$, we have $\frac{1}{q} \phi(b'_i) > \nabla \bar{\mu}_{t+1}(C_t - s_k + 1)$. All these results suggest that $\frac{1}{q} \nabla \bar{\gamma}'_t(s_k) > \nabla \bar{\mu}_{t+1}(C_t - s_k + 1)$. By Proposition 1, this means that all top k bidders win again because the allocated capacity is sufficient to accommodate all requests, i.e., $Q_t^* \geq s_k$.

For the second statement, we consider the following two scenarios. (a) $b'_i < b_{k+1}$. In this case, b'_i is no longer within the top k prices, and is at most the $k + 1$ -th highest bid. However, even in this case i is rejected, as the allocated capacity cannot accommodate all requests of the top $k + 1$ bidders ($Q_t^* < s_{k+1}$). To see this, we have $\frac{1}{q} \bar{\gamma}'_t(s_{k+1}) = \frac{1}{q} \phi(b'_i) < \frac{1}{q} \phi(b_{k+1}) = \frac{1}{q} \nabla \bar{\gamma}_t(s_{k+1}) \leq \frac{1}{q} \nabla \bar{\mu}_{t+1}(C_t - s_{k+1} + 1)$, where the last inequality is derived from the fact that $Q_t^* < s_{k+1}$ (see Algorithm 1). By Proposition 1, $Q_t^* < s_{k+1}$. Therefore, i loses by bidding b'_i . (b) $b'_i < \hat{b}_{s_k}$, or equivalently, $\frac{1}{q} \phi(b'_i) < \nabla \bar{\mu}_{t+1}(C_t - s_k + 1)$. From the results of (a), it suffices to assume $b'_i \geq b_{k+1}$ (i is in top k). With the similar argument

made in the proof of the first statement, we see that for every bidder $j \leq k$ other than i , $\frac{1}{q}\phi(b_j) > \nabla\bar{\mu}_{t+1}(C_t - s_k + 1)$. Therefore, $b_j > b'_i \geq b_{k+1}$, which means that i ranks in the k -th place in the new submission. However, since $\frac{1}{q}\nabla\bar{\gamma}'_t(s_k) = \frac{1}{q}\phi(b'_i) < \frac{1}{q}\nabla\bar{\mu}_{t+1}(C_t - s_k + 1)$, we see $Q_t^* < s_k$. As a result, i is rejected because the allocated capacity cannot accommodate all requests of top k bidders. ■

APPENDIX C REVENUE ANALYSIS FOR APPROXIMATE SOLUTION

To analyze the revenues performance of our design, we first prove Lemma 7.

Proof of Lemma 7: It suffices to prove the first inequality. Suppose k bidders win in Algorithm 1. We have $\bar{\gamma}_t(Q_t^*) - \gamma_t(Q_t^*) = (Q_t^* - \sum_{i=1}^k n_i)\phi(v_{k+1})$. (Note that $(r_i, b_i) = (n_i, v_i)$ due to the truthfulness.) We consider the following two cases.

Case 1: $n_{k+1} = \infty$. In this case, bidder $k+1$ is the virtual bidder with $\phi(v_{k+1}) = 0$. We see the statement holds with $\bar{\gamma}_t(Q_t^*) = \gamma_t(Q_t^*)$.

Case 2: $n_{k+1} < \infty$. In this case, bidder $k+1$ is a regular bidder with $n_{k+1} \leq \bar{n}$. We see the statement holds because $\bar{\gamma}_t(Q_t^*) - \gamma_t(Q_t^*) \leq \bar{n}\phi(v_{k+1}) \leq Q_t^*\phi(v_{k+1})/\alpha \leq \bar{\gamma}_t(Q_t^*)/\alpha$. Here, the last inequality is derived from the fact that $\phi(v_1) \geq \dots \geq \phi(v_{k+1}) \geq \dots \geq \phi(v_{N_t+1})$. ■

We now show that our design is asymptotically optimal by proving Lemma 8.

Proof of Lemma 8: Given α , we prove the statement by induction.

Basis: Show the statement holds for $\tau = T+1$. This is indeed the case due to the boundary conditions, i.e., $V_{T+1} = V_{T+1}^* = \bar{V}_{T+1} = 0$.

Inductive step: Suppose the statement holds for $\tau+1$. We show it also holds for τ . To see this, we first show $\gamma_\tau(Q_\tau^*) \geq (1 - \frac{1}{\alpha})\bar{\gamma}_\tau(Q_\tau^*)$ w.p.1 for all Q_τ^* by considering the following two cases.

Case 1: $Q_\tau^* \geq \alpha\bar{n}$. In this case, $\gamma_\tau(Q_\tau^*) \geq (1 - \frac{1}{\alpha})\bar{\gamma}_\tau(Q_\tau^*)$ because of Lemma 7.

Case 2: $Q_\tau^* < \alpha\bar{n}$. In this case, we show $\gamma_\tau(Q_\tau^*) = \bar{\gamma}_\tau(Q_\tau^*)$ w.p.1. Let X be the number of top bidders with unit demand, i.e., $n_i = 1$ and $v_i = \bar{v}$. Suppose $P(n_i = 1, v_i = \bar{v}) = p > 0$. We know $X \sim \text{Bin}(N_t, p)$. For any finite k , we have $\lim_{N_t \rightarrow \infty} P(X = k) = \lim_{N_t \rightarrow \infty} B(N_t, k, p) = \lim_{N_t \rightarrow \infty} \mathcal{N}(N_t p, kp(1-p)) = 0$, where the second equality is derived from the central limit theorem. This essentially indicates that $P(X \geq Q_\tau^*) = 1$ for any finite Q_τ^* . That is, w.p. 1, all Q_τ^* instances can be allocated to top bidders with unit demand, which implies $\gamma_\tau(Q_\tau^*) = \bar{\gamma}_\tau(Q_\tau^*)$ w.p.1.

We therefore conclude that $\gamma_\tau(Q_\tau^*) \geq (1 - \frac{1}{\alpha})\bar{\gamma}_\tau(Q_\tau^*)$ w.p.1. Also, from the induction assumptions, $\mathbf{E}_K[\bar{V}_{\tau+1}(C_\tau - Q_\tau^* + K)] \geq (1 - \frac{1}{\alpha})\mathbf{E}_K[\bar{V}_{\tau+1}(C_\tau - Q_\tau^* + K)]$ w.p.1. Substituting all these results back to (10), we see $V_\tau(C_\tau) \geq (1 - \frac{1}{\alpha})\bar{V}_\tau(C_\tau)$ for all $C_\tau = 0, 1, \dots, C$. ■

REFERENCES

- [1] Amazon EC2 Pricing, <http://aws.amazon.com/ec2/pricing/>.
- [2] GoGrid Cloud Hosting, <http://www.gogrid.com>.
- [3] RackSpace Cloud Hosting, <http://www.rackspace.com/cloud>.
- [4] M. Stokely, J. Winget, E. Keyes, C. Grimes, and B. Yolken, "Using a market economy to provision compute resources across planet-wide clusters," in *IEEE Intl. Symp. on Parallel & Distributed Processing*, 2009.
- [5] A. AuYoung, P. Buonadonna, B. Chun, C. Ng, D. Parkes, J. Shneidman, A. Snoeren, and A. Vahdat, "Two auction-based resource allocation environments: Design and experience," *Market-Oriented Grid and Utility Computing*, pp. 513–539, 2009.
- [6] K. Lai, L. Rasmusson, E. Adar, L. Zhang, and B. Huberman, "Tycoon: An implementation of a distributed, market-based resource allocation system," *Multiagent and Grid Systems*, vol. 1, no. 3, pp. 169–182, 2005.
- [7] S. Zaman and D. Grosu, "Combinatorial auction-based allocation of virtual machine instances in clouds," in *Proc. IEEE CloudCom*, 2010.
- [8] Q. Zhang, E. Gürses, R. Boutaba, and J. Xiao, "Dynamic resource allocation for spot markets in clouds," in *Proc. HOT-ICE*, 2011.
- [9] A. Danak and S. Mannor, "Resource allocation with supply adjustment in distributed computing systems," in *Proc. IEEE ICDCS*, 2010.
- [10] Q. Wang, K. Ren, and X. Meng, "When cloud meets eBay: Towards effective pricing for cloud computing," in *Proc. IEEE INFOCOM*, 2012.
- [11] R. B. Myerson, "Optimal auction design," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 58–73, 1981.
- [12] J. Bulow and J. Roberts, "The simple economics of optimal auctions," *Journal of Political Economy*, vol. 97, no. 5, pp. 1060–1090, 1989.
- [13] G. Aggarwal and J. Hartline, "Knapsack Auctions," in *Proc. 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006, pp. 1083–1092.
- [14] A. Archer and É. Tardos, "Truthful Mechanisms for One-Parameter Agents," in *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, 2001.
- [15] D. C. Parkes and S. Singh, "An MDP-Based Approach to Online Mechanism Design," in *Proc. 17th Annual Conf. on Neural Information Processing Systems (NIPS)*, 2003.
- [16] G. Vulcano, G. Van Ryzin, and C. Maglaras, "Optimal dynamic auctions for revenue management," *Management Science*, vol. 48, no. 11, pp. 1388–1407, 2002.
- [17] A. Gershkov and B. Moldovanu, "Dynamic revenue maximization with heterogeneous objects: A mechanism design approach," *American Economic Journal: Microeconomics*, vol. 1, no. 2, pp. 168–198, 2009.
- [18] M. Said, "Auctions with dynamic populations: Efficiency and revenue maximization," in *Proc. AMMA*, 2009.
- [19] M. Balcan, A. Blum, J. Hartline, and Y. Mansour, "Mechanism Design via Machine Learning," in *Proc. 46th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2005, pp. 605–614.
- [20] V. Vapnik, *Statistical Learning Theory*. Wiley-Interscience, 1998.
- [21] I. Segal, "Optimal Pricing Mechanisms with Unknown Demand," *The American Economic Review*, vol. 93, no. 3, pp. 509–529, 2003.
- [22] E. Caron, F. Desprez, and A. Muresan, "Forecasting for grid and cloud computing on-demand resources based on pattern matching," in *Proc. IEEE CloudCom*, 2010.
- [23] R. Randhawa and S. Kumar, "Usage restriction and subscription services: Operational benefits with rational users," *Manufacturing & Service Operations Management*, vol. 10, no. 3, pp. 429–447, 2008.
- [24] H. Fujiwara and K. Iwama, "Average-case competitive analyses for skilrental problems," *Algorithmica*, vol. 42, no. 1, pp. 95–107, 2005.
- [25] Y. Xu and W. Xu, "Competitive algorithms for online leasing problem in probabilistic environments," in *Advances in Neural Networks (ISNN)*, 2004.
- [26] W. Wang, B. Li, and B. Liang, "Towards optimal capacity segmentation with hybrid cloud pricing," in *Proc. IEEE ICDCS*, 2012.
- [27] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 2005.
- [28] P. Klemperer, *Auctions: Theory and Practice*. Princeton University Press, 2004.
- [29] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of Finance*, vol. 16, no. 1, pp. 8–37, 1961.
- [30] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*. Cambridge University Press, 2007.
- [31] J. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer Verlag, 2001.