An Auction Mechanism for Cloud Spot Markets

ADEL NADJARAN TOOSI, University of Melbourne, Australia
KURT VANMECHELEN, University of Antwerp, Belgium
FARZAD KHODADADI and RAJKUMAR BUYYA, University of Melbourne, Australia

Dynamic forms of resource pricing have recently been introduced by cloud providers that offer Infrastructure as a Service (IaaS) capabilities in order to maximize profits and balance resource supply and demand. The design of a mechanism that efficiently prices perishable cloud resources in line with a provider's profit maximization goal remains an open research challenge, however. In this article, we propose the Online Extended Consensus Revenue Estimate mechanism in the setting of a recurrent, multiunit and single price auction for IaaS cloud resources. The mechanism is envy-free, has a high probability of being truthful, and generates a near optimal profit for the provider. We combine the proposed auction design with a scheme for dynamically calculating reserve prices based on data center Power Usage Effectiveness (PUE) and electricity costs. Our simulation-based evaluation of the mechanism demonstrates its effectiveness under a broad variety of market conditions. In particular, we show how it improves on the classical uniform price auction, and we investigate the value of prior knowledge on the execution time of virtual machines for maximizing profit. We also developed a system prototype and conducted a small-scale experimental study with a group of 10 users that confirms the truthfulness property of the mechanism in a real test environment.

Categories and Subject Descriptors: C.4.7 [**Organization and Design**]: Distributed Systems; K.6.0 [**General**]: Economics; K.6.2 [**Installation Management**]: Pricing and Resource Allocation; C.2.4 [**Distributed Systems**]: Distributed Applications

General Terms: Design, Algorithms, Economics

Additional Key Words and Phrases: Pricing, cloud computing, auction, mechanism design, multi-unit, online, reserve price, PUE

ACM Reference Format:
Adel Nadjaran Toosi, Kurt Vanmechelen, Farzad Khodadadi, and Rajkumar Buyya. 2016. An auction mechanism for cloud spot markets. ACM Trans. Auton. Adapt. Syst. 11, 1, Article 2 (February 2016), 33 pages.
DOI: http://dx.doi.org/10.1145/2843945

1. INTRODUCTION

The increased adoption and maturity of cloud computing offerings has been accompanied by a growing role and significance of pricing mechanisms for trading computational resources. Especially Infrastructure as a Service (IaaS) cloud providers that offer computational services in the form of Virtual Machine (VM) instances with specific resource characteristics have gradually expanded their pricing plans in order to maximize their profits and further attract demand. Currently, the most widely used model remains a fixed *pay-as-you-go* pricing plan wherein the consumer is charged the amount of time a

Authors' addresses: A. N. Toosi, F. Khodadadi, and R. Buyya, Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computing and Information Systems, The University of Melbourne; emails: adel.nadjaran@unimelb.edu.au, fkhodadadi@student.unimelb.edu.au, rbuyya@unimelb.edu.au; K. Vanmechelen, Department of Mathematics and Computer Science, University of Antwerp; email: kurt.vanmechelen@uantwerpen.be.

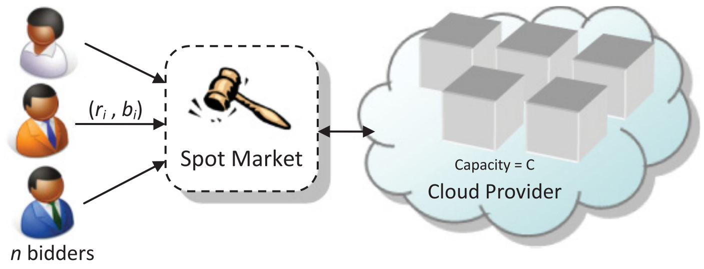


Fig. 1. Spot market and auction mechanism.

VM instance was used at a fixed rate. However, the fact that computational resources sold by a cloud provider can be characterized as a *non-storable* or *perishable* commodity,[1] combined with the fact that demand for computational resources is nonuniform over time, motivates the use of dynamic forms of pricing in order to optimize revenue [Lee and Szymanski 2005]. Through price adjustment based on actual (and possibly forecasted) supply and demand conditions, consumers can be incentivized to acquire spare capacity or shift demand from on-peak to off-peak hours. Consequently, both profits and consumer satisfaction can be increased.

Market-based pricing mechanisms that solicit reports (*bids*) from consumers and subsequently use *allocation* and *pricing* rules are well fit to realize such dynamic forms of pricing; recently, they have received significant attention for selling underutilized capacity of cloud infrastructures [Toosi et al. 2011; Macías and Guitart 2011]. Well-designed auction mechanisms can be particularly effective since they (i) incentivize users to reveal the true value of their required resources (i.e., report the price they are willing to pay for resources), (ii) ensure resources are allocated to those who value them the most, and (iii) correctly price resources in line with supply-and-demand conditions by creating competition among buyers.

Amazon Web Services (AWS) has adopted an auction-like approach to expand its pricing plans with Spot Instances for the Amazon Elastic Compute Cloud (EC2). In this scheme, consumers communicate their bids for a VM instance hour to AWS. Subsequently, AWS reports a market-wide *spot price* at which VM instance use is charged while terminating any instances that are executing under a bid price that is lower than the market price. Although Amazon is not the only provider to offer dynamic pricing, it is currently the only IaaS provider that publicly offers an auction-like mechanism for selling IaaS resources. Nevertheless, attempts to create such mechanisms have already been reported by other companies [Stokely et al. 2009] and have also received attention by academia [Ben-Yehuda et al. 2013; Danak and Mannor 2010; Wang et al. 2012; Zhang et al. 2011].

AWS has revealed no detailed information regarding its auction mechanism and the calculation of the spot price. At present, the design of an efficient, fair, and profit-maximizing auction mechanism for pricing cloud computing resources is an open research challenge and of great interest to cloud providers.

In this article, we design such an auction mechanism aimed at generating additional profit from the spare capacity of non-storable resources available in cloud data centers. Note that we refer to the marketplace in which this mechanism is used to sell VMs as the cloud *spot market* (Figure 1).

[1]Note that resources tied to a VM are qualified as non-storable (perishable) because a non-used hour of CPU time or memory space can never be reclaimed and therefore wastes data center capacity.

The spare capacity that can be offered by an IaaS cloud provider in the spot market is usually much larger than the demand.[2] Therefore, a provider is potentially able to accept all consumer requests. In this context, popular auction mechanisms such as the second-price Vickrey [1961] auction may fail to generate a reasonable revenue for the provider [Ausubel and Milgrom 2006]. In general, when supply exceeds demand, bidders are less motivated to bid competitively, which can prevent providers from collecting the optimal revenue. Providers therefore require an auction mechanism that can maximize revenue while incentivizing bidders to reveal their true value.

An auction mechanism is truthful if for each bidder i and any choice of order values by all other bidders, bidder i's *dominant strategy* is to truthfully report her private information with respect to her order. A strategy is dominant if a bidder cannot increase the payoff derived from participating in the mechanism by diverging from it. Such a truthful and revenue-maximizing auction mechanism can be designed [Wang et al. 2012] if perfect knowledge about the distribution from which the bidders' valuations are drawn is available. Unfortunately, this is not always the case, and pricing depends heavily on the accuracy of the underlying market analysis. Such analysis also needs to be updated frequently in order to adapt to changes in the market. Moreover, since customers of cloud services are distributed globally and experience different latency for the same service, it might be invalid to assume that the valuations for all bidders are drawn *independent and identically distributed* (i.i.d.) from some underlying distribution.

In addition, as shown by Green and Laffont [1986] and Hurwicz [1975], it is impossible for a mechanism to simultaneously attain the properties of being truthful, efficient, individually rational, and budget-balanced. In fact, our proposed mechanism will only achieve truthfulness with high probability, and not in the absolute sense. In practice, this means that there is a probability that bidders can improve their outcome through misreporting, albeit that probability becomes lower when the market size grows.

This article focuses on designing an auction mechanism for cloud spot markets aimed at maximizing the cloud provider's profit. The cloud spot market context influences our auction design in the sense that the design needs to support multiunit bids, operate in an online recurrent manner, result in a single market-wide price and fair outcomes, operate under a limitation of the maximal quantity a consumer can request, operate without prior knowledge on the distribution of bidders' valuations, and, finally, allow for reserve prices to be set during oversupply conditions. The article's key contributions are:

—The design and application of a multiunit, online, recurrent auction mechanism called *Online Extended Consensus Revenue Estimate mechanism* (Online Ex-CORE) within the context of IaaS resource trading. The proposed auction mechanism is envy-free, truthful with high probability, and generates near optimal profit for the provider in a single round of auction. It adopts a greedy approach for maximizing provider profits in the online setting. It is initially designed for the unlimited supply case and is subsequently extended to the limited supply case.

—The evaluation of the proposed mechanism with respect to revenue generation, truthfulness, and bid rejection rates. Extensive simulation results are presented that demonstrate that it achieves near optimality with respect to maximizing revenue without requiring prior knowledge on the order distributions. It is also shown to achieve low bid rejection rates, thus mitigating the *bidder drop problem* in online

[2]This can be explained by the promise of clouds providing infinite capacity of resources [Armbrust et al. 2010], and recent reports that suggest the overall utilization in large data centers is lower than 30% most of the time [Goiri et al. 2011].

mechanisms [Lee and Szymanski 2005]. We compare the proposed mechanism to a clairvoyant and nonclairvoyant variant of the Optimal Single Price Auction and to the Uniform Price Auction.

—A clairvoyant optimal auction mechanism (HTA-OPT) that uses dynamic programming to calculate the set of accepted bids. HTA-OPT serves as a benchmark that is used to quantify the efficiency loss caused by the lack of information on the amount of time a bidder wants to run a VM when applying the allocation rule in a single auction round.

—The presentation of a method for dynamically computing a *reserve price* based on a coarse-grained data center power usage model that can be used by the provider within the proposed auction mechanism. The resulting prices are shown to correspond to actual minimal spot prices observed on the EC2 spot market.

—The design and implementation of a system prototype of the proposed auction mechanism using the OpenStack cloud platform and conducting a small-scale experimental study by employing group of 10 participants competing to acquire IaaS cloud resources in a spot market.

The remainder of this article is organized as follows: After reviewing related work in Section 2, we introduce required terminology and notations in Section 3. Sections 4, 5, and 6 discuss, respectively, the competitiveness, truthfulness, and envy-freeness properties for our auction design. Section 7 describes the proposed auction mechanism, whereas Section 8 focuses on the limited supply setting and the computation of the reserve price in that setting. Section 9 describes the online version of the proposed auction mechanism and mechanisms used in the comparative analysis. Our experimental evaluation of the mechanism can be found in Section 10. Using simulation-based experiments, we compare its performance to the Optimal Single Price Auction and the Uniform Price Auction and investigate the impact of perfect knowledge on the execution time of a VM. We provide results concerning the probability that any bidder can benefit from an untruthful reporting of the number of VM instances required. We also evaluate the truthfulness property of the proposed auction mechanism using an experimental study in a real system implementation. Our conclusions and future directions follow in Section 11.

2. RELATED WORK

The use of an auction-like mechanism to sell spare capacity in cloud data centers was pioneered in late 2009 by Amazon. In Amazon's spot market, customers bid the maximum hourly price they are willing to pay to obtain a VM instance.[3] All instances incur a uniform charge, the *spot market price*. According to Amazon, this price is set dynamically based on the relationship of supply and demand over time. A unique feature of spot instances is that the provider has the right to terminate them when their associated bid falls below the spot market price. As a result, the resulting Quality of Service (QoS) may be lower compared to *on-demand* and *reserved* instances, depending on the bid made. Current spot market data show customers can acquire VMs at price reductions of between 50% and 93% compared to on-demand instances.

Amazon has revealed little information on the pricing and allocation rules of its mechanism. Ben-Yehuda et al. [2013] examined the price history of the EC2 spot market through a reverse engineering process and found that the mechanism was not completely driven by demand and supply. Their analysis suggests that spot prices are usually drawn from a tight, fixed price interval and reflect a random nondisclosed reserve price. In this article, we propose an auction mechanism with transparent

[3]http://aws.amazon.com/ec2/spot-instances/.

allocation and pricing rules while sharing similar properties with the EC2 spot market.

Several authors have presented strategies for customers to utilize Amazon spot instances (cost-)effectively [Chohan et al. 2010; Javadi et al. 2011; Song et al. 2012; Voorsluys and Buyya 2012; Yi et al. 2010]. However, as yet, a limited amount of work has been conducted that focuses on the design of auction mechanisms to the benefit of cloud providers and the associated algorithms for allocating resources and capacity planning to maximize the provider's revenue. The problem of dynamically allocating resources to different spot markets in order to maximize a cloud provider's revenue has been investigated by Zhang et al. [2011]. Danak and Manno [2010] present a uniform-price auction for resource allocation that suits the dynamic nature of grid systems. Mihailescu and Teo [2012] investigate Amazon EC2's spot market as a case in a federated cloud environment. They argue that spot pricing used by Amazon is truthful only in a market with a single provider and show that rational users can increase their utility by being untruthful in a federated cloud environment. Zaman et al. [2013] have investigated the applicability of combinatorial auction mechanisms for allocation and pricing of VM instances in cloud computing. The same authors [Zaman and Grosu 2012] propose an online auction mechanism for VM provisioning and allocation. In their model, a user requests a bundle of VMs expressing that she is only interested in the whole bundle and not a subset of it. Moreover, they assume that the user submits a holding time period for her request and, once she receives an allocation, continues to hold the resources for that time period (i.e., no preemption). By contrast, our mechanism performs similarly to the Amazon EC2 market for spot instances in which partial fulfillment of the request is acceptable and VM terminations by provider happen for *out-of-bid* requests. Recently, Zhang et al. [2014] studied combinatorial auctions of heterogeneous VMs. They focused on social welfare maximization, whereas we aim at profit maximization for the cloud provider.

Xu and Li [2013] proposed an infinite horizon stochastic dynamic program to dynamically price the IaaS cloud provider's resources based on stochastic demand arrivals and departures of cloud users. They aim at maximizing revenue specifically for the spot market and adopt a revenue management framework from economics to build their model without any discussion of the truthfulness of the proposed strategy. Wang et al. [2012] proposed an optimal recurrent auction for a spot market based on the seminal work of Myerson [1981]. The mechanism was designed in the context of optimally segmenting the provider's data center capacity between on-demand and spot market requests. Their work differs from ours since they adopt a *Bayesian* approach wherein it is assumed that the customers' private values are drawn from a known distribution. They also propose a truthful dynamic auction [Wang et al. 2013] that periodically computes the number of instances to be auctioned off in order to maximize providers' revenue. Unlike the EC2 spot marketplace, their approach offers guaranteed services (i.e., instances are never terminated by the provider) and constant price over time (i.e., because the price is set for the user, it remains constant as long as the user holds the instance). Their auction charges each user a different price and does not generate a market-wide single price. Moreover, their auction mechanism requires a priori known distribution of valuations and near future demand prediction.

In contrast, we propose an auction mechanism designed to maximize profit based on the *competitive framework* proposed by Goldberg et al. [2006]. The mechanism computes a uniform price outcome and focuses on maximizing profit when the seller knows very little about the bidders valuations. To achieve truthfulness in this context, we rely on a *consensus estimation* technique [Goldberg and Hartline 2003a]. Our work differs from that of Goldberg et al. in several respects. First, their analysis relies on the assumption that each customer is restricted to formulated unit demand, which is not the case for cloud consumers. Consequently, we revisit the definition and truthfulness analysis of the mechanism for the multiunit case. Second, their auction mechanism is designed for offline single-round scenarios. The context of a cloud spot market, however, requires an online auction where customers arrive over time and resources allocated by VM instances can be released and subsequently reused by other consumers. We adopt a greedy approach in realizing the online character of the auction, and we investigate its performance compared to a clairvoyant optimal mechanism that relies on dynamic programming. Finally, the production cost of goods is not taken into account in their work. In the IaaS setting, taking this cost into account is important because a seller has the option to either shut down server capacity or sell it at a given *reserve price*. We add such reserve pricing to the mechanism and introduce a coarse-grained cost model for determining it.

Lee and Szymanski [2005] have proposed an auction mechanism for time-sensitive e-services in which services must be repeatedly resold for future time periods. They investigated the *bidder drop problem* in recurrent auctions that occurs when the least wealthy bidders tend to withdraw from the future auction rounds due to repeatedly losing the auction. Our proposed auction is not specifically designed to address this issue; however, our evaluation shows that it rejects a lower number of requests compared to the Optimal Single Price auction while generating near optimal revenue.

3. PRELIMINARIES AND NOTATION

Consider a cloud provider with capacity C for a specific type of VM. That is, at a given time t, up to C instances of the specific type can be hosted simultaneously. The provider runs a sealed-bid auction $\mathcal{A}$ to sell this capacity. First, we assume the case that the provider's capacity far exceeds the total demand, in line with the cloud's promise of delivering an unlimited supply of resources. Subsequently, we generalize the results to a scenario in which supply is limited and lower than total demand.

Suppose there are n customers joining the auction at time t. Each bidder i $(1 \leq i \leq n)$ requires q_i VM instances and has a private valuation v_i, denoting the maximum amount i is willing to pay for each VM instance per time slot (e.g., 1 hour). Customers submit an order (request) (r_i, b_i) where r_i represents the number of required VM instances and b_i the bid price. We denote by $\mathbf{d}$ the vector of all submitted orders. The ith element of $\mathbf{d}$, d_i, is the order by customer i.

Given $\mathbf{d}$, the provider (auctioneer) computes an allocation vector, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, and a price vector, $\mathbf{p} = (p_1, p_2, \ldots, p_n)$. The ith component x_i of the allocation vector indicates whether bidder i receives the r_i VMs requested in its order (if $x_i = 1$) or not ($x_i = 0$). A bidder for which $x_i = 1$ is called a *winner* and pays the corresponding price p_i; otherwise, the bidder is called a *loser* and does not make any payment to the mechanism. Because we mostly focus on *single price* auctions, all p_i are equal for all winning bidders and we therefore refer to the sale price as p.

Definition 3.1. A *single price auction* is defined as follows: Let $\mathbf{d}$ be the vector of all submitted orders. Auction $\mathcal{A}$ is called a *single price auction* if, on input $\mathbf{d}$, it determines only a single sale price p at which all bidders with $b_i \geq p$ win and pay a price p and all remaining bidders lose and pay 0.

Note that whenever we discuss a single price auction (e.g., Uniform Price, Optimal Single Price, or Ex-CORE), the determination of the allocation outcome is omitted because it can be simply deduced from the definition. *Partial fulfillment* of requests, in which only a fraction of the number of VM instances requested is allocated to a winning bidder, is only considered in the case of limited supply and when $b_i = p$. We allow for partial fulfillment for those orders in line with the behavior of the EC2 spot market, and we discuss it in later parts of this article.

Bidders are individually rational users who try to maximize their utility. Therefore, as long as it is deemed beneficial, a customer will strategically misreport her bid or the required number of VMs (i.e., $b_i \neq v_i$ or $r_i \neq q_i$), where v_i and q_i are private information known only to customer i. We define customer i's utility at time t for one time slot of VM usage as follows:

$$u_i(r_i, b_i) = \begin{cases} (q_i v_i - r_i p_i)x_i, & \text{if } b_i \geq p_i \text{ and } r_i \geq q_i; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The values of r_i and v_i for each customer are drawn from distributions that are unknown to the provider. Customer i's optimal bidding strategy must be defined so that it maximizes i's utility over all time slots. However, assuming that customers are not aware of the future and have no time-dependent valuation for resources, we define the utility function in Equation (1) based on a single time slot. Winners in an auction round are awarded their requested VM instances and automatically attend the next round of the auction until they cancel their requests on their own account or they lose the auction. In the latter case, VMs held by an out-of-bid customer are terminated by the provider without any prior notice.

The *holding time* of a VM is the specific amount of time a customer wants to run the VM. The VM's actual holding time might be smaller than the expected time if it is terminated by the provider instead of the owner. The holding time of a VM by the customer is not known to the provider (or to the mechanism) in advance. Therefore, in our model, a provider acts in a greedy manner to maximize revenue according to the arriving requests and the current existing requests in each round of auction. This can be modeled as a single round auction that is recurrently conducted by the provider as new requests arrive or current requests are terminated. In Section 10.3, we compare the performance of this greedy strategy to the optimal strategy that has prior knowledge of the VM holding time. From this point onward, we limit our discussion only to a single round of the auction. In Section 9, we introduce the recurrent version of the mechanism.

4. COMPETITIVE FRAMEWORK

The revenue generated by auction $\mathcal{A}$ in a time slot equals:

$$\mathcal{A}(\mathbf{d}) = \sum_i r_i p_i. \tag{2}$$

The problem of maximizing revenue in an auction for cloud resources can be solved optimally if the seller knows the distribution from which the bidders' valuations are drawn [Wang et al. 2012]. In conventional economics, this is called *Bayesian optimal mechanism design* [Myerson 1981; Nisan et al. 2007]. However, we assume that the distributions from which the bidder's private information are drawn are unknown to the provider. Therefore, we base our approach on the competitive mechanism design framework proposed by Goldberg et al. [2006]. We will compare the revenue attained by our mechanism to that of the *Optimal Single Price auction* for the unlimited capacity case.

Definition 4.1. The *Optimal Single Price auction* $\mathcal{F}$ is defined as follows: Let $\mathbf{d}$ be an order vector. Without loss of generality, suppose the components of $\mathbf{d}$ are sorted in descending order by bid values. So, (r_i, b_i) is the ith largest bid in $\mathbf{d}$ regardless of r_i. The auction $\mathcal{F}$ on input $\mathbf{d}$ determines the value k such that $b_k \sum_{i=1}^{k} r_i$ is maximized. We denote by $\sigma_k(\mathbf{d})$ the sum of the number of requested instances in the sorted vector of orders from the first order to kth order ($\sigma_k(\mathbf{d}) = \sum_{i=1}^{k} r_i$). All bidders with $b_i \geq b_k$ win at price b_k, and all remaining bidders lose. Thus, the revenue of $\mathcal{F}$ on input $\mathbf{d}$ is

$$\mathcal{F}(\mathbf{d}) = \max_i \; b_i\sigma_i(\mathbf{d}). \tag{3}$$

If more than one value of i maximizes $b_i\sigma_i(\mathbf{d})$, choosing the price point that results in a lower transacted volume is preferable considering the cost of accommodating VM instances (e.g., electricity cost). From this point forward, we assume $\mathbf{d}$ is sorted decreasingly by bids values (b_i) unless otherwise mentioned.

We are interested in an auction mechanism that is competitive with $\mathcal{F}$ on every possible input; however, if a single bidder's utility dominates the total utility of the other bidders, no auction can compete with $\mathcal{F}$, as shown by Goldberg et al. [2006]. We do not consider this to be an issue in our setting because the cloud environment can be viewed as a *mass market* in which the number of winners of the optimal single price auction is typically large. In a mass market, removing one order does not significantly change the maximum extractable profit.

Definition 4.2 (*Mass Market*). Let $\mathcal{F}(\mathbf{d})$ be the revenue of $\mathcal{F}$ and $h_b(\mathbf{d})$ denote the maximum value of b in $\mathbf{d}$; then $\mathcal{F}(\mathbf{d}) \gg h_b(\mathbf{d})$ in mass-markets, which implies that $\mathcal{F}$ sells $m \gg 1$ units.

We say that auction $\mathcal{A}$ is competitive if there exists a constant β such that $\mathcal{A}(\mathbf{d}) \geq \mathcal{F}(\mathbf{d})/\beta$. For a randomized mechanism,[4] the previous equation for competitiveness becomes:

$$\mathbf{E}[\mathcal{A}(\mathbf{d})] \geq \frac{\mathcal{F}(\mathbf{d})}{\beta}. \tag{4}$$

Assuming the fact that $\mathcal{F}$ sells at least m units, we define $\beta(m)$-competitiveness for a mass market as below:

Definition 4.3. Auction $\mathcal{A}$ is $\beta(m)$-competitive for a mass market if, for all order vectors $\mathbf{d}$ such that $\mathcal{F}$ sells at least m units, we have:

$$\mathbf{E}[\mathcal{A}(\mathbf{d})] \geq \frac{\mathcal{F}(\mathbf{d})}{\beta(m)}. \tag{5}$$

5. TRUTHFULNESS

Let $\mathbf{d}_{-i}$ denote the vector of orders $\mathbf{d}$ with $(r_i,\ b_i)$ removed; that is, $\mathbf{d}_{-i} = ((r_1, b_1), \ldots, (r_{i-1}, b_{i-1}), (r_{i+1}, b_{i+1}), \ldots, (r_n, b_n))$, and further introduce the notation $\mathcal{F}((r_i, b_i), \mathbf{d_{-i}}) = \mathcal{F}(\mathbf{d})$.

Proposition 1. *$\mathcal{F}$ is not truthful.*

Proof. Suppose $\mathcal{F}$ is truthful, then utility for each bidder i is maximized if $b_i = v_i$ and $r_i = q_i$ for any choice of $\mathbf{d}_{-i}$.

Consider $\mathbf{d}$ as any arbitrary vector of orders and assume $\mathcal{F}(\mathbf{d})$ is the maximum revenue by $\mathcal{F}$ and b_k is the sale price. Suppose $\mathcal{F}_2(\mathbf{d})$ is the second largest revenue that can be obtained by $\mathcal{F}$, and we limit $\mathbf{d}$ to those vectors such that $\mathcal{F}(\mathbf{d}) > \mathcal{F}_2(\mathbf{d})$. Given a fixed $\mathbf{d}_{-k}$, r_k, q_k, bidder k is able to reduce her bid from b_k to b'_k and still be the winner as long as $\mathcal{F}((r_k, b'_k), \mathbf{d}_{-k}) > \mathcal{F}_2(\mathbf{d})$. As a result, fixing other variables and considering that bidder k is a winner ($x_k = 1$), bidder k is able to increase her utility from $q_kv_k - r_kb_k$ to $q_kv_k - r_kb'_k$. So, there exists a $\mathbf{d}_{-i}$ and a bidder i such that u_i can be increased by misreporting i's true value (i.e., $b_i \neq v_i$). This contradicts the supposition that $\mathcal{F}$ is

[4]The mechanism's allocation and/or pricing rule procedure has a randomized component.

truthful. A similar proof can be constructed for the number of requested instances, which we omit here for space considerations. □

The following example will give us a better understanding of the preceding proof. Let $\mathbf{d} = ((1, \$8), (2, \$7), (4, \$2))$ be the truthful vector of orders. $\mathcal{F}(\mathbf{d})$ equals to \$21, and the sale price is \$7. Interested readers can check that this is the highest revenue that the auctioneer can generate with given $\mathbf{d}$. The utility value for the second bidder is $2 \times \$7 - 2 \times \$7 = 0$. Assuming other bidders will not change their orders, the second bidder can reduce her bid price to \$5 and still be the winner in the auction. In this case, the utility value for the second bidder will be $2 \times \$7 - 2 \times \$5 = \$4$. Therefore, the second bidder can increase her utility by misreporting her bid price, which shows $\mathcal{F}$ is not truthful.

To create a truthful auction, an intuitive idea is to design the mechanism in a way that a bidder believes that her own order does not affect the price she pays. This is called an order-independent auction since the price the bidder is offered in the auction is independent of the bidder's bid value. An order-independent auction can be viewed as a function that maps $\mathbf{d}_{-i}$ to a price for each bidder.

Definition 5.1. The *order-independent auction* offers a price p_i to bidder i computed by the function f according to the order vector $\mathbf{d}_{-i}$ (i.e., $p_i = f(\mathbf{d}_{-i})$). If bidder i's bid is greater or equal to p_i ($b_i \geq p_i$), the bidder wins the auction ($x_i = 1$) and pays p_i; otherwise, the bidder loses the auction ($x_i = 0$) and pays zero.

The *order-independent auction* is truthful. Following Definition 5.1, bidder i's order does not affect the price she ends up paying, so the bidder is not able to increase her utility by changing her order. As a result, the bidder has no incentive to misreport her bid or quantity levels because this does not change the amount she pays.

Following Goldberg et al. [2006], we introduce the *optimal order-independent auction*. To define it, first we define the notion of the optimal single sale price for a set of orders.

Definition 5.2. Let $\mathbf{d}$ be a sorted vector of orders by descending values of bids. Denote $opt(\mathbf{d})$ the optimal single sale price for $\mathbf{d}$ that maximizes the revenue for the auctioneer; that is,

$$opt(\mathbf{d}) = \underset{b_i}{\operatorname{argmax}} \; b_i \sigma_i(\mathbf{d}). \tag{6}$$

Now we can define the *optimal order-independent auction*, which is a truthful auction, as follows:

Definition 5.3. The *optimal order-independent auction* is defined by the order-independent function f such that $f(\mathbf{d}_{-i}) = opt(\mathbf{d}_{-i})$.

Unfortunately, even though the optimal order-independent auction is truthful, it has two main characteristics that make it unsuitable for our purposes. First, it is not single-price (no market-wide spot price), and, second, a bidder j might lose the auction while bidder i with $b_i < b_j$ wins and is charged $p_i < b_j$. In this case, the auction's outcome is not fair, and the losing bidder envies the winning bidder's outcome. This might happen because the sale price for bidder i is computed based on $\mathbf{d}_{-i}$ which is different for each bidder. Proof of Lemma 6.2 provides examples of these outcomes.

6. ENVY-FREENESS

In an *envy-free* auction, no bidder can increase its utility by adopting another bidder's outcome. Envy-freeness has been suggested as a highly desirable property of auction mechanisms in the literature [Goldberg and Hartline 2003b; Guruswami et al. 2005], because it provides equal treatment of bidders and induces the perception of a fair allocation. If bidders are envious, the outcome of the auction might not be stable. Nonetheless, revenue maximization and client classification based on price discrimination has been suggested by other research works [Macías and Guitart 2014].

For our case, an envy-free auction requires a single sale price. All bidders willing to pay this price are provided with VM instances and are uniformly charged at that price. This is the same practice that a reputable cloud provider such as AWS uses in its spot market.

Definition 6.1. The outcome of an auction is *envy-free* if there is a single sale price p, such that every winning bidder pays p. All bidders with bid value greater than p win, whereas all bidders with value lower than p lose. Bidders with a bid value equal to p may either win or lose.

In this work, it will be irrelevant how bids that equal the sale price are treated; however, we assume that they are always provided with VM instances if the provider's capacity allows for it. Note that, according to the utility function in Equation (1), the utility value (u_i) is always zero for those bidders with true bid values (v_i) equal to p, irrespective of them winning or losing. Therefore, those bidders are assumed to have no preference over the two possible outcomes.

LEMMA 6.2. *The optimal order-independent auction is not envy-free.*

PROOF. It suffices to construct an example showing that the optimal order-independent auction is not single-price. Consider three bidders with the following orders $d_1 = (1, \$8)$, $d_2 = (2, \$7)$, and $d_3 = (4, \$2)$. To calculate the sale price for each bidder i, first we obtain $\mathbf{d_{-i}}$ by removing bidder i's order from $\mathbf{d}$. Then $opt(d_{-i})$ is computed according to Equation (6). Performing these process for all bidders, we obtain the outcome for each bidder as follows. Bidders 1 and 2 win the auction and pay \$7 and \$2, respectively, whereas bidder 3 loses the auction and pays zero. This shows that the optimal order-independent auction is not single-price.

In addition, the order-independent auction is not fair because there exists situations in which a bidder might lose the auction while another bidder with a lower bid price wins the auction. Consider four bidders with orders $d_1 = (2, \$13)$, $d_2 = (5, \$3)$, $d_3 = (1, \$2)$, and $d_4 = (20, \$1)$. Bidders 1 and 3 win the auction, and both pay Bidder 4's bid price (i.e., \$1 per instance) while Bidder 2 with a bid price higher than Bidder 3's ($\$3 > \$2$) loses the auction. □

Goldberg and Hartline [2003b] showed that no truthful, envy-free auction can be constant competitive, and they provided the lower bound of $log(n)/log\ (log(n))$ with n the number of bidders. To obtain a constant competitive auction mechanism, we relax the assumption of truthfulness and propose the Ex-CORE auction for our case. The proposed auction is envy-free but is only truthful with *high probability*.

Definition 6.3. An auction is truthful with probability $1 - \epsilon$ if the probability that any bidder can benefit from an untruthful bid is at most ϵ. If ϵ is inverse polynomial in some specified parameters of the auction (such as the number of items or bidders), then we say the mechanism is *truthful with high probability*.

In the following section, we show that the proposed auction mechanism is truthful with high probability with respect to the bid price dimension. We also provide simulation results concerning the probability that any bidder can benefit from an untruthful reporting of the number of VM instances required.

7. EXTENDED CONSENSUS REVENUE ESTIMATE AUCTION

Recall that the optimal order-independent auction in Section 5 is truthful since it is order-independent. Due to the fact that it is not single-price, and therefore not envy-free, it is not suitable for our problem context. The question therefore arises as to how a single price can be computed for an order-independent auction while attaining the revenue of the optimal auction (i.e., $\mathcal{F}(\mathbf{d})$). It is clear that $\mathcal{F}(\mathbf{d})$ cannot be computed from $\mathbf{d}_{-i}$ and, consequently, a function f that generates the optimal sale price based on $\mathbf{d}_{-i}$ cannot be built. Therefore, we are interested in a mechanism that provides us with a sufficiently accurate estimate of $\mathcal{F}(\mathbf{d})$ that is constant on $\mathbf{d}_{-i}$ for all i (i.e., it achieves *consensus*). If $\mathcal{F}(\mathbf{d}_{-i})$ is limited by a constant fraction of $\mathcal{F}(\mathbf{d})$, it is possible to pick a good estimate of $\mathcal{F}(\mathbf{d})$ such that it achieves consensus with high probability [Goldberg and Hartline 2003b]. In the remainder of this section, we outline how this estimate is computed.

In mass markets such as clouds, $\mathcal{F}(\mathbf{d})$ is much larger than the highest bid. Let $h_b(\mathbf{d})$ denote the maximum bid value in $\mathbf{d}$, then $\mathcal{F}(\mathbf{d}) \geq \alpha h_b(\mathbf{d})$ in mass markets, which implies that $\mathcal{F}$ sells at least α units.

Let m ($m \geq \alpha$) be the number of sold units in $\mathcal{F}$. If m is sufficiently large and the maximum number of units that can be requested by a customer is limited, removing an order does not change $\mathcal{F}(\mathbf{d})$ considerably. We show this in Lemma 7.1.

Enforcing a restriction on the maximum number of VM instances that can be simultaneously acquired by a customer is reasonable and done by public cloud providers such as Amazon.[5] Such restriction reduces the chance that the system stability being threatened by very large unpredicted requests. In addition, it reduces the risk of starvation for customers with small requests in the presence of wealthy customers.

LEMMA 7.1. *Let r denote the supremum of the number of requested units in $\mathbf{d}$ (i.e., $r_i \leq r$ for all bidders, $1 \leq i \leq n$). If m, the number of sold units in $\mathcal{F}$, is sufficiently large, then for any i,*

$$\frac{m-r}{m}\mathcal{F}(\mathbf{d}) \leq \mathcal{F}(\mathbf{d}_{-i}) \leq \mathcal{F}(\mathbf{d}). \tag{7}$$

PROOF. Without loss of generality, suppose $\mathbf{d}$ is sorted in descending order of bids (b_i) (i.e., $b_1 \geq b_2 \geq \cdots \geq b_n$). Suppose k is the rank of the bidder in $\mathbf{d}$ whose bid maximizes $b_i\sigma_i(\mathbf{d})$; that is, $\mathcal{F}(\mathbf{d}) = b_k\sigma_k(\mathbf{d})$. By removing order i from $\mathbf{d}$, the maximum reduction in $\mathcal{F}(\mathbf{d})$ is r_ib_k (when $i \leq k$), and the minimum reduction is zero (when $i > k$). Therefore,

$$\mathcal{F}(\mathbf{d}) - r_ib_k \leq \mathcal{F}(\mathbf{d}_{-i}) \leq \mathcal{F}(\mathbf{d}).$$

$$m = \sum_{j=1}^{k} r_j \Rightarrow b_k = \frac{\mathcal{F}(\mathbf{d})}{m},$$

$$r_i \leq r \Rightarrow r_ib_k \leq r\frac{\mathcal{F}(\mathbf{d})}{m} \Rightarrow$$

$$\frac{m-r}{m}\mathcal{F}(\mathbf{d}) \leq \mathcal{F}(\mathbf{d}_{-i}) \leq \mathcal{F}(\mathbf{d}). \quad \square$$

We introduce ρ for $\frac{m}{m-r}$. In mass markets, $\frac{1}{\rho}\mathcal{F}(\mathbf{d}) \leq \mathcal{F}(\mathbf{d}_{-i}) \leq \mathcal{F}(\mathbf{d})$, meaning that $\mathcal{F}(\mathbf{d}_{-i})$ is at least a constant fraction of $\mathcal{F}(\mathbf{d})$.

Ex-CORE combines two general ideas, as its name implies: *consensus estimation* and *revenue extraction*. For consensus estimation, it picks a function that estimates

[5]http://aws.amazon.com/ec2/faqs/#How_many_Spot_Instances_can_I_request.

$\mathcal{F}(.)$ with high quality and achieves consensus with high probability. A function that works well in our case is g, defined as $g(\mathcal{F}(.)) = \mathcal{F}(.)$ rounded down to the nearest c^{l+u}, where $c > \rho$ is a constant chosen to maximize the quality of the estimation, u is a uniform random value on [0, 1], and l is the largest integer so that $c^{l+u} \leq \mathcal{F}(.)$.

LEMMA 7.2 [GOLDBERG AND HARTLINE 2003A]. *For $c > \rho$ and any $\mathbf{d}$ with $\frac{1}{\rho}\mathcal{F}(\mathbf{d}) \leq \mathcal{F}(\mathbf{d}_{-i}) \leq \mathcal{F}(\mathbf{d})$, the probability that g outputs a value that is constant on all $\mathbf{d}_{-i}$ (i.e., achieves consensus) is* $1 - \log_c \rho$.

LEMMA 7.3 [GOLDBERG AND HARTLINE 2003A]. *If payoff for g, Υ_g, is defined as:*

$$\Upsilon_g(\mathcal{F}(.)) = \begin{cases} g(\mathcal{F}(.)), & \textit{if } g \textit{ achieves consensus;} \\ 0, & \textit{otherwise.} \end{cases} \tag{8}$$

then, for all $\mathcal{F}(.)$, we have:

$$\mathbf{E}[\Upsilon_g(\mathcal{F}(.))] = \frac{\mathcal{F}(.)}{ln(c)}\left(\frac{1}{\rho} - \frac{1}{c}\right). \tag{9}$$

Let us now discuss how to choose the value of c. We are interested in the expected payoff to be large relative to $\mathcal{F}(.)$; that is, $\mathbf{E}[\Upsilon_g(\mathcal{F}(.))]/\mathcal{F}(.)$ is large over different values of $\mathcal{F}(.)$. For a fixed value of ρ, we can choose the value of c that maximizes $\frac{1}{ln(c)}(\frac{1}{\rho} - \frac{1}{c})$. This function is differentiable on $c \in (1, \infty)$, and it has an absolute maximum on that interval. Therefore, by taking the derivative of it with respect to c and setting it to zero, we have:

$$\begin{gathered} \frac{\partial \mathbf{E}[\Upsilon_g(\mathcal{F}(.))]/\mathcal{F}(.)}{\partial c} = 0 \Rightarrow \\ \frac{\rho\, ln(c) + \rho - c}{\rho\, c^2\, ln^2(c)} = 0, \quad \rho > 1, \quad c > \rho \Rightarrow \\ \rho\, ln(c) + \rho - c = 0. \end{gathered} \tag{10}$$

Note that Equation (10) does not have an exact solution and needs to be solved by numerical methods.

The second component of Ex-CORE, a revenue extraction mechanism, extracts a target revenue from the set of bidders if this is possible. The algorithm is based on the cost-sharing mechanism proposed by Moulin and Shenker [2001]]. Given an order vector $\mathbf{d}$ sorted in descending order of bids and a target revenue R, the revenue extractor function $e_R(\mathbf{d})$ finds the largest k such that $R/\sigma_k(\mathbf{d}) \leq b_k$. In other words, it finds the k bidders with the highest bid values that allow for the extraction of R. R is then shared among these k bidders based on the number of requested instances by each bidder; that is, each of these bidders are charged $R/\sigma_k(\mathbf{d})$. If no subset of bidders can share R, the auction has no winners.

Figure 2 illustrates an example of a revenue extraction mechanism for the target revenue of \$18 (i.e., $R = \$18$) when four bidders with the following orders are available: $d_1 = (1, \$8)$, $d_2 = (2, \$7)$, $d_3 = (1, \$5)$, and $d_4 = (4, \$2)$. We verify $18/\sigma_k(\mathbf{d}) \leq b_k$ for all values of $k = 1, 2, 3, 4$. For $k = 2$ and $k = 3$, the revenue of R can be extracted by k winning orders with the highest bid values (i.e., $18/(1+2) \leq 7$ and $18/(1+2+1) \leq 5$)). But this cannot be done for other values of k (i.e., $18/(1) > 8$ and $18/(1+2+1+4) > 2$ for $k = 1$ and $k = 4$, respectively). Therefore, based on the revenue extraction mechanism, the highest value of k satisfying the given relation ($k = 3$) is chosen, and the single sale price is set to $18/4 = \$2.5$.

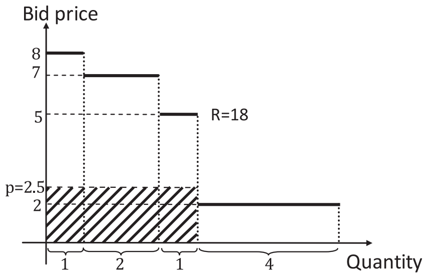
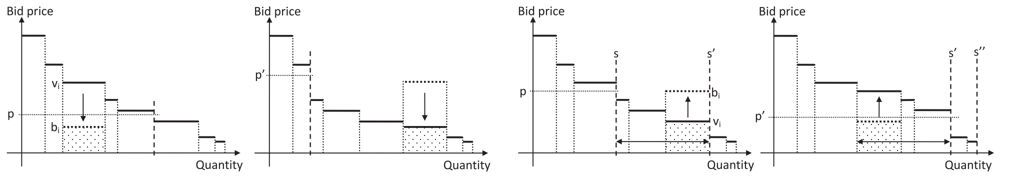


Fig. 2. Example of the revenue extraction mechanism for a target revenue and a sample vector of orders.



(a) Reporting $b_i < v_i$ increases the price to $p' > v_i$. (b) Reporting $b_i > v_i$ decreases the price to $p' > v_i$.

Fig. 3. Effect of misreporting true value on the sale price. Truthful submission leads to (a) winning and (b) losing.

LEMMA 7.4. *Given a target revenue R, the revenue extraction mechanism is truthful for the price dimension but not for the quantity dimension.*

PROOF. Without loss of generality, we consider $\mathbf{d}$ as sorted. The revenue extraction mechanism is truthful if $u_i(q_i, v_i) \geq u_i(r_i, b_i)$ for all values of b_i and r_i and for every bidder i, $1 \leq i \leq n$.

First, we show that given a fixed r_i any untruthful submission of the bid price (i.e., $b_i \neq v_i$) decreases Bidder $i's$ utility. It suffices to consider the following two cases:

Case 1: Suppose the truthful submission ($v_i = b_i$) leads to Bidder i winning the auction, it is easy to verify that reporting $b_i > v_i$ only decreases the rank of Bidder i in $\mathbf{d}$, assuming $\mathbf{d}$ remains unchanged except for Bidder i. Therefore, it does not change the sale price, and, as a result, Bidder i's utility also remains unchanged.

If Bidder i reports $b_i < v_i$, as long as $b_i \geq p$ (p is the sale price), p remains unchanged. Hence, Bidder i's utility does not increase or decrease. However, as soon as $b_i < p$, Bidder i loses the auction, the sale price rises, and the bidder's utility drops to zero. This is illustrated in Figure 3(a). Consequently, submitting $b_i < v_i$ might not improve Bidder i's utility and might reduce it to zero.

Case 2: Suppose the truthful submission ($v_i = b_i$) leads to the bidder losing the auction; then, reporting $b_i < v_i$ would clearly not change the zero utility of the bidder.

If reporting $b_i = v_i$ leads to Bidder i losing the auction, it follows that $p > v_i$. Assume $p = R/s$, where s is the sum of the number of requested units by largest group of k bidders with the highest bid values that can at least generate a revenue of R. Consider $s' = \sigma_i(\mathbf{d})$, as a result $s' > s$, since we know b_i is a losing bid.

Suppose Bidder i reports her bid $b_i > v_i$, we argue that new sale price p' is always larger than v_i ($p' > v_i$). That is, increasing b_i might increase s up to s' at most. This is shown in Figure 3(b).

Using reductio ad absurdum, assume by increasing b_i, s can be increased to a value $s'' > s'$. Hence, we know that there is a bidder j whose bid price, b_j, is larger than R/s'' ($R/s'' \leq b_j$). We know that $i < j$ and $b_j \leq v_i$ because $s'' > s'$ requires j to be placed after i in the sorted vector. If $R \leq s''b_j$ after increasing b_i, then $R \leq s''b_j$ before increasing as well, because Bidder i is placed in the lower rank either bidding at $b_i = v_i$ or $b_i > v_i$ in the sorted vector of orders. That is, Bidder i is a winner in either of cases. This contradicts our initial assumption that reporting $b_i = v_i$ leads to Bidder i losing the auction. So, $s'' \leq s' \Rightarrow p' > v_i$.

Hence, bidding $b_i > v_i$ leads to negative utility for Bidder i, and Bidder i would be worse off.

Second, we provide an example that demonstrates that the revenue extraction mechanism is not truthful for the quantity dimension. That is, bidders are able to increase their utility by misreporting their required number of instances. Assume $R = \$7$ and an order vector $\mathbf{d} = \{(1, \$8), (5, \$1)\}$. Bidder 1 is charged $7/1 = 7$ and Bidder 2 loses according to the revenue extraction mechanism. The utility for Bidder 1 is then computed as follows: $1 \times 8 - 1 \times 7 = 1$. Now, consider that Bidder 1 misreports 2 as the required number of instances. Then, the largest group of bidders able to share R includes the orders of both bidders. Therefore, the price for Bidder 1 is $7/7 = 1$, and its utility is computed as: $1 \times 8 - 2 \times 1 = 6$. □

Definition 7.5. Extended Consensus Revenue Estimate Auction (Ex-CORE): For constant c, and a random value u uniformly chosen from $[0, 1]$ find $g(.)$ as $\mathcal{F}(.)$ rounded down to nearest c^{l+u} for integer l. The sale price by Ex-CORE is then defined as $p = e_R(\mathbf{d})$ where $R = g(\mathcal{F}(\mathbf{d}))$.

LEMMA 7.6. *For order vector* $\mathbf{d}$*, constant* c*, and a choice of* u*, if* $g(\mathcal{F}(\mathbf{d}_{-i})) = R$ *for all* i*,* $1 \leq i \leq n$ *(i.e., it is a consensus), then the Ex-CORE auction is truthful with respect to bid prices.*

PROOF. It suffices to show that if $g(\mathcal{F}(\mathbf{d}_{-i})) = R$ for all i, no bidder can increase her utility by bidding any value other than the true bid value. Note that if $g(\mathcal{F}(\mathbf{d}_{-i})) = R$ for all i, then $g(\mathcal{F}(\mathbf{d})) = R$. Now consider that Bidder i submits an order (r_i, b_i) where $b_i \neq v_i$ resulting in $\mathbf{d}'$ ($\mathbf{d}'$ is identical to $\mathbf{d}$ except for Bidder i's bid price).

As long as $g(\mathcal{F}(\mathbf{d}')) = g(\mathcal{F}(\mathbf{d})) = R$, Bidder i is not able to benefit from misreporting. Because the sale price p is computed as $p = e_R(\mathbf{d})$, and according to Lemma 7.4, the revenue extraction mechanism is truthful. Therefore, Bidder i's utility cannot be improved by misreporting v_i; thus, Bidder i's best strategy is to bid at v_i.

The proof is in fact very straightforward. For every user i, since $\mathcal{F}(\mathbf{d}_{-i}) = \mathcal{F}(\mathbf{d})$, changing bid b_i to b_i' will lead to a new order vector $\mathbf{d}'$ the same as the original $\mathbf{d}$ except for component i. As a result, $\mathbf{d}'_{-i} = \mathbf{d}_{-i}$. Hence, $g(\mathcal{F}(\mathbf{d}'_{-i})) = g(\mathcal{F}(\mathbf{d}_{-i})) = g(\mathcal{F}(\mathbf{d})) = R$. This essentially implies that user i will be given exactly the same price as before. Consequently, the sale price cannot be decreased, and Bidder i's utility cannot be increased. □

PROPOSITION 2. *Ex-CORE is envy-free, truthful with probability* $1 - \log_c \rho$ *for the bid price dimension, and* $\frac{1}{ln(c)}(\frac{1}{\rho} - \frac{1}{c})$*-competitive for mass markets.*

PROOF. Definition 7.5 and Lemmas 7.2, 7.3, and 7.6 are enough to prove the proposition. □

7.1. Discussion

The Ex-CORE auction is not two-dimensionally truthful because the revenue extraction mechanism is not truthful for the quantity dimension (Lemma 7.4). In the cloud spot market, however, no customer has an incentive to request fewer instances than needed (because $u_i(r_i, b_i) = 0$ whenever $r_i < q_i$). Our detailed investigation of the proposed mechanism shows that bidders are able to increase their utility in some cases by requesting a higher number of instances than what they actually require. Devising a two-dimensional truthful mechanism for this potentially highly complex strategy space remains as a future work. Nevertheless, we believe that the proposed mechanism retains its practical value due to several key reasons.

First, users who misreport the required number of instances end up paying for a higher number of instances. In order to increase utility, the increment in r_i must cause a sufficient reduction in the market price to compensate for the surplus cost a bidder pays for the additional instances. Considering that the bidder is not aware of the other orders, there is always a risk of decrease in utility by misreporting.

Second, $\mathcal{F}(\mathbf{d})$ is monotonically increasing with respect to r_i (the rationale is intuitive), and Ex-CORE calculates the sale price based on the estimation of $\mathcal{F}(\mathbf{d})$. Given that r (the maximum number of requested instances) is a constant and $m \to \infty$ in a cloud mass market, in expectation, R (the estimated value of $\mathcal{F}(\mathbf{d})$) rises as the bidder increases demand. The revenue extraction mechanism computes the price by $R/\sigma_k(\mathbf{d})$. Therefore, the risk of increasing the market price increases by misreporting the number of required instances because the numerator of the fraction (R) is increasing while there is no certainty about the decrease or increase in the denominator ($\sigma_k(\mathbf{d})$).

Last, but not the least, r is constrained by a limit. Because the number of sold instances in the cloud market is usually high, the effect on the market price of a bidder misreporting demand is typically low given the assumption of *noncollusive* behavior of bidders. In Section 10, we demonstrate through simulation that, in markets of sufficient size, an individual bidder indeed has a very low probability of gaining utility by misreporting VM demand.

Even though the Ex-CORE auction is designed to be truthful with high probability, we assume that there is no *collusion* among the bidders, in which a group of bidders collaborate to gain an unfair market advantage or to limit fair competition. In practice, collusion, at least on a small scale, is a possibility, and mechanism designers would like to ensure that their auction mechanisms are at least somewhat *collusion-resistant*. In our model, even though Ex-CORE is not designed to be collusion-resistant, constraining r by a limit provides resistance against small-scale collusion (i.e., a large group of bidders are required to collude to significantly impact the market price). There is a large body of literature on the design of collusion-resistant auction mechanisms (e.g., Goldberg and Hartline [2005]), and, as a future direction for this work, we are interested in investigating a collusion-resistant property for our proposed mechanism.

8. LIMITED SUPPLY AND RESERVE PRICE

Up to this point, we have considered an unlimited capacity setting. In reality, however, situations arise wherein a cloud provider needs to reject requests due to lack of supply. We modify the auction mechanisms to take into account that $C(t)$, the number of VM instances available for sale at time t, can be lower than the demand.

Because the provider wishes to maximize revenue, it can select a set of high-value bidders such that the total amount of requested VMs by this set is smaller or equal to $C(t)$. This set of bidders subsequently participates in the auction mechanism for the limited supply case, while the remaining orders are rejected. Figure 4 depicts how supply is limited by $C(t)$. This method allows us to extend our discussion into the bounded supply case. In order to be envy-free in the bounded supply case, we need to ensure that none of the bidders wins at a price lower than the highest losing bid order. Therefore, we ensure that $p = max(b_{lost}, p)$, with b_{lost} the highest losing bid.

If a bidder accepts partial fulfillment of an order, the fraction of required instances that fits in the provider's available capacity can be allocated. When multiple winning consumers are subject to such partial delivery, ties can be broken randomly.

We believe that allowing partial fulfillment of requests for the case of limited supply would not significantly affect the *truthfulness with high probability* and *near-optimal profit* properties of the proposed mechanism. In practice, for a large cloud provider,

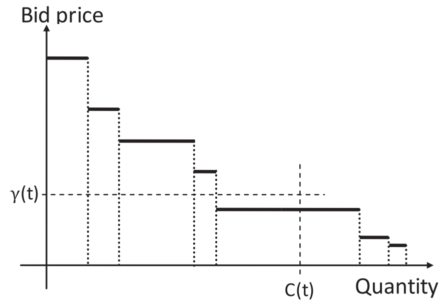


Fig. 4. Supply limited by capacity and reserve price at time t.

the effect is significantly minor and negligible because there is always a huge amount of customers requesting virtual instances ($m \rightarrow \infty$), and the maximum number of instances that can be requested by a customer is limited and much lower than m (i.e., $m \gg r$). In fact, even for cases where there are not too many customers bidding in the recurrent auction, the proposed mechanism still generates near-optimal profit for the cloud provider. We later verify this point via extensive simulations in Section 10.

8.1. Reserve Price

If profit instead of revenue is of concern, the provider needs to take its costs for delivering a VM instance into account. Let $\gamma(t)$, the *reserve price* at time t, be the lowest possible price that the provider accepts for one slot of usage of a VM instance at time t; orders with bids below this level are ignored by the auction. In this section, we propose a method for a provider to compute $\gamma(t)$. Figure 4 depicts how the order vector is shaped by $\gamma(t)$.

The reserve price for most perishable goods and services is considerably low at their expiration time. For instance, the reserve price for flight seats is theoretically negligible; as soon as boarding is closed on a particular flight, all the unsold seats on that flight are completely wasted. Thus, selling a remaining seat at a reasonable low price is often a better option compared to wasting the seat capacity without generating any revenue.

However, there is a fundamental difference between cloud resources and other perishable goods and services. A significant part of the service cost in cloud data centers is related to power consumption of physical servers. The cost of power drawn by servers and associated cooling systems is comparable to the amortized capital investments for purchasing the servers themselves [Patterson 2008]. Thus, when considering the perishable nature of VM services, taking into account the *marginal cost* of instantiating a VM is important in this case.[6]

The overall cost of the data center, $C_{overall}$, can be divided into capital and operational costs, $C_{overall} = C_{cap} + C_{opr}$. The parameter C_{cap} includes upfront investments and all one-time expenses that are depreciated over the lifetime of the data center (e.g., those related to the purchase of land, buildings, construction, buying physical servers and software, installing power delivery, and cooling infrastructures etc.). C_{opr} includes electricity costs, staff salaries, and ISP costs. Operational costs can further be categorized as being *fixed* or *variable*, $C_{opr} = C_{opr_{fixed}} + C_{opr_{var}}$. The parameter $C_{opr_{fixed}}$ includes costs that remain identical whether the data center is operating at full capacity or not (e.g.,

[6]In economics, the marginal cost is the change in total cost that arises as a result of one additional unit of production.

staff salaries). However, components of $C_{opr_{var}}$ may increase or decrease depending on data center utilization (e.g., electricity costs).

The provider is not able to avoid incurring C_{cap} and $C_{opr_{fixed}}$, whereas $C_{opr_{var}}$ can be avoided to a large extent. $C_{opr_{var}}$ over any specific time period is dominated by the cost of power consumption, C_{pwr}, and can be strongly approximated by it ($C_{opr_{var}} \approx C_{pwr}$).

Cloud providers are able to measure instant power consumption in the data center. Knowing the power consumption and electricity prices, C_{pwr} can be easily calculated. We argue that the cloud provider should define the reserve price in a way that accommodating a VM with a specific bid must at least generate sufficient revenue to offset the contribution of VM to $C_{opr_{var}}$. Assuming all VMs are of the same type, $\gamma(t)$ can be derived as follows:

$$\gamma(t) = C_{pwr}/VM_n(t), \tag{11}$$

where $VM_n(t)$ is the number of running VMs in the data center at time t, and C_{pwr} is the cost of power consumption at that time. Knowing the electricity price, φ, and total data center power consumption $Power_{total}$, C_{pwr} can be computed as $C_{pwr} = Power_{total} \times \varphi$. Because $\gamma(t)$ is primarily affected by factors such as IT load, electricity price, data center outside air temperature, and humidity [Pelley et al. 2009], it should vary dynamically.

Because we need to resort to simulation for the experimental performance evaluation of our proposed solution, we require a model for C_{pwr}. Detailed modeling of data center power usage is, however, difficult because of the complexity and diversity of the infrastructure [Pelley et al. 2009]. Consequently, we propose an abstract model based on the concept of Power Usage Effectiveness (*PUE*).

8.2. Power Usage Efficiency Model

PUE is a measure of how efficiently a data center consumes its power. It is computed as the ratio of total data center power consumption, $Power_{total}$, to IT load power, $Power_{IT}$ (i.e., power consumed by servers, storage, and network equipment):

$$PUE = Power_{total}/Power_{IT}. \tag{12}$$

PUE measures the power overhead consumed in supporting the IT load. The overhead is caused by cooling and humidification systems (e.g., chiller), power distribution (e.g., PDU), power conditioning system (e.g., UPS), and lighting. Ideally $PUE = 1$. Inefficient data centers have a *PUE* of 2.0 to 3.0, whereas *PUE* scores lower than 1.14 are advertised by leading companies such as Facebook and Google [Greenberg et al. 2008]. *PUE* reported in this way is usually an average value over a specific period (e.g., one year), whereas instant *PUE* is not a constant value. The efficiency of the data center varies over time depending on changes in the data center conditions.

One of the most important conditions is the outside ambient temperature [Rasmussen 2011] because the energy required to remove heat generated within the data center grows with it [Pelley et al. 2009]. To some degree, outside air humidity affects cooling power as well, but we do not consider it in this work in order to limit model complexity.

A second important condition that changes over time and affects *PUE* is the IT load. This follows from the fact that the efficiency of power conditioning system and cooling equipment increases under higher load [Greenberg et al. 2006]. We represent IT load by the percentage of ON physical servers in the data center (referred to as *data center utilization*). We model *PUE* as a function of load and outside ambient temperature (i.e., $PUE = f(load, temp)$). To simplify the model, we assume that every server in the data center on average consumes a constant rate of power if it is ON, and none otherwise. $Power_{IT}$, is therefore computed according to Equation (13):

$$Power_{IT} = N_{Srv-ON} \times Power_{Srv}, \tag{13}$$

ALGORITHM 1: The Online Ex-CORE Auction

```
Input: $\mathbf{d}$, $p_{cur}$, $p_{optprv}$    ▷ $\mathbf{d}$ is the list of orders, sorted in descending order of bids, $p_{cur}$ is current market price, $p_{optprv}$ is the optimal single price in the previous round.
Output: $p$    ▷ sale price
1  $p_{opt} \leftarrow opt(\mathbf{d})$;
2  if $p_{opt} = p_{optprv}$ then
3      return $p_{cur}$;
4  end
5  $r \leftarrow$ the largest $r_i$ in $\mathbf{d}$;
6  $m \leftarrow \underset{\sigma_i(\mathbf{d})}{\text{argmax}}\ b_i \sigma_i(\mathbf{d})$ ;
7  if $m \leq r$ then
8      return $p_{opt}$;    ▷ single optimal price
9  else
10     $\rho \leftarrow \frac{m}{m-r}$;
11     Find $c$ in $\rho\, ln(c) + \rho - c = 0$;
12     $u \leftarrow rnd(0, 1)$;    ▷ chosen uniformly random on [0,1]
13     $l \leftarrow \lfloor log_c(\mathcal{F}(\mathbf{d})) - u \rfloor$;
14     $R \leftarrow c^{(l+u)}$;
15     $j \leftarrow$ the largest $k$ such that $\frac{R}{\sigma_k(\mathbf{d})} \geq b_k$;
16     return $\frac{R}{\sigma_j(\mathbf{d})}$;
17 end
```

where N_{Srv-ON} is the number of non-idle servers in the data center, and $Power_{srv}$ is the average rate of power consumption by servers. The contribution of networking equipment in Equation (13) is not taken into account because it is small and its power draw does not vary significantly with data center load [Pelley et al. 2009].

In this study, we assume that the provider commits to provide the actual amount of resources required by a VM regardless of the actual resource usage pattern of the applications it executes. Moreover, we assume the cloud provider periodically packs the data center's workload into a minimum number of servers, powering off any inactive ones.

9. AUCTION MECHANISMS AND BENCHMARKS

In this section, we review the different auction mechanisms that are included in our experimental evaluation.

Optimal Single Price Auction (OPT): The extractable revenue in a single-round, single-price auction is at most $\mathcal{F}(\mathbf{d})$, which can be achieved by an optimal price choice. Since we are interested in maximizing the provider's revenue, we use the OPT described in Definition 4.1 as a benchmark. In the online version of OPT, the auction is executed upon every arrival of an order or termination of an instance.

Online Extended Consensus Revenue Estimate Auction (Online Ex-CORE): Details of the Ex-CORE auction can be found in Section 7. Our online version of Ex-CORE (outlined in Algorithm 1) records the optimal sale price computed by OPT in the previous round of auction and updates the sale price using the Ex-CORE algorithm only when the optimal sale price calculated in the current round differs from the one in the previous round of the auction (Lines 1–3). This prevents the market from being exposed to a high number of price fluctuations due to randomness in the Ex-CORE algorithm. Note that this does not violate a possibly existing consensus established in the previous round of the Ex-CORE auction because arriving or leaving orders have not changed the optimal price.

Lines 5 and 6 compute r, the maximum number of requested units in the order list, and m, the maximum number of units sold by OPT. Because our mechanism is designed to work for mass-market scenarios, it requires m to be larger than r ($m \gg r$). In the rare event when this condition would not hold, the algorithm returns the price computed by OPT.

On Line 10, ρ is computed, followed by the computation of the optimal value for c, for which we use Newton-Raphson method. Subsequently, c is used to generate an estimation of $\mathcal{F}(.)$ that achieves the consensus with high probability (Lines 12–14). Finally, the estimated value is converted to the market clearing price through the revenue extraction mechanism.

Holding Time Aware Optimal Auction (HTA-OPT): Due to a lack of prior knowledge on the holding time of VMs, the online version of the Ex-CORE auction operates in a greedy manner because it attempts to maximize revenue given the newly arriving order and the existing orders at a given time. In order to quantify the efficiency loss caused by this lack of information, we use HTA-OPT as a benchmark algorithm that uses prior knowledge on VM holding times. HTA-OPT takes into account the fact that an order with a long holding time and a low bid can potentially generate more revenue than a short order with a high bid.

Algorithm 2 calculates the optimal sale price using dynamic programming. The price is computed based on the maximum possible revenue that can be generated by current orders in the system and the corresponding remaining time of these orders. The main reasoning is that if the algorithm sets the price at a specific bid price, all orders with bid prices lower than that price are not available for the next time slot. Assuming bidders are charged on an hourly basis of VM usage, we express duration similarly in an hourly basis. Each partial hour is considered as a full hour (e.g., 2.5 hours is considered as 3 hours of usage).

Algorithm 2 has the following input arguments: the list of orders **d**, sorted in descending order of bids; an order index i set to the number of orders in the first call of the function; a time slot index t set to 1 for the first call; and a boolean argument $firstCall$ indicating that it is the first call to the function. Lines 5–15 initialize the revenue array rev such that each element in rev is set to the revenue that can be generated in that time slot, provided that the price is set to a corresponding bid price. Line 17 ends the recursion when the termination conditions are reached.

In Lines 24–32, the algorithm chooses the most profitable path given two choices for dealing with order i at time t. This is done by recursively computing the total revenue in case the market price is kept below the order's bid at time t ($ans1$) and computing the revenue in case the decision is made to let the market price exceed the order's bid at t ($ans2$).

The most profitable decision path is stored in the dp array with the aggregated revenue for the checked paths. Finally, we find the highest possible revenue within the first column of dp and return the corresponding price. We break ties by favoring the market price with the lowest transaction volume.

The HTA-OPT algorithm determines the optimal prices because both cases of acceptance or rejection of all orders for each time slot are verified by the algorithm. Hence, a formal proof seems to be unnecessary.

Uniform Price Auction: In the uniform price auction, the provider serves the highest bidder first by allocating the requested number of instances. This is followed by an allocation for the second highest bidder and so forth until supply is exhausted or there are no more orders. All bidders are charged with the lowest winning bid.

10. PERFORMANCE EVALUATION

Our evaluation is based on both *simulation* and a *real-world experimental study*. Our simulation-based evaluation of the proposed auction framework includes three parts and is conducted via an extension of CloudSim [Calheiros et al. 2011] with support for our auction framework. First, we simulate Ex-CORE in a single-round, unlimited

ALGORITHM 2: Holding Time-Aware Optimal Auction

```
Input: $\mathbf{d}, i, t, firstCall$
Output: $p$                                                        ▷ Sale Price
1  $maxDuration \leftarrow \max_i(\text{DURATION}(d_i))$;  ▷ The duration function returns the time remaining
                                                     from the holding time of the orders in unit of hours.
2  $dp[|\mathbf{d}|][maxDuration] \leftarrow \{-1\}$;
3  $rev[|\mathbf{d}|][maxDuration] \leftarrow \{0\}$;   ▷ Create $dp$ and $rev$ arrays with $|\mathbf{d}|$ (size of $\mathbf{d}$) rows and
                                                     $maxDuration$ columns, and initialize all cells with $-1$
                                                     and 0 respectively.
4  function HTA-OPT($\mathbf{d}, i, t, firstCall$)
5      if $firstCall$ then
6          for $j \leftarrow 1$ to $maxDuration$ do
7              $prvCount \leftarrow 0$;
8              for $k \leftarrow 1$ to $|\mathbf{d}|$ do
9                  if $j \le \text{DURATION}(d_k)$ then
10                     $rev[k][j] = d_k \times (r_k + prvCount)$;
11                     $prvCount \leftarrow prvCount + r_k$;
12                 end
13             end
14         end
15     end
16     if $i = 0$ or $t > \text{DURATION}(d_i)$ then
17         return 0;
18     end
19     if $dp[i][t] = -1$ then
20         $ans1 \leftarrow 0, ans2 \leftarrow 0$;
21         if $t \le \text{DURATION}(d_i)$ then
22             $ans1 \leftarrow \text{HTA-OPT}(\mathbf{d}, i, t+1, false) + rev[i][t]$;
23         end
24         if $i \ge 1$ then
25             $ans2 \leftarrow \text{HTA-OPT}(\mathbf{d}, i-1, t, false)$;
26         end
27         if $ans1 > ans2$ then
28             $dp[i][t] \leftarrow ans1$;
29         else
30             $dp[i][t] \leftarrow ans2$ ;
31         end
32     end
33     if $firstCall$ then
34         $k \leftarrow \text{argmax}_i(dp[i][0])$;      ▷ In case of ties, pick lowest $i$, i.e., higher price and
                                                     selling less instances.
35         return $b_k$;
36     end
37     return $dp[i][t]$;
38 end
```

supply setting using several order distributions. In the second part, the impact of misreporting the number of required instances on the utility obtained by an individual bidder is explored. The last part evaluates the auction framework under bounded supply. Auctions then occur recurrently by arriving and finishing orders, and the marginal cost of VM production changes dynamically over time.

An experimental study is also conducted to evaluate a system prototype of the proposed auction framework. The study is performed with a group of 10 people competing in a spot market to acquire VM instances. The results of the experimental study confirms the truthfulness property of the proposed auction mechanism.

10.1. Single Round Evaluation

10.1.1. Order Generation. Due to the lack of real-world data on bidder valuations and order sizing, we need to resort to a synthetic generation of orders. In line with Goldberg et al. [2006], we adopt the following four distributions for the generation of bids:

(1) **uniform (l, h):** Bid prices are drawn from a uniform distribution bounded by l and h.
(2) **normal (μ, σ):** Bid prices are drawn from a normal distribution with mean μ and standard deviation σ. Bids less than or equal to zero are discarded, and a new bid is drawn from the distribution. This causes the normal distribution to be skewed as zero, and negative bid values are not permitted.
(3) **Zipf (h, θ):** Bid prices are drawn from a Zipf distribution with parameters h as the highest bid price and parameter θ. This distribution is a generalization of the Pareto principle that 80% of the total bid value originates from 20% of the bidders.
(4) **bipolar (l, h):** Bid prices are generated by randomly choosing either l or h with equal probability.

For requested number of instances in each order, we consider three types of distributions:

(1) **constant (ζ):** The number of instances for all orders equals $\zeta \leq r$ where r is the supremum on the number of requested units.
(2) **uniform (l, h):** The number of instances for an order is drawn from a uniform distribution between $l = 1$ and $h = r$.
(3) **normal (μ, σ):** The number of instances for an order is drawn from a normal distribution as a discrete value with mean μ and standard deviation σ. Values smaller than 1 or larger than r are discarded.

We investigate the generated revenue for different combinations of distributions for the bid price and the number of requested units per order. We set parameters of the distributions such that the quantity falls in the range [1,50] and bid prices are in the range [1,60], where the value 60 is derived from the Amazon EC2 price of on-demand small instances for the eastern US region (in 0.1 cents) at the time of writing.[7] The number of orders varies between 10 and 100,000, and the ratio of generated revenue by Ex-CORE (R) to $\mathcal{F}$ is reported ($R/\mathcal{F}$). Each experiment is carried out 30 times, and the mean value of $R/\mathcal{F}$ is reported. Figure 5 shows the simulation results when $r = 50$. As the number of orders increases, $R/\mathcal{F}$ approaches 1 regardless of the distribution used for order generation, as we expected. Although there is a small difference between the revenue obtained by Ex-CORE for different distributions, as shown in Figure 5, the distribution of orders does not have a significant effect on the generated revenue, especially when the number of orders in the market is large. Figure 6 shows separate box plots of $R/\mathcal{F}$ for different order distributions when the number of orders equals 100. Statistical analysis certifies that the performance does not change significantly under different order distributions. By design, Ex-CORE does not require a priori knowledge about the order distribution. Therefore, the provider does not need to rely on frequent investigation and monitoring of changes in market conditions to maximize its revenue.

A sensitivity analysis with respect to r showed that its value does significantly impact the results as long as r is sufficiently small compared to the total demand volume and

[7]http://aws.amazon.com/ec2/pricing/.

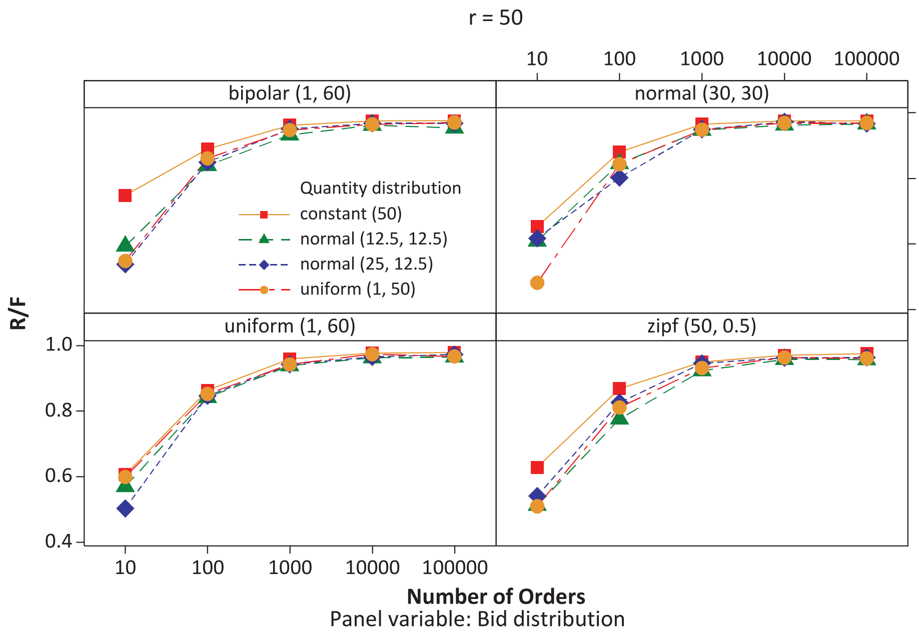


Fig. 5. Ratio of gained revenue by the Ex-CORE auction to optimal auction under different distributions of orders.

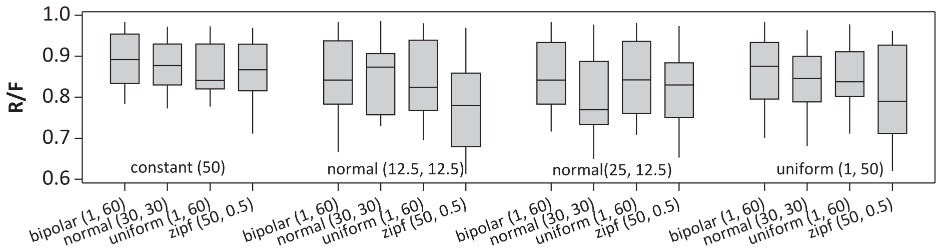


Fig. 6. Ratio of gained revenue by the Ex-CORE auction to optimal auction under different distributions of orders when number of orders is 100.

the total supply volume in the market is sufficiently large. This can be easily justified by Lemma 7.1. For brevity, we omit a discussion on these experiments.

10.2. Evaluation of Misreporting Quantity

Because the Ex-CORE mechanism is not two-dimensionally truthful, we investigate the potential for a bidder to gain utility by misreporting the number of required units in her order. First, we generate lists of orders with the same settings used for Experiment 1 and assume each generated order to be truthful. The utility obtained by every bidder is subsequently calculated according to Equation (1).

Assuming there is no collusion among bidders, we increase the requested number of instances (r_i) for an individual bidder i up to the maximum number of instances that can be requested (r) by the step size of 1 while keeping the orders of the other bidders unchanged. For every stepwise increase, we calculate the bidder's utility and compare it with the utility attained under a truthful report. We then compute the probability of a bidder increasing her utility through a misreport of quantity by dividing the number of cases in which utility increased by the total number of steps.

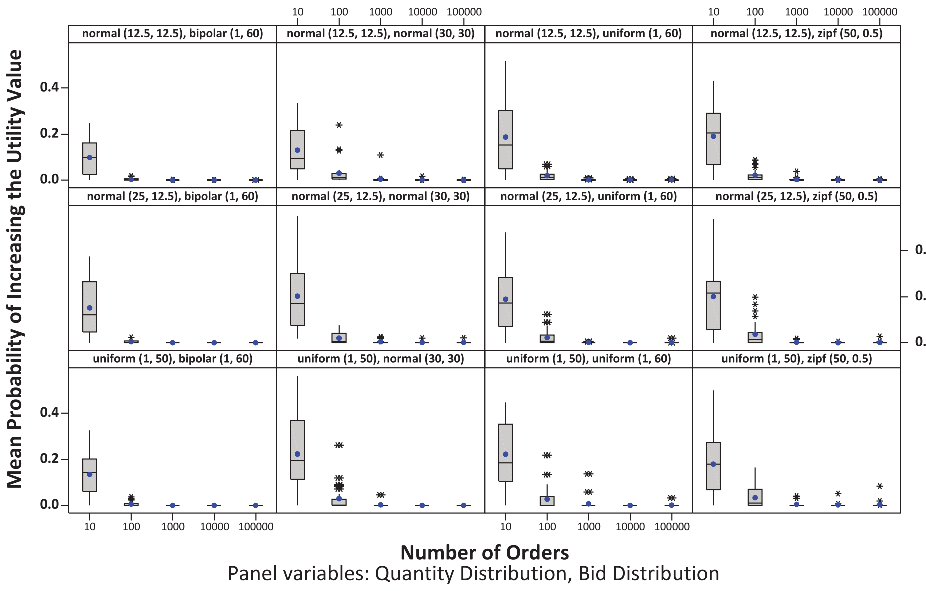


Fig. 7. Mean probability of increase in the utility value for bidders by Ex-CORE under different distributions of orders when $r = 50$. A blue dot denotes the mean value.

The experiment is repeated for every bidder with the same random number seed for each step and is subsequently carried out 30 times with different seeds. Figure 7 shows the mean and box plot of the mean probability of increase in the utility by misreporting the quantity after 30 runs under different order distributions. The constant distribution of the requested units is removed from the set of quantity distributions because there is no opportunity to change r_i. As one can observe in the figure, the probability of gaining utility through misreports converges to zero as the market size grows under all order distributions. Moreover, there is no predictable pattern for the user to increase the utility value due to the implicit random component of Ex-CORE and the lack of knowledge about other bidders. Figure 8 shows the maximum increase of the utility value among all bidders for the same experiment.

10.3. Online Auction Framework Evaluation

We evaluate the profit of the online Ex-CORE auction through simulation. We consider the case where capacity is bounded at $C = 8 \times 10^4$ throughout the simulation. In real-world scenarios, a provider may offer several pricing plans (e.g., on-demand, reserved pricing plans) or different types of VMs (e.g., small, medium, large). Under such circumstances, the capacity allocated for a specific VM type in the auction market can be dynamically adjusted to maximize profit [Wang et al. 2012; Zhang et al. 2011]. Our proposed auction mechanism can be run separately for each type of VMs.

We simulate the market for 24 hours. Customers submit their orders independently following a Poisson process, with λ set at the total number of requests in the whole simulation divided by 24. Because the distribution of the bid prices and the quantity of requested units do not significantly impact the revenue results, we use uniform distributions for both.

Bid prices in dollars are drawn from a uniform distribution on [0, 0.06]. The maximum bid price is derived from the Amazon EC2 price of on-demand small instances in the eastern US region. Considering the lower QoS for VMs in the spot market and the

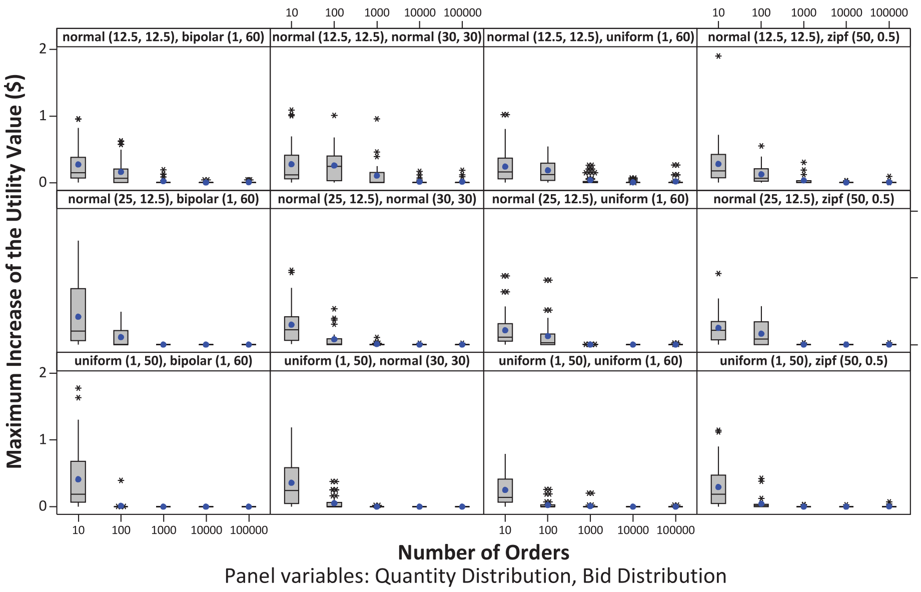


Fig. 8. Maximum increase of the utility value among all bidders, achieved through misreporting quantity in Ex-CORE auction under different distributions of orders when $r = 50$. A blue dot denotes the mean value.

truthfulness property of the mechanism, bidding higher than the on-demand price seems unreasonable. However, in real-world scenarios, there might be orders with a bid higher than the on-demand price, as observed on the EC2 spot market. This is not of concern in our model.

The requested number of instances per order for each bidder is modeled by i.i.d. random variables uniformly distributed on [1, 50]. Amazon EC2 similarly imposes a limit of 100 VM instances per region that can be acquired by a customer in the spot market.[8]

Following Mills et al. [2011], the holding time of the VM instances by users is modeled by i.i.d Pareto distributed random variables, with shape parameter 1 and location parameter 1. Each generated random value represents the time in hours that VM instances remain in the system. If the order's bid price is lower than the current market price, the order remains in the queue for a maximum time period of half an hour. The order is considered in every auction round during this period. If the order is not serviced at the end of this period, it is labeled as rejected. The VMs that are instantiated following the acceptance of an order can be terminated at any time if the market price exceeds the order's bid. Upon termination, these VMs are not charged for their last partial hour. VMs that are terminated by their owner are charged for a discrete number of hours, with a partial hour of usage accounted for as a full hour.

To model the marginal cost of running VMs, we assume that the data center is populated with BL460c G6 blade servers that host a quad-core Intel Xeon E5504 2.0GHz processor. The average power usage per blade server is rated at 400W. Using the Amazon EC2 small instances type, each server is able to host up to eight VMs. Two sets of electricity prices are considered for the data center, one for "on-peak" hours from 7 a.m. to 9 p.m. and another for "off-peak" hours from 9 p.m. to 7 a.m. Following work by Le et al. [2011], we adopt a peak price of 0.108$/KWh and an off-peak price reduction of 50%.

[8]http://aws.amazon.com/ec2/spot-instances/.

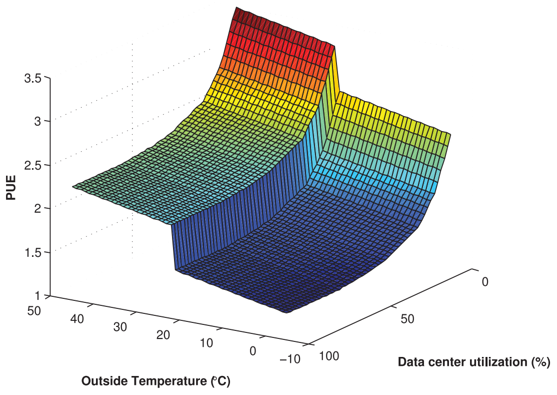


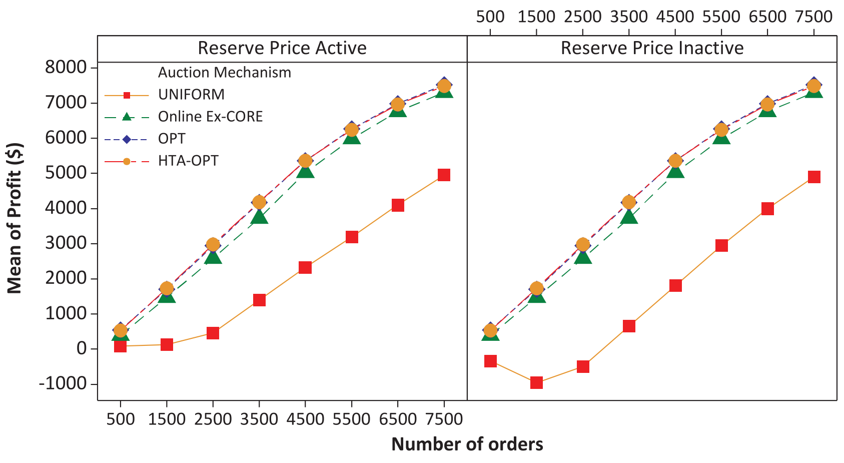
Fig. 9. *PUE* as related to load and outside temperature.

We compute *PUE* based on data center load and outside air temperature. Figure 9 illustrates our modeling of the PUE as a function of outside temperature. This is an interpolation of the models by Goiri et al. [2011] and Rasmussen [2011]. *PUE* increases drastically due to chiller activation when outside temperature rises above $20^{\circ}C$ [Goiri et al. 2011]. We consider a relatively warm day with minimum temperature of $14^{\circ}C$ and maximum temperature of $33^{\circ}C$. We estimate hourly temperatures throughout the day based on a method by Campbell and Norman [2012].

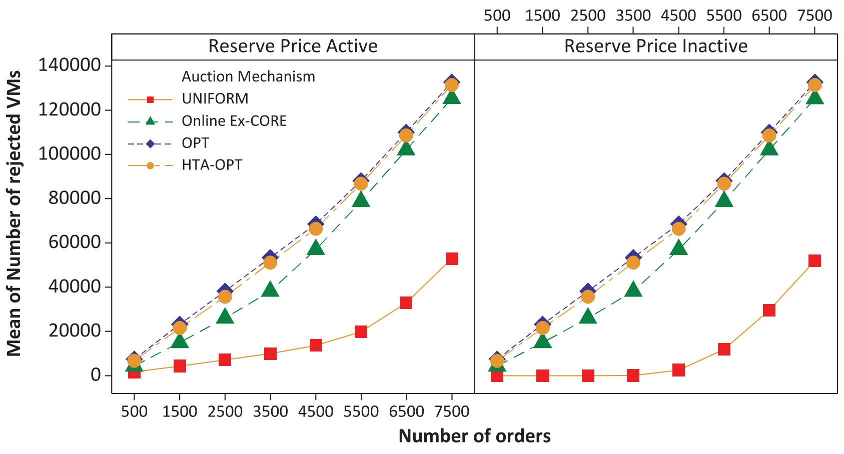
The green dashed line in Figure 11 depicts the reserve price generated based on our model in a sample simulation run. Our investigation on the historical price data of spot instances for the past 90 days prior to 14 November 2013 shows that the spot market price never goes below $0.007 for the small instances in the eastern US region. The value complies with our computed reserve price for that instance type when the physical server characteristics as well as the electricity prices and outside air temperature parameters are based on realistic data for an Amazon data center in the US-East region. The same holds true for the modeled and observed minimum spot price for the other instance types in the `m1` instance class because they are based on hardware with similar power-draw characteristics.

10.3.1. Experimental Results. We evaluate the online Ex-CORE auction by comparing profit and number of rejected VM requests to the other auction mechanisms outlined in Section 9. The computed profit is the total generated revenue minus the cost of electricity. The capital cost and all other fixed cost are not considered because they are identical for all mechanisms. Each experiment is carried out 30 times, and the mean value is reported. The results are illustrated in Figure 10(a) and Figure 10(b), where the number of orders in a 24-hour simulation is increased from 500 to 7,500, and scenarios with or without the adoption of a reserve price are shown.

Figure 10(a) shows that gained profit by all mechanisms increases with the number of orders. The OPT, HTA-OPT, and online Ex-CORE auctions generate comparable profits, whereas there is a big gap between the uniform price auction and the other mechanisms. When supply is higher than demand and there is no competition among bidders, all orders are accepted by the uniform price auction; consequently, the uniform price auction performs poorly under such circumstances. This supports the idea that the traditional auction mechanisms such as the Vickrey auction [Vickrey 1961] or uniform price auction are not suitable for the cloud spot market in which supply is often higher than demand.



(a)



(b)

Fig. 10. (a) Average profit gained and (b) number of rejected VM instances with different auction mechanisms.

The benefit of using the online Ex-CORE auction is that, in spite of a small difference in generated profit compared to OPT and HTA-OPT (6% lower on average), it accommodates a considerably higher number of VMs (17% and 14% less rejections on average, respectively). This reduces the impact of the bidder drop problem, introduced by Lee and Szymanski [2005] that can be caused by frequent rejection of customers with low valuations.

As illustrated in Figure 10(a) and Figure 10(b), the reserve price only affects the outcome of the uniform price auction. Considering the range and distribution of bid values used in the simulation, the market price generated by online Ex-CORE, OPT, and HTA-OPT is always higher than the reserve price. To exemplify further, Figure 11 provides the reserve price and the market price generated by online Ex-CORE in a sample simulation run. This does not mean, however, that the reserve price is of no importance and can be ignored in real-world scenarios.

To show the impact of the reserve price, various highest submitted bid prices are used to decrease the average market price, as shown in Figure 12. As can be seen in

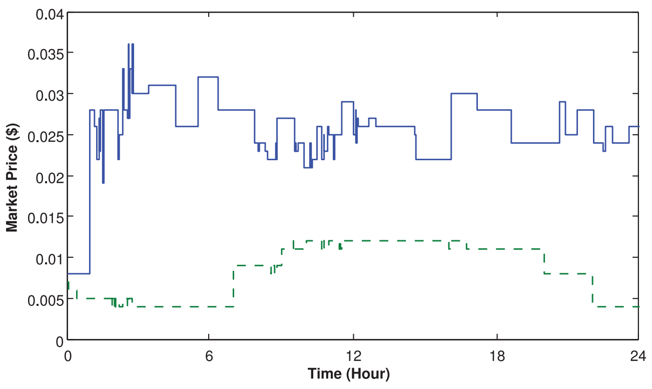


Fig. 11. Reserve price (green dashed line) and spot market price generated by online Ex-CORE (blue solid line) in a sample simulation run when the number of orders is 1,500.

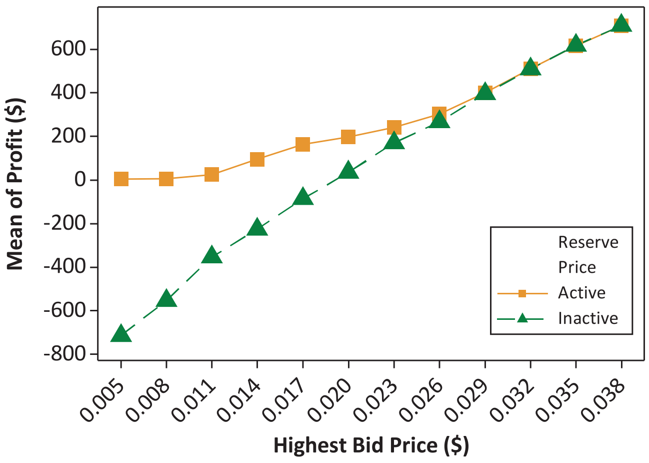


Fig. 12. Average profit gained by Ex-CORE when the number of orders is 1,500.

Table I. Paired T-test with 95% Confidence Interval (CI) for Comparison of Difference in Mean of Profit and Number of Rejected VMs Generated by OPT and HTA-OPT (OPT − HTA-OPT) when the Number of Orders Is 4,500

	OPT	HTA-OPT	Difference (95% CI)	*P*-value
Profit	5358.7	5361.6	−2.9 (−33.6, 27.7)	=0.846
Rejected	68575	66423	2152 (1192, 3111)	<0.001

the figure, when the highest submitted bid price is low and therefore the market price is lower on average, the absence of reserve price can lead to loss or lower profit due to execution of VMs at a price below their variable cost.

Because we are interested in the importance of a priori knowledge on the holding time of VMs by customers, the profit and the number of rejected VMs by OPT and HTA-OPT are investigated further. The results of a paired T-test comparing the profit performance of OPT with HTA-OPT when the number of orders is 4,500 and the holding time of VMs is distributed i.i.d. based on a Pareto distribution with both shape and location parameters equal to 1 are shown in Table I. Given a null hypothesis of no statistically significant difference in mean profit by OPT and HTA-OPT, the p-value is relatively high ($p = 0.846$), suggesting that there is no strong evidence that the null hypothesis

Table II. True Private Values of Experiment Participants

User	Price Value ($)	Quantity
T1, C1	0.0691	2
T2, C2	0.0092	1
T3, C3	0.0475	1
T4, C4	0.0232	2
T5, C5	0.0184	1

is false (i.e., there is no credible evidence that OPT and HTA-OPT on average generate different profits). However, there is a statistically significant difference in the mean number of rejected VMs. HTA-OPT rejected 2,152 fewer VMs on average as it results in outcomes with a lower market price. Considering the reported 95% Confidence Interval (CI), we can state that knowing the holding time of VMs by itself does not change the amount of profit a provider generates because it is not aware of upcoming orders' bid prices.

10.4. Experimental Study Using A System Protoype

10.4.1. Experimental Design. In this section, we evaluate our proposed framework by conducting an experimental study in a real environment. We designed and implemented a prototype of the proposed auction mechanism using the OpenStack cloud platform. We extended Horizon – *the OpenStack dashboard project* – to price and allocate VM resources using the online Ex-CORE auction mechanism. We added a section visible to standard OpenStack users, labeled Spot Instances, that allows users to submit their bid prices and the number of required VM instances (i.e., orders) to the system. The Ex-CORE auction is then used to compute the market price according to the submitted orders. If the user's bid price is greater than the current market price, which means the request can be fulfilled, requested instances are created by the OpenStack platform; otherwise, a proper error message will be provided to the user. Interested readers are referred to Toosi et al. [2015] for detailed information regarding the system implementation.

Using our system prototype, we conducted a 20-minute experiment with a group of 10 people who want to acquire VM instances on a local private cloud. Participants were divided into two groups of five: (i) Group T or truthful bidders and (ii) Group C or counterpart bidders who have the freedom to misreport their true private values to maximize their utility.

We provided participants in the experiment with a pair of uniformly random generated quantity and price values that must be considered as their true private values. Considering the scale of the experiment, we limited r to two, i.e., the maximum number of simultaneous VM instances each user can run in the system is 2. Accordingly, we chose uniformly random true quantity values from {1, 2}. For price values, we adopted the pricing details of Amazon EC2 *m3.medium* in the Asia Pacific-Sydney region at the time of the experiment, while we replaced per-hour charging period with 30 seconds in our experiment. True price values in dollars are drawn from a uniform distribution of the interval [0.0081, 0.0700]. Table II shows true private price and quantity values for participants of each group.

Participants of the experiment were provided with the details of the online Ex-CORE auction algorithm (i.e., Algorithm 1) and their utility function (i.e., Equation (1)). Participants were asked to submit their orders using the OpenStack dashboard to acquire VM instances of a specific type as much as and as long as they could, according to their true private values. Participants of group T were obliged to submit their true values to acquire instances through the whole experiment regardless of the market price fluctuation. Participants of group C were asked to try to maximize their utility

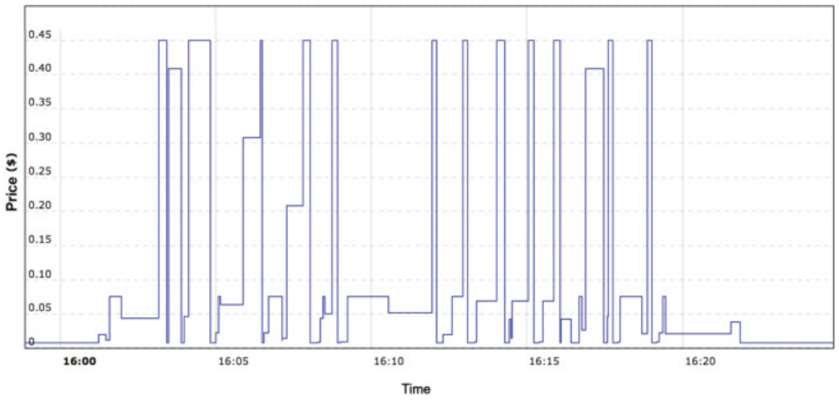


Fig. 13. Spot market price fluctuation during the experiment.

Table III. Total Cost, the Number of Full Time Slots Usage, and Utility Values of Experiment Participants

User	Total Cost ($)	Number of Full Time Slots	Utility ($)
T1	1.2964	16	**0.9148**
C1	1.8216	17	0.5278
T2	0.0000	0	**0.0000**
C2	10.0227	18	−9.8571
T3	0.1896	6	0.0954
C3	0.2280	8	**0.1520**
T4	0.0436	1	**0.0030**
C4	3.6810	5	−3.4490
T5	0.0000	0	**0.0000**
C5	0.0738	2	−0.0370

based on the rules of the experiment and given pricing information. Therefore, if it is deemed beneficial, a participant of group C might strategically misreport his or her bid price or the quantity (i.e., $b \neq v$ or $r \neq q$). To provide enough incentives for a participant of group C to act rationally in the experiment, we considered a prize for the winner of the experiment. The winner of the experiment is the one who can achieve the highest positive difference of the utility value with his or her counterpart truthful bidders.

10.4.2. Results and Analysis. Figure 13 depicts the market price fluctuation during the experiment. As shown in the figure, the price reaches the maximum bid price on multiple occasions. This happened because some participants with low private values (e.g., $C4$ and $C2$, who were starving in the market) tried to terminate other participants' instances by submitting very high bid prices. This affected their utility, however, since they were charged multiple times higher than their true values.

As theoretically proved in Section 7, the Ex-CORE auction mechanism is truthful with high probability in the price dimension. Therefore, as we expected, excluding $T3$, all truthful users (i.e., participants of group T) achieved higher utility than their counterpart users who misreported their true values. Table III shows the total cost and achieved utility values by all users based on the utility function in Equation (1). In order to investigate how user $C3$ could achieve the highest positive difference in comparison with his or her paired truthful participant, we analyzed the submitted orders by all users. The result of our analysis shows that $C3$ is the most truthful user among the participants of group C; he or she continuously submitted the true quantity value and bid price values significantly close to his or her true value. The only reason

$C3$ achieved the highest difference is that he or she was quicker in submitting orders and could obtain two additional full time slots of instance usage. The truthful user $T3$ was also able to do the same if he or she would have submitted his or her true values fast enough at the same time.

As can be seen in Table III, $T2$ and $T5$ could not acquire instances for a full time slot at all since the market price was often higher than their true price values. $T4$ similarly ends up running instances for only one time slot. Comparatively, paired users from group C acquired instances for higher number of time slots. However, their overall utility values were negative because they ended up paying more than their true values. Results of the experiment reported in Table III support the theoretically proven supposition that the Ex-CORE algorithm is truthful with high probability in the price dimension. This confirms the fact that rational users' dominant strategy in a truthful auction mechanism is to report their true private values. Moreover, our investigation on the historical price data of spot instances in Amazon EC2 shows that price spikes similar to what happened in our experiment are occurring in Amazon's spot market as well. This happens either due to the same experience we had in our experiment, where some users submit very high bids, or possibly sudden spikes in demand. Intuitively, without knowing how the spot market mechanism works, no user has the incentive to strategize over its bid. This has been suggested by other studies as well [Ben-Yehuda et al. 2013; Wang et al. 2012].

11. CONCLUSION AND FUTURE WORK

With the rapid adoption of cloud computing environments, balancing supply and demand for cloud resources through dynamic forms of pricing is quickly gaining importance. In this article, we presented an envy-free auction that is truthful with high probability and generates a near optimal profit for the cloud provider. The auction operates under conditions similar to the EC2 spot market. The truthfulness of the mechanism frees bidders from understanding its intricacies, thereby lowering the complexity of participation and the options for strategic behavior. At the same time, the mechanism aims to achieve a maximal profit for the provider and achieves envy-freeness through the use of a uniform price. The mechanism is a generalization and extension of the Consensus Revenue Estimate (CORE) auction that enables its application in the cloud computing setting, which requires an online recurrent auction with multiunit requests. To incorporate marginal costs of production in the resource trading process, we pair the auctioning scheme with a method that calculates dynamic reserve prices based on a cost model that incorporates data center *PUE*, load, and electricity cost.

An important benefit of the proposed auction design is that it achieves near optimality with regard to maximizing revenue without requiring prior knowledge on the bid distributions. Our evaluation demonstrates its performance in this regard under a variety of order distributions. The proposed mechanism is shown to significantly outperform the uniform price auction and to closely approximate the profit outcome of the revenue maximizing, but nontruthful, optimal single-price auction in an online setting (within 6% in our experiments) while improving on the number of rejected VMs (up to 17% in our experiments). Our results also show that the generated revenue does not differ significantly from the revenue attained by a mechanism based on dynamic programming that relies on prior knowledge regarding the holding time of VMs. Finally, a small-scale experimental study using a system prototype confirms the truthfulness property of the proposed mechanism in a real test environment.

As future work, we intend to explore the collusion-resistant property for our proposed auction mechanism. We would also like to investigate an extensive cost model for calculation of the reserve price. Penalty costs can be considered as part of variable operational costs, as when a *Service Level Agreement (SLA)* violation happens due to a VM consolidation process, for example. Other factors must be taken into account such as network-related costs, dynamic changes in price of electricity, or the amount of on-site renewable energy usage if they exist. The bidder drop problem might result in disappointed customers who diverge from the auction market to other pricing models or even services from other providers. We are interested in studying how the bidder drop problem can be minimized while aligned with the profit maximization goal of cloud providers.

REFERENCES

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. 2010. A view of cloud computing. *Communications of the ACM* 53, 4 (2010), 50–58.

Lawrence M. Ausubel and Paul Milgrom. 2006. The lovely but lonely Vickrey auction. *Combinatorial Auctions* 17 (2006), 22–26.

Orna Agmon Ben-Yehuda, Muli Ben-Yehuda, Assaf Schuster, and Dan Tsafrir. 2013. Deconstructing Amazon EC2 spot instance pricing. *ACM Transactions on Economy Computing* 1, 3, Article 16 (Sept. 2013), 20 pages.

Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. 2011. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 41, 1 (Jan. 2011), 23–50. DOI:http://dx.doi.org/10.1002/spe.995

Gaylon S. Campbell and John M. Norman. 2012. *An Introduction to Environmental Biophysics*. Springer Science & Business Media.

Navraj Chohan, Claris Castillo, Mike Spreitzer, Malgorzata Steinder, Asser Tantawi, and Chandra Krintz. 2010. See spot run: Using spot instances for mapreduce workflows. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*. USENIX Association.

Amir Danak and Shie Mannor. 2010. Resource allocation with supply adjustment in distributed computing systems. In *Proceedings of the 30th International Conference on Distributed Computing Systems (ICDCS'10)*. 498–506. DOI:http://dx.doi.org/10.1109/ICDCS.2010.60

Ínigo Goiri, Kien Le, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. 2011. Intelligent placement of datacenters for internet services. In *Proceedings of the 31st IEEE International Conference on Distributed Computing Systems (ICDCS'11)*. 131–142. DOI:http://dx.doi.org/10.1109/ICDCS.2011.19

Andrew V. Goldberg and Jason D. Hartline. 2003a. Competitiveness via consensus. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'03)*. 215–222.

Andrew V. Goldberg and Jason D. Hartline. 2003b. Envy-free auctions for digital goods. In *Proceedings of the 4th ACM Conference on Electronic Commerce (EC'03)*. 29–35. DOI:http://dx.doi.org/10.1145/779928.779932

Andrew V. Goldberg and Jason D. Hartline. 2005. Collusion-resistant mechanisms for single-parameter agents. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'05)*. Society for Industrial and Applied Mathematics, Vancouver, British Columbia, 620–629.

Andrew V. Goldberg, Jason D. Hartline, Anna R. Karlin, Michael Saks, and Andrew Wright. 2006. Competitive auctions. *Games and Economic Behavior* 55, 2 (2006), 242–269. DOI:http://dx.doi.org/10.1016/j.geb.2006.02.003

Jerry R. Green and Jean-Jacques Laffont. 1986. Partially verifiable information and mechanism design. *Review of Economic Studies* 53, 3 (1986), 447–456. http://restud.oxfordjournals.org/content/53/3/447.abstract.

Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. 2008. The cost of a cloud: Research problems in data center networks. *SIGCOMM Computing Communication Review* 39, 1 (2008), 68–73. DOI:http://dx.doi.org/10.1145/1496091.1496103

Steve Greenberg, Evan Mills, Bill Tschudi, Peter Rumsey, and Bruce Myat. 2006. Best practices for data centers: Lessons learned from benchmarking 22 data centers. *ACEEE Summer Study on Energy Efficiency in Buildings in Asilomar, CA* 3 (2006), 76–87.

Venkatesan Guruswami, Jason D. Hartline, Anna R. Karlin, David Kempe, Claire Kenyon, and Frank McSherry. 2005. On profit-maximizing envy-free pricing. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*. 1164–1173.

Leonid Hurwicz. 1975. On the existence of allocation systems whose manipulative Nash equilibria are pareto-optimal. *Presented at the 3rd World Congress of the Econometric Society*. Toronto, Canada.

Bahman Javadi, Ruppa K. Thulasiram, and Rajkumar Buyya. 2011. Statistical modeling of spot instance prices in public cloud environments. In *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing (UCC'11)*. 219–228.

Kien Le, Richardo Bianchini, Jingru Zhang, Yogesh Jaluria, Jiandong Meng, and Thu D. Nguyen. 2011. Reducing electricity cost through virtual machine placement in high performance computing clouds. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11)*. Article 22, 22:1–22:12 pages. DOI:http://dx.doi.org/10.1145/2063384.2063413

Juong-Sik Lee and B. K. Szymanski. 2005. A novel auction mechanism for selling time-sensitive e-services. In *Proceedings of 7th IEEE International Conference on E-Commerce Technology, (CEC'05)*. Hong Kong, 75–82. DOI:http://dx.doi.org/10.1109/ICECT.2005.7

Mario Macías and Jordi Guitart. 2011. A genetic model for pricing in cloud computing markets. In *Proceedings of the 2011 ACM Symposium on Applied Computing*. 113–118. DOI:http://dx.doi.org/10.1145/1982185.1982216

Mario Macías and Jordi Guitart. 2014. SLA negotiation and enforcement policies for revenue maximization and client classification in cloud providers. *Future Generation Computer Systems* 41 (2014), 19–31.

M. Mihailescu and Yong-Meng Teo. 2012. The impact of user rationality in federated clouds. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'12)*. Ottawa, Canada, 620–627. DOI:http://dx.doi.org/10.1109/CCGrid.2012.127

Kevin Mills, James Filliben, and Chris Dabrowski. 2011. Comparing VM-placement algorithms for on-demand clouds. In *Proceedings of 3rd International Conference on Cloud Computing Technology and Science (CloudCom'12)*. 91–98. DOI:http://dx.doi.org/10.1109/CloudCom.2011.22

Hervé Moulin and Scott Shenker. 2001. Strategyproof sharing of submodular costs: Budget balance versus efficiency. *Economic Theory* 18, 3 (2001), 511–533. DOI:http://dx.doi.org/10.1007/PL00004200

Roger B. Myerson. 1981. Optimal auction design. *Mathematics of Operations Research* 6, 1 (1981), 58–73. DOI:http://dx.doi.org/10.1287/moor.6.1.58

Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. 2007. *Algorithmic Game Theory*. Cambridge University Press.

Michael K. Patterson. 2008. The effect of data center temperature on energy efficiency. In *Proceedings of 11th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITHERM'08)*. Orlando, FL, 1167–1174. DOI:http://dx.doi.org/10.1109/ITHERM.2008.4544393

Steven Pelley, David Meisner, Thomas F. Wenisch, and James W. VanGilder. 2009. Understanding and abstracting total data center power. In *Proceedings of the Workshop on Energy-Efficient Design (WEED'09) held in Conjunction with the 36th International Symposium on Computer Architecture (ISCA'09)*. Austin, Texas, USA.

Neil Rasmussen. 2011. Electrical efficiency measurement for data centers. *White Paper by Schneider Electric - Data Center Science Center* 154 revision 2 (2011).

Yang Song, Murtaza Zafer, and Kang-Won Lee. 2012. Optimal bidding in spot instance market. In *Proceedings of the 31st International Conference on Computer Communications (INFOCOM'12)*. Orlando, Florida, USA, 190–198. DOI:http://dx.doi.org/10.1109/INFCOM.2012.6195567

Murray Stokely, Jim Winget, Ed Keyes, Carrie Grimes, and Benjamin Yolken. 2009. Using a market economy to provision compute resources across planet-wide clusters. In *Proceedings of IEEE International Symposium on Parallel Distributed Processing (IPDPS'09)*. 1–8. DOI:http://dx.doi.org/10.1109/IPDPS.2009.5160966

Adel Nadjaran Toosi, Farzad Khodadadi, and Rajkumar Buyya. 2015. SipaaS: Spot instance pricing as a Service framework and its implementation in OpenStack. *Concurrency Computation: Practice and Experiences*. DOI:http://dx.doi.org/10.1002/cpe.3749

Adel Nadjaran Toosi, Rodrigo N. Calheiros, Ruppa K. Thulasiram, and Rajkumar Buyya. 2011. Resource provisioning policies to increase IaaS provider's profit in a federated cloud environment. In *Proceedings of the 13th IEEE International Conference on High Performance Computing and Communications (HPCC'11)*. Banff, Canada, 279–287. DOI:http://dx.doi.org/10.1109/HPCC.2011.44

William Vickrey. 1961. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance* 16, 1 (1961), 8–37.

William Voorsluys and Rajkumar Buyya. 2012. Reliable provisioning of spot instances for compute-intensive applications. In *Proceedings of 26th International Conference on Advanced Information Networking and Applications (AINA'12)*. Fukuoka, Japan, 542–549. DOI:http://dx.doi.org/10.1109/AINA.2012.106

Wei Wang, Baochun Li, and Ben Liang. 2012. Towards optimal capacity segmentation with hybrid cloud pricing. In *Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems (ICDCS'12)*. 425–434. DOI:http://dx.doi.org/10.1109/ICDCS.2012.52

Wei Wang, Ben Liang, and Baochun Li. 2013. Revenue maximization with dynamic auctions in IaaS cloud markets. In *Proceedings of the 21st IEEE/ACM International Symposium on Quality of Service (IWQoS)*. 1–6. DOI: http://dx.doi.org/10.1109/IWQoS.2013.6550265

Hong Xu and Baochun Li. 2013. Dynamic cloud pricing for revenue maximization. *IEEE Transactions on Cloud Computing* 1, 2 (July 2013), 158–171.

Sangho Yi, Derrick Kondo, and Artur Andrzejak. 2010. Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud. In *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing (Cloud'10)*. 236–243. DOI: http://dx.doi.org/10.1109/Cloud.2010.35 http://dx.doi.org/10.1109/Cloud.2010.35

Sharrukh Zaman and Daniel Grosu. 2012. An online mechanism for dynamic VM provisioning and allocation in clouds. In *5th IEEE International Conference on Cloud Computing (CLOUD'12)*. 253–260.

Sharrukh Zaman and Daniel Grosu. 2013. Combinatorial auction-based allocation of virtual machine instances in clouds. *Journal of Parallel and Distributed Computing* 73, 4 (2013), 495–508. DOI: http://dx.doi.org/10.1016/j.jpdc.2012.12.006

Linquan Zhang, Zongpeng Li, and Chuan Wu. 2014. Dynamic resource provisioning in cloud computing: A randomized auction approach. In *Proceedings of IEEE INFOCOM*. Toronto, Canada, 433–441. DOI:http://dx.doi.org/10.1109/INFOCOM.2014.6847966

Qi Zhang, Quanyan Zhu, and R. Boutaba. 2011. Dynamic resource allocation for spot markets in cloud computing environments. In *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing (UCC'11)*. Melbourne, Australia, 178–185. DOI: http://dx.doi.org/10.1109/UCC.2011.33

Received Februray 2015; revised September 2015; accepted November 2015