Improving the Revenue, Efficiency and Reliability in Data Center Spot Market: A Truthful Mechanism

Kai Song
Department of Computer Science
University of Southern California
Los Angeles, CA, USA
kaisong@usc.edu

Yuan Yao
Oracle Corporation
Redwood Shores, CA, USA
yuan.x.yao@oracle.com

Leana Golubchik
Department of Computer Science
University of Southern California
Los Angeles, CA, USA
leana@usc.edu

***Abstract*—Data centers are typically over-provisioned, in order to meet certain service level agreements (SLAs) under worst-case scenarios (e.g., peak loads). Selling unused instances at discounted prices thus is a reasonable approach for data center providers to off-set the maintenance and operation costs. Spot market models are widely used for pricing and allocating unused instances. In this paper, we focus on mechanism design for a data center spot market (DCSM). Particularly, we propose a mechanism based on a repeated uniform price auction, and prove its truthfulness. In the mechanism, to achieve better quality of service, the flexibility of adjusting bids during job execution is provided, and a bidding adjustment model is also discussed. Four metrics are used to evaluate the mechanism: in addition to the commonly used metrics in auction theory, namely, revenue, efficiency, slowdown and waste are defined to capture the Quality of Service (QoS) provided by DCSMs. We prove that a uniform price action achieves optimal efficiency among all single-price auctions in DCSMs. We also conduct comprehensive simulations to explore the performance of the resulting DCSM. The result show that (1) the bidding adjustment model helps increase the revenue by an average of 5%, and decrease the slowdown and waste by average of 5% and 6%, respectively; (2) our model with repeated uniform price auction outperforms the current Amazon Spot Market by an average of 14% in revenue, 24% in efficiency, 13% in slowdown, and by 14% in waste. Parameter tuning studies are also performed to refine the performance of our mechanism.**

***Keywords*-spot instance, mechanism design, evaluation**

I. Introduction

The emergence of data centers and cloud computing enables users to dynamically provision required IT resources. However, at the providers' end, due to the heterogeneity and variability of resource requests over time and space, data centers are typically over-provisioned, in order to meet certain service level agreements (SLAs) under worst-case scenarios (e.g., peak loads) [15]. Consequently, most of the time, large amounts of resources are unused and unsold, which results in significant waste for the cloud service providers. Given this, service providers have incentives to encourage customers to purchase the unsold cloud capacity to off-set the maintenance and operation costs as well as to increase profit. At the same time, customers desire to obtain such resources at discounted prices. Therefore, Data Center Spot Markets (DCSMs) driven by supply-demand curves, naturally become a good solution that can result mutual benefit to both, service providers and their customers. Use of DCSMs has been shown to result in average cost-savings of $50\% - 66\%$ under various scenarios including web crawling, Geo-spatial Analysis and Image Processing and so on [1]. A DCSM could have a single [1] or multiple [2] [3] providers. In this paper, we focus on a DCSM with a single provider, and use the term DCSM to mean a single-provider DCSM. To better understand the nature and properties of DCSMs, we first introduce and discuss the Amazon Spot Market (ASM).

In ASM, diverse instance types [1] are offered to meet customers' various computing needs. Each type of instance has a predictable amount of compute capacity, with cost charged on per instance-hour basis. In this paper, we focus on designing a mechanism for the market with only one type of instance at one location, where each instance is sold at the same price. We note that, in DCSMs, instances with various types are allocated from resources in physical machines at different locations, and such allocation algorithm is outside the scope of this paper. Figure 1 outlines the framework of ASM. Here,

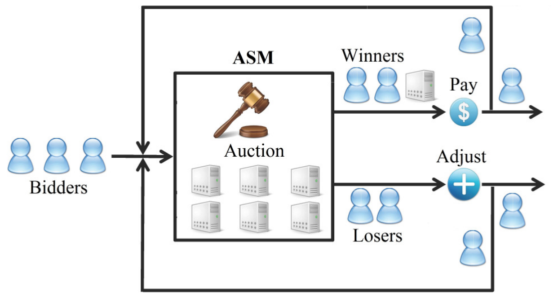


Fig. 1. ASM Overview

customers request unsold instances (i.e., spot instances) with a *bid*, which is the maximum price ($/hour) they are willing to pay for one such instance. Based on submitted bids that reflect the demands of the market, Amazon determines a time-variant selling price, namely the *Spot Price* (SP), as the cut-off price between winners (those who obtain resources) and

[1]An instance is an atomic unit for which a customer can bid at a cloud provider (e.g., Amazon, Google, Microsoft), refer [4] for descriptions of instance types in ASM.

losers (those who do not). SP fluctuates in accordance with the highly volatile market supply-demand curve at the granularity of inter-price time. Whenever SP exceeds customers' bids, out-of-bid events are triggered, and Amazon may terminate out-of-bid instances without notification. When an out-of-bid event occurs, bidders whose bids are below SP lose their access to instances and need to submit bids again for instance access, while other bidders remain in the market with instance access. Despite the fact that customers' bids differ, those whose bids exceed the spot price pay the same price for each instance, which is typically lower than their bids. Such form of an auction is referred to as a *single-price auction* [14]. Two forms of single-price auction, namely, *uniform-price auction* [18] [20] and *optimal single-price auction* [14] have been widely used in mechanism design, and in Section II, we discuss respective pros and cons when applying them to a DCSM.

Although ASM provides customers with inexpensive service, such monetary benefit is attained at the cost of service reliability, since requests could be terminated by Amazon without notification. Thus, in addition to revenue and efficiency, which are commonly used metrics to evaluate markets, reliability is also a critical performance metric in DCSM. Indeed, the discussions in [5], indicates that both, the customers and Amazon are looking for a more reliable ASM.

Unfortunately, the mechanism adopted in ASM is not publicly known, and hence customers are left guessing how ASM works. Recent work [11], [25] and [17] has been conducted to discover the underlying mechanism, and it shows that ASM has the following problems.

- Pricing mechanism is unknown to customers, and thus the use of ASM is complicated by the fact that a good bidding strategy heavily relies on an understanding of strategies of other buyers as well as an estimate of available instances, both of which are difficult to obtain. In addition, with current ASM, malicious bidders have incentives to game the system through their bids [6].
- Once instances are obtained, winners are unable to change their bids. For example, given the opportunity of adjusting their bids, bidders are likely to increase their bids when jobs are near completion, to avoid last-minute service interruption. At the same time, Amazon might be able to increase its revenue by allowing winners to pay higher price at later stage of job execution.

Towards this end, we argue that a truthful auction mechanism is more attractive in DCSM to avoid complications mentioned above. Particularly, we propose a truthful mechanism adapted from *repeated uniform price auction*. In this mechanism, observing that bidders are likely to adjust true valuation during the job's execution, we incorporate *bidding flexibility* in our mechanism by allowing bid adjustments for all bidders. To evaluate how bidding flexibility helps improve the performance of our mechanism, a bidding adjustment model is discussed in Section III-D; the use of this model is validated in Section V.

Evaluation of the proposed mechanism is conducted using four metrics, namely *revenue*, *efficiency*, *slowdown* and *waste*. The first two metrics are commonly used metrics in auction theory, and the last two metrics we define in the context of DCSM in evaluation of service reliability. We prove that a uniform price achieves optimal efficiency among all single-price auctions in DCSMs. A comprehensive comparison study is conducted to show, the performance improvements gained using the four metrics stated above, over the mechanism used in ASM . Parameter tuning studies are also performed to refine the performance of our mechanism. Our contributions can be summarized as follows:

- We propose a truthful mechanism based on a uniform price auction, which eliminates the complications and malicious behaviors that exist in the current mechanism in ASM. In the proposed mechanism, to achieve better performance, the flexibility of adjusting bids during job execution is incorporated, and a bidding adjustment model is introduced accordingly to make use of such flexibility and validated in Section V.
- We define and introduce two new performance metrics, namely, slowdown and waste, to better reflect the defined Quality of Service in DCSM.
- We prove that a uniform price action achieves optimal efficiency among all single-price auctions in DCSMs, and comprehensive comparison studies are conducted to show the performance improvements over the mechanism used in ASM using the four metrics mentioned above. The results show that (1) The bidding adjustment model helps increase the revenue by an average of 5.4%, and decrease the slowdown and waste by an average of 4.9% and 5.7%, respectively; (2) Our model with a repeated uniform price auction outperforms the current Amazon Spot Market by averagely 14.2% in revenue, 24.3% in efficiency, 13.6% in slowdown, and by 14.1% in waste.

II. Background

In this section we present fundamentals in auction theory as background needed in the remainder of the paper.

Definition 1: An auction is called incentive-compatible if, for every player i, the dominant strategy is to provide the true valuation v^i. Incentive compatibility is also called strategy-proofness or truthfulness [21].

Definition 2: In a single-price auction, each winner pays the same per unit price.

A single-price auction is commonly used in digital goods markets, where identical units of a homogeneous commodity are sold for the same price. There are two commonly used forms of a single-price auction, namely, uniform price auction and optimal single-price auction, depending on how an auctioneer determines the selling price. Suppose we have n customers bidding for M units in a uniform price auction. Each bidder has a demand of m^i units, and the bid for each unit is b^i. Assume, without loss of generality, $b^1 \geq b^2 \geq \cdots \geq b^n$.

Definition 3: Let k^u be the smallest number such that $\sum_{1 \leq i \leq k^u} m^i > M$, in the uniform-price auction, the unit selling price is b^{k^u}, i.e., the highest losing bid.[2] All bids that are greater than b^{k^u} will be accepted, and all other bids will be rejected.

It has been proven that a *uniform-price* auction is always incentive-compatible, if each bidder is only allowed to bid for one unit. But, truthfulness fails in the multiple-demand case since bidders have monetary incentives to lower their actual bids, which is termed *demand reduction* [18]. It is interesting to consider the case where $\sum_{1\le i\le n} m^i < M$ (i.e., k^u does not exist). We will return to this issue in Section III.

Definition 4: Let $k^{opt} = \arg\max_{1\le i\le n} (i-1)\cdot b^i$, in the optimal single-price auction, the unit selling price is $b^{k^{opt}}$. All bids that are greater than $b^{k^{opt}}$ will be accepted, and all other bids will be rejected.

We argue that an optimal single-price auction is not truthful even in the context of a single-unit demand. As an illustrating example, consider three bidders A, B, and C who bid \$10, \$5, and \$2, respectively, each for a single instance. The utility function of each bidder is defined as 0 for losers, and the difference between true valuation and selling price for winners. Here, with the optimal single-price auction, one instance will be sold for a revenue of \$5, where only bidder A is the winner, and bidder B has a utility of \$0. However, if the second highest bidder B decreases its bid from \$5 to \$3, two instances are sold for a revenue of \$4, where B would win one instance at the cost of \$2, with a utility of \$3. Therefore, B has sufficient incentive to hide its true valuation, to increase its utility.

Definition 5: In a single-price auction, if m units are sold at the unit price of p, then the revenue is $m\cdot p$.

Definition 6: In a single-price auction, assume the true valuation of each unit of bidder i is v^i, and let the value to the seller of each instance be a small constant c, where instances are sold to n_w winners with a unit price p , then efficiency (social welfare) is defined as $\sum_{1\le i\le n_w} m^i\cdot(v^i-p)+(p-c)\cdot\sum_{1\le i\le n_w} m^i = \sum_{1\le i\le n_w} m^i\cdot(v^i-c)$.

III. Mechanism Design

In this section, we focus on mechanism design for a DCSM. After justifying our selection of single-price auction, we propose our truthful mechanism based on repeated uniform-price auction. In addition to proving the truthfulness and discussing the mechanism details, we also model how bidders adjust their true valuations as job execution progresses and job completion approaches. Table I summarizes the notation used in the paper.

A. Why Single-price Auction

In a DCSM, multiple types of instances are sold at different unit prices, instances of the same type in various geographic locations are sold the different unit prices as well; however, instances of the same type in the same location are sold at the same unit price. To better capture and understand the economic properties of a DCSM, we focus on designing a mechanism for a market with only one type of instance at one location, where a single-price auction becomes an appropriate mechanism. We note that, in DCSMs, instances with various types are allocated from resources on physical machines at different locations; such allocation algorithms are outside the scope of this paper.

[2]Some literature takes the highest losing bid as the unit selling price.

TABLE I
NOTATIONS

N_j	Total number of job requests at t_j
M	Number of available instances
m^i	Number of request instances of job i
l^i	Initial length of job i
v^i	Initial true valuation of job i
b^i_j	Job i's bid at t_j
v^i_j	Job i's true valuation at t_j
l^i_j	Job i's length at t_j
k^u_j	Index of the highest losing bid at t_j
p_j	Spot price at t_j
α^i_j	Number of instances allocated to job i at t_j
T_{to}	Job requests' timeout

B. Repeated Uniform-price Auction Model

In many cases, the use of auctions is complicated by the fact that a good bidding strategy for a bidder requires understandings of strategies and utilities of other bidder. Thus, truthful auction mechanisms are attractive in this context because they avoid such complications. Particularly in DCSMs, both, bidders' strategies and their heterogeneous job requests are private. In addition, DCSMs are highly volatile due to new job arrivals and completed job departures. Therefore, a truthful mechanism is favored by both the provider and bidders, with the goal of building a sustainable and profitable market. As discussed in Section II, uniform-price auction and optimal single-price auction are two widely used variations. In our work, we select uniform-price auction as the model due to its truthfulness in DCSMs' settings. Details are described below.

In our model, time is slotted, where one time slot is an atomic scheduling unit. At each time slot t_j, each job request is associated with a bid, based on which a uniform-price auction is performed to determine selling price, namely *Spot Price* (SP). Any request whose bid is greater than SP obtains instances it needs in time slot t_j, whereas other requests fail to obtain any instances in t_j. Formally, let (b^i_j, m^i) characterize job request $R^i (1\le i\le N_j)$ at t_j, where N_j is the total number of job requests at t_j, b^i_j is R^i's bid in t_j, and m^i is the number of requested instances, which remains the same during job execution. Without loss of generality, let $b^1_j \ge b^2_j \ge \cdots \ge b^{N_j}_j$, and suppose that there are M available instances. Based on the uniform-price auction, the spot price at t_j, $p_j = b^{k^u_j}_j$, where $k^u_j = \arg\min_k \sum_{1\le i\le k} m^i > M$. If k^u_j does not exist, i.e., $\sum_{1\le i\le N_j} m^i \le M$, for ease of exposition, we add a dummy job $R^d(b^{N_j}_j - \varepsilon, +\infty)$, which requests infinite number of instances with a bid $b^{N_j}_j - \varepsilon$ (ε is a small constant). In this case, the spot price is $b^{N_j}_j - \varepsilon$. Note that for each request R^i, either all the m^i requested instances are granted, or none are granted. In summary, payment and allocation rules are given as:

$$p_j^i = \begin{cases} m^i \cdot b_j^{k_j^u}, i < k_j^u \\ 0, i \geq k_j^u \end{cases} \qquad \alpha_j^i = \begin{cases} m^i, i < k_j^u \\ 0, i \geq k_j^u \end{cases} \tag{1}$$

where p_j^i and α_j^i are B^i's payment and allocated number of instances at t_j, respectively. Note that, $p_j^i = m^i \cdot p_j$ for winners.

In each time slot t_j, the market also has job arrivals and departures - new job requests arrive and existing job requests depart upon completion. During the execution of a job, i.e., R^i, if the spot price exceeds R^i's bid, then R^i is interrupted and thus might be executed again from the beginning or from a certain check point. Details of service interruption are discussed in Section IV. In addition, some bidders might have less competitive bids due to budget constraints, and thus fail to gain access to requested instances for a long time. To avoid accumulating losers in the market, we assume that such bidders will voluntarily leave the market after experiencing failure for T_{to} consecutive time slots. We study how T_{to} impacts system performance in Section V.

C. Truthfulness

Generally, uniform-price auction with multiple demands is untruthful because of two reasons: 1) bidders are allowed to bid for the same instance using different price; 2) bidders are allowed to obtain a fraction of requested instances. For example, a bid vector of (10, 6) indicates that bidder A wants to bid for the first instance at \$10 and the second one at \$6. Consider another bidder B with bid vector (8, 7), and suppose two instances are available. If both A and B bid truthfully, each of them obtains one instance with a unit price of \$7. However, if B just needs one instance and changes the bid vector to (8, 1), then each of them obtains one instance with a unit price of \$6. From this example, we can see the bidders might have incentives to shade their bids, which causes the mechanism to be untruthful. Nonetheless, the aforementioned issues do not exist in DCSMs, where either all instances are granted when bids are higher than the spot price or none otherwise.

Theorem 1: In DCSMs, the uniform-price auction described in Section III-B is truthful.

Proof: We prove that no bidder has incentive to lie to improve its utility. Suppose that we have n customers bidding for M units. Each bidder has a demand of m^i units, with true valuation of v^i for each instance. Without loss of generality, let $v^1 \geq v^2 \geq \cdots \geq v^n$. If everyone bids truthfully, $k^u = \arg\min_k \sum_{1 \leq i \leq k} m^i > M$. We discuss three cases below:

- If the loser B^{k^u} lies by setting the bid to b^i, there are two cases: i) if $b^i \leq v^{k^u-1}$, then B^{k^u} remains a loser and thus has no incentive to lie; ii) $b^i > v^{k^u-1}$, then B^{k^u} becomes a winner and B^{k^u-1} becomes a loser, which implies that the spot price becomes v^{k^u-1}. Since $v^{k^u} < v^{k^u-1}$, B^{k^u}'s utility becomes $v^{k^u} - v^{k^u-1} \leq 0$. Therefore, B^{k^u} has no incentive to lie.
- If a loser $B^i (i > k^u)$ lies by setting the bid to b^i, there are also two cases: i) if $b^i \leq v^{k^u}$, then B^i is still a loser, and thus has no incentive to lie; ii) if $b^i > v^{k^u}$, then B^i could become a winner, in which case the spot price is at least v^{k^u}. Since $v^i < v^{k^u} (i > k^u)$, B^i's utility becomes $v^i - v^{k^u} \leq 0$. Therefore, B^i has no incentive to lie.
- If a winner $B^i (i < k^u)$ lies by setting the bid to b^i, there are two cases: i) if $b^i > v^{k^u}$, obviously the spot price remains the same, B^i is still the winner, and B^i's utility remains the same, i.e., $v^i - v^{k^u}$; ii) if $b^i \leq v^{k^u}$, B^i cannot be a winner, otherwise we have at least k^u winners, namely $B^k (1 \leq k \leq k^u)$. Therefore, $\sum_{1 \leq k \leq k^u} n^k \leq M$, which contradicts with the fact that $\sum_{1 \leq k \leq k^u} n^k > M$. Thus, in this case, B^i must be a loser, so B^i's utility decrease from $v^i - v^{k^u} > 0$ to 0, which means there is no incentive to lie.

■

D. True Valuation Adjustment

As shown in Section III-C, bidders will always bid their true valuation in every time slot. However, true valuation is likely to change as the job execution progresses and completion approaches. For instance, when a job approaches completion, the true valuation of the job's remaining instances increases as the cost of being interrupted and starting from the beginning (or a recent checkpoint) increases. We note that, checkpointing [27] techniques have been widely used to save state periodically so that interrupted jobs do not have to re-start from the beginning, that is, once interrupted, a job can start from the most recent check point. Particularly, we assume check points are set every t_c time slots, i.e., $t_1, t_{c+1}, t_{2c+1}, \cdots$. For ease of exposition, we only consider the first c time slots since the analysis of time slots between any two consecutive check points is similar. Check points in DCSMs are usually created and stored inexpensively [27]. Thus, we assume that the overhead of creating and retrieving a check point from storage is negligible. We assume bidders are risk-neutral. Thus in any time slot $t_j (1 \leq j \leq c)$, bidder i should select the true valuation v_j^i with the goal of minimizing the expected cost of processing the j-th time slot of the job. However, in a real system, bidding information is private. And bidders are unable to obtain the spot prices in advance, which is needed in the computation of the expected cost. In this case, one reasonable approach for bidders is to minimize the expected *maximum* cost of processing the j-th time slot of the job (note that the expected cost can always be upper bounded by this quantity). The effectiveness of this model is evaluated in Section V. In the remainder of Section III-D, we drop the superscript i for clarity of presentation. Particularly, let $\overline{C_j(b_j)}$ be the expected maximum cost of processing the j-th time slot of the job given its bid b_j. Accordingly, let $\overline{C_j^*}$ be the minimum of this expectation by properly selecting b_j. Note that $\overline{C_1^*} = v_1$ since we are at a checkpoint at t_1, and thus an out-of-bid event would not cost us anything. Define the function $f(b) : \mathbb{R}^+ \to [0, 1]$ as the probability function of failure (i.e., out-of-bid) given a bid b, and let $D_j^* = \sum_{1 \leq k \leq j} \overline{C_k^*}$. Note that the maximum cost for a bidder at t_j is b_j. Then we have

$$\overline{C_j}(b_j) = f(b_j)\cdot(D^*_{j-1}+b_j)+(1-f(b_j))\cdot b_j = f(b_j)\cdot D^*_{j-1}+b_j$$

Therefore, the true valuation v_j should be set as

$$v_j = b^*_j = \arg\min_{b_j}\{f(b_j)\cdot D^*_{j-1}+b_j\} \tag{2}$$

Accordingly,

$$D^*_j = D^*_{j-1}+\overline{C^*_j} = D^*_{j-1}+\overline{C_j}(b^*_j) = D^*_{j-1}+f(b^*_j)\cdot D^*_{j-1}+b^*_j$$

By calculating v_j and D^*_j using the above two equations iteratively, for each job, we can obtain the true valuations for each time slot (i.e., the computation sequence is $v_1, D^*_1, v_2, D^*_2, v_3, D^*_3\cdots$).

The above model has the following property.

Property 1: $\forall j_1, j_2, 1\le j_1 < j_2 \le c$, we have $v_{j_1}\le v_{j_2}$.

Proof: Suppose $\exists j_1, j_2, 1\le j_1 < j_2\le c$, such that $v_{j_1} > v_{j_2}$. Thus $f(v_{j_1})\le f(v_{j_2})$. At time slot t_{j_1}, according to (2), we have

$$f(v_{j_1})\cdot D^*_{j_1-1}+v_{j_1}\le f(v_{j_2})\cdot D^*_{j_1-1}+v_{j_2} \tag{3}$$

similarly at t_{j_2}, we have

$$f(v_{j_2})\cdot D^*_{j_2-1}+v_{j_2}\le f(v_{j_1})\cdot D^*_{j_2-1}+v_{j_1} \tag{4}$$

Adding (3) and (4), we have

$$(f(v_{j_2})-f(v_{j_1}))\cdot D^*_{j_1-1}\ge(f(v_{j_2})-f(v_{j_1}))\cdot D^*_{j_2-1} \tag{5}$$

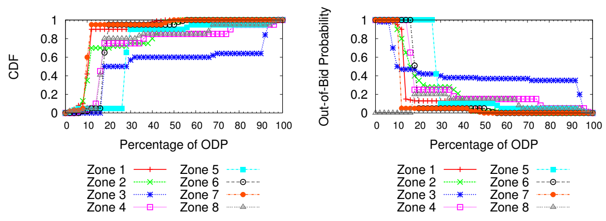
Since $D^*_{j_2-1} > D^*_{j_1-1}$ and $f(v_{j_2})-f(v_{j_1})\ge 0$, the only possibility for (5) to hold is $f(v_{j_2}) = f(v_{j_1})$. However, if we put this condition back to (3) and (4), we get $v_{j_1}\ge v_{j_2}$ and $v_{j_1}\le v_{j_2}$, which result in $v_{j_1} = v_{j_2}$. Contradiction. ■

This property of our model indicates that the true valuations is monotonically non-decreasing in time slots between two consecutive checkpoints. Such property is reflected in the experiment results in Section V.

The function $f(b)$ represents the probability of out-of-bid with a bid b, and thus guides bidders to adjust their true valuations during job's execution. The function can be approximated by analyzing the historical spot prices (e.g., In ASM, the spot prices for the latest 3 months are provided [7].). Figure 2(a) depicts the 8 cumulative distribution functions of the spot prices of the selected 8 zones[3] of ASM from January 1st, 2013 to April 1st, 2013, based on which we derive the out-of-bid probability functions as depicted in Figure 2(b). Note that each zone sells multiple types of instances. Here, we select 8 different types of instances from 8 zones, one from each zone, where each instance type forms a DCSM. In addition, the unit price of instances of different types varies due to differences in resources. For clarity of presentation, we pre-processed the historical spot prices in each zone by normalizing them by the on-demand price (ODP)[4] in the corresponding zone. This normalization eliminates the inherent price differences between various zones, thus giving us a clearer (although relative) representation of the spot prices.

[3]ASM consists of eight regions (e.g., US East, EU, South America), each of which has multiple zones (e.g., us-east-1a, eu-west-1a, sa-east-1a).

[4]ODP is a fixed price per instance-hour, with resources allocated using on-demand basis.



(a) CDFs of spot price (b) Out-of-bid probabilities

Fig. 2. Spot price of 8 zones in ASM from 1/1/2013 to 4/1/2013

IV. Performance Evaluation

In this section, we evaluate the proposed mechanism using *revenue* and *efficiency*, which are the two commonly used metrics in auction theory. We develop an algorithm for revenue computation and prove that our approach achieves optimal efficiency among all single-price auctions in DCSMs. In addition, we also evaluate the Quality of Service of the proposed mechanism, which can be measured by two metrics, namely *slowdown* and *waste*. We define the two metrics and develop algorithms to compute them.

A. Revenue

As discussed in Section III-B, the spot price at time slot t_j is equal to the highest losing bid, i.e, $p_j = b_j^{k^u_j}$, where $k^u_j = \arg\min_k \sum_{1\le i\le k} m^i > M$, and the revenue gained in t_j is $b_j^{k^u_j}\cdot(k^{u_j}-1)$ (see Definition 5 in Section II). Therefore, the key factor in calculating the revenue is k^u_j The index k^u_j is determined by the order of all the bids at t_j along with corresponding demand m^i, which is determined by three factors: 1) completed job requests in t_j no longer participate in the bidding; 2) winning job requests and losing job requests in the previous time slot change the bids based on the model discussed in Section III-D; 3) new arriving job requests submit new bids. In time slot t_j, let $\mathcal{R}_j = \{(m^i, b^i_j, l^i_j)(1\le i\le N_j)\}$ be the set of all job requests (including arrivals) at t_j, where m^i, b^i_j and l^i_j are the number of requested instances, bid and remaining job length, respectively of job request i at t_j. Note that m^i is a constant for job request i. Accordingly, let $\mathcal{R}^a_j\subseteq\mathcal{R}_j$ be the set of all arriving jobs at t_j. The algorithm for computing the revenue at t_j is given by Algorithm 1. Lines 3, 5 and 8 deal with completed jobs, remaining jobs, and newly arriving jobs, respectively. The revenue generated by our approach is studied in detail through simulation-based experiments in Section V.

B. Quality of Service

Recall that service could be interrupted without any notification in a DCSM, in which case jobs need to be rolled back to the most recent check point. Service interruptions result in at least two drawbacks for customers: 1) The completion time is postponed, which is bad for delay-sensitive jobs; 2) The total number of instance-hour required to complete a job increases, and thus the cost of completing jobs increases. To capture and evaluate the above two drawbacks, we use *slowdown* and

Algorithm 1 Revenue Calculation

Input: M_j, $\mathcal{R}_{j-1}$ and $\mathcal{R}_j^a$
Output: Revenue generated in t_j

```
1:  for all $r^i \in \mathcal{R}_{j-1}$ do
2:      if $l_j^i = 0$ then
3:          remove $r^i$ from $\mathcal{R}_j$
4:      else
5:          $b_j^i \leftarrow$ output from Eq. (2) in Section III-D
6:      end if
7:  end for
8:  $\mathcal{R}_j = \mathcal{R}_{j-1} \cup \mathcal{R}_j^a$
9:  Sort all $b_j^i$ in $\mathcal{R}_j$ in descending order.
10: $k_j^u \leftarrow \arg\min_k \sum_{1 \le k \le i} m^i > M_j$
11: return $b_j^{k_j^u} \cdot (k_j^u - 1)$
```

waste to evaluate the Quality of Service (QoS) in DCSMs. The definition of slowdown of a job in DCSMs is similar with that defined in [16]. The expected slowdown measures the expected delay of job completion in a DCSM.

Definition 7: A job J of length l enters the market at t_{j_1}, and leaves the market at t_{j_2} upon completion, then the slowdown of J is $\frac{l_s}{l}$, where $l_s = j_2 - j_1$.

Service interruptions not only result in delay of job completion, but also result in cost increases of job completion in the sense that a fraction of a job is executed in a duplicated manner when recovered from a check point. We use waste to measured the cost increase of completing a job in a DCSM.

Definition 8: Given Definition 7, suppose during $[t_{j_1}, t_{j_2}]$, J gains access to instances for l_w time slots[5], then the waste for J is $\frac{l_w}{l}$.

Accordingly, the slowdown and waste in a system are defined as the average slowdown and waste over all jobs in that system.

C. Efficiency

As discussed in Definition 6 of Section II, efficiency reflects social welfare generated by auctions. Efficiency might not be of great interest for providers in a short term. However, in the long run, a more efficient auction tends to be sustainable and attracts more customers. We have the following theorem regarding efficiency in DCSMs.

Theorem 2: Among all single-price auctions, uniform-price auction achieves optimal efficiency in DCSMs.

Proof: Suppose we have n customers bidding for M units. Each bidder has a demand of m^i units, with bid b^i for each instances. Without loss of generality, let $b^1 \ge b^2 \ge \cdots \ge b^n$. In uniform-price auction, the efficiency, as defined in Definition 6, $E^u = \sum_{1 \le i < k^u} m^i \cdot (b^i - c)$. The efficiency in any other single-price auction $\mathcal{S}$, $E^s = \sum_{1 \le i < k^s} m^i \cdot (b^i - c)$, where k^s is index of the highest losing bidder in $\mathcal{S}$. Note that $k^u = \arg\min_k \sum_{1 \le i \le k} m^i > M$, and thus $k^u \ge k^s$, which proves $E^u \ge E^s$. ■

[5]In the other $j_2 - j_1 - l_w$ time slots, as a loser, J does not need to pay.

V. Experiments

In this section, we evaluate our proposed mechanism through extensive simulations. We conduct a comparison study against ASM using four performance metrics, namely, revenue, efficiency, slowdown, and waste. Revenue and efficiency are introduced in Section II; slowdown and waste are defined in Section IV-B. The main objectives of these simulation-based experiments are i) gain understanding of how the inter-checkpointing time t_c and timeout T_{to} affect our proposed mechanism's performance. ii) quantify the gained performance improvements over the mechanism currently used in ASM; iii) evaluate how bidding flexibility helps improve the performance in our mechanism.

A. Datasets and Simulation Settings

We first describe the datasets and simulation settings used throughout the remainder of the paper.

We use a tuple (v^i, l^i, m^i) to represent a job request, where v^i, l^i, and m^i are initial true valuation, initial length, and number of requested instances, respectively, of job i. The job length l^i is modeled using a Geometric distribution $G(p)$ ($\frac{1}{p}$ is the expected job length), and m^i is uniformly distributed in $[1, 100]$[6]. We also vary the true valuation v^i using 4 distributions, denoted as HIST, DET(0.5,1), $\mathcal{N}(0.5, 0.09)$, and $\mathcal{N}(0.5, 0.01)$. Note that v^i is normalized using On-Demand Price as discussed in Section III-D. In HIST (setting 1-4 in Table II), the distribution of v^i is generated from the histogram in Figure 3, which is reported by ASM, and replicated from [1]). In DET(0.5,1) (setting 5-8 in Table II, as modeled

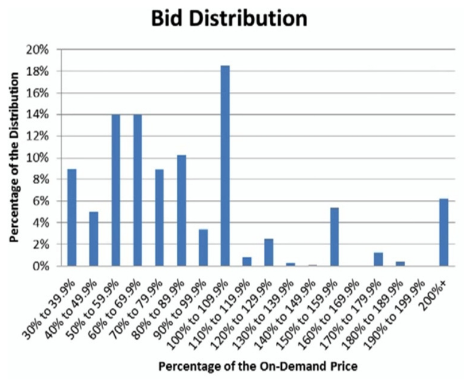


Fig. 3. Bid distribution replicated from [1].

in [11]), v^i is deterministic given a l^i with r^i increasing as l^i increases; specifically, it is obtained using a linear function which maps job lengths to values in the interval $[0.5, 1]$, i.e., for a job of length l^i, $v^i = \frac{0.5(l_{max} + l^i) - l_{min}}{l_{max} - l_{min}}$.[7] In these settings customers' bids are higher for longer jobs, which is intended to represent customers' aversion to having long-running instances

[6]In ASM, customers are not allowed to submit a job request with more than 100 instances [8].

[7]If $l_{max} = l_{min}$, every v^i has the same value, and thus can be selected from any value in [0.5,1], say 0.5.

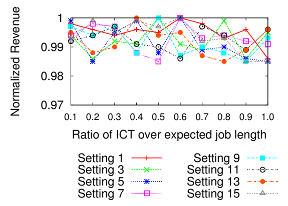
terminated [11]. In settings 9-16 in Table II, v^i is drawn from a normal distributions $\mathcal{N}(0.5, 0.01)$ and $\mathcal{N}(0.5, 0.09)$. The mean is set to 0.5 based on the report from Amazon that ASM results in average cost-savings of $50\% \sim 66\%$ as compared to that using ODP [1]. The two different standard deviations represent scenarios where v^i has large or small variance. In addition, we model the number of jobs arrival during a time slot using a Poisson distribution, i.e., $n_j^a \sim Poisson(\lambda)$, where n_j^a is the number of jobs arrival between t_{j-1} and t_j, with λ being the average number of jobs arrival in each time slot. We treat all jobs arrival between t_{j-1} and t_j as arriving at t_j. We note that $\lambda = 10$ represents a buyer's market (where supply exceeds demand), and $\lambda = 1000$ represents a seller's market (where demand exceeds supply). We generated 16 simulation settings in total by combining the job model with the job arrival model as summarized in Table II. For each setting, to obtain statistically significant results, we generated a sequence of job arrival batches for 100,000 time slots. In each time slot, the number of arrival job requests and the characteristics of jobs are specified by n_j^a and (v^i, m^i, l^i). In all simulations below, the number of available instances M is set to 50,000.

TABLE II
SIMULATION SETTINGS

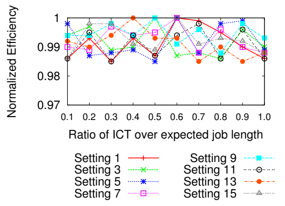
Setting ID	v^i	λ	p
1	HIST	10	0.1
2	HIST	1000	0.1
3	HIST	10	0.02
4	HIST	1000	0.02
5	DET(0.5, 1)	10	0.1
6	DET(0.5, 1)	1000	0.1
7	DET(0.5, 1)	10	0.02
8	DET(0.5, 1)	1000	0.02
9	$\mathcal{N}(0.5, 0.01)$	10	0.1
10	$\mathcal{N}(0.5, 0.01)$	1000	0.1
11	$\mathcal{N}(0.5, 0.01)$	10	0.02
12	$\mathcal{N}(0.5, 0.01)$	1000	0.02
13	$\mathcal{N}(0.5, 0.09)$	10	0.1
14	$\mathcal{N}(0.5, 0.09)$	1000	0.1
15	$\mathcal{N}(0.5, 0.09)$	10	0.02
16	$\mathcal{N}(0.5, 0.09)$	1000	0.02

B. Performance Study

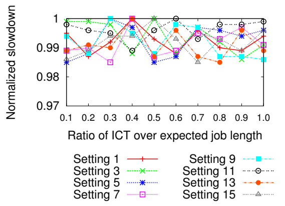
Experiment 1: Here, we study how the inter-checkpointing Time (ICT, see Section III-D) affects the performance of our mechanism, and how sensitive is the performance of our mechanism to the parameter settings. In determining the simulation settings of our experiments, rather than using the same range of t_c values for each experiment, we determine the range of values to use relative to the mean job length (i.e., $\mathbb{E}\{l\}$). In a real system, a larger $\mathbb{E}\{l\}$ is likely to result in a larger t_c, and a smaller $\mathbb{E}\{l\}$ is likely to result in a smaller t_c. We ran 16 simulations using the 16 setting in Table II, and computed the four metrics of interest. For each setting, we vary 10 values of t_c, and obtain the average values (defined in Section IV) of the four metrics for each value of t_c. Simulations are performed under both buyer's (settings with λ = 10) and seller's (setting with λ = 1000) markets, and the results are shown in Figure 4 and Figure 5, respectively. For clearer depiction, we present our results across different settings as a function of $\frac{t_c}{\mathbb{E}\{l\}}$. We also normalized the results, corresponding to the 10 different values of $\frac{t_c}{\mathbb{E}\{l\}}$, by their maximum values within each setting. The normalization is done for each metric in each setting. All simulation results are obtained with $95\% \pm 6\%$ confidence intervals.



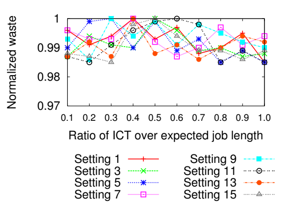
(a) Revenue



(b) Efficiency

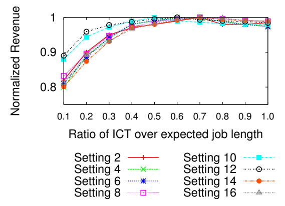


(c) Slowdown

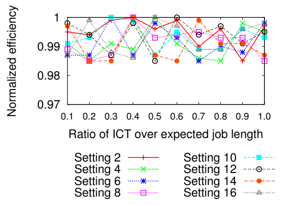


(d) Waste

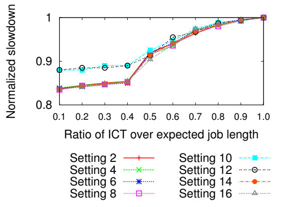
Fig. 4. Parameter tuning of ICT with $\lambda = 10$



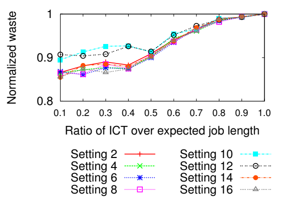
(a) Revenue



(b) Efficiency



(c) Slowdown



(d) Waste

Fig. 5. Parameter tunning of ICT with $\lambda = 1000$

From Figure 4, we observe that, in buyer's markets (settings with $\lambda = 10$), all four metrics are fairly insensitive to the value of $\frac{t_c}{\mathbb{E}\{l\}}$. For revenue, slowdown and waste, such insensitivity can be explained by the rare occurrences of interruptions occurred in the market, where available instances are essentially sufficient; the insensitivity of efficiency can be explained by the fact that efficiency is mainly determined by the number of instances sold (see Section II, Definition 6).

In seller's markets, we observe that the revenue (see Figure 5(a)) gradually increases as $\frac{t_c}{\mathbb{E}\{l\}}$ increases. The explanation is as follows: when the value of $\frac{t_c}{\mathbb{E}\{l\}}$ increases, according to Property 1 proved in Section III-D, bidders have a greater number of time slots (between two consecutive check points) in which to increase their bids. Therefore, it is likely that the lowest losing bid in the market increases, which results in the increase in revenue. Figures 5(b) shows that the efficiency is fairly insensitive to the value of $\frac{t_c}{\mathbb{E}\{l\}}$, which can be explained as in the case of buyer's markets. Slowdowns in seller's markets are shown in Figures 5(d), where we observe that slowdown grows slowly when $\frac{t_c}{\mathbb{E}\{l\}}$ is below a certain value, and then grows fast after that. The slow-growing part can be explained by the trade-off between reduction in the interruption probability and the cost of the interruption, once it happens. That is, under a relatively large value of $\frac{t_c}{\mathbb{E}\{l\}}$, the interruption probability is low, but the cost of an interruption, once it occurs, is relatively high; while under a relatively small value of $\frac{t_c}{\mathbb{E}\{l\}}$, the interruption probability is high, but corresponding cost of an interruption, once it occurs, is relatively low. After a certain value of $\frac{t_c}{\mathbb{E}\{l\}}$, the steep increase in the slowdown indicates that the cost of an interruption occurrence outweights the benefit due to reducing the interruption probability. Figure 5(d) shows similar results for waste, which can be explained similarly to the slowdown, because both metrics are impacted by interruptions.

Experiment 2: In a real system, some bidders might have less competitive bids due to budget constraints, and thus fail to gain access to requested instances for a long time. To avoid accumulating losers in the market, we assume such bidders will voluntarily leave the market after failure for T_{to} consecutive time slots of failing to obtain an instance. In this experiment, we study how the timeout (see Section III-D) affects the performance of our mechanism, and how sensitive is the performance of our mechanism to the parameter settings.

In our simulations, instead of using a specific range of T_{to}, we use values that are proportional to the mean job length (i.e., $\mathbb{E}\{l\}$). In a real system, a larger $\mathbb{E}\{l\}$ is likely to result in a larger T_{to}, and a smaller $\mathbb{E}\{l\}$ is likely to result in smaller T_{to}. Again, we ran simulations using the 16 setting in Table II, and computed the four metrics of interest. For each setting, we varied T_{to} over 10 different values and obtained the normalized average values (defined in Section IV) as was done in Exp. 1. To better depict the results across different settings, we present them as a function of $\frac{T_{to}}{\mathbb{E}\{l\}}$. The simulation results are shown in Figure 6 and Figure 7. All simulation results are obtained with $95\% \pm 4\%$ confidence intervals.

Again, from the four figures in Figure 6, we observe that, in buyer's markets (settings with $\lambda = 10$), all four metrics are fairly insensitive to the value of $\frac{T_{to}}{\mathbb{E}\{l\}}$. The explanation is the same with the one given in Exp. 1. In a seller's market (see 7(a)), we observe that revenue grows fast when $\frac{T_{to}}{\mathbb{E}\{l\}}$ is below a certain value, and then grows slowly. In the system, out-of-bid could be caused by two reasons: i) arrival of jobs with

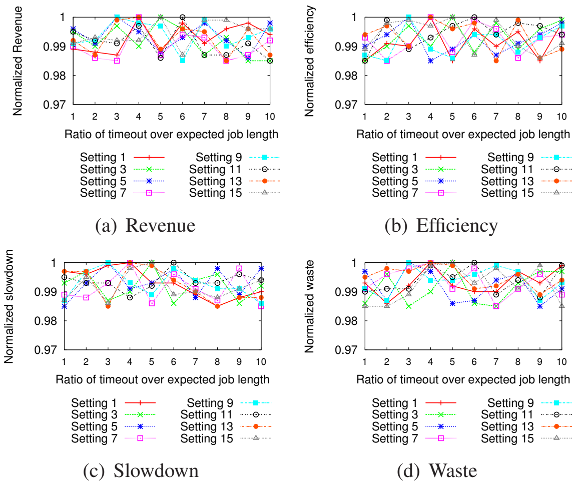


Fig. 6. Parameter tunning of timeout with $\lambda = 10$

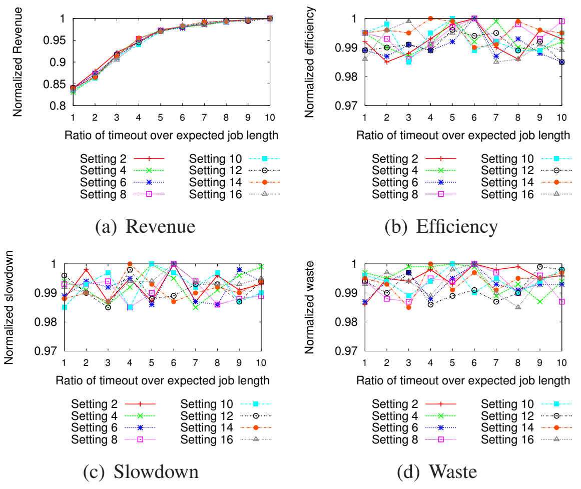


Fig. 7. Parameter tunning of timeout with $\lambda = 1000$

high bids; ii) bids are too low to get access to instances. The fast-growing part corresponds to reason i), where out-of-bid jobs just need to wait for jobs with high bids to complete, to gain access to instances. The slow-growing part corresponds to reason ii), where bids of out-of-bid jobs are too low to become winners, and thus jobs corresponding to those bids leaving does not significantly affect the system revenue.

Efficiency, slowdown and waste (see Figure 7(b), 7(c) and 7(d)) are fairly insensitive to the value of $\frac{T_{to}}{\mathbb{E}\{l\}}$. The case of efficiency can be explained by the same reason given in Exp. 1, i.e., efficiency is mainly determined by the number of instances sold. For the cases of slowdown and waste, note that the two metrics are computed over completed jobs, thus removing jobs with low bids that can hardly be completed from the system barely affects the two metrics.

Experiment 3: In this experiment, we compare the system performance under the following three mechanisms: 1) the mechanism suggested in [11], denoted as *ASM*, which is

TABLE III
PERFORMANCE COMPARISON

Setting ID	RUA vs. ASM				RUA vs. RUA-BF			
	Revenue	Efficiency	Slowdown	Waste	Revenue	Efficiency	Slowdown	Waste
1	10.28%	19.42%	11.75%	13.45%	4.71%	0.21%	5.41%	5.72%
2	18.62%	27.56%	16.19%	17.25%	6.03%	0.29%	6.11%	5.93%
3	10.58%	19.8%	11.94%	10.16%	5.12%	0.15%	5.82%	5.63%
4	17.74%	30.96%	15.14%	15.79%	5.95%	0.21%	6.25%	6.04%
5	11.83%	19.37%	9.38%	12.16%	5.06%	0.15%	5.08%	5.12%
6	17.96%	28.87%	15.94%	16.94%	5.48%	0.15%	6.15%	5.73%
7	11.99%	19.89%	11.33%	11.73%	4.61%	0.28%	5.84%	5.51%
8	15.60%	29.65%	15.31%	16.32%	5.97%	0.24%	6.31%	6.12%
9	11.09%	18.53%	12.44%	11.39%	4.81%	0.10%	5.52%	5.17%
10	15.86%	30.12%	16.90%	15.70%	6.14%	0.12%	6.41%	6.31%
11	12.78%	19.01%	10.96%	12.96%	4.56%	0.27%	5.22%	5.17%
12	15.76%	26.83%	16.63%	16.58%	5.68%	0.24%	6.21%	5.96%
13	10.71%	19.07%	9.54%	11.50%	5.49%	0.26%	5.22%	5.29%
14	18.18%	29.96%	16.87%	16.44%	5.90%	0.16%	5.73%	5.81%
15	10.85%	19.9%	10.82%	10.97%	5.71%	0.24%	5.32%	5.12%
16	16.78%	29.14%	16.37%	16.80%	6.52%	0.29%	5.91%	6.52%

likely the current mechanism used in ASM; 2) our proposed mechanism, denoted as *RUA*; 3) our proposed mechanism without bidding flexibility, denoted as *RUA-BF*. We ran simulations using the 16 setting in Table II under each of the three mechanisms, and computed the four metrics of interest. Columns 2-5 in Table III show the performance improvements achieved by RUA relative to ASM, where RUA outperforms ASM by an average of 14.2% in revenue, 24.3% in efficiency, 13.6% in slowdown, and 14.1% in waste. In particular, more significant improvements are observed in all seller's markets (i.e., settings with $\lambda = 1000$), where spot price better reflects the results of an auction. In a buyer's market, job requests are satisfied most of the time, and thus the benefits due to RUA are not as great; while in a seller's market, jobs are competing with each other for instances, and there are greater benefits from our approach. Columns 6-9 in Table III show the performance improvements achieved by RUA against RUA-BF. We observe that RUA outperforms RUA-BF by around 5% in revenue, slowdown and waste. This improvement implies that, from the improvements achieved by RUA against ASM, more than 35% of the revenue improvement (i.e., $\frac{5\%}{14.2\%}$), 36% (i.e., $\frac{5\%}{13.6\%}$) of the slowdown improvement, and 35% (i.e., $\frac{5\%}{14.1\%}$) of the waste improvement is contributed by the bidding flexibility. The performance improvements due to the bidding flexibility can be explained by the fact that bidders are likely to reduce the out-of-bid probability by gradually increasing their bids, which will result in interruption reduction and revenue increase. We also observe that efficiency is barely changed. The explanation is that high efficiency is achieved by a property of uniform-price auction, i.e., sell as many instances as possible. RUA does not impact efficiency significantly. All results are obtained with a confidence interval of $95\% \pm 7\%$.

In summary, we have the following four takeaways from the experiments. 1) In a buyer's market, the performance of our mechanism is fairly insensitive to the discussed two parameters (i.e., t_c and T_{to}), which indicates that our mechanism is fairly robust under various parameter settings; 2) In a seller's market, revenue, slowdown and waste are affected by the value of $\frac{t_c}{\mathbb{E}\{l\}}$, and t_c needs to be properly selected in a seller's market for desired trade-off between revenue and reliability; $\frac{T_{to}}{\mathbb{E}\{l\}}$ only significantly impacts revenue, which increases as the value of $\frac{T_{to}}{\mathbb{E}\{l\}}$ increases. However, such increase is negligible once the value of $\frac{T_{to}}{\mathbb{E}\{l\}}$ exceeds a certain value; 3) In both buyer's and seller's markets, efficiency in our mechanism is not significantly affected by the above two parameters; 4) Our mechanism outperforms ASM by an average of 14.2% in revenue, 24.3% in efficiency, 13.6% in slowdown, and 14.1% in waste, to which the bidding flexibility contributes 35% of the revenue improvement, 36% of the slowdown improvement, and 35% of the waste improvement.

VI. RELATED WORK

Since the start of the Amazon's Spot Market (ASM) in December 2009, its cost-saving property inspired researchers to develop provisioning, scheduling, and allocation algorithms for client applications. Meanwhile, its reliability problem (i.e., unexpected service termination due to out-of-bid events) motivated researchers to investigate fault-tolerant mechanisms for better quality of service.

A number of works focus on DCSMs' pricing strategy as well as on trying to understand the underlying principles. We give a brief overview of these efforts. [11] shows that the spot price is typically generated at random within a tight bound of a hidden reserve price, thus cannot reflect the real market demand and supply. Javadi *et al* [17] studied the price pattern of ASM for a one year period, and observed the bi-modality in the price distributions for all instance types, based on which they proposed a model with 3 or 4 components to capture characteristics of spot prices. They also discovered that inter-price time is around 60 minutes. In [22], the researchers study the profit-reliability trade-off in ASM using an order-statistics approach. In addition, they also illustrate and prove the hardness of pricing in ASM with the goal of profit maximization. In [13], Gelenbe *et al* studied the local/remote cloud in consideration of Quality-of-Service.

From customers' prospective, several works investigate how to utilize ASM for reliable and cost-effective service (e.g., MapReduce [12] and migration [26]). In [25], Wee shows that Spot price is 52.3% lower on average than the on-demand price.

In the ASM, cost-reduction can be achieved by optimal resource allocation [28] and appropriate provisioning [10][24]. Meanwhile, due to relatively low reliability in the ASM environment, fault-tolerant mechanisms (e.g., checkpointing [27]) are also studied. Other research directions include exploiting statistical approaches for market prediction, with the goal of performance and availability guarantees [19] and SLA satisfaction [9].

Other spot markets have also emerged in recent years. In [23], Google implemented a similar idea using a *clock auction*, and spotcloud.com [2] is a Priceline-style online cloud market for multiple providers to sell their unused capacity, where users pay what they bid for the instances instead of a uniform price.

Our work differs from all the works mentioned above in the sense that we are proposing an *improved* mechanism for DCSMs in consideration of four performance metrics, namely revenue, efficiency, slowdown and waste. Such mechanism is truthful and featured with bidding flexibility, with the goal of improving the performance of DCSMs. We conduct experimental studies with extensive settings, as well as a comprehensive comparison study against current ASM mechanism. The results show that our mechanism achieves significant performance improvement over ASM, and provide service providers with insights for better design and implementation of auction mechanisms for DCSMs.

VII. Conclusions and Future Work

In this paper, we focus on mechanism design for DCSMs with a single service provider. A truthful mechanism based on a repeated uniform-price auction is proposed, and bidding flexibility is also incorporated in this mechanism. Accordingly, a bidding adjustment model is introduced to make use of such flexibility. Our proposed mechanism is evaluated by four performance metrics, namely revenue, efficiency, slowdown and waste. The first two metrics are commonly used metrics in auction theory, and the last two metrics are introduced to evaluate the Quality of Service of a DCSM with our proposed mechanism. We prove that our approach achieves optimal efficiency among all single-price auctions in DCSMs. We conduct parameter tuning studies and a comprehensive comparison study against ASM, and the results showed our mechanism achieves significant performance improvement over ASM. Specifically, The results show that (1) The bidding adjustment model helps increase the revenue by an average of 5.4%, and decrease the slowdown and waste by an average of 4.9% and 5.7%, respectively; (2) Our model with a repeated uniform price auction outperforms the current Amazon Spot Market by averagely 14.2% in revenue, 24.3% in efficiency, 13.6% in slowdown, and by 14.1% in waste.

Future work includes two directions. From the auction theory perspective, we plan to consider markets with multiple types of instances and design corresponding approximation algorithm that can be used to estimate bounds on the four metrics (as defined in this paper). Moreover, we are developing performance models that can capture the dynamics of winners and losers in DCSMs and consequently can be used to analytically compute the four performance metrics.

References

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

[2] http://spotcloud.com/.

[3] http://www.njvc.com/news-and-events/news-releases/2012/njvc-cloudcuity-virtustream-government-marketplace.

[4] http://aws.amazon.com/ec2/instance-types/.

[5] https://forums.aws.amazon.com/thread.jspa?threadID=76964.

[6] https://forums.aws.amazon.com/thread.jspa?messageID=280850.

[7] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances-history.html/.

[8] http://aws.amazon.com/ec2/faqs/.

[9] A. Andrzejak, D. Kondo, and S. Yi. Decision model for cloud computing under SLA constraints. In *MASCOTS'10*, pages 257–266, 2010.

[10] D. Ardagna, B. Panicucci, and M. Passacantando. Generalized nash equilibria for the service provisioning problem in cloud systems. *IEEE Transactions on Services Computing*, 99(PrePrints):1, 2012.

[11] O. A. Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir. Deconstructing Amazon EC2 spot instance pricing. *Economics and Computation (TEAC), ACM Transactions on*.

[12] N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A. Tantawi, and C. Krintz. See spot run: using spot instances for mapreduce workflows. In *Proceedings of HotCloud'10*, pages 7–7, Berkeley, CA, USA, 2010.

[13] E. Gelenbe, R. Lent, and M. Douratsos. Choosing a local or remote cloud. In *NCCA*, pages 25–30, 2012.

[14] A. V. Goldberg, J. D. Hartline, and A. Wright. Competitive auctions and digital goods. In *Proceedings of SODA '01*, pages 735–744, 2001.

[15] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The cost of a cloud: research problems in data center networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73, Dec. 2008.

[16] M. Harchol-Balter, K. Sigman, and A. Wierman. Understanding the slowdown of large jobs in an m/gi/1 system. *SIGMETRICS Perform. Eval. Rev.*, 30(3):9–11, Dec. 2002.

[17] B. Javadi, R. K. Thulasiram, and R. Buyya. Characterizing spot price dynamics in public cloud environments. *Future Gener. Comput. Syst.*, 29(4):988–999, June 2013.

[18] V. Krishna. *Auction Theory*. Academic Press, 2002.

[19] M. Mazzucco and M. Dumas. Achieving performance and availability guarantees with spot instances. In *HPCC 2011*, pages 296 –303.

[20] P. Milgrom. *Putting Auction Theory to Work*. Cambridge University Press, 2004.

[21] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007.

[22] K. Song, Y. Yao, and L. Golubchik. Exploring the profit-reliability trade-off in amazon spot instance market: A better pricing mechanism. In *IWQoS*, 2013.

[23] M. Stokely, J. Winget, E. Keyes, C. Grimes, and B. Yolken. Using a market economy to provision compute resources across planet-wide clusters. In *IPDPS '09*, pages 1–8, Washington, DC, USA, 2009.

[24] W. Voorsluys, S. K. Garg, and R. Buyya. Provisioning spot market cloud resources to create cost-effective virtual clusters. In *ICA3PP (1)*, pages 395–408, 2011.

[25] S. Wee. Debunking real-time pricing in cloud computing. In *Proceedings of IEEE/ACM CCGRID'11*, pages 585–590, Washington, DC, USA.

[26] S. Yi, A. Andrzejak, and D. Kondo. Monetary cost-aware checkpointing and migration on amazon cloud spot instances. *Services Computing, IEEE Transactions on*, (99):1, 2011.

[27] S. Yi, D. Kondo, and A. Andrzejak. Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud. In *Proceedings of the IEEE CLOUD'10*, pages 236–243, Washington, DC, USA, 2010.

[28] Q. Zhang, E. Gürses, R. Boutaba, and J. Xiao. Dynamic resource allocation for spot markets in clouds. In *Proceedings of Hot-ICE'11*, Berkeley, CA, USA.