

Combinatorial Auction-Based Allocation of Virtual Machine Instances in Clouds

Sharrukh Zaman
Department of Computer Science
Wayne State University
Detroit, MI 48202
Email: sharrukh@wayne.edu

Daniel Grosu
Department of Computer Science
Wayne State University
Detroit, MI 48202
Email: dgrosu@cs.wayne.edu

Abstract—The current cloud computing platforms allocate virtual machine instances to their users through fixed-price allocation mechanisms. We argue that combinatorial auction-based allocation mechanisms are especially efficient over the fixed-price mechanisms since the virtual machine instances are assigned to users having the highest valuation. We formulate the problem of virtual machine allocation in clouds as a combinatorial auction problem and propose two mechanisms to solve it. We perform extensive simulation experiments to compare the two proposed combinatorial auction-based mechanisms with the currently used fixed-price allocation mechanism. Our experiments reveal that the combinatorial auction-based mechanisms can significantly improve the allocation efficiency while generating higher revenue for the cloud providers.

I. INTRODUCTION

Cloud computing enables individuals and small to medium enterprises satisfy their computational needs with no or minimum upfront costs of acquiring hardware and software. On the other hand, cloud providers benefit by commercializing their huge computing resources through the cloud computing platform. A cloud computing platform abstracts the underlying physical resources from the users by providing them with the view of virtual machines (VMs). This enables easy management and pricing of the resources. Currently, the majority of cloud providers price their computing resources based on the ‘size’ of the VM instances offered. They define different types of VM instances by specifying the number and speed of processors, the memory size, the bandwidth allocation, etc. There are two ways to ‘purchase’ VM instances: pay as you go and long term contract. In both cases users pay fixed prices per unit time for using the resources; the only difference is that by committing to a long term contract they usually pay less per unit of time for using the same resource.

We argue that the currently used fixed-price schemes for allocation and pricing of resources have several drawbacks. First, they are not economically efficient [1], that is, they cannot guarantee that the user that values a bundle of VM instances the most, gets it. Second, fixed prices do not necessarily reflect the equilibrium prices that arise from market demand and supply. This may lead to lower than optimal revenue for the cloud providers. Finally, since in cloud computing platforms resources are sold for a period of time, it is desirable that user requests be evenly distributed throughout the day (over a

larger unit of time). Fixed-price methods do not provide users with any incentive for selecting their execution time-frames in such a way that the system load is balanced over time.

The inefficiencies in solving the resource allocation problem in clouds mentioned above can be best addressed by auction-based mechanisms. Among different types of auctions, the *combinatorial auctions* are the most suitable for solving the VM pricing and allocation problem in clouds. In combinatorial auctions, the participants bid for bundles of items rather than individual items [2]. This enables bidders to express their valuations in a more meaningful way, especially when the items they require are complementary to each other. To illustrate this, let us consider the following example. A cloud service provider offers ‘small’ and ‘large’ VM instances. A user needs to run an application which requires two small and two large VM instances. It is more meaningful for her to be able to bid for the entire bundle she needs rather than bidding for each VM instance separately. Bidding for each VM instance separately involves the risk of ending up acquiring just a subset of her required set of instances. The motivation behind our work is that by designing and deploying combinatorial auction-based mechanisms for allocating VM instances, the cloud providers can guarantee fairness to their users as well as enjoy higher revenues and a balanced load on their systems.

Application of auctions, however, is not entirely new to the cloud computing community. After allocating computing resources for the long-term and on demand users, Amazon EC2 sells the remaining virtual machines (instances) through an auction called Spot Instances [3]. In this auction, the bidders specify their demand (i.e., the number and type of the instances) and the maximum price they are willing to pay. Amazon periodically runs the auction with active bidders to determine the current price and then users with bids higher than that price are provided with their desired instances. All users pay the same price per instance which is computed by the auction. A user getting the allocation may be terminated at a later point if the auction-determined price goes beyond her bid. This approach is different from combinatorial auctions because one single price is determined based on market supply and demand (i.e., equilibrium) and all bidders pay the same price per item regardless of how much they value

the item. On the other hand, in combinatorial auctions, each winning bidder's payment is calculated based on her and other bidders' valuations. Therefore, combinatorial auctions are capable of capturing the competition between users for resources and generating higher revenue for the providers. From Amazon's initial effort of using auction-based allocation, it is reasonable to expect that cloud providers will be interested in more efficient allocation and pricing schemes in the near future. Combinatorial auctions will clearly be one of the most desirable allocation schemes in this regard. This is supported by their successful application in various fields ranging from selling wireless spectrum to transportation procurement for large industries [2].

A. Our Contribution

We formulate the problem of allocating VM instances in clouds as a combinatorial auction problem. The objective of this problem is to efficiently allocate VM instances of several types to several users requesting a set of VM instances of different types. To solve this problem, we propose two combinatorial auction-based allocation mechanisms. These two mechanisms are obtained by extending the mechanisms proposed in [4] and [5]. The mechanism proposed in [4] considers a combinatorial auction problem where a user can include at most one item of a particular type in her requested bundle. We relax this condition to allow users requesting more than one item of a given type. Also, this mechanism is suitable for combinatorial auctions with many types of items where each type of items has few instances. We extend the mechanism so that it can be applied to the VM allocation problem where there are few types of items and each type has many instances.

The other mechanism we propose is an extension of the greedy mechanism presented in [5]. This mechanism determines the allocation based on the valuation of the users and the total number of items they request. We extend the mechanism in [5] so that it considers the relative sizes of the VM instances and show that the properties of the original mechanism are maintained. We compare the two proposed combinatorial auction-based mechanisms with the fixed-price based allocation mechanism used by Microsoft in their Windows Azure platform [6]. We investigate the relative performance of these three allocation mechanisms by performing extensive simulation experiments. The experiments show that the proposed combinatorial auction-based mechanisms clearly outperform the fixed-price mechanism in terms of resource utilization, generated revenue, and allocation efficiency. We analyze the results and provide recommendations on where to use the proposed mechanisms.

B. Related Work

Several researchers investigated economic models for resource allocation in computational grids. Wolski *et al.* [7] compared commodities markets and auctions in grids in terms of price stability and market equilibrium. Gomoluch and Schroeder [8] simulated the double auction protocol for

resource allocation in grids and showed that it outperforms the conventional round-robin approach. Das and Grosu [9] proposed a combinatorial auction-based protocol for resource allocation in grids. They considered a model where different grid providers can provide different types of computing resources. An 'external auctioneer' collects these information about the resources and runs a combinatorial auction-based allocation mechanism where users participate by requesting bundles of resources. The major difference between the present work and the one presented in [9] is that we are considering allocation of VM instances of a single cloud provider whereas in [9], the authors considered the problem of allocating different types of physical resources from multiple grid providers.

Wang *et al.* [10] studied different economic and system implications of pricing resources in clouds. Buyya *et al.* [11] proposed an infrastructure of federated clouds for auction-based resource allocation across multiple clouds. Altmann *et al.* [12] proposed a marketplace for resources where the allocation and pricing are determined using an exchange market of computing resources. In this exchange, the service providers and the users both express their ask and bid prices and matching pairs are granted the allocation and removed from the system. We are considering a combinatorial auction mechanisms instead of an exchange in our paper. In this case, the cloud providers need not express any valuations of the items (or ask prices). Also, combinatorial auctions are more useful in the cloud computing context since usually users require a particular bundle of resources in order to run their applications.

The complexity of solving the combinatorial auctions, specifically the winner determination problem, was first addressed in [13]. Sandholm [14] proved that solving the winner determination problem is computationally hard. Lehmann *et al.* [5] studied combinatorial auctions with single-minded bidders and devised a greedy mechanism for combinatorial auctions. In this paper, we extend this mechanism in order to solve the VM allocation and pricing in clouds problem. Archer *et al.* [4] considered another case of single-minded bidders where multiple identical copies are available for different types of items. They provided a mixed integer programming based algorithm for winner determination and showed theoretically that their solution performs better than generalized solutions for this special case. We extend this mechanism such that it can be used to solve the VM allocation and pricing problem we consider in this paper. Cramton *et al.* [2] provides good foundational knowledge on combinatorial auctions.

C. Organization

The rest of the paper is organized as follows. We formally define the problem we are considering in the next section. In Section III, we present the mechanisms we consider for solving the VM allocation problem. In Section IV, we describe the experimental results. In Section V, we conclude the paper and present possible directions for future research.

II. VIRTUAL MACHINE ALLOCATION PROBLEM

The cloud providers set different configurations of VM instances that the users can request. A user requests VM instances of different types and pays the cloud provider for the time she uses them. The prices for different types of instances for short-term use are fixed by the cloud providers in advance. Another possibility is that a user sets up a long-term contract if she requires the resources for a long period of time, in which case she may obtain them for a lower price. Here we consider the problem of efficient allocation and pricing of VM instances for short-term use.

We define the *virtual machine allocation problem (VMAP)* as follows. Assume that the allocation and prices are decided periodically by a given mechanism. Let the interval between two such decisions be ‘one unit of time’. VMAP considers allocating the VM instances for one unit of time. Assume that a cloud provider has m different types of virtual machines $VM_1, \dots, VM_m$. The relative computing capabilities (based on number and speed of CPUs, memory, etc.) of these VMs are characterized by a vector $\mathbf{w} = (w_1, \dots, w_m)$ where $w_i \in \mathbb{R}_+$, $i = 1, \dots, m$. We also assume that $w_1 = 1$ and $w_1 \leq w_2 \leq \dots \leq w_m$. To illustrate this, we consider the types of instances currently offered by Microsoft Azure Platform: Small (CPU 1.6 GHz, Memory: 1.75 GB, Storage: 225 GB), Medium (CPU 2x1.6 GHz, Memory: 3.5 GB, Storage: 490 GB), Large (CPU 4x1.6 GHz, Memory: 7 GB, Storage: 1 TB), and Extra large (CPU 8x1.6 GHz, Memory: 14 GB, Storage: 2 TB). In this example, VM_1, VM_2, VM_3, VM_4 are the Small, Medium, Large, and respectively Extra large VM instances. The weight vector characterizing the VM instances is $\mathbf{w} = (1, 2, 4, 8)$.

Let us assume that k copies of each type of VMs are available for allocation at a given instance of time. To simplify the presentation we assume that the system has the same number of each type of VM instances available. This assumption does not affect the generality of the results presented in this paper. There are n users $u_1, \dots, u_n$, each requesting a set of VM instances and revealing how much she values that particular set. That is, a user u_j is requesting VMs from the cloud provider by placing a bid $B_j = (r_1^j, r_2^j, \dots, r_m^j, v_j)$, where $r_i^j \in \{0, 1, \dots, k\}$ is the number of instances of type VM_i user u_j requires in her bundle and v_j is her valuation for this bundle, i.e., the maximum price she is willing to pay for using the requested VMs for one unit of time. Here we consider the users to be single-minded bidders. A single-minded bidder u_j desires only a specific bundle of items S_j , values that bundle at v_j , and has the following valuation function [5],

$$v(S) = \begin{cases} v_j & \text{if } S_j \subseteq S \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The goal of the VM allocation problem, given the set of users U and their bids, is to determine the set of winners $W \subseteq U$ and the price the winners have to pay to the cloud provider. User u_j is a winner if she receives her requested bundle of VM instances. The price user u_j pays to the cloud provider is denoted by p_j . We formally define the VM allocation problem as follows:

Virtual Machine Allocation Problem (VMAP)

Determine the set of winners, $W \subseteq U$, and payment p_j for each user u_j , $j = 1, \dots, n$, such that

$$\sum_{j: u_j \in W} r_i^j \leq k \quad i = 1, \dots, m \quad (2)$$

$$0 \leq p_j \leq v_j \quad \text{if } u_j \in W \quad (3)$$

$$p_j = 0 \quad \text{if } u_j \notin W \quad (4)$$

The constraint in Equation (2) ensures that the users are allocated at most k copies of each type of VMs. Equations (3) and (4) ensure that the winners pay at most their valuations and the losers do not pay at all.

Note that VMAP does not have an objective function. The most reasonable objective function would be to maximize the cloud provider’s revenue, but very little is known about revenue maximization in the context of combinatorial auctions [15]. Combinatorial auctions are usually designed to maximize the sum of the bidders’ valuations, i.e., $\max \sum_{j=1}^n v_j$. Since valuation is a measure of willingness to pay, maximizing the sum of the valuations usually generates more revenue for the resource provider than a fixed-price allocation does. On the other hand, given the prices of each type of VM instance, a fixed-price allocation mechanism does not have an objective function to maximize. Therefore, VMAP is formulated here as a feasibility problem with the constraints that are to be satisfied by all types of solutions. We shall introduce other constraints and/or objective functions when we discuss the proposed mechanisms for solving VMAP.

III. VIRTUAL MACHINE ALLOCATION MECHANISMS

In this section, we present three mechanisms that solve VMAP. The first, called FIXED-PRICE, is the fixed-price mechanism currently used by several cloud service providers [16], [17]. The next two mechanisms are the proposed combinatorial auction mechanisms, CA-LP (Combinatorial Auction - Linear Programming) and CA-GREEDY (Combinatorial Auction - Greedy). CA-LP is an extended version of the mechanism proposed by Archer *et al.* [4]. The mechanism proposed in [4] solves a problem similar to VMAP by using linear programming relaxation and randomized rounding. We extend that mechanism so that it is able to solve VMAP. CA-GREEDY is an extension of the mechanism proposed by Lehmann *et al.* [5]. The mechanism proposed in [5] is the best possible approximate solution of the combinatorial auction with single-minded bidders. However, this is a general purpose mechanism that does not assume any relative importance of the items being allocated. We extend this mechanism by incorporating the weights of different types of VMs as described in Section II. We now describe each mechanism in detail.

A. FIXED-PRICE Mechanism

The FIXED-PRICE mechanism defines a fixed-price vector $\mathbf{f} = \{f_1, \dots, f_m\}$, where f_i is the price a user has to pay to use one instance of VM_i for one unit of time. The mechanism

allocates VM instances to the users in a first-come, first-served basis until the resources are exhausted. It also makes sure that in order to get the requested bundle, the valuation of user u_j is at least F_j , where F_j is the sum of the fixed prices of each VM instance in her bundle. It also makes sure that the allocation does not exceed the number of available VM instances of each type. The set of users receiving the requested bundle is denoted by W . A user pays the sum of the fixed-prices of each VM instance in her allocated bundle. We formally define the mechanism as follows:

FIXED-PRICE Mechanism

1. *Receive requests from users.*
 - a. For $j = 1 \dots n$,
Receive $(r_1^j, \dots, r_m^j, v_j)$ from user u_j
2. *Allocation.*
 - a. Sort users according to their time of placing the request, from earliest to latest. (Here we assume $u_1, u_2, \dots, u_n$ as the order.)
 - b. Initialize $W \leftarrow \phi$
 - c. For $j = 1 \dots n$,

$$F_j \leftarrow \sum_{i=1}^m r_i^j f_i$$
if $(v_j \geq F_j)$ and

$$(r_i^j + \sum_{u_{j'} \in W} r_i^{j'} \leq k, i = 1, \dots, m)$$
then $W \leftarrow W \cup \{u_j\}$
3. *Payment.*
User u_j pays, $p_j = F_j$, if $u_j \in W$; and $p_j = 0$, if $u_j \notin W$.

B. Combinatorial Auction-Based Mechanisms

The general combinatorial auction problem can be informally stated as determining the allocation and prices of bundle of items such that the sum of the user's valuations is maximized. In a combinatorial auction user valuations are expressed on bundles of items rather than on individual items.

A desired property of a combinatorial auction mechanism is *truthfulness*. A mechanism is truthful if the participants benefit most when they reveal their true valuations to the mechanism. A participant's benefit in a combinatorial auction is expressed by her *utility*, which is defined as the difference between the valuation she receives from the resource allocation and the price she pays to the mechanism. An ideal truthful mechanism determines the optimal allocation that maximizes the sum of the valuations and computes payments such that each participant maximizes her utility only by reporting her true valuation to the mechanism. A truthful mechanism helps the bidders in that they do not need to compute a complex strategy or assume other users' strategies while making their bids.

The winner determination problem of combinatorial auctions is an NP-hard problem [14]. Therefore, research has been conducted to find approximate solutions to combinatorial auctions. In order to obtain a truthful approximation mechanisms that solve the winner determination problem, few issues need to be addressed [18]. The approximation

algorithm needs to be *monotone*. In a monotone allocation algorithm, a bidder can only increase her chance of getting her requested bundle by reporting a higher valuation or by requesting fewer items in her bundle. A monotone allocation algorithm allows finding the so called *critical value* of a winning bidder, which is the minimum she needs to bid in order to get her requested bundle. A user has to pay her critical value to the mechanism if she wins. For some combinatorial auction problems, randomization is involved in the winner determination and/or the payment calculation algorithm. In that case, the goal of the resulting mechanism is to ensure that the participants maximize their expected utility by bidding their true values. Such mechanisms are *truthful in expectation*. We discuss the useful properties of our proposed mechanisms as we describe them below.

We now describe the two proposed mechanisms CA-LP and CA-GREEDY, the modifications to the original mechanisms from [4] and [5] and the reason behind those modifications. We also show that the mechanisms are truthful.

1) *CA-LP Mechanism*: Archer *et al.* [4] considered a combinatorial auction problem similar to VMAP. The difference is that in their case bidders can request at most one copy of each item type (i.e., $r_i^j \in \{0, 1\}$), whereas in the VMAP, users can request multiple copies of each type of item (i.e., $r_i^j \in \{0, 1, \dots, k\}$). We modify the winner determination algorithm of the original mechanism such that it is able to solve VMAP. The algorithm for the calculation of payment is kept the same as in [4] because it maintains its property when applied to VMAP with the modified winner determination algorithm. We present it here for completeness. The CA-LP mechanism is given below.

CA-LP Mechanism

1. *Collect Bids.*
 - a. For $j = 1 \dots n$,
Collect bid $B_j = (r_1^j, \dots, r_m^j, v_j)$ from user u_j
2. *Winner Determination.*
 - a. Set $k' \leftarrow (1 - \epsilon)k$, where $0 < \epsilon < 1$
 - b. Solve the following linear program

$$\max \sum_{j=1}^n x_j v_j \quad (5)$$

subject to

$$\sum_{j=1}^n x_j r_i^j \leq k' \quad i = 1, \dots, m \quad (6)$$

$$0 \leq x_j \leq 1 \quad j = 1, \dots, n \quad (7)$$

- c. Initialize $W \leftarrow \phi$
- d. For each user u_j , taken in descending order of x_j ,
 - Generate a random number $y_j \in [0, 1]$
 - If $y_j \leq x_j$ and

$$r_i^j + \sum_{j': u_{j'} \in W} r_i^{j'} \leq k \quad i = 1, \dots, m$$

then $W \leftarrow W \cup \{u_j\}$

3. *Payment.* (same as in [4])

- a. For each user $u_j \in W$,
 - Perform binary search for v'_j in the range $[0, v_j]$
 - Set valuation of u_j as v'_j in Equation (5)
 - Solve the LP, let x'_j be the fractional allocation computed for u_j
 - Until a v'_j is found such that, setting valuation of u_j less than v'_j generates $x'_j < y_j$ and setting the valuation greater than v'_j generates $x'_j > y_j$. This v'_j is the ‘critical value’.
 - $p_j \leftarrow v'_j$
- b. For each user $u_j \notin W$, $p_j \leftarrow 0$

In the above mechanism, the objective of the linear program is to find a vector of ‘fractional allocations’ $\mathbf{x} = \{x_1, \dots, x_m\}$ that maximizes the sum of the users’ valuations (Equation (5)). In step 2.a., the total number of available VM instances of each type is reduced to k' , which is then used in the constraint in Equation (6). This constraint limits the allocation of each type of VMs to k' . Using k' instead of k in this constraint helps reducing the probability of over allocating the VMs during the randomized rounding in step 2.d. This constraint is a modification of the constraint used in the mechanism presented in [4] by letting r_i^j take any value rather than only 0 and 1. The next constraint bounds the fractional allocation values between 0 and 1.

In step 2.d., user u_j is selected as a winner with probability x_j , if this allocation does not violate any constraint in Equation (6). This operation is executed in order of decreasing x_j values so that if there is a violation in the constraint, the user assigned a lower x_j value is not included in W . This step is another modification of the winner determination algorithm presented in [4]. In the original algorithm, users are first included in W with a probability of x_j and if constraint (6) is violated for any item, all users requesting that item are excluded from W . This method is suitable for auctions where many different types of items are sold and each type of item has only a few copies. But in the context of VMAP, this approach will significantly affect the allocation since each type of VM has many copies and there are only a few different types of VMs. For example, let a cloud provider offer four types of virtual machines, 500 instances of each type. Suppose that after rounding, VM_1 becomes over allocated. According to [4] we need to discard all the users that requests any instance of VM_1 in her bundle. This results in 500 unsold VM instances. The other VMs requested by those users also becomes unallocated. This is the reason we cannot use the original winner determination algorithm from [4] to solve VMAP.

The payment in CA-LP mechanism is calculated by finding a bid for a winning user u_j that is minimum for her to be a winner based on the y_j value generated in step 2.d. This component of the mechanism is the same as in the original mechanism.

The authors of [4] proved that the original mechanism is truthful in expectation. We claim that CA-LP maintains this property.

Claim 1: CA-LP mechanism is truthful in expectation.

Proof: (Sketch.) It is shown in [4] that the x_j values calculated by the LP in step 2.b. are monotone with respect to the user valuations, i.e., a user u_j can increase her probability of winning by increasing her valuation. Now, in step 2.d., we exclude the users with the lowest x_j values from being a winner in case an item is oversold. Since this decision is made with respect to the x_j value and a user can increase her x_j value by bidding higher, we claim that the winner determination algorithm of CA-LP is monotone.

The payment calculated by CA-LP is the critical value that is the minimum a user must bid to get her requested bundle allocated. Her reported valuation only helps decide whether she will be a winner, but she has to pay this critical value when she wins, no matter how large her valuation is. Because of these and following the results given in [4] the mechanism maintains the property of truthfulness in expectation. ■

2) *CA-GREEDY Mechanism:* Lehmann *et al.* [5] proposed a $\sqrt{M}$ -approximation mechanism for combinatorial auctions with single-minded bidders where the total number of items is M . We extend this mechanism by redefining M to be the weighted total number of VM instances, i.e., $M = k \sum_{i=1}^m w_i$. We define the ‘size’ s_j of the bundle in bid B_j requested by user u_j as $s_j = \sum_{i=1}^m w_i r_i^j$. In the original mechanism, s_j is defined as $s_j = \sum_{i=1}^m r_i^j$ i.e., the total number of items requested in B_j . Here we present our CA-GREEDY mechanism.

CA-GREEDY Mechanism

1. *Collect Bids.*

- a. For $j = 1 \dots n$,

Collect bid $B_j = (r_1^j, \dots, r_m^j, v_j)$ from user u_j

2. *Winner Determination.*

- a. Initialize $W \leftarrow \phi$
- b. For $j = 1 \dots n$, $s_j \leftarrow \sum_{i=1}^m r_i^j w_i$
- c. Reorder user bids such that

$$v_1/\sqrt{s_1} \geq v_2/\sqrt{s_2} \geq \dots \geq v_n/\sqrt{s_n}$$
- d. For $j = 1 \dots n$,
 - If for all $i = 1 \dots m$, $r_i^j + \sum_{u_{j'} \in W} r_i^{j'} \leq k$,
then $W \leftarrow W \cup \{u_j\}$

3. *Payment.*

- a. For each $u_j \in W$,
 - $W'_j \leftarrow \{u_l : u_j \notin W \Rightarrow u_l \in W\}$
 - $l \leftarrow$ minimum index in W'_j
 - if $W'_j \neq \phi$ then $p_j \leftarrow (v_l/\sqrt{s_l})\sqrt{s_j}$
 - else $p_j \leftarrow 0$
- b. For each $u_j \notin W$, $p_j \leftarrow 0$

The mechanism determines the winners by first ranking the users in decreasing order of their bid density (i.e., $v_1/\sqrt{s_1}$) and then greedily allocating them starting from the top of the list. Before allocating a new bundle the mechanism verifies that the

new allocation does not exceed the number of available VM instances of each type (step 2.d.). The payment p_j a winner u_j pays is calculated by multiplying $\sqrt{s_j}$ with the highest bid density among the losing players who would win if u_j would not be a winner (step 3.a and 3.b). That is, the winner pays the critical value.

Our mechanism is different from the original in the way the bid density is calculated. The original mechanism computes s_j as the total number of items in B_j , while in our case we consider s_j to be the weighted sum of the number of VM instances requested in B_j . Another difference is in the way our mechanism verifies if the capacity is exceeded for each type of VM instance (step 2.d). We claim that CA-GREEDY has the same approximation ratio as the original greedy mechanism and it is truthful as well.

Claim 2: CA-GREEDY is a truthful mechanism that computes a $\sqrt{M}$ -approximate solution to VMAP, where $M = k \sum_{i=1}^m w_i$.

Proof: (Sketch.) The mechanism presented in [5] is a $\sqrt{M}$ -approximation of a general combinatorial auction, where M is the total number of items. In the case of VMAP, $M = k \sum_{i=1}^m w_i$. According to the definition of $\mathbf{w}$, w_i is the number of VM_1 instances equivalent to one VM_i instance. Therefore, in VMAP, M is the total number of equivalent instances of VM_1 that are available. The mechanism presented in [5] provides a $\sqrt{M}$ -approximation solution when there are M items in total, therefore the CA-GREEDY mechanism also generates an $\sqrt{M}$ -approximation solution to the VMAP.

Now, we show that the winner determination algorithm of CA-GREEDY is monotone and the payment calculated for a winner is the minimum payment she has to pay to win her bid. From step 2.c. of the mechanism, it is clear that a user can increase her chance of winning by increasing her bid. Also, a user can increase her chance to win by decreasing the weighted sum of the items. For example, a user requesting two small and two large VM instances will be higher in the order than a user requesting one small and three large instances for the same valuation, although the numbers of VMs requested are the same. Therefore, the winner determination algorithm of CA-GREEDY is monotone with respect to user bids considering the relative computing capability of different types of VMs. Also, the payment of a winning bidder is set to the minimum payment she has to pay in order to get the bundle, that is the critical value payment. Therefore, the mechanism is truthful. ■

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

The simulation for one instance of VMAP runs for five simulation days. During each simulation, a maximum of 100,000 users are generated. A group of users are created five times an hour, i.e., every twelve minutes. Therefore, an average of about 167 users are generated every twelve minutes. We add a deviation randomly chosen from [-20%, +20%] to this number to determine the actual number of users generated at a particular time. We invoke all three mechanisms every hour,

TABLE I: Simulation Parameters

Parameter	Description	Value(s)
m	Types of VMs	4
k	Available VMs of each type	500, 1000, 2000
$\mathbf{w}$	Relative weight of VMs	(1, 2, 4, 8)
N	Maximum number of users	100000
r^{min}, r^{max}	Min. & Max. VM instances of each type in a bundle	0, 5
v^{max}	Maximum valuation	1, 2, 5, 10
t^{min}, t^{max}	Min. & Max. execution time	1, 10
d^{min}, d^{max}	Min. & Max. deadline	2, 10
π	Distribution of users	(10%, 40%, 50%), (20%, 30%, 50%), (20%, 40%, 40%), (30%, 30%, 40%)
ρ	Scale factor for bundle size	(2, 1.5, 1), (3, 2, 1)
λ	Scale factor for valuation	(2, 1.5, 1), (3, 2, 1)
τ	Scale factor for execution time	(2, 1.5, 1), (3, 2, 1)
δ	Scale factor for deadline	(.5, .67, 1), (.33, .5, 1)

with all the users generated during that hour and all users from previous time slots that are still active. An active user is one whose task has not been finished or the task deadline has not been reached. Each mechanism computes the allocation and pricing for the next one hour and keeps track of the users' status separately.

Users are of three categories: type-1, type-2, and type-3. Type-1 users are the most demanding, type-3 the least, and type-2 fall in between. User demands are characterized by four factors: number of requested VMs, valuation, duration for which the bundle is requested, and a deadline by which the task has to be finished. For example, type-1 users request more VMs than the other two types of users, request the VMs for longer period of time, have the highest valuations, and have stricter deadlines than the others. Also, each category of users are generated at particular times of the day. A simulation day is divided into three time slots: peak (8am–4pm), off-peak (4pm–midnight), and night (midnight–8am). Type-1 users are generated (and hence submit their bids) during the peak hours only. Type-2 users submit bids during peak and off-peak hours and type-3 users can be generated any time of the day. To compare with real life scenarios, we can roughly consider that type-1 users are the big corporations, type-2 are the large and medium businesses, and type-3 are the small businesses and individual users.

We assume that the cloud provider offers four types of VM instances: small, medium, large, and huge (VM_1, VM_2, VM_3 , and VM_4). We also set their relative weights to $\mathbf{w} = (1, 2, 4, 8)$ and their fixed prices to $\mathbf{f} = (0.12, 0.24, 0.48, 0.96)$. This corresponds to the fixed-price model used in Microsoft's Windows Azure Platform [6]. Therefore, each user u_j 's bid is a 5-tuple $(r_1^j, r_2^j, r_3^j, r_4^j, v_j)$, where r_i^j is the number of requested instances of VM_i and v_j is her valuation. User u_j 's task is characterized by the tuple (t_j, d_j) , where t_j is the unit of time the resources are requested for and d_j is the time by which u_j 's job needs to be completed.

To generate user bids, first the type of the user is randomly

chosen from the user distribution. Then, random numbers are generated from the ranges $[r_i^{min}, r_i^{max}]$, $[0, v_i^{max}]$, and $[t_i^{min}, t_i^{max}]$ and assigned to the r_i^j s, v_j , and t_j , respectively. These values are then scaled with a factor associated with the category of the user. For example, user factors for r_i^j s are given by the vector ρ . Therefore, after generating r_i^j values from the given range, they are multiplied with ρ_1 to determine the actual value when the user is of type-1. The vector λ denotes the scaling factors for valuation for different types of users. After generating a random number within the range $[0, v_i^{max}]$, we multiply that value with the λ value according to the type of the user generated. Similarly, vectors τ and δ denote the user factors for scaling the time required and the deadline. The deadline is determined by selecting a random number, scaling it, and then adding the result to t_j . We list all simulation parameters in Table I.

To create different instances of VMAP, we vary the parameters that have effect on the user distribution, demand, and payment resulting from the allocation. Thus, we choose four different distributions of type-1, type-2, and type-3 users given by the following tuples: (10%, 40%, 50%), (20%, 30%, 50%), (20%, 40%, 40%), and (30%, 30%, 40%). We consider four values of v_i^{max} , 1, 2, 5, and 10, which give four ranges of valuations (0-1), (0-2), (0-5), and (0-10). Table I lists the parameters, their description, and the range of values they take. Combining all these values with each other, our simulation experiment runs 768 different instances of VMAP.

B. Analysis of Results

The experimental results show that the proposed combinatorial-based auction mechanisms have clear advantages over the fixed-price mechanism for solving the VMAP. Here we discuss their overall performance and then investigate the effect of different parameters on various performance metrics such as generated revenue, utilization, runtime and the number of users served by the system.

We show the average performance of the three mechanisms over all combination of parameter values in Figure 1. We see that CA-LP outperforms the other two mechanisms in all three metrics except the running time. Around 8% of the 100,000 users could complete their tasks when CA-LP mechanism is used. We also see that the overall utilization of the resources and the revenue generated are the best for CA-LP. This is because the linear program has as objective maximizing the sum of the valuations, which eventually generates higher revenue by utilizing as many machines as possible while satisfying the constraint given in Equation (6). Utilizing more machines allocates more users and therefore more users can finish their tasks. On the other hand, CA-GREEDY allocates users based on their relative valuation. Therefore, it cannot always utilize resources as much as CA-LP can. But the running time of the CA-LP is prohibitively high because the payment calculation involves repeated solving of the linear program. The FIXED-PRICE mechanism obviously has the lowest running time because it only allocates users on a first-come, first-served basis. The CA-GREEDY mechanism has

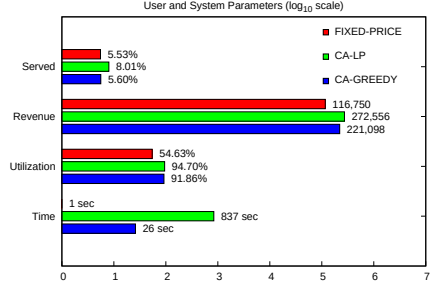


Fig. 1: Overall performance of the mechanisms

very low running time compared to CA-LP since its only major computation is to sort the list of users.

We now investigate different performance metrics with varying simulation parameters. In Figure 2a, we show the revenue generated for different ranges of user valuations. We see that low user valuations most adversely affect the FIXED-PRICE mechanism. This is because it does not allocate requested bundles to users having valuations below the fixed-price range. But the auctions can generate revenues because they calculate the payments from the user valuations. The revenue increase at the same rate from valuation ranges (0-1) to (0-5). Then, for the valuation range (0-10), we see a sharp rise in revenue generated by the auction mechanisms, while the FIXED-PRICE mechanism's revenue does not increase that much. This is because the price for an average-sized bundle is 4.5 according to the fixed prices we set. FIXED-PRICE mechanism's revenue is bounded by the fixed prices, therefore it cannot take advantage of higher user valuations. As shown in Figure 2b, our experiments reveal that the rate of resource utilization obtained by the auction-based mechanisms is not affected by the valuation ranges. The utilization obtained by FIXED-PRICE increases as the valuation range increases, and it is lower than that obtained by combinatorial auction mechanisms for all the ranges of valuations except for (1-10) range.

We now examine how the three mechanisms deal with different types of users. Recall that type-1 users are the most demanding and type-3 users are the least demanding. In Figure 2c we plot the percentage of users that could complete their tasks. We observe that the FIXED-PRICE mechanism serves type-3 users the most. This is because it only considers the order in which users arrive. Type-1 and type-2 users have shorter deadlines and therefore leave the system if they do not get the allocation within a few allocation events. On the other hand, type-3 users have longer deadlines, and therefore they are active longer and eventually get the allocation once the users that entered the system earlier finish their tasks. CA-LP also served more users of type-3 than users of other types, yet it served more users compared to the FIXED-PRICE mechanism in every category. Here we see a nice property of CA-GREEDY that is, it serves more type-1 users than other mechanisms. It also serves more users of type-1 than other

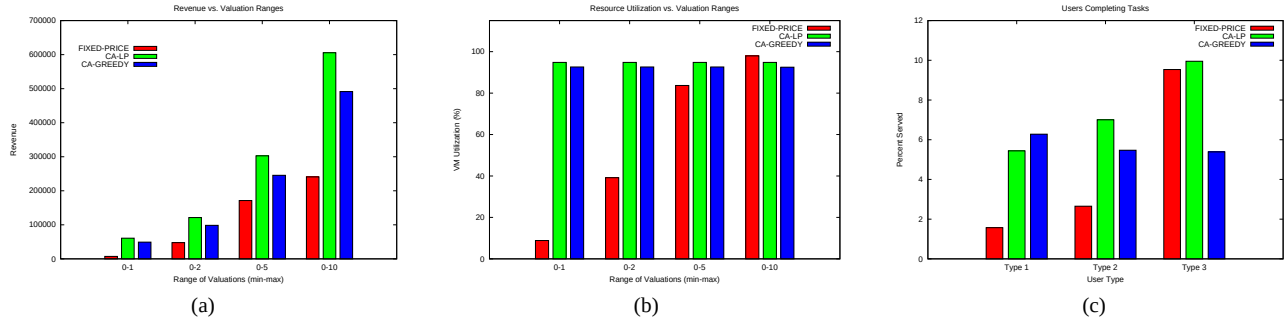


Fig. 2: (a) Revenue vs. ranges of valuation; (b) Effect of valuation on machine utilization; (c) Percentage of served users.

user types. This is because CA-GREEDY takes care of the relative valuations, which are on average the highest for type-1 users. The FIXED-PRICE mechanism has the least partially served users because of the first-come, first-served nature.

In summary, we can conclude that combinatorial auction-based allocation and pricing mechanisms are more desirable over the fixed-price based ones currently used by cloud providers. CA-LP is a better choice when the objective is to obtain higher revenue and higher utilization of resources. However, we have to limit the application of CA-LP to instances with a small number of users, because otherwise the execution time will be prohibitive. CA-LP can also be a choice when auctions are run at longer intervals. On the other hand, CA-GREEDY can be applied to cloud systems with any number of users being able to generate high revenue and resource utilization with very low execution time. CA-GREEDY is a better choice when the objective of VMAP is to maximize the social welfare. It is also worth mentioning that the CA-LP mechanism is designed for bidders with known bundles [4]. Therefore, this mechanism is vulnerable to users who bid for false bundles if that can benefit them. On the other hand, CA-GREEDY is a truthful mechanism with respect to both valuations and the bundles requested. Considering all the above aspects, we recommend using CA-GREEDY for general purpose VM allocation problems in clouds.

V. CONCLUSION

We investigated the applicability of combinatorial auction-based mechanisms for allocation and pricing of VM instances in cloud computing platforms. We performed extensive simulation experiments and concluded that combinatorial auction-based mechanisms are clearly a better choice for VM allocation in clouds. Based on experimental data and on the theoretical properties of the mechanisms, we recommend that the CA-GREEDY mechanism should be the choice for general purpose VM instance allocation problems while the CA-LP mechanism can be reserved for special scenarios. Future work includes the deployment of the proposed mechanisms on an experimental cloud computing testbed and the development of other market-based VM allocation protocols.

ACKNOWLEDGMENT

This work was supported in part by NSF grant DGE-0654014.

REFERENCES

- [1] R. Wang, "Auctions versus posted-price selling," *The American Economic Review*, vol. 83, no. 4, pp. 838–851, 1993.
- [2] P. Cramton, Y. Shoham, and R. Steinberg, *Combinatorial Auctions*. The MIT Press, 2005.
- [3] Amazon, "Amazon EC2 spot instances." [Online]. Available: <http://aws.amazon.com/ec2/spot-instances/>
- [4] A. Archer, C. Papadimitriou, K. Talwar, and E. Tardos, "An approximate truthful mechanism for combinatorial auctions with single parameter agents," *Internet Mathematics*, vol. 1, no. 2, pp. 129–150, 2005.
- [5] D. Lehmann, L. I. O'Callaghan, and Y. Shoham, "Truth revelation in approximately efficient combinatorial auctions," *Journal of the ACM*, vol. 49, no. 5, pp. 577–602, 2002.
- [6] Microsoft, "Windows Azure FAQ." [Online]. Available: <http://www.microsoft.com/windowsazure/faq/>
- [7] R. Wolski, J. S. Plank, J. Brevik, and T. Bryan, "Analyzing market-based resource allocation strategies for the computational grid," *Intl. J. of High Performance Comp. Appl.*, vol. 15, no. 3, pp. 258–281, 2001.
- [8] J. Gomoluch and M. Schroeder, "Market-based resource allocation for grid computing: A model and simulation," in *Proc. 1st Intl. Workshop on Middleware for Grid Computing*, 2003, pp. 211–218.
- [9] A. Das and D. Grosu, "Combinatorial auction-based protocols for resource allocation in grids," in *Proc. 6th IPDPS Workshop on Parallel and Distributed Scientific and Engineering Computing*, 2005.
- [10] H. Wang, Q. Jing, R. Chen, B. He, Z. Qian, and L. Zhou, "Distributed systems meet economics: Pricing in the cloud," in *Proc. 2nd USENIX Workshop on Hot Topics in Cloud Computing*, 2010.
- [11] R. Buyya, R. Ranjan, and R. N. Calheiros, "InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Proc. 10th Intl. Conf. on Algorithms and Architectures for Parallel Processing*, 2010, pp. 13–31.
- [12] J. Altmann, C. Courcoubetis, G. D. Stamoulis, M. Dramitinos, T. Rayna, M. Risch, and C. Bannink, "GridEcon: A market place for computing resources," in *Proc. Workshop on Grid Economics and Business Models*, 2008, pp. 185–196.
- [13] M. H. Rothkopf, A. Pekec, and R. M. Harstad, "Computationally manageable combinatorial auctions," *Management Science*, vol. 44, no. 8, pp. 1131–1147, 1998.
- [14] T. Sandholm, "Algorithm for optimal winner determination in combinatorial auctions," *Artificial Intelligence*, vol. 135, no. 1-2, pp. 1–54, 2002.
- [15] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*. Cambridge University Press, 2007.
- [16] Amazon, "Amazon EC2 pricing." [Online]. Available: <http://aws.amazon.com/ec2/pricing/>
- [17] Microsoft, "Windows Azure offers." [Online]. Available: <http://www.microsoft.com/windowsazure/offers/>
- [18] A. Archer and E. Tardos, "Truthful mechanisms for one-parameter agents," in *Proc. 42nd IEEE Symp. on Foundations of Computer Science*, 2001, pp. 482–491.