Maximizing Cloud Provider Profit from Equilibrium Price Auctions

Ulrich Lampe, Melanie Siebenhaar, Apostolos Papageorgiou, Dieter Schuller, Ralf Steinmetz
Multimedia Communications Lab (KOM)
Technische Universität Darmstadt
Darmstadt, Germany
Email: ulrich.lampe@KOM.tu-darmstadt.de

***Abstract*—Auctioning constitutes a market-driven scheme for the allocation of cloud-based computing capacities. It is practically applied today in the context of Infrastructure as a Service offers, specifically, virtual machines. However, the maximization of auction profits poses a challenging task for the cloud provider, because it involves the concurrent determination of equilibrium prices and distribution of virtual machine instances to the underlying physical hosts in the data center. In the work at hand, we propose an optimal approach, based on linear programming, as well as a heuristic approach to tackle this *Equilibrium Price Auction Allocation Problem* (EPAAP). Through an evaluation based on realistic data, we show the practical applicability and benefits of our contributions. Specifically, we find that the heuristic approach reduces the average computation time to solve an EPAAP by more than 99.9%, but still maintains a favorable average solution quality of 96.7% in terms of cloud provider profit, compared to the optimal approach.**

***Keywords*-Cloud Computing; IaaS; Equilibrium; Auction; Allocation; Optimization; Optimal; Heuristic**

I. Introduction

A. Motivation

In the past few years, cloud computing has gained tremendous interest both among practitioners and researchers. One of the essential ideas of this novel paradigm consists in the provision of Information Technology (IT) in a utility-like manner [1]. In the context of an envisioned cloud computing market, the decision whether to supply IT capacities in-house or lease them from the cloud is – aside from strategical considerations – largely determined by the price [2]. One potential instrument to achieve a market-based pricing and thus, efficient allocation of computing capacities, are auctions [3]. In this respect, Amazon Web Services[1] has not only been one of the pioneers in the cloud computing domain, but also among the first to employ auctions as a complement to traditional fixed-price schemes.

Specifically, in the *Spot Instances* system[2], cloud users submit bids to Amazon for Infrastructure as a Service (IaaS) offers in the form of Virtual Machines (VMs). Each bid states the desired number of instances and the maximum willingness to pay for a specific VM type. At a periodic time interval, Amazon determines an equilibrium price for each type. Users whose bids exceed (or meet) the equilibrium price are (partially) served with the desired instances. All users pay the identical equilibrium price – rather than their respective bid price – per instance.

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

While many researchers have previously assumed that the price-setting mechanism is market-driven, i. e., determined by supply and demand [3]–[5], more recent research indicates that this is not the case. According to Agmon Ben-Yehuda et al. [6], the publicly posted equilibrium prices are likely selected at random from a small predefined interval. In the opinion of the aforementioned authors, this indicates a low demand in the Spot Instance system at present.

However, it is safe to assume that with increasing acceptance and utilization of cloud computing offers, auctions will gain in popularity for the efficient allocation of computing capacities. Based on this notion, in the work at hand we examine how cloud providers can maximize their profit using equilibrium price auctions.

B. Research Problem

When applying equilibrium price auctions for the allocation of capacities, a cloud provider faces two distinct yet linked challenges. First, the provider has to determine specific equilibrium prices for each offered VM type. Second, the provider has to distribute the VM instances, which have been requested in the served (or satisfied) bids, among the Physical Machines (PM) instances in her/his data center. In this process, the cloud provider will commonly pursue the objective of maximizing profit. This profit is given by the difference between the revenue from the served bids and the operating costs of the PM instances. In the remainder of this paper, the combination of both challenges is referred to as *Equilibrium Price Auction Allocation Problem* (EPAAP).

We assume that the cloud provider operates a limited number of PM types, which provide restricted resource supplies. These supplies are specified in terms of certain resource types, e. g., processor power or memory. Of each PM type, a restricted number of instances are available in the cloud provider's data center(s). These PM instances may be independently powered on or off. Each active instance leads to a fixed operating cost due to, e. g., idle power consumption and maintenance demands.

We further assume that the cloud provider offers a predefined set of VM types, which exhibit certain resource demands. Due to these resource demands, each individual VM instance imposes additional variable operating costs on the PM instance that hosts it, due to, e. g., increased power consumption.

Lastly, we assume that in accordance with the Spot Instances system, the cloud users may continuously submit or cancel bids. In contrast, the allocation process (i. e., the pricing of VM types and distribution of VM instances) is only periodically conducted by the cloud provider.

The remainder of this paper is structured as follows: In Section II, we introduce a formal notation for the EPAAP and subsequently describe two optimization approaches, an optimal and a heuristic approach. Section III describes our evaluation procedure and the obtained results. In the subsequent Section IV, we provide an overview of related work. Lastly, Section V concludes the paper with a summary and outlook on future work.

II. Optimization Approaches

A. Formal Notations

As a basis for the optimization approaches that are presented in the following subsections, we introduce a formal notation for the EPAAP. First, we define the basic entities:

- $V \subset \mathbb{N}$: Set of offered VM types.
- $P \subset \mathbb{N}$: Set of available PM types.
- $R \subset \mathbb{N}$: Set of regarded resource types.

Based on the previous definitions, the characteristics of and relations between the basic entities can be further specified:

- $RD_{vr} \in \mathbb{R}^+$: Resource demand of VM type $v \in V$ for resource type $r \in R$.
- $RS_{pr} \in \mathbb{R}^+$: Resource supply of PM type $p \in P$ for resource type $r \in R$.
- $CF_p \in \mathbb{R}^+$: Fixed operating cost of PM type $p \in P$ per utilized instance.
- $CV_{pv} \in \mathbb{R}^+$: Variable operating cost of PM type $p \in P$ per hosted instance of VM type $v \in V$.
- $n_p \in \mathbb{N}$: Available instances of PM type $p \in P$ in the cloud provider's data center(s).

The bids that have been submitted by the cloud users are formalized using the following constructs:

- $B \subset \mathbb{N}$: Set of submitted bids.
- $W_b \in \mathbb{R}^+$: Specified price for bid $b \in B$, i. e., maximum willingness to pay.
- $T_b \in V$: Specified VM type for bid $b \in B$.

For reasons of simplicity and without loss of generality, we assume that each bid $b \in B$ specifies an individual request for *one* VM instance. Thus, if a user bids for $\eta \in \mathbb{N}$ VM instances of the same type at an identical price – as it is the case in the Spot Instances system – this results in η individual bids in the set B.

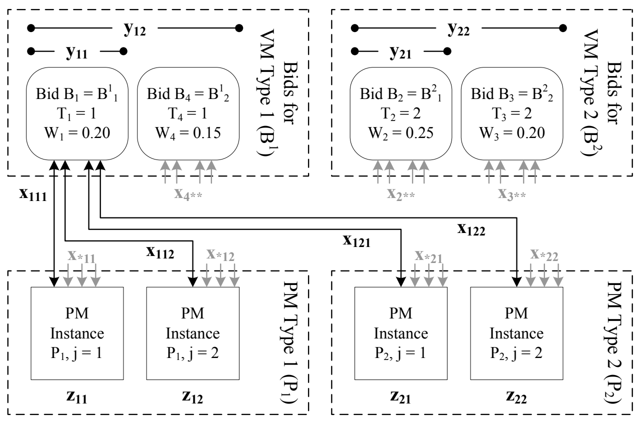

Figure 1. Schematic overview of the optimization model, depicting the decision variables (in bold) and most relevant entities. A portion of the links has solely been sketched (in gray) to improve readability.

Lastly, for reasons of convenience, we infer the following definitions from the previous specifications:

- $B^v \subseteq B$: Set of submitted bids for VM type $v \in V$.
- $W^v \subseteq W$: Set of prices for the bids in set B^v.
- $m_v \in \mathbb{N}$: Overall number of submitted bids for VM type $v \in V$.

Without loss of generality, we establish that the bids in the set B are given in *monotonically decreasing* order of the corresponding prices. That is, it holds that $W_b \geq W_{b'}$ for $b, b' \in B, b < b'$. The same applies for the respective subsets of bids, i. e., B^v.

B. Optimal Allocation Approach

To compute an optimal solution to the EPAAP, we transfer the problem definition from Subsection I-B into a mathematical optimization model. The result is given in Model 1 and will be explained in detail in the following. In order to promote an easier interpretation of the model, we first discuss the significance of the decision variables.

In Equation 12, x, y, and z are defined as binary decision variables. x is the primary decision variable, whereas y and z can be considered auxiliary decision variables. Specifically, x_{bpj} indicates whether the bid b has been served and assigned to a PM instance of type p with the running index j or not. y_{vk} indicates whether for a VM type v, exactly the first k bids are served or not. Lastly, z_{pj} represents whether a PM instance of type p with the running index j is utilized, i. e., powered on, or not. An overview of the optimization model, which highlights the relations between the decision variables and the most important entities, is depicted in Figure 1.

Equation 1 specifies the objective of the optimization model, namely the maximization of profit. The profit comprises three components. The first component is the revenue that is generated through the served bids. For that matter, y_{vk} also indicates whether the k-th bid for VM type v corresponds to the equilibrium price or not. The second component is the fixed operating cost of all utilized PM instances, while the third component represents the additional variable operating cost due to the hosted VM instances.

Equations 2 and 3 link the decision variables x and y and x and z respectively. Equation 4 assures that the resource demands of all VM instances are met by the resource supplies of the respective PM instance that host them. Equation 5 guarantees that an individual bid cannot be served more than once. Equation 6 ensures that solely one equilibrium price for each VM type can be set.

Equation 7, for performance reasons, restricts the solution space by preferring PM instances of the same type with a lower running index over those with a higher running index. Equation 8, also for performance reasons, excludes dominated solutions from the solution space. A solution is referred to as dominated if another solution exists that would yield a higher or equal revenue, even if a smaller number of bids for a specific VM type is served.

Equations 9 through 11 define valid running indices for VM and PM instances. The definition of the latter is based on two observations: First, the number of utilized PM instances cannot exceed the available number of instances of this type, n_p. Second, in the theoretical case that all bids were served and the requested VM instances were each assigned to an individual PM instance of the same type, no more than $|B|$ instances would be required.

As can be seen, Model 1 constitutes a Linear Program (LP), or more specifically, Binary Integer Program (BIP). This class of optimization problems can be solved using well-known methods from the field of Operations Research, most notably, the *Branch and Bound* (B&B) algorithm [7]. While the B&B algorithm can be very efficient in some cases, it is still based on the principle of enumeration, i. e., in the worst case, all potential solutions have to be examined [8].

Specifically, for a BIP, the solution space grows exponentially with the number of decision variables. As can be observed from Model 1 (notably, Equations 1 and 12), the number of decision variables increases quadratically with the number of bids and linearly with the number of PM types. Accordingly, the computational complexity of the optimal allocation approach is exponential and corresponds to $\mathcal{O}(2^{|B|^2 * |P|})$.

C. Heuristic Allocation Approach

For real-life application scenarios involving thousands of bids, the optimal allocation approach may be problematic due to its exponential growth in computational complexity. Thus, we have developed a heuristic approach that trades reductions in computation time against potentially sub-optimal solutions.

The principle idea is to initially determine the equilibrium prices for all VM types, such that the *expected* profit from the served bids is maximized. Subsequently, these served bids – or more specifically, the VM instances that have been requested in these bids – are cost-efficiently distributed across the physical hosts. Accordingly, the approach is split into two phases, *VM pricing* and *VM distribution*.

Model 1 Optimal Allocation Model

$$\text{Maximize } Pr(x,y,z) = \sum_{v\in V, k\in K_v} y_{vk} * k * W_k^v \quad (1)$$
$$- \sum_{p\in P, j\in J_p} z_{pj} * CF_{pj} - \sum_{b\in B, p\in P, j\in J_p} x_{bpj} * CV_{pT_b}$$

subject to

$$k * y_{vk} \le \sum_{1\le i\le k, p\in P, j\in J_p} x_{B_i^v pj} \quad \forall v\in V, k\in K_v \quad (2)$$

$$z_{pj} \ge x_{bpj} \quad \forall b\in B, p\in P, j\in J_p \quad (3)$$

$$\sum_{b\in B} x_{bpj} * RD_{T_b r} \le RS_{pr} \quad \forall p\in P, j\in J_p, r\in R \quad (4)$$

$$\sum_{p\in P, j\in J_p} x_{bpj} \le 1 \quad \forall b\in B \quad (5)$$

$$\sum_{k\in K_v} y_{vk} \le 1 \quad \forall v\in V \quad (6)$$

$$z_{pj} \ge z_{pj'} \quad \forall p\in P, j\in J_p, j'\in J_p, j<j' \quad (7)$$

$$y_{vk} = 0 \quad \text{if } k * W_k^v \le k' * W_{k'}^v$$
$$\forall k\in K_v, k'\in K_v, k>k' \quad (8)$$

$$K_v = \begin{cases} \{1,...,m_v\} & \text{if } m_v > 0 \\ \emptyset & \text{else} \end{cases} \quad \forall v\in V \quad (9)$$

$$J_p = \begin{cases} \{1,...,\min(n_p,|B|)\} & \text{if } n_p > 0 \\ \emptyset & \text{else} \end{cases} \quad \forall p\in P \quad (10)$$

$$J_p, K_v \subset \mathbb{N} \quad (11)$$

$$\begin{aligned} x_{bpj} &\in \{0,1\} \quad \forall b\in B, p\in P, j\in J_p \\ y_{vk} &\in \{0,1\} \quad \forall v\in V, k\in K_v \\ z_{pj} &\in \{0,1\} \quad \forall p\in P, j\in J_p \end{aligned} \quad (12)$$

The procedure for the second phase is inspired by a heuristic for the distribution of software services across VM instances, which we have introduced in our previous work [9]. This heuristic, in turn, adapts concepts that are frequently applied for solving the well-known Knapsack problem [7]. For additional details, we refer to our previous publication.

The pseudo code for the two phases is provided in Listing 1 and 2. In accordance with the previous subsection, x_{bpj} represents the main binary decision variable. $Q_v \in \mathbb{R}^+$

Listing 1 Algorithm for VM Pricing

```
1:  $S \leftarrow \emptyset$
2:  for all $v \in V$ do
3:      $g_v \leftarrow 0, CS_v \leftarrow 0$
4:      for all $p \in P$ do
5:          if $p.\mathsf{canHost}(v) =$ true then
6:              $g_{vp} \leftarrow \min(n_p, |B|)$
7:              $g_v \leftarrow g_v + g_{vp}$
8:              $CS_{vp} \leftarrow CV_{pv} + CF_p * \frac{1}{|R|} \sum_{r \in R} \left( \frac{RD_{vr}}{RS_{pr}} \right)$
9:              $CS_v \leftarrow CS_v + g_{vp} * CS_{vp}$
10:         end if
11:     end for
12:     if $g_v > 0$ then
13:         $CS_v \leftarrow CS_v / g_v$
14:         $\hat{k}_v \leftarrow 0$, $\hat{E}_v \leftarrow 0$, $Q_v \leftarrow \infty$
15:         for $k = 1 \rightarrow m_v$ do
16:             $E_v \leftarrow k * (W_k^v - CS_v)$
17:             if $E_v > \hat{E}_v$ then
18:                 $\hat{E}_v \leftarrow E_v$, $\hat{k}_v \leftarrow k$, $Q_v \leftarrow W_k^v$
19:             end if
20:         end for
21:         $S \leftarrow S \cup \{B_1^v, ..., B_{\hat{k}_v}^v\}$
22:     end if
23: end for
```

denotes the equilibrium price for each VM type v. Lastly, the set $S \subseteq B$ contains the bids that will be served.

In the first phase, VM pricing, we initially estimate the cost of serving an individual bid for each VM type v. This serving cost CS_{vp} of a VM type v on each PM type p corresponds to the partial fixed and additional variable operating cost of a respective PM instance. The partial fixed operating cost, in this context, is the fixed operating cost of a PM instance, multiplied by the ratio between the resource demands of VM type v and the resource supplies of PM type p. The individual serving costs are aggregated into an weighted average serving cost CS_v across all suitable PM types for each VM type v, using the respective weights g_{vp}. Suitable, in this respect, means that a PM type can host a VM type subject to the given resource constraints (lines 3-13). On the basis of these serving costs and the initial bid prices, we determine a favorable number $\hat{k}_v \in \mathbb{N}$ of bids for each VM type v that should be served. The number is considered favorable if the expected profit $E_v \in \mathbb{R}$, i. e., the difference between the revenue from the bids and the expected serving costs, becomes maximal. In the same step, the equilibrium price Q_v for each VM type v is inferred. The served bids are stored in the set S, which, in accordance with the set B, we assume to be ordered by monotonically decreasing bid prices (lines 14-21). The corresponding VM instances are considered for distribution in the following phase.

Listing 2 Algorithm for VM Distribution

```
1:  for all $p \in P$ do
2:      $j_p \leftarrow 0$
3:  end for
4:  repeat
5:      $\hat{p} \leftarrow \emptyset$, $\hat{U} \leftarrow 0$
6:      for all $p \in P$ do
7:          if $j_p < n_p$ then
8:              $L_p \leftarrow \emptyset$, $I_p \leftarrow \emptyset$
9:              for all $b \in S$ do
10:                 if $T_b \notin I_p$ then
11:                     if $p.\mathsf{canHost}(L_p \cup b) =$ true then
12:                         $L_p \leftarrow L_p \cup b$
13:                     else
14:                         $I_p \leftarrow I_p \cup T_b$
15:                     end if
16:                 end if
17:             end for
18:             $U_p \leftarrow \left( \sum_{b \in L_p} Q_{T_b} \right) / \left( CF_p + \sum_{b \in L_p} CV_{pT_b} \right)$
19:             if $U_p > \hat{U}$ then
20:                 $\hat{U} \leftarrow U_p$, $\hat{p} \leftarrow p$
21:             end if
22:         end if
23:     end for
24:     if $\hat{p} \neq \emptyset$ then
25:         $j_{\hat{p}} \leftarrow j_{\hat{p}} + 1$
26:         for all $b \in L_{\hat{p}}$ do
27:             $x_{bvj_{\hat{p}}} \leftarrow 1$
28:         end for
29:         $S \leftarrow S \setminus L_{\hat{p}}$
30:     end if
31: until $S = \emptyset \vee \hat{p} = \emptyset$
```

In the second phase, VM distribution, we first initialize an instance count $j_p \in \mathbb{N}$ for each PM type $p \in P$ (lines 1-3). Subsequently, the following packing process is conducted: We scan the list of PM types to determine a favorite $\hat{p} \in P$. For this purpose, we initially create a packing list $L_p \subseteq S$ for each type p, unless the maximum number of instances n_p of this type has already been reached. A packing list constitutes a set of bids that could be hosted by a new instance of the regarded PM type. For that matter, we scan the list of served bids S. For each bid b, we check whether it could – in addition to the current packing list L_p – be hosted by a new instance of type p, based on the given resource constraints. If the bid b meets this condition, it is added to the packing list. If not, the associated VM type T_b is added to an ignore list $I_p \subseteq V$, and bids of the identical type are ignored during the remainder of the packing list creation (lines 8-17). In the following, we compute the utility U_p of the current PM type p. Utility is defined as the ratio between the revenue from the packing list L_p and the resulting fixed and variable operating costs of a new PM instance. If a PM type exhibits higher utility than the current favorite $\hat{p}$, it is stored as new favorite (lines 18-21). After all PM types have been scanned and a favorite has been identified, all bids in the packing list $L_{\hat{p}}$ are assigned to a new instance of type $\hat{p}$ with the previously incremented index $j_{\hat{p}}$. The bids are also removed from the set S (lines 26-29). The packing process is repeated until all served bids have been assigned or no suitable (i. e., favorite) PM instance remains for assignment.

In terms of computational complexity, the heuristic has substantial advantages over the optimal approach: As it can be observed from Listing 1, the computational complexity of the first phase grows linearly with the number of PM types and VM types. The latter is commonly substantially smaller than the number of bids and thus negligible. For the second phase, according to Listing 2, the maximum number of iterations corresponds to the number of bids. In each iteration, the computational complexity relates linearly to the number of bids and PM types again. Thus, the computational complexity of the heuristic allocation approach is polynomial and corresponds to $\mathcal{O}(|B|^2 * |P|)$.

III. Evaluation

Both optimization approaches from the previous section have been implemented in a prototypical Java program. In order to solve the optimization model for the first allocation approach, we map it into a programmatic representation using the *JavaILP* framework[3]. As actual solvers, the commercial *IBM CPLEX Optimizer*[4] and the free *lpsolve*[5] frameworks may be employed, with the first constituting the default choice in our evaluation.

A. Approach and Methodology

The aim of our evaluation lies in a quantitative assessment of the two optimization approaches. Thus, the evaluation complements the brief and solely qualitative analyses of computational complexity from Subsections II-B and II-C. Our focus lies on two metrics that are of practical relevance in the context of auction-based VM allocation: First, the metric *computation time* demonstrates the overall scalability of the approaches. It also expresses the delay that is introduced into the allocation process through the application of the optimization approaches. Second, the metric *profit* represents the absolute solution quality of the approaches. It thus expresses the utility of an optimized allocation to the cloud provider in monetary terms.

For the evaluation, we have created 18 distinct classes of EPAAPs. Each class contains 100 individual problems. Across the different classes, we vary the problem dimension with respect to the regarded number of bids and PM types. The number of VM types and resource types are fixed across all classes. Each problem is randomly generated based on realistic data, which has been obtained as described in the following.

For the data of the VM types in the evaluation, we use the specifications provided by Amazon Web Services for its Elastic Computing Cloud (EC2) and Spot Instances offers[6]. Excluding special-purpose and non-deterministic types (namely, *Cluster* and *Micro*), we infer eight different VM types. Each of these types exhibits specific resource demands with respect to three resource types, namely processor (CPU), memory (RAM), and storage (HDD).

Unfortunately, to the best of our knowledge, Amazon Web Services has not published detailed information about the PMs in its data centers to date. However, empirical results by an industry researcher indicate that the most resource intensive VM types are run on dedicated PMs [10]. Based on this notion, the specifications of the *High-Memory Quadruple Extra Large* VM type, which exhibits the highest demands for each resource type, is assumed as baseline for the definition of five different PM types. Based on calculations by Walker [2], the fixed operating costs of these PM types are conservatively estimated to range from \$0.20 to \$0.40 per hour. The specific figure for each PM type is correlated with its resource supply.

The bid prices that the cloud users submit to the cloud provider were modeled using specific distribution functions F_v for each VM type v. For the choice of these distribution functions, we reason as follows: The lowest observed bid price, α_v, will most likely correspond to the lowest permissible bid price, i. e., $\alpha_v = 0.01$. Given that the users act rational, the highest observed bid price, γ_v, will not exceed the price $O_v \in \mathbb{R}$ of a so-called *On-Demand* VM instance, which imposes a static usage fee, but essentially offers guaranteed availability in return. Thus, we assume $\gamma_v = O_v - 0.01$. Lastly, according to empirical findings by Wee [5] and statements by Amazon Web Services [11], the average savings from Amazon Spot Instances – compared to On-Demand instances – amount to 52.3% and between 50% and 66% respectively. Thus, we assume that the most frequently observed bid price, β_v, corresponds to 50% of the On-Demand price, i. e., $\beta_v = 0.5 * O_v$. The underlying notion is that the cloud users would like to maximize their savings from the auction mechanism, but also maintain a reasonable chance of actually being allocated VM instances. Based on the previous reasoning, we believe that the use of a triangle distribution, i. e., $F_v \sim Tr(\alpha_v, \beta_v, \gamma_v)$, best reflects a realistic bidding behavior.

Based on past research by Greenberg et al. [12] and Barroso and Hölzle [13], we estimate that the additional variable operating cost of a fully utilized server amounts to about 25% of its fixed operating cost. As a first approximation, we further assume that the cost of hosting a certain VM type

[3]http://javailp.sourceforge.net/
[4]http://www.ibm.com/software/integration/optimization/cplex-optimizer/
[5]http://sourceforge.net/projects/lpsolve/

[6]http://aws.amazon.com/en/ec2/#instance

linearly relates to the resource utilization on the underlying PM. Accordingly, the variable operating costs of each VM type on every PM type are determined using Equation 13.

$$CV_{pv} = 0.25 * CF_p * \frac{1}{|R|} * \sum_{r \in R} \frac{RD_{vr}}{RS_{pr}} \quad \forall p \in P, v \in V \quad (13)$$

B. Results and Discussion

Following the problem generation, each EPAAP was solved using both optimization approaches, *optimal* and *heuristic*. In the process, the respective computation time and resulting profit for the cloud provider were recorded. In order to complete the evaluation within an acceptable amount of time, a timeout period of 5 minutes (i. e., 300 seconds) was imposed per problem and approach. The evaluation was conducted on a desktop computer with an Intel Core 2 Quad Q9450 processor and 4 GB of memory, operating under Microsoft Windows 7.

Table I provides an overview of the evaluated EPAAP classes. As it has been previously outlined, the number of bids (d_B) and PM types (d_P) was varied for each problem class, whereas the number of VM types and resource types was fixed ($d_V = 8$, $d_R = 3$). The table also indicates the percentage of problems that could be solved within the specified timeout period by each optimization approach. Only those problems that could be solved by both approaches served as sample for the evaluation results provided hereafter. Please note that problem classes E and F constitute an exception, because they were exclusively solved using the heuristic approach in order to show its applicability to large-scale problems.

To begin with, Figure 2 depicts the absolute average computation times per problem across all considered problem classes. In accordance with the qualitative discussion, the computation times for the optimal approach grow roughly exponentially with the number of bids. The effect of an increasing number of PM types is less accentuated, but still well observable. For the problem classes involving 20 bids (B_{1-4}), the absolute computation time of the optimal approach reaches the magnitude order of one second. For problem classes that involve 30 bids or more (C_{1-4} and D_{1-4}), the computation time reaches and even exceeds the magnitude order of ten seconds. Accordingly, a substantial share of problems cannot be solved at all within the specified timeout period (cf. Table I). In contrast, for the heuristic approach, the average computation time does not exceed the magnitude order of one millisecond up to problem class D. For problem classes E and F, the average computation time lies in the magnitude order of ten milliseconds and one second respectively. In general, the increase with growing problem size is rather moderate and roughly corresponds to the square of the number of bids. In accordance, the timeout is not of practical relevance to the heuristic approach.

Table I
OVERVIEW OF EVALUATED EPAAP CLASSES AND SHARE OF SOLVED PROBLEMS PER CLASS.

Problem class			Solved problems (%)		
ID	d_P	d_B	Opt.	Heur.	Both
A_1	1	10	100	100	100
A_2	1	20	100	100	100
A_3	1	30	100	100	100
A_4	1	40	100	100	100
B_1	2	10	100	100	100
B_2	2	20	100	100	100
B_3	2	30	100	100	100
B_4	2	40	100	100	100
C_1	3	10	95	100	95
C_2	3	20	97	100	97
C_3	3	30	85	100	85
C_4	3	40	80	100	80
D_1	4	10	84	100	84
D_2	4	20	72	100	72
D_3	4	30	54	100	54
D_4	4	40	46	100	46
E	4	1000	–	100	–
F	4	10000	–	100	–

The performance difference between the two approaches is further highlighted by Figure 3, which depicts the ratio of computation time between the heuristic and optimal approach. For the smallest problem classes involving 10 bids (A_{1-4}), the ratio amounts to approximately 0.2% or less. This is equivalent to a reduction of more than 99.8% in computation time. For the larger problem classes involving additional bids, the ratio converges toward 0, indicating reductions of more than 99.9% by the heuristic approach.

The results for the second metric, profit, are given in Figure 4. As can be seen, the heuristic approach achieves favorable and consistent results, ranging between about 95.2% and 97.6% compared to the optimal solution. On average across all classes, the figure corresponds to approximately 96.7%. That means, due to the application of the heuristic allocation approach, the cloud provider would incur an average reduction in profit of 3.3% compared to the optimal solution.

In summary, the results indicate that the computation of an optimal solution to the EPAAP is difficult to achieve under practical conditions. A cloud provider will usually receive thousands of bids that have to be regarded in the allocation process. At the same time, the cloud provider underlies stringent time constraints, given that the timespan between accepting the last bids in an auction period and the announcement of the resulting equilibrium prices and VM allocations should be minimal. For such application scenarios, our proposed heuristic approach presents a viable option. It achieves substantial reductions in computation time, but also

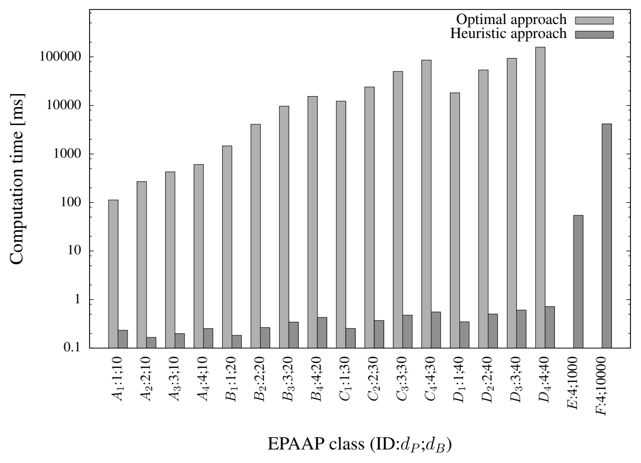

Figure 2. Mean absolute computation times per problem for both optimization approaches.

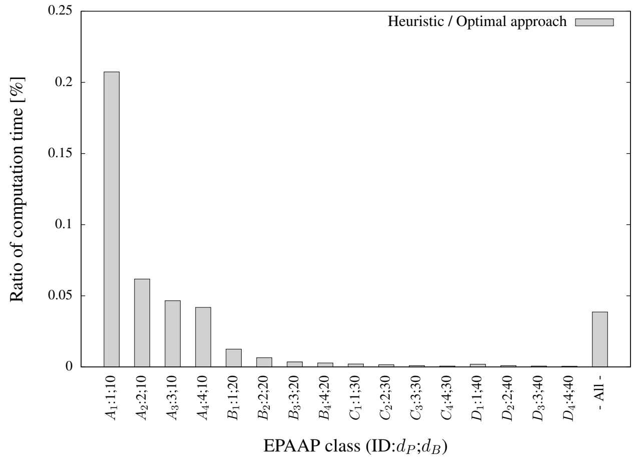

Figure 3. Ratios of computation times between both optimization approaches (based on micro-average).

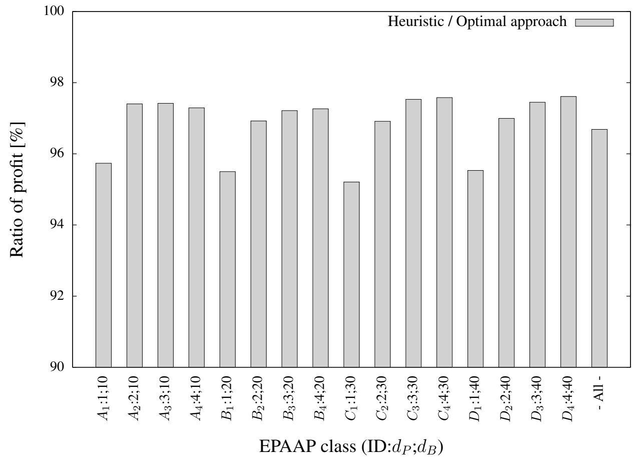

Figure 4. Ratios of profits between both optimization approaches (based on micro-average).

retains a favorable solution quality in terms of profit for the cloud provider.

IV. Related Work

To the best of our knowledge, we are the first to scientifically examine the Equilibrium Price Auction Allocation Problem, i. e., the challenge of concurrently pricing and distributing VM instances based on an auction scheme. However, in the broader context of cloud computing, a substantial amount of work has been conducted with respect to related topics. In the following, we focus on a set of papers that we consider representative for each topic area.

Breitgand et al. [14], for instance, have proposed an optimization model and corresponding heuristic for the distribution of VM instances on physical hosts through a cloud provider. The authors take into account various constraints, including resource demands and supplies, and also permit the definition of different objectives, including profit maximization. However, their research does not address the aspect of pricing the VMs in an auction-based setting.

Korupolu et al. [15] have proposed an optimization scheme for the placement of applications, which comprise compute and storage components, in data centers. In accordance with our work, their heuristic approach is inspired by the Knapsack problem. However, the work of Korupolu et al. does not involve an auction-based pricing mechanism.

Zaman and Grosu [3] have examined the allocation of VM instances to physical machines based on combinatorial auctions, where cloud users bid for arbitrary bundles of VM instances. The authors propose multiple heuristic allocation strategies, which are evaluated with respect to different objectives, including maximization of revenues. In contrast to our work, the prices of identical VM types may be discriminated between different users, which is not the case in equilibrium price auctions. Furthermore, the authors do not explicitly regard the distribution among physical hosts under resource constraints.

Zaman and Grosu [16] have additionally addressed the issue of auction-based VM allocation in a more recent paper. However, the focus of this work lies on bidding strategies for the cloud user, rather than optimization approaches for the cloud provider.

Lin et al. [17], in accordance with our work, have proposed an allocation mechanism for second-price (i. e., equilibrium price) auctions. However, the authors solely focus on the optimal pricing of resources, but do not consider the concurrent distribution of VM instances across physical hosts.

Özer and Özturan [18] have proposed an optimal approach, as well as different heuristics for the allocation of grid and cloud resources. In contrast to our work, the authors assume combinatorial auctions, where users submit bids for bundles of resources, which results in a different pricing approach. In addition, Özer and Özturan do not consider the distribution to physical hosts as part of the allocation process.

In our own previous research [9], we have examined the *Software Service Distribution Problem*. This challenge concerns the cost-minimal distribution of Software as a Service instances across leased VM instances under resource constraints. We have presented an optimal, as well as heuristic solution approach. However, these approaches only address the distribution process. They do not cover the concurrent pricing of entities, which constitutes a major challenge in the EPAAP.

V. Summary and Outlook

In the work at hand, we have introduced the Equilibrium Price Auction Allocation Problem (EPAAP), a challenge in the context of cloud computing. This problem concerns cloud providers and involves the concurrent pricing and distribution of virtual machines across physical machines based on equilibrium price auctions.

As first major contribution, we have introduced a mathematical formulation of the EPAAP as binary integer program. This model serves as the basis of an optimal allocation approach, which permits the computation of profit-maximizing solutions to the EPAAP. As second major contribution, given the computational complexity of the optimal approach, we have developed a heuristic approach. This heuristic trades substantial reductions in computation time against small reductions in overall profit.

Through an evaluation based on realistic data from the cloud computing domain, we have demonstrated the practical applicability of our approaches. Specifically, we have shown that the heuristic is able to achieve reductions in computation time of more than 99.9% compared to an optimal approach. At the same time, it achieves profits that correspond to about 96.7% of the optimal solution on average. Thus, our work is not only of scientific interest, but can also provide a foundation for the practical application of equilibrium price auctions in the cloud computing domain.

In our future work, we will aim at the further improvement of the heuristic approach with respect to the solution quality. In addition, we plan to extend the proposed approaches such that the specific requirements of an optimization across multiple subsequent auction periods, such as the live migration of virtual machines, are supported.

Acknowledgments

This work has partly been sponsored by E-Finance Lab e. V., Frankfurt a.M., Germany (www.efinancelab.de).

References

[1] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] E. Walker, "The Real Cost of a CPU Hour," *Computer*, vol. 42, no. 4, pp. 35–41, 2009.

[3] S. Zaman and D. Grosu, "Combinatorial Auction-Based Allocation of Virtual Machine Instances in Clouds," in *IEEE 2nd Int. Conf. on Cloud Computing Technology and Science*, 2010.

[4] R. T. Bahman Javadi and R. Buyya, "Statistical Modeling of Spot Instance Prices in Public Cloud Environments," in *4th IEEE/ACM Int. Conf. on Utility and Cloud Computing*, 2011.

[5] S. Wee, "Debunking Real-Time Pricing in Cloud Computing," in *11th IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Computing*, 2011.

[6] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing Amazon EC2 Spot Instance Pricing," in *IEEE 3rd Int. Conf. on Cloud Computing Technology and Science*, 2011.

[7] W. Domschke and A. Drexl, *Einführung in Operations Research*, 6th ed. Springer, 2004, in German.

[8] F. Hillier and G. Lieberman, *Introduction to Operations Research*, 8th ed. McGraw-Hill, 2005.

[9] U. Lampe, T. Mayer, J. Hiemer, D. Schuller, and R. Steinmetz, "Enabling Cost-Efficient Software Service Distribution in Infrastructure Clouds at Run Time," in *2011 IEEE Int. Conf. on Service Oriented Computing & Applications*, 2011.

[10] H. Liu, "Amazon's Physical Hardware and EC2 Compute Unit," http://huanliu.wordpress.com/2010/06/14/amazons-physical-hardware-and-ec2-compute-unit/, last accessed January 4, 2013.

[11] Amazon Web Services LLC, "Using Amazon EC2 Spot Instances for Scientific Computing," http://aws.amazon.com/en/ec2/spot-and-science/, last accessed January 4, 2013.

[12] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel, "The Cost of a Cloud: Research Problems in Data Center Networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.

[13] L. Barroso and U. Hölzle, "The Case for Energy-Proportional Computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[14] D. Breitgand, A. Maraschini, and J. Tordsson, "Policy-Driven Service Placement Optimization in Federated Clouds (TR H-0299)," IBM Research, Tech. Rep., 2011.

[15] M. Korupolu, A. Singh, and B. Bamba, "Coupled Placement in Modern Data Centers," in *2009 IEEE Int. Symp. on Parallel & Distributed Processing*, 2009.

[16] S. Zaman and D. Grosu, "Efficient Bidding for Virtual Machine Instances in Clouds," in *2011 IEEE Int. Conf. on Cloud Computing*, 2011.

[17] W. Lin, G. Lin, and H. Wei, "Dynamic Auction Mechanism for Cloud Resource Allocation," in *10th IEEE/ACM Int. Conf. on Cluster, Cloud and Grid Computing*, 2010.

[18] A. Özer and C. Özturan, "An Auction Based Mathematical Model and Heuristics for Resource Co-Allocation Problem in Grids and Clouds," in *5th Int. Conf. on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control*, 2009.

ERRATA

Please note that the following errata were identified in the originally published version of this paper:

- In Figure 1, the price of bid 1 should correspond to $W_1 \geq 0.25$, given the assumption that all bids are ordered by descending prices.
- In Model 1, Equation 1, the term "CF_{pj}" should read "CF_p".
- In Algorithm 2, Line 27, the variable "$x_{bvj_{\hat{p}}}$" should read "$x_{bpj_{\hat{p}}}$".
- In Table I, the values for columns d_P and d_B have been incorrectly stated. For problem classes A, B, C, and D, the number of bids d_B corresponds to 10, 20, 30, and 40 respectively, whereas the subscript denotes the actual number of PM types d_P. The labels in Figures 2 through 4 are correct.
- In Section IV, concerning the discussion of the work by Zaman and Grosu [3], the expression "to physical machines" should be dropped from the first sentence.