Bill Estimation in Simplified Memory Progressive Second Price Auctions

Danielle Movsowitz-Davidow[1](✉), Nir Lavi[2], and Orna Agmon Ben-Yehuda[2,3](✉)

[1] Computer Science Department, University of Haifa, Haifa, Israel
dmovsowi@campus.haifa.ac.il

[2] Computer Science Department, Technion—Israel Institute of Technology, Haifa, Israel
dl8.nir@gmail.com, ladypine@cs.technion.ac.il

[3] Caesarea Rothschild Institute for Interdisciplinary Applications of Computer Science, University of Haifa, Haifa, Israel

Abstract. Vertical elasticity, the ability to add resources on-the-fly to a virtual machine or container, improves the aggregate benefit clients get from a given cloud hardware, namely the social welfare. To maximize the social welfare in vertical elasticity clouds, mechanisms which elicit resource valuation from clients are required. Full Vickrey-Clarke-Groves (VCG) auctions, which allocate resources to optimize the social welfare, are NP-hard and too computationally-complex for the task. However, VCG-like auctions, which have a reduced bidding language compared with VCG, are fast enough. Such is the Simplified Memory Progressive Second Price Auction (SMPSP). A key problem in VCG-like auctions is that they are not completely truthful, requiring participants, who wish to maximize their profits, to estimate their future bills. Bill estimation is particularly difficult since the bill is governed by other participants' (changing) private bids.

We present methods to estimate future bills in noisy, changing, VCG-like auction environments. The bound estimation method we present leads to an increase of 3% in the overall social welfare.

Keywords: Bill estimation · Progressive second price auction · Resource allocation · Multi armed bandit problem

1 Introduction

Cloud providers give their clients the illusion of elastic resources: "just ask for more, and the cloud shall provide". However, virtual machines (VMs) or containers in the cloud are located on physical machines. Unless they are live-migrated,

Nir Lavi—The work was done while the author was a student at the Computer Science Dept., Technion.

their vertical elasticity—the ability to extend the resources on the same system—is limited first by the physical boundaries of the machine, and furthermore by the resource consumption of neighboring clients.

During 2017, the major providers introduced vertical CPU elasticity using the term "burstable" (Amazon [4], Azure [12], and Google [8]). The industry offered vertical elasticity for CPU first, because it is easy to evict and allocate on-the-fly. This is done using mature tools at the system's level, such as cgroups and CFS [5]. Vertical elasticity of storage resources, such as RAM and SSD, is the hardest, since their eviction takes a toll. Hence, the allocation of storage resources should be modified less often, to allow clients to benefit from the resources prior to its eviction.

Resources that are divided to a small number of units can be auctioned to optimize the social welfare—the aggregate benefit that all clients draw from the resource—using a VCG auction [6,9,13], as done for last level cache (LLC) ways [7]. In this auction, the clients bid with their valuation for each good—how much each good is worth to them. The host chooses the allocation which maximizes the social welfare and charges each client according to the damage it causes to its neighbors, such that it is in the client's best interest to tell the truth and enable the host to optimize the sum of true valuations.

A RAM auction is more complicated. RAM is a divisible good, which can be viewed as continuous or as composed of millions of small chunks. When viewed as millions of chunks, the complexity of a full VCG auction is prohibitive for practical purposes.

The Simplified Memory Progressive Second Price (SMPSP) [1,2] auction has a reasonable complexity on the provider's side ($O(n \log(n))$), because it is only VCG-like. In a VCG-like mechanism, clients do not bid using their full valuation functions. Instead, they use simplified bids, representing their specific resource requests: a maximal quantity q and a unit price p. The client's choice of a bid price p as a function of q is usually truthful, and simple [3]. However, the choice of quantity is harder—this is where the computational burden lies. Each client wants to choose a bid that wins the auction and optimizes its performance and profit goals. Since the unit price and quantity are coupled, the chosen quantity must correspond to a unit price that is likely to win, and that is likely to yield the maximal profit. Only by estimating the bill, can the client predict its profit and optimize it by choosing its bid quantity.

Our contribution is a method for clients to estimate their next bill in an SMPSP auction. Future bills depend on private bids made by other auction participants, which might change their preferences and consequently their bids over time. We detect such changes in a noisy environment by tracking and analyzing historical data, and focusing on the notable effect of environmental changes on the bill and minimal winning unit price. As a result, clients in an SMPSP auction now successfully respond to changing conditions in a noisy environment, improving the overall social welfare by 3% on average.

The code is free and available from https://bitbucket.org/danimovso/ginseng-open/.

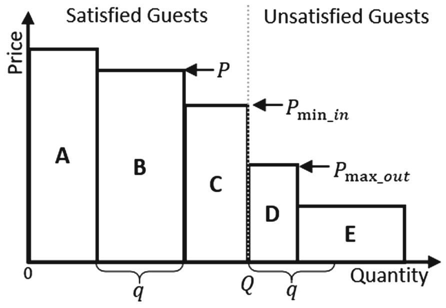


Fig. 1. Auction results. Each bid is represented by a rectangle whose height and width are the client's p and q.

2 Allocation and Payment in the SMPSP Auction

In the SMPSP auction the host (the software running on the physical machine, on behalf of the provider) sorts the participating clients' bids in a descending price order (see Fig. 1) and allocates RAM to those in the interval $[0, Q]$. This is the allocation which optimizes the social welfare. The allocation's social welfare is the sum of areas that belong to clients who were allocated RAM. To determine a client's bill the host first calculates the social welfare, and subtracts the specific client's valuation for the allocation of the resources it got. The host then repeats the process, excluding the specific client. The client's bill is the difference between these two results. Visually, if the client was allocated a quantity of q, its payment will be the area under the plot in the interval $[Q, Q + q]$.

3 Client Strategy

SMPSP clients need to translate their valuation into a bid. Unlike PSP [10] clients, SMPSP clients do not hear all the bids and therefore need to estimate what their bill would be for every bid they make.

To help clients estimate their future bills, at the end of each auction round the host provides two *borderline bid results*, as depicted in Fig. 1. p_{min_in} denotes the minimal unit price offered by a client that was allocated a positive amount of RAM. p_{max_out} denotes the maximal unit price offered by a client that did not receive its full requested quantity of RAM.

We consider a single auction round in which Q MB of RAM are offered for rent, and focus on a single client with a valuation function $V(\cdot)$ for RAM. For simplicity, V depends only on the quantity of RAM allocated to the client, and w.l.o.g., $V(0) = 0$.

The client examines options for a bid quantity q in the interval $[0, Q]$. For each option q, it performs the following stages:

1. Compute the truthful bid price, the mean unit valuation $p(q) = \frac{V(q)}{q}$, which is the best choice in a steady state, according to [3].

2. Filter out q if the resulting $p(q)$ is unlikely to win any resources.
3. Estimate $bill(q)$, the bill for the quantity q, assuming the auction is won.

Of the remaining options, the client bids with the $(q, p(q))$ pair with the smallest q that maximizes the profit:

$$q = min\left(argmax_q\left(V\left(q\right) - bill\left(q\right)\right)\right). \tag{1}$$

For the bill estimation in Step 3 we examine three approaches: (1) Use the last unit price paid by the client. (2) Use a weighted average (with a time-dependent decay) of the latest unit prices paid by the client. (3) Perform additional computations on each historical data piece; use a feedback loop to learn and adjust the results (in Sect. 4).

4 Bound Estimation

To estimate its own bill, denoted by $bill$, a client gathers data about the effect other clients' have on its bill. After each auction round, the client can adjust its estimation using the announced borderline bids. E.g., clients who where allocated some RAM in the previous round can use p_{max_out} together with their actual bill (denoted by $bill'$) and allocation (denoted by q') to deduce the average unit price in the interval $[Q, Q + q']$.

In the bound estimation approach, the client bounds its future bill from above and below by extrapolating data which was possibly recorded under different circumstances, and is not necessarily accurate. Finally, the client learns to correct the interpolation between the bounds according to the general shape of the allocation plot.

The analysis in this section relies on the following statements, which result from the concavity and monotonicity of the valuation functions: The first, $p_{max\_out} \leq p_{min\_in}$. The second, if $q_1 < q_2$, then $p\left(q_1\right) \leq p\left(q_2\right)$.

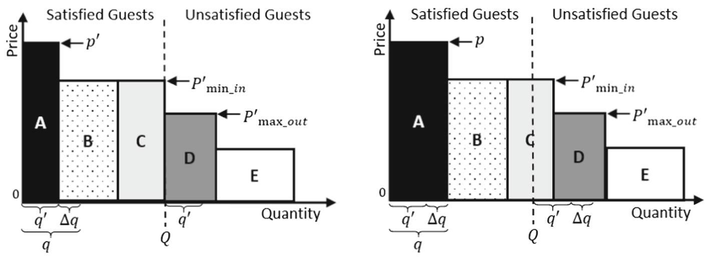
Let us denote the client's change in the requested amount of RAM by $\Delta q = q - q'$. The estimated bill can be bound on the basis of Δq, assuming that bids from the rest of the clients stay the same. We will analyze the cases where Δq is positive or negative separately.

In the following sections, "our" client, whose bill we are estimating, is described by the rectangle marked "A" in the figures.

4.1 Increased Demand

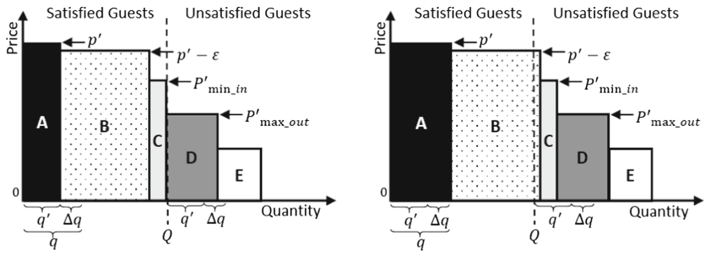
When $\Delta q > 0$ and the client's request is fully satisfied, the client's bill increases. The client's estimated bill is the lowest (Fig. 2) when all the newly unsatisfied clients (indicated by the rectangle "C") are clients who bid with unit price p_{min_in}. Hence, the estimated lower bill bound is:

$$bill \geq bill' + \Delta q \cdot p_{min\_in} \tag{2}$$



(a) Before incrementing by Δq **(b)** After incrementing by Δq

Fig. 2. Visualizing client A's estimation of the lowest bill bound when the rest of the clients do not change their bid. $\Delta q > 0$.



(a) Before incrementing by Δq **(b)** After incrementing by Δq

Fig. 3. Visualizing client A's estimation of the highest bill bound when the rest of the clients do not change their bid. $\Delta q > 0$.

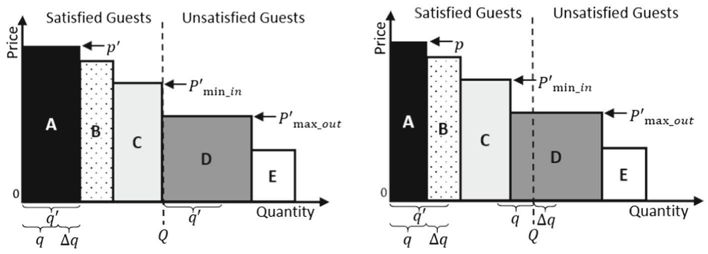
The client's estimated bill is the highest (Fig. 3) if only one client bid unit price p_{min_in} and received only δMB, such that $\delta \to 0$ (the rectangle "C"), and another client (the rectangle "B") received an amount of memory when bidding for the unit price $p' - \epsilon$, such that $\epsilon \to 0$. Hence, the estimated upper bill bound is:

$$bill \leq bill' + \Delta q \cdot p\left(q\right). \tag{3}$$

4.2 Decreased Demand

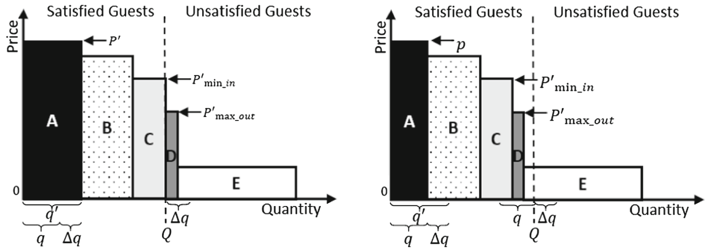
In cases where $\Delta q < 0$, the client's bill decreases. The client's estimated bill is the lowest (Fig. 4) when all the unsatisfied clients affecting our client's bill bid with the unit price p_{max_out} (the rectangle "D"). If our client decreases its RAM request by $|\Delta q|$, then the rest of the clients in the interval $[Q, Q + \Delta q]$ are allocated RAM. Since the clients are sorted in a descending order, the average unit price decreases. Hence, the estimated lower bill bound is:

$$bill \geq bill' + \Delta q \cdot p_{max\_out}. \tag{4}$$



(a) Before decreasing by Δq **(b)** After decreasing by Δq

Fig. 4. Visualizing client A's estimation of the lowest bill bound when the rest of the clients do not change their bid. $\Delta q < 0$.



(a) Before decreasing by Δq **(b)** After decreasing by Δq

Fig. 5. Visualizing client A's estimation of the highest bill bound when the rest of the clients do not change their bid. $\Delta q < 0$.

The client's estimated bill is the highest (Fig. 5) when the client who bid p_{max_out} receives only δMB, such that $\delta \to 0$ (rectangle "D"), and the rest of the clients in the interval $[Q, Q + \Delta q]$ bid with the unit price $\epsilon \to 0$. Hence, the estimated upper bill bound is:

$$bill \leq bill' \cdot \frac{q}{q'}. \tag{5}$$

This is because the unit price can only drop, since the bids with the higher unit price affect the bill less.

4.3 Interpolating the Bounds

Where between the bounds would the actual bill be? The interpolation depends on the shape of the allocation plot in the vicinity of the Q boundary which affects the bill change computation. The shape of the allocation plot is the shape of the function formed by the top of the allocation rectangles. The concavity of the allocation plot affects the distance of the bill from the bounds. When $\Delta q < 0$, the shape of the allocation plot to the right of the Q boundary affects

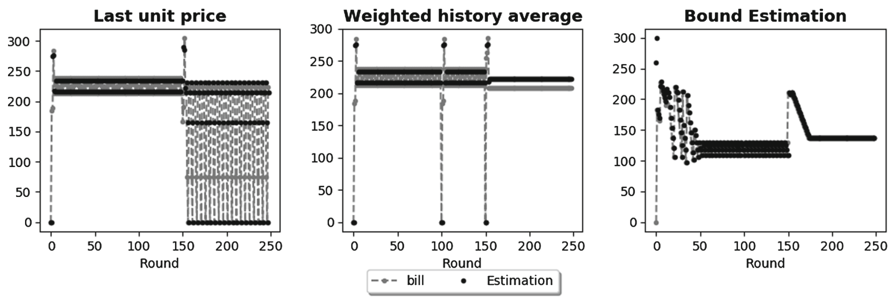


Fig. 6. The accuracy of the estimation algorithms. A typical trace of one VM's actual bill and its estimation, for each of the methods.

the interpolation: if the allocation plot is concave (downward), the lower bound will be a better estimate (Fig. 4), if the allocation plot is convex (downward), the upper bound will be better (Fig. 5). When $\Delta q > 0$, it is the shape of the plot to the left of the Q boundary that matters. The upper bound dominates when it is concave (Fig. 3), and the lower—when it is convex (Fig. 2).

The client does not need to learn the exact shape of the allocation plot—it is enough to learn its effect on the interpolation. Hence, the client validates its prior estimates of upper and lower bounds against its actual bill: it expresses the previous actual bill as a linear interpolation:

$$\frac{bill'}{q} = (1-\alpha)\frac{L_{n-1}}{q'_{n-1}} + \alpha\frac{U_{n-1}}{q'_{n-1}}, \tag{6}$$

where L_{n-1}, U_{n-1} and q'_{n-1} denote the lower bound, upper bound and requested RAM quantity of the previous round, respectively. The interpolation coefficient, α, is extracted from the validation and used to predict the future bill,

$$bill = (1-\alpha)L_n + \alpha U_n. \tag{7}$$

The reuse of a past value of the interpolation coefficient α relies on the assumption that the environment changes slowly, and thus the shape of the allocation plot remains more or less the same, at least for a small quantity change $|\Delta q|$.

5 Evaluation

To evaluate the bill estimation methods, we conducted a series of experiments, each with a different estimation method used by all guests. In each experiment, Ginseng [3] auctioned RAM among 10 VM clients using an SMPSP auction. Each VM ran the elastic version of memcached, a key-value storage application which is widely used on clouds. The elastic version[1] can dynamically adjust its RAM footprint on-the-fly, so its valuation function for RAM is concave, monotonically

[1] Available from https://github.com/ladypine/memcached.

rising. Its performance, defined by the rate of successful query responses, was measured using memaslap, which reports its progress every second. The valuation function of each guest was the performance multiplied by a factor, which was drawn from a Pareto distribution: a characteristic economic distribution. We used an index of 1.36, according to Levy and Solomon [11] and as used in earlier work [1,3].

Each experiment lasted 150 auction rounds, each taking 12 seconds. The experiments all started after a warm-up time of 100 rounds, in which auctions did not take place, allowing memcached's cache to stabilize. During each experiment, the valuation functions of 5 of the 10 participating VMs changed, once, to introduce noise.

The accuracy of the estimation methods is presented in Fig. 6. The previous unit price method and the weighted history average method do not converge, and induce fluctuations in the bill. Adjoined by needless allocation changes, this hurts the VMs' ability to utilize the RAM. The bound estimation method converges, and the bill it induces on the system is more stable.

Bill prediction inaccuracy leads to a sub-optimal allocation, which takes its toll. First, it prevents the SMPSP auction from optimizing the social welfare: the RAM is not allocated to the best possible clients. Second, an instable allocation means that RAM has to be reclaimed more often. Frequent RAM reclamation hurts the application's ability to make use of the RAM, thus hurting the overall performance and social welfare achieved by the physical machine. In this set of experiments, the "previous unit price" method increases the social welfare by 0.2%, compared with the "weighted history average" method. The "bound estimation" method increases it by 3% compared with the "weighted history average" method.

6 Conclusion and Future Work

An accurate bill prediction algorithm is essential for the stability and social welfare optimization of a VCG-like auction. The bounds estimation algorithm predicts the bill in an SMPSP auction better than others, and converges to the actual bill. Improving the bound estimation algorithm by gathering additional and relevant historical data, remains for future work.

Acknowledgments. This work was partially funded by the Amnon Pazi memorial research foundation, and supported by the Israeli Ministry of Science & Technology. We thank Orr Dunkelman for fruitful discussions. We also thank the Caesarea Rothschild Institute for Interdisciplinary Applications of Computer Science in the University of Haifa for their support. This research was also partially supported by the Center for Cyber, Law and Privacy and the Israel National Cyber Directorate.

References

1. Agmon, S., Agmon Ben-Yehuda, O., Schuster, A.: Preventing collusion in cloud computing auctions. In: Coppola, M., Carlini, E., D'Agostino, D., Altmann, J., Bañares, J.Á. (eds.) GECON 2018.: preventing collusion in cloudcomputing auctions, vol. 11113, pp. 24–38. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-13342-9_3
2. Movsowitz, D., Funaro, L., Agmon, S., Agmon Ben-Yehuda, O., Dunkelman, O.: Why are repeated auctions in RaaS clouds risky? In: Coppola, M., Carlini, E., D'Agostino, D., Altmann, J., Bañares, J.Á. (eds.) GECON 2018. LNCS, vol. 11113, pp. 39–51. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-13342-9_4
3. Agmon Ben-Yehuda, O., Posener, E., Ben-Yehuda, M., Schuster, A., Mu'alem, A.: Ginseng: market-driven memory allocation. In: Proceedings of the 10th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE 2014, pp. 41–52. ACM, New York (2014). https://doi.org/10.1145/2576195.2576197
4. Amazon: Amazon EC2 burstable performance instances. https://aws.amazon.com/ec2/instance-types/#burst. Accessed 25 July 2018
5. CFS scheduler. https://www.kernel.org/doc/Documentation/scheduler/sched-design-CFS.txt. Accessed 22 Oct 2017
6. Clarke, E.H.: Multipart pricing of public goods. Public Choice **11**(1), 17–33 (1971)
7. Funaro, L., Agmon Ben-Yehuda, O., Schuster, A.: Ginseng: market-driven LLC allocation. In: Gulati, A., Weatherspoon, H. (eds.) 2016 USENIX Annual Technical Conference, USENIX ATC 2016, Denver, CO, USA, 22–24 June 2016, pp. 295–308. USENIX Association (2016). https://www.usenix.org/node/196287
8. Google: Google cloud compute engine pricing. https://cloud.google.com/compute/pricing. accessed 07 June 2019
9. Groves, T.: Incentives in teams. Econ.: J. Econ. Soc. **41**(4), 617–631 (1973)
10. Lazar, A., Semret, N.: The progressive second price auction mechanism for network resource sharing. International Symposium on Dynamic Games and Applications 05 1999
11. Levy, M., Solomon, S.: New evidence for the power-law distribution of wealth. Phys. A: Stat. Mech. Appl. **242**(1), 90–94 (1997). https://doi.org/10.1016/S0378-4371(97)00217-3. http://www.sciencedirect.com/science/article/pii/S0378437197002173
12. Microsoft: Microsoft azure AKS b-series burstable VM. https://azure.microsoft.com/en-us/blog/introducing-b-series-our-new-burstable-vm-size/. accessed 25 July 2018
13. Vickrey, W.: Counterspeculation, auctions, and competitive sealed tenders. J. Financ. **16**(1), 8–37 (1961)