Research Article

Optimization Approach for Resource Allocation on Cloud Computing for IoT

Yeongho Choi and Yujin Lim

1Department of Computer Science, University of Suwon, San 2-2, Wau-ri, Bongdam-eup, Hwaseong, Gyeonggi-do 445-743, Republic of Korea
2Department of Information Technology Engineering, Sookmyung Women's University, Cheongpa-ro 47-gil 100, Yongsan-gu, Seoul 04310, Republic of Korea

Correspondence should be addressed to Yujin Lim; yujin91@sookmyung.ac.kr

Received 28 December 2015; Accepted 28 January 2016

Academic Editor: Fan Wu

Combinatorial auction is a popular approach for resource allocation in cloud computing. One of the challenges in resource allocation is that QoS (Quality of Service) constraints are satisfied and provider’s profit is maximized. In order to increase the profit, the penalty cost for SLA (Service Level Agreement) violations needs to be reduced. We consider execution time constraint as SLA constraint in combinatorial auction system. In the system, we determine winners at each bidding round according to the job’s urgency based on execution time deadline, in order to efficiently allocate resources and reduce the penalty cost. To analyze the performance of our mechanism, we compare the provider’s profit and success rate of job completion with conventional mechanism using real workload data.

1. Introduction

Over recent years, the growth of the IoT (Internet of Things) is expected to create a pervasive connection of things such as embedded devices, sensors, and actuators. This will inevitably result in the generation of enormous amount of data, which have to be autonomously stored, processed, accessed, and managed. Cloud computing has been recognized as a paradigm for the big data problem [1]. Cloud computing allows the sensing data to be stored and used intelligently for smart applications.

One of challenges which arose when IoT meets cloud is resource allocation by which a cloud provider efficiently allocates its resources to cloud users with SLA constraints. The dominating performance factors in resource allocation include provider’s and user’s profit, resource utilization, and QoS (Quality of Service) [2]. For resource allocation in cloud, auction-based model is a popular approach in resource allocation and pricing [3]. In particular, combinatorial auction is preferred in cloud computing because it allows a user to buy a package of resources rather than an individual resource. Since the auction is done on group of resources, it provides efficient resource allocation and helps to improve the profit to both a provider and a user.

The other issue of resource allocation in cloud computing is meeting SLA (Service Level Agreement) established with a user. Before a provider provisions a service to a user, a provider and a user need to establish SLA contract. The SLA is an agreement that specifies QoS between a provider and a user [4]. We define two-levels of SLA to represent different objectives: class-based SLA and job-based SLA [5]. In class-based SLA, for each job class, QoS is measured based on performance metrics. In job-based SLA, QoS is measured using the metrics of individual jobs. Any single job with a poor service quality immediately affects the measured QoS and incurs some SLA penalty cost. We believe the job-based SLA is a more robust type of SLAs from the users’ perspective than providers’ perspective and we focus on this. In general, SLA is defined in terms of various performance metrics, such as service latency, throughput, consistency, and security. In our paper, SLA is defined in terms of deadline along with execution time of each job. The deadline violation is
important from QoS guarantees perspective. It causes loss of profit for a provider due to incompletion to execute the certain jobs and SLA penalty cost [6].

Considering SLA guarantee, we propose a resource allocation mechanism in combinatorial auction system for cloud computing. For efficient resource allocation, winner determination problem in the auction should be solved. The conventional winner determination mechanisms are proposed to maximize resource utilization and a provider's profit for each time interval. However, the profit can be maximized by reducing the penalty cost for SLA violation.

To reduce SLA violations, the deadline constraints need to be considered when winner is determined in the auction. Thus, we propose a winner determination mechanism with consideration for the deadline constraints to maximize the provider’s profit.

The rest of the paper is organized as follows. In Section 2, we discuss the related work. In Section 3, we present our winner determination mechanism. In Sections 4 and 5, we evaluate the performance of our mechanism and conclude the paper with the future research plan.

2. Related Work

We investigate resource allocation strategies for the SLA-based cloud computing framework. Several proposals in the literature are based on utility functions to dynamically allocate resources. In [7], a two-tier resource management approach based on utility functions is presented. It defines the VM (Virtual Machine) utility function as a linear function of the CPU resources. In [8], an autonomic resource manager is presented to control the virtualized environment which decouples the provisioning of resources from the dynamic placement of VMs. It finds an optimal number and size of VMs to allocate CPU resources to applications while maximizing the utility function.

There are several proposals that dynamically manage VMs by optimizing objective functions. Reference [9] determines dynamic placement of VMs based on minimizing an objective cost function using linear programming. The function is to provide an economic model between infrastructure providers and users. The pMapper [10] system includes a power-aware application placement controller to optimize a cost-performance function. It uses the bin-packing algorithm based on a modified version of FFD (First-Fit Decreasing) to place the VMs on the servers, while trying to meet the target utilization.

Researchers have investigated maximizing the profit under SLA constraints. In [11], a multilevel generalized assignment problem is formulated. To solve the problem, the first-fit heuristic is used for maximizing the profit under SLA and the power budget constraint. In [12], a combinatorial optimization problem is defined to maximize the provider’s profit from SLA compliant placement. The problem is presented as a multiunit combinatorial auction. To solve the problem, a column generation method is presented to obtain near optimal solutions.

A combinatorial auction allows bidders to bid for a combination of multiple resources. In other words, bids are submitted for the whole bundle as a single unit. After the auction, each bidder receives either the whole bundle or nothing. In the auction, the goal of a cloud provider is to efficiently allocate the available resources to users and to maximally generate its revenue. In [13], CA-provision is proposed as an allocation mechanism to maximize the revenue for a cloud provider as well as maximize the resource utilization. The provider’s revenue has three elements: the revenue of the resource, the cost of the VM instances that are allocated to the users, and the cost of keeping the remaining resources idle. The purpose of considering the two costs is to reduce the cloud provider’s losses. Reference [14] proposes a model, called ABRA (Auction-Based Resource Co-Allocation) to solve the resource co-allocation problem. It imposes penalty costs on unallocated resources after an auction in order to improve the resource utilization. First, a new neighborhood structure is proposed to define an ordering as a permutation of the bids in the auction. Then, the search space is defined as the set of all possible orderings to find the optimum solution. Finally, various neighbor selection methods are used to find the solution.

The conventional resource allocation mechanisms in combinatorial auction system present winner determination algorithms to reduce the penalty cost in each bidding round. However, since they do not consider the probability of SLA violation when deadline gets to end, the performance is limited. We present a new winner determination algorithm with consideration for deadline constraints of jobs (i.e., urgency) to reduce the penalty cost for SLA violation and maximize the profit of the provider.

3. Proposed Mechanism

A cloud provider offers computing services to users through $m$ different types of VM instances, $V_{M_1}, V_{M_2}, \ldots, V_{M_m}$. The computing power of a VM instance of type $V_{M_i}$, $i = 1, \ldots, m$, is $w_i$, where $w_1 = 1$, $w_2 < w_3 < \cdots < w_m$, and $w = (w_1, w_2, \ldots, w_m)$ [13]. We consider that $w$ is determined mainly by the number of cores, like Amazon EC2 [15]. Let $k_i$ be the number of VM instances provisioned by the provider. Let $M$ be the maximum number of VM instances provisioned by the provider. The provider provisions a combination of instances given by the vector $(k_1, k_2, \ldots, k_m)$ as long as $\sum_{i=1}^{m} w_i k_i \leq M$. We consider $n$ users $u_1, \ldots, u_n$ who request computing resources from the provider specified as bundles of VM instances. A user $u_i$ requests VM instances for its job job, by submitting a bid $B_i = (r_{i1}, \ldots, r_{im}, v_i)$ to the provider, where $r_{ij}$ is the number of requested instances of type $V_{M_j}$ and $v_i$ is the price user $u_i$ is willing to pay to use the requested bundle of VMs for a unit of time. In addition, let us denote by $p_j$ the amount paid by user $u_i$ for using its requested bundle of VMs, and $p_j$ and $v_i$ can be different (usually $p_j \leq v_i$). We assume that the users are single minded which means a user bids for only one bundle.

The provider runs auction mechanism periodically to allocate the VM instances. Thus, users bid for obtaining
In (4), we define $E_j^\text{profit}$ the expected value of the provider's profit for each user to determine the winners at a bidding round. The profit is divided into two cases. When the job is completed before deadline and SLA is violated, the provider should pay the penalty cost $C_j^{\text{penalty}}$. With the probability of SLA violation in (4), we define $E_j^\text{profit}$ the expected value of the provider's profit for $u_j$ as follows:

$$E_j^\text{profit} = Pr_j \left( v_j - C_R \sum_{i=1}^{m} w_i r_i^j \right) + \left( 1 - Pr_j \right) \left( v_j - C_R \sum_{i=1}^{m} w_i r_i^j - C_j^{\text{penalty}} \right).$$

Using (6), winners are determined at each round.

Figure 1 shows that $Pr_j$ increases as $k_j$ or $f_j$ increases in our mechanism. We set $d_j$ and $K_j$ to 15 and 10, respectively. Thus, when $f_j > 5$, job is not completed before the deadline. In the figure, $Pr_j$ gets to 1 in order to complete the job before the deadline, when $f_j = 5$. Besides, as you can see, $f_j$ has the bigger impact than $k_j$ on $Pr_j$. Because $E_j^\text{profit}$ increases as $Pr_j$ increases, the probability that $u_j$ is to be a winner also increases. Thus, the jobs with impending deadlines are likely to be determined as winners. Through the determination, deadline violation can be reduced and the profit of the provider can be improved.

## 4. Performance Analysis

To evaluate the performance of our mechanism, we conduct simulation with real workload data and compare the performance of the conventional mechanism (CA-provision [13]). In preliminary experiments, we use UniLu-Gaia-2014 workload logs from the Parallel Workload Archive [17]. In the logs, the average number of submitted jobs per round is set to about 24 and the average number of processors required per job is 9.97. In the experiments, we set $C_R$ to 0.5 and $C_j^{\text{penalty}}$ to 10 percent of $v_j$. $K_j$ is randomly selected from $(1, 20)$ and the average execution time of a job is multiplying bidding time interval $\Delta t$ with $K_j$.

Figure 2 shows the probability of bidding success at current round $\theta_j$ when the remaining bidding rounds before $d_j$ vary. In CA-provision, the probability does not show...
the big change. In our mechanism, the probability increases as the remaining round decreases. By considering the job’s urgency, the probability gets to 1 as the deadline gets to end, and the jobs with impending deadlines are likely to be determined as winners.

Figure 3 shows the success rate of job completion for each user. The success rate of job completion indicates the number of jobs completed before the deadline to the number of jobs submitted by user $u_j$ in the total simulation time. The success rate of ours is about 27.4% higher than that of CA-provision. In Figure 4, the deadline of a job is determined by multiplying a deadline factor $\tau_j$ with $K_j$; thus the deadline is $d_j = \tau_j K_j$. Here, $\tau_j$ is chosen from 1.5, 2.0, 2.5, and 3.0. The figure shows the success rate of job completion with varying the deadline factors. As you can see, the success rate increases as the deadline increases. The success rate of our mechanism is about 2%, 13.3%, 19.8%, and 27.4% higher than that of CA-provision, respectively. Figures 3 and 4 show that our mechanism effectively reduces the deadline violation and increases the success rate of job completion.

Figure 5 shows the profit of a cloud provider with varying the deadline factors. The profit of our mechanism is about 10.5%, 12%, 10%, and 13% higher than that of CA-provision, respectively. This figure shows that, through the winner determination by considering the job’s urgency, deadline violation decreases and the profit of the provider increases.

To evaluate the performance in different system scenarios, we use eight workload logs from the Parallel Workload Archive [17]. In Table 1, we show a brief description of the workload files. The table describes the log file name, the average execution time of the jobs ($T_{wl}$), the average number of jobs submitted for a unit of time ($J_{wl}$), the average number of processors required per job ($P_{wl}$), and the total number of processors in the system ($M_{wl}$). From the real workload data, we use several data such as job number, execution time, the number of allocated processors, average CPU time used, and user ID. Some records in a log file are not specified because the original files had missing information. So if a record data is missing, we randomly
### Table 1: Real workload data.

| Log file   | $T_{wl}$ (hours) | $J_{wl}$ (jobs/hr) | $P_{wl}$ | $M_{wl}$ |
|------------|------------------|--------------------|----------|----------|
| SDSC-DS    | 13               | 10.26              | 62.36    | 8192     |
| LLNL-Thunder| 5                | 33.62              | 42.36    | 4008     |
| LLNL-Atlas | 8                | 7.6                | 401      | 9216     |
| RICC       | 5                | 119.87             | 16.45    | 4096     |
| PIK-Iplex  | 40               | 20.46              | 34.49    | 2560     |
| LCG        | 1                | 261.16             | 34.52    | 4096     |
| LLNL-uBGL  | 7                | 22.34              | 576      | 2048     |
| UniLu-Gaia | 3                | 24.06              | 9.97     | 64       |

Figure 6: The success rate of job completion with varying the workloads.

Figure 7: The provider’s profit with varying the workloads.

To increase the provider’s profit, the penalty cost for SLA violations needs to be considered. To reduce the cost, we consider the job’s urgency based on the deadline constraint when winners are determined in the combinatorial auction. Taking the urgency into consideration, we calculate the probability of deadline violation for each job. Then, using the probability, we calculate the expected value of the provider’s profit when the corresponding user is selected as a winner at a bidding round. The user with the larger expected value is likely to be determined as a winner. Thus, the penalty cost decreases by the decrease of SLA violation, and the provider’s profit increases. The experimental results show that our mechanism has higher profit and success rate of job completion than these of the conventional mechanism. We look forward to compare the other winner determination mechanisms in the combinatorial auction and demonstrate effectiveness of our mechanism.

5. Conclusion

As a result, our mechanism shows the better performance with respect to the success rate and the profit than CA-provision in various workload scenarios.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.
Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2015R1D1A1A09057141).

References

[1] H.-L. Truong and S. Dustdar, “Principles for engineering IoT cloud systems,” IEEE Cloud Computing, vol. 2, no. 2, pp. 68–76, 2015.
[2] U. Lampe, M. Siebenhaur, A. Papageorgiou, D. Schuller, and R. Steinmetz, “Maximizing cloud provider profit from equilibrium price auctions,” in Proceedings of the IEEE 5th International Conference on Cloud Computing (CLOUD ’12), pp. 83–90, Honolulu, Hawaii, USA, June 2012.
[3] S. R. Shirley and P. Karthikeyan, “A survey on auction based resource allocation in cloud environment,” International Journal of Research in Computer Applications and Robotics, vol. 1, no. 9, pp. 96–102, 2013.
[4] S. Son, G. Jung, and S. C. Jun, “An SLA-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider,” Journal of Supercomputing, vol. 64, no. 2, pp. 606–637, 2013.
[5] H. J. Moon, Y. Chi, and H. Hacıgümüş, “SLA-aware profit optimization in cloud services via resource scheduling,” in Proceedings of the 6th International Conference on World Congress on Services (SERVICES ’10), pp. 152–153, IEEE, Miami, Fla, USA, July 2010.
[6] M. Alrokayan, A. V. Dastjerdi, and R. Buyya, “SLA-aware provisioning and scheduling of cloud resources for big data analytics,” in Proceedings of the 3rd IEEE International Conference on Cloud Computing for Emerging Markets (CCEM ’14), pp. 1–8, Bangalore, India, October 2014.
[7] D. Minarolli and B. Freisleben, “Utility-based resource allocation for virtual machines in cloud computing,” in Proceeding of the 16th IEEE Symposium on Computers and Communications (ISCC ’11), pp. 410–417, Kerykyra, Greece, July 2011.
[8] H. N. Van, F. D. Tran, and J.-M. Menaud, “Autonomic virtual resource management for service hosting platforms,” in Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing (CLOUD ’09), pp. 1–8, Vancouver, Canada, May 2009.
[9] J.-G. Park, J.-M. Kim, H. Choi, and Y.-C. Woo, “Virtual machine migration in self-managing virtualized server environments,” in Proceeding of the 11th IEEE International Conference on Advanced Communication Technology (ICACT ’09), pp. 2077–2083, February 2009.
[10] A. Verma, P. Ahuja, and A. Neogi, “pMapper: power and migration cost aware application placement in virtualized systems,” in Middleware 2008: ACM/IFIP/USENIX 9th International Middleware Conference Leuven, Belgium, December 1–5, 2008 Proceedings, vol. 5346 of Lecture Notes in Computer Science, pp. 243–264, Springer, Berlin, Germany, 2008.
[11] W. Shi and B. Hong, “Towards profitable virtual machine placement in the data center,” in Proceedings of the 4th IEEE International Conference on Cloud and Utility Computing (UCC ’11), pp. 138–145, IEEE, Victoria, Australia, December 2011.
[12] D. Breitgand and A. Epstein, “SLA-aware placement of multi-virtual machine elastic services in compute clouds,” in Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM ’11), pp. 161–168, Dublin, Ireland, May 2011.
[13] S. Zaman and D. Grosu, “A combinatorial auction-based mechanism for dynamic VM provisioning and allocation in clouds,” IEEE Transactions on Cloud Computing, vol. 1, no. 2, pp. 129–141, 2013.
[14] A. H. Özer and C. Özturan, “An auction based mathematical model and heuristics for resource co-allocation problem in grids and clouds,” in Proceedings of the 5th International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control (ICSCCW ’09), pp. 1–4, Famagusta, Cyprus, September 2009.
[15] Amazon EC2 Spot Instances, http://aws.amazon.com/ec2/spot-instances/.
[16] Y. Choi and Y. Lim, “Resource management mechanism for SLA provisioning on cloud computing for IoT,” in Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC ’15), pp. 500–502, IEEE, Jeju Island, South Korea, October 2015.
[17] D. G. Feitelson, “Parallel Workloads Archive,” http://www.cs.huij.ac.il/ labs/parallel/workload/logs.html.