An Estimation-Based Task Load Balancing Scheduling in Spot Clouds

Daeyong Jung, Heeseok Choi, Daewon Lee, Heonchang Yu, Eunyoung Lee

An Estimation-based Task Load Balancing Scheduling in Spot Clouds

Daeyong Jung[1], HeeSeok Choi[1], DaeWon Lee[2], Heonchang Yu[1], Eunyoung Lee[3*]

[1] Dept. of Computer Science Education, Korea University, Seoul, Korea
[2] Division of General Education, SeoKyeong University, Seoul, Korea
[3] Dept. of Computer Science, Dongduk Women's University, Seoul, Korea
[1] {karat, hsrangken, yuhc}@korea.ac.kr, [2] daelee@skuniv.ac.kr, [3] elee@dongduk.ac.kr

Abstract. Cloud computing is a computing paradigm in which users can rent computing resources from service providers according to their requirements. Cloud computing based on the spot market helps a user to obtain resources at a lower cost. However, these resources may be unreliable. In this paper, we propose an estimation-based distributed task workflow scheduling scheme that reduces the estimated generation compared to Genetic Algorithm (GA). Moreover, our scheme executes a user's job within selected instances and stretches the user's cost. The simulation results, based on a before-and-after estimation comparison, reveal that the task size is determined based on the performance of each instance and the task is distributed among the different instances. Therefore, our proposed estimation-based task load balancing scheduling technique achieves the task load balancing according to the performance of instances.

1 Introduction

In recent years, due to the increased interest in cloud computing, many cloud projects and commercial systems, such as the Amazon Elastic Compute Cloud (EC2) [1] and FlexiScale [2], have been implemented. Cloud computing provides high utilization and high flexibility for managing computing resources. In addition, cloud computing services provide a high level of scalability of computing resources combined with Internet technology that are distributed among several customers [3, 4]. In most cloud services, the concept of an instance unit is used to provide users with resources in a cost-efficient manner.

Spot-market-based cloud environment configures the spot instance. In the spot instance environment, spot prices changes depending on the supply and demand of spot instances. The environment affects the success or failure of task completion according to the changing spot prices. Spot prices have a market structure and follow the law of demand and supply. Therefore, cloud services (Amazon EC2) provide a spot instance when a user's bid is higher than the current spot price. Furthermore, a running instance stops when a user's bid becomes less than or equal to the current

* Corresponding author

spot price. After a running instance stops, it restarts when a user's bid becomes greater than the current spot price.

We analyze the task and instance information from the price history data, and estimate the task size and instance availability from the analyzed data. A workflow is created using each available instance and the task size. However, the created workflow has a problem in that it does not consider the failure time of each instance. To solve this problem, we propose a scheme to change the task size of each instance using an estimation algorithm, such as Genetic Algorithm (GA).

2 Estimation Method

In this paper, using environment expands workflow scheduling scheme from our previous paper [5]. Our task distribution method determines the task size in order to allocate a task to a selected instance. Based on a compute-unit and an available state, the task size of an instance I_i (T_{I_i}) is calculated as

$$T_i = \left(\frac{U_i \times A_i}{\sum_{i=1}^{N} (U_i \times A_i)} \right) \times \frac{1}{U_i} \times T_{request} \times U_{baseline} \quad (1)$$

where $T_{request}$ represents the total size of tasks required for executing a user request. In an instance I_i, U_{I_i} and A_{I_i} represent the compute-unit and the available state, respectively. The available state A_{I_i} can be either 0 (unavailable) or 1 (available). The baseline represents the standard of the instance.

In our scheduling scheme, chromosome is defined as an assigned task to an instance. The length of chromosome composes the number of task. If available instances allocate the same length of chromosome, each instance is different task completion time. This reason, each instance has different the performance and the occurrence frequency of out-of-bid situation. The problem solution is the length of each chromosome varies to consider each instance condition (the performance, the occurrence frequency of out-of-bid situation, etc.). Therefore, we have designed a new crossover and mutation scheme for scheduling tasks that is based on the performance of each instance.

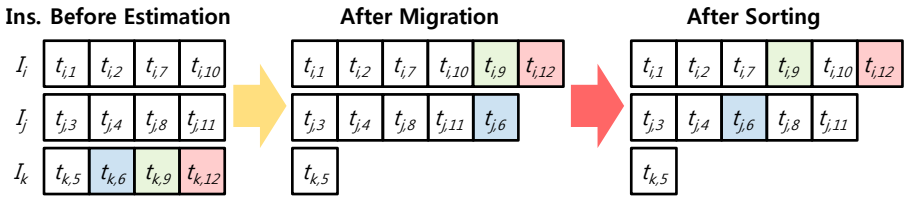

Fig. 1. Processing of migration and sorting

The scheduling scheme is depicted in Fig. 1. The instances I_i, I_j, and I_k have high, medium, and low performance, respectively. The instance I_k belongs to a positive

group and the other two instances (I_i, I_j) belong to a negative group. In the crossover operation, we select an instance to find the target instances that belong to the positive group. Next, we calculate the size of tasks in the positive group that are to be sent to the negative group (e.g., I_k). Finally, the calculated tasks are distributed to instances in the negative group (e.g., I_i and I_j) according to the performance of each instance. In mutation, we perform the re-arrangement of tasks. The re-arrange method sorts tasks in the increasing order of their indices.

3 Performance Evaluation

The simulations were conducted using the history data obtained from Amazon EC2 spot instances [6]. The history data before 10-01-2010 was used to extract the expected execution time and failure occurrence probability for our checkpointing scheme. The applicability of our scheme was tested using the history data after 10-01-2010. Table 1 shows the parameters and values for the simulation.

Table 1. Simulation parameters and values for instances

Simulation parameter	Task time interval	Distribution time	Merge time	Checkpoint time	Recovery time
Value	43,200(s)	300(s)	300(s)	300(s)	300(s)

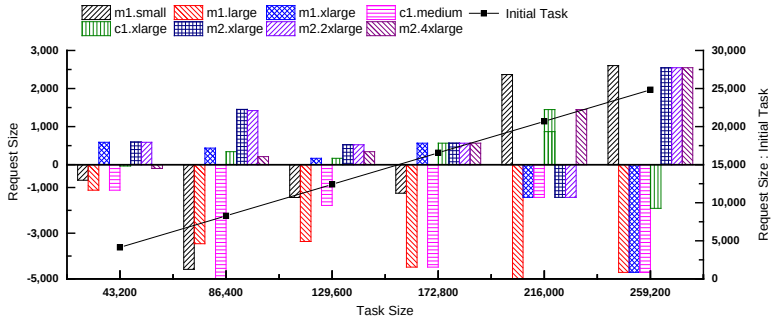

Fig. 2. Size variations in requested tasks

Fig. 2 shows the size variations of requested tasks in each instance before and after using the proposed estimation. Initial Task stands for the initial task size before using the estimation in all instances. The task size is determined based on the performance of each instance, and the task is distributed among the different instances. Each instance type (m1.small, m1.large, etc.) indicates the task size.

Fig. 3 shows the variation of the allocated task size in each instance I_i when task size is 86,400. In each instance, as the task size grows, the instance with high performance increased the task size, whereas the instance with low performance reduced the task size. It is due to the failure time of each instance. Therefore, we reduced the failure time of low performance instances in order to achieve similar estimated failure times across all instances.

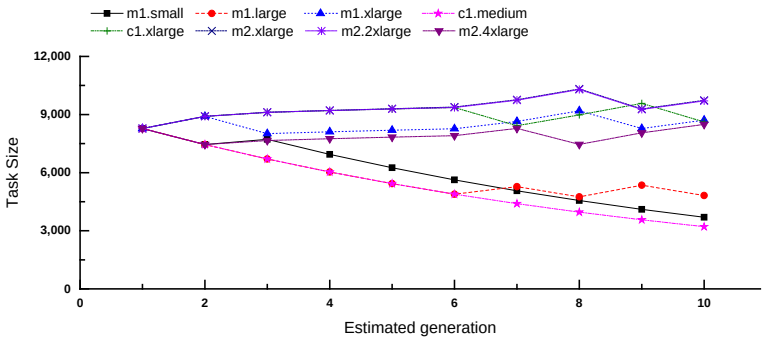

Fig. 3. Task size variation in each estimated generation

4. Conclusion

In this paper, we proposed an estimation-based task load balancing scheduling in unreliable cloud computing environments. The proposed scheduling technique achieves the task load balancing according to the performance of instances. In our scheme, we reduced the failure time of low performance instances in order to achieve similar estimated failure times across all instances.

Acknowledgments. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2013R1A1A3007940).

References

1. Elastic Compute Cloud (EC2), http://aws.amazon.com/ec2, (2013)
2. Ferraris, F.L., Franceschelli, D., Gioiosa, M.P., Lucia, D. and Ardagna, D., Di Nitto, E. and Sharif, T.: Evaluating the Auto Scaling Performance of Flexiscale and Amazon EC2 Clouds. In Proceedings of 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC). (2012) 423–429
3. Van, H.N., Tran, F.D., Menaud, J.M.: SLA-Aware Virtual Resource Management for Cloud Infrastructures. In Proceedings of the 2009 Ninth IEEE International Conference on Computer and Information Technology. Vol. 2. IEEE Computer Society. (2009) 357-362
4. Komal, M., Ansuyia, M. and Deepak, D.: Round Robin with Server Affinity: A VM Load Balancing Algorithm for Cloud Based Infrastructure. Journal of Information Processing Systems, Vol. 9. No. 3. (2013) 379-394
5. Daeyong Jung, JongBeom Lim, Heonchang Yu, JoonMin Gil, and EunYoung Lee.: A Workflow Scheduling Technique for Task Distribution in Spot Instance-Based Cloud. Proceeding CUTE2013 (2013) 409-416
6. Cloud exchange, http://cloudexchange.org, (2013)