An Algorithm in SwinDeW-C for Scheduling Transaction-Intensive Cost-Constrained Cloud Workflows

Yun Yang[1], Ke Liu[1, 2], Jinjun Chen[1], Xiao Liu[1], Dong Yuan[1], Hai Jin[2]

[1] *Faculty of Information and Communication Technologies*
Swinburne University of Technology
Hawthorn, Melbourne, Australia 3122
{yyang, kliu, jchen}@ict.swin.edu.au

[2] *School of Computer Science & Technology*
Huazhong University of Science and Technology
Wuhan, Hubei, China 430074
hjin@hust.edu.cn

The work presented focuses on scheduling for transaction-intensive cost-constrained cloud workflows, which are workflows with a large number of workflow instances (i.e. transaction intensive) bounded by a certain budget for execution (i.e. cost constrained) on a cloud computing platform (i.e. cloud workflows). First, the scheduling algorithms should take execution cost of 'pay for use' as one key factor. Second, as an important criterion of transaction-intensive workflows, mean execution time will be taken as another key factor. Third, the algorithms should facilitate multiple strategies for compromising execution time and cost with user input enabled on the fly. Finally, they should conform to the nature of cloud computing.

In general, there are two major types of workflow scheduling: market-driven and performance-driven. The former aims at achieving 'optimal' execution performance, normally without considering cost, by mapping workflow tasks onto resources according to some specific strategies, such as the Heterogeneous Earliest-Finish-Time algorithm [1] and a throughput maximisation strategy [2]. The latter tries to allocate resources for workflow tasks according to market models with cost imposed, such as Back-tracking [3], a genetic algorithm [4], the LOSS and GAIN approach [5] and the deadline distribution algorithm [6].

Because cloud workflow executions are unlikely free, cloud workflows scheduling usually belongs to market-driven strategies. Most existing market-driven strategies mentioned above are designed for scheduling a single workflow instance in general. However, for transaction-intensive workflows on a cloud computing platform, fierce competition on servers may occur and failures may happen from time to time. Thus the scheduling strategy needs to incorporate this situation accordingly. In addition, it also needs to consider the characteristics of cloud computing by compromising execution time and cost with user input enabled on the fly which is not considered in other algorithms.

SwinDeW-C System Design

SwinDeW-C (**Swin**burne **De**centralised **W**orkflow for **C**loud) cloud workflow system includes the following major parts: service clouds, cloud workflow execution agents, services catalogue, and user interface.

- *Service Cloud*: The servers with the same service are organised dynamically as a service cloud. Every server will automatically join the service clouds according to the service it can provide. In case when a cloud is too big, it is divided into several sub-clouds according to servers' geographical positions in order to reduce communication delay.
- *Cloud Workflow Execution Agent*: In order to access the services, every service is managed by a cloud workflow execution agent. This agent manages the services for workflow execution, including monitoring, data management and coordination with other agents which manage the services of the same type.
- *Cloud Services Catalogue*: Cloud services are a foundation of cloud computing. They are a variety of services available over the Internet that deliver compute functionality on the service provider's infrastructure. To access the services, there is a Cloud Services Catalogue which is a list of services that a user can request.
- *User Interface*: SwinDeW-C provides a user interface for users to monitor the workflow execution

978-0-7695-3535-7/08 $25.00 © 2008 IEEE
DOI 10.1109/eScience.2008.93

IEEE computer society

status and input the setting of compromised time and cost on the fly if needed. The input from the user on time and cost will be taken into account for scheduling the next round for better user satisfaction.

Our Scheduling Algorithm

We have proposed an algorithm for scheduling transaction-intensive cost-constrained cloud workflows. It has the following characteristics. (1) Considering the "pay for use" feature of cloud workflows, the algorithm takes execution cost and execution time as the two key considerations. The current primary purpose of the algorithm is to minimise the cost under certain user-designated deadlines. (2) The algorithm always enables the compromises of execution cost and time. It provides a just-in-time graph of time-cost relationship during workflow execution in the user interface for users to choose an acceptable compromise before the next round scheduling begins.

The algorithm can be described briefly as follows:

Pre-Step: Check uncompleted tasks and schedule them first in this round.

Step 1: Allocate sub-deadlines to tasks for each instance.

Step 2: Calculate estimated execution time and cost on each service.

Step 3: Allocate tasks to proper services.

Step 4: Provide just-in-time time-cost relationship graph for user to optionally choose an updated compromised deadline for scheduling.

Step 5: Sleep until next round scheduling.

Simulation and Comparison

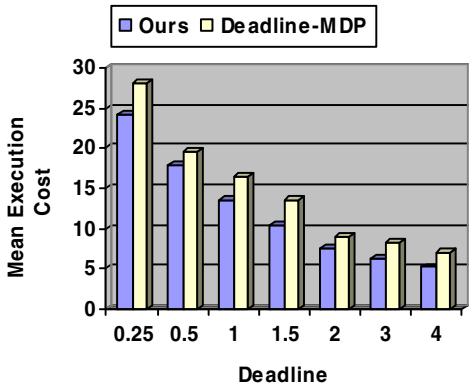

Figure 1: Comparison on mean execution cost

The simulation calculates the actual completion time and cost within 0.25 to 4 times of the input deadlines, using our algorithm and the currently most effective Deadline-MDP algorithm [6].

Figure 1 demonstrates the comparison results on the mean execution cost within different deadlines. Given both algorithms meet the deadlines, it can be seen that the mean execution cost of our algorithm is always lower than that of the Deadline-MDP algorithm in all circumstances. On average, the saving on the mean execution cost is over 15%.

Conclusions and Future Work

This paper has proposed an innovative transaction-intensive cost-constraint cloud workflow scheduling algorithm which takes cost and time as the main concerns with user input on the fly and incorporates the characteristics of cloud computing. The simulation has demonstrated that our algorithm can achieve lower cost than others while meeting the user-designated deadline.

In the future, we will try to apply our scheduling algorithm to some real world applications such as insurance claiming workflow systems.

Acknowledgement

The research work reported in this paper is partly supported by Australian Research Council under Discovery Grant DP0663841.

References

[1] T. Tannenbaum, D. Wright, K. Miller, and M. Livny. "Condor – A Distributed Job Scheduler", *Computing with Linux*, MIT Press, 2002.

[2] K. Liu., J. Chen, Y. Yang and H. Jin, "A Throughput Maximization Strategy for Scheduling Transaction Intensive Workflows on SwinDeW-G", *Concurrency and Computation: Practice and Experien.2008;20:1807-1820.*

[3] D. A. Menasc and E. Casalicchio, "A Framework for Resource Allocation in Grid Computing", *Proc. of the 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)*, 2004; 259-267.

[4] J. Yu and R. Buyya, "Scheduling Scientific Workflow Applications with Deadline and Budget Constraints using Genetic Algorithms", *Scientific Programming Journal*, IOS Press, 2006; 14(3-4): 217 – 230.

[5] R. Sakellariou, H. Zhao, E. Tsiakkouri, and M. D. Dikaiakos., "Scheduling Workflows with Budget Constraints", *CoreGRID Workshop on Integrated research in Grid Computing*, Technical Report TR-05-22, University of Pisa, Dipartimento Di Informatica, Pisa, Italy, November 2005; 347-357.

[6] J. Yu, R. Buyya, and C. K. Tham, "A Cost-based Scheduling of Scientific Workflow Applications on Utility Grids", *Proc. of the 1st IEEE International Conference on e-Science and Grid Computing*, Melbourne, Australia, December 2005; 140-147.