Task Profiling Algorithm for Scheduling Highly Heterogeneous Tasks on Cloud Computing

Thiar Hasbiya Ditanaya*, Royyana M Ijtihadie *, Bagus Jati Santoso *

* Department of Informatics, Institut Teknologi Sepuluh Nopember

Abstract- The use of cloud computing becomes more mature for large-scale system. To optimally utilize cloud computing resource, task scheduling algorithm needed. Task scheduling algorithm approach try to minimize completion time and optimally utilize cloud computing resource. The biggest challenge of task scheduling algorithm not only how to distribute tasks to workers fairly, but also distribute task to the right worker at the right time. On the real world, the tasks may have different resource usage. There exists task with high computational usage and a task with high memory usage. The different type of task may affect performance of task scheduling algorithm. Grouping or classification of task may increase scheduling performance [1]. In this paper, task scheduling algorithm based on task profiling is introduced. The task profiling algorithm profiles the task based on its resource need, then classify it to HIGH_CPU for high computing task, HIGH_MEMORY for high memory task or LIGHT for task which not consume much resource. Using profiled tasks, proposed system is able to choose worker more precisely. Result shown, the proposed task profiling algorithm gives better completion time compared to traditional scheduling algorithm.

Keywords- Cloud Computing; Task Scheduling; Task profiling; Load balance;

I. INTRODUCTION

Cloud computing technology becomes more mature for a large scale system. The use of cloud computing on large scale system may increase user satisfaction. User satisfaction can be measured by quality of service (QoS) [2] [3] [4]. There are many techniques to provide quality of service to the cloud applications, *Scheduling, admission control and dynamic resource provisioning* [5]. Scheduling manage cloud resource by scheduling and distributing task to worker in cloud computing datacenter. Worker can be physical or virtual machine [6].

A million of tasks may be executed at a time. That make completion time become considerably high [7]. The response time of each task that user may tolerate is 2 up to 8 seconds [8]. Minimizing response time can be done by reducing task completion time. Traditional scheduling algorithm minimize completion time by distributing task to the worker based on worker specification only. This approach works well in the homogeneous task or on the tasks with little difference resource usage. The biggest drawback of this approach is imbalance load where the tasks have different resource usage. for example, there exist task A and B. Task A needs 8 seconds computation time and 50 megabytes memory in order to run and finished. On the other hand, task B needs 4 second computation time and 500 megabytes memory in order to run and finished. If there are a bunch of task A and B scheduled and distributed normally only based on worker specification. There will be a worker with a lot of task B and a lot of task A. The worker with a lot of task A will finish slower and increase completion time, and worker with a lot of task B will fail due to run out of memory. High completion time and low success rate can reduce QoS and impact to user satisfaction.

To overcome the drawback, more advanced scheduling algorithm is required. We need scheduling algorithm that made decision not only based on worker specification but also based on resource usage of task. In this paper, we proposed new scheduling algorithm based on resource usage of task and worker specification. This new algorithm proposed to minimize task completion time and increase success rate. Hereafter in this paper, will be organized as: Section II present overview of previous works about task scheduling algorithm. In Section III present the new task profiling algorithm. In Section IV presents about implementation of algorithm and the tests scenario. Later in Section V will be a discussion of the test results. Finally, Section VI give a conclusion of the papers and possibility of future works.

II. RELATED WORK

Traditional task scheduling intends to minimize task completion time and increase success rate. The scheduling algorithm known as NP-Complete problem. Task scheduling architecture shown in Figure 1.

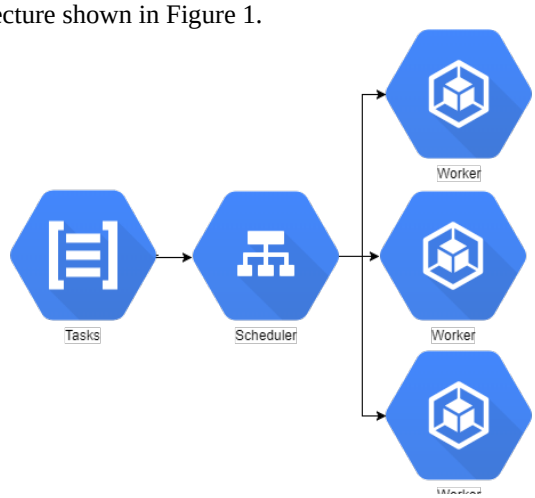

Figure 1: Task Scheduling Algorithm

In cloud environments, scheduling problem becomes more challenging because of the characteristics of high heterogeneity in resources and high heterogeneity of tasks [9]. Several task scheduling algorithms already proposed. Kumari and Saxena [10] have a study about traditional task scheduling based on round robin. They make comparison of round robin, weighted round robin, least connections, weighted least connections and random. The result shown that the round robin approach has good performance on homogeneous task and homogeneous worker. Later Kaur and Luthra [11] give a study of performance min-min and max-min algorithm. They compare performance min-min and max-min to first come first Served algorithm (FCFS) and round robin scheduling algorithm (RR). The result shown, min-min and max-min have better resource utilization and performance compared to another algorithm.

1) Round-Robin

Round robin approach distributes task evenly across all workers. In round robin, tasks will be distributed normally without considering capacity of workers. Suppose, there are Server A and Server B. Server A have 4 cores of CPU and 2 Gigabytes memory. Server B have 1 cores of CPU and 512 Megabytes memory. We know that Server A have resources twice than Server B. So, Server A may handle request capacity twice larger than Server B. But using round robin approach, task will be distribute evenly on Server A and Server B. Because of task evenly distributed, Server B will get overloaded faster than server A. Overloaded servers are at risk of failure and may go down. According to round robin behavior, it is fit to be used in homogenous worker only.

2) Min-Min

Min-min scheduling algorithm complete the shortest task first. Algorithm work with following step: First, the algorithm calculates completion time of each task and sort it from the shortest task. Then, worker sorted from the fastest finished the task. Finally, task will be executed sequentially, that means one worker can only executed one task at the same time.

3) Max-Min

Max-min algorithm work like min-min algorithm. While min-min algorithm execute task from the shortest completion time first, max-min algorithm execute task from the longest one. This approach improves min-min algorithm by execute longest task first. Max-min algorithm best suite in environments with more longest task than shortest task.

III. PROPOSED METHOD

Task profiling algorithm consists of following steps: collecting worker available resource, collecting task resource usage, profiling and task scheduling.

A. Collecting worker available resources

Before run the profiling, we need to collect worker available resources and task resource usage. Worker available resources are collected by measuring number of CPU cores and total physical memory. After collecting the data, we will get Worker Resource (*WR*) variable. *WR* variable consist of WR_{CPU} and WR_{mem}.

1. **For** worker in worker list;
2. Get WR_{CPU} and Get WR_{mem}
3. End **For**;

Figure 2: Collecting worker available resources

B. Collecting task resource usage

Collecting task resource usages can be done by executing each task to every worker and get the resource usages. The resource measured is computation time of CPU and memory usage. Computation time measured in second, while memory usage in megabytes. After collecting data, we will get Task Resource (*TR*) variable. *TR* variable consist of CPU computation time, memory usage and worker list sorted from the fastest.

1. **For** task in task list; T_i
2. **For** worker in worker list; W_j
3. Execute T_i on W_j
4. Get $TR_{i(\mathrm{CPU})}$ and $TR_{\mathrm{i(mem)}}$
5. End **For**;
6. Sort *W* based on the fastest finished the task
7. End **For**;

Figure 3: Collect task resource usage

C. Profiling

In order to get task executed to the right server, tasks needs to be classified. Classification of task can be done by profiling step. The output of the profiling step is the task classification, HIGH_CPU for task with high computational time, HIGH_MEM for task with high memory and LIGHT for task which not consume much resource. Classification of task aims to prevent task with same classification executed on the same worker at the same time. For example, Worker A execute a HIGH_MEMORY task. Currently worker A lose a lot of memory due to executing HIGH_MEMORY task, if worker A execute another HIGH_MEMORY task it may lead worker A to a failure. The Profiling use variable Worker Resource (*WR*) and Task Resource (*TR*) from the previous step. The following rule used to determine the classification:

- If average completion time of task ($\overline{TR_{cpu}}$) more than 8 s [8], the task is HIGH_CPU.
- If average memory usage of task ($\overline{TR_{mem}}$) more than average all worker memory ($\overline{WR_{mem}}$), the task is HIGH_MEM.
- Else the task is LIGHT.

1. **For** task resource in task resource list; TR_i
2. **If** $\overline{TR_{cpu}} > 8$ s then
3. TR_iweight = HIGH_CPU
4. **Else If** $\overline{TR_{mem}} > \overline{WR_{mem}}$ then
5. $TR_i weight$ = HIGH_MEM

6. **Else**
7. $TR_i weight$ = LIGHT
8. End **If**;

Figure 4: Task Profiling

D. *Task scheduling*

Task scheduling distributes tasks using *TR* and *WR* as reference. The main idea is to avoid overloaded worker and utilize resource optimally. Scheduling algorithm works as follow

1. **For** task in scenario; T_i
2. Search TR_i corresponding to T_i
3. Get W_j from TR_i
4. **If** $TR_i weight$ not LIGHT
5. **If** $TR_i weight$ not same as $W_j current\_task\_weight$
6. Execute T_i
7. Update $W_j current\_task\_weight$
8. End **If**;
9. Update W_j and TR_i resource
10. **Else**
11. Execute T_i
12. End **If**;

Figure 5: Task Profiling Scheduling Algorithm

IV. EXPERIMENT AND IMPLEMENTATION

To evaluate performance of this new scheduling algorithm, we setup the experiment environment. We implement the scheduling algorithm by using python and psutil library. We use virtual machine provide by proxmox virtual environment for worker machine. We design experiment architecture shown in Figure 6.

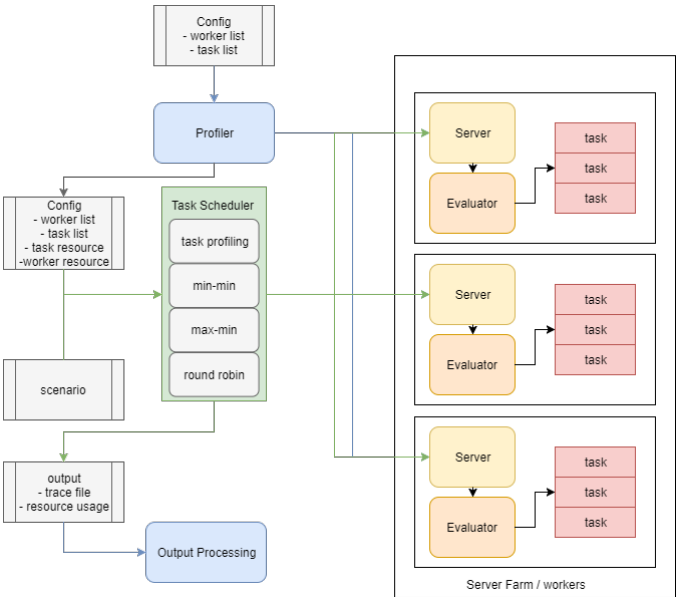

Figure 6: Experiment Architecture

Server farm or workers will run under proxmox virtual environment. Each worker will run server and evaluator. Server waits for incoming connection and execute the task through evaluator. Evaluator executes the incoming task and get the resource usage and completion time of the task. Before running task scheduler, profiler must be initially executed to get Worker Resource (*WR*) and Task Resource (*TR*).
Specification of machine and software are used in experiment as following:

- Python version 2.7 and psutil version 5.4.2
- Proxmox PVE Manager Version pve-manager/5.1-35/722cc488
- Physical server to run proxmox pve with specification:
  - 8 Core CPU 8 x Intel(R) Core(TM) i7-2700K CPU @ 3.50GHz (1 Socket)
  - RAM 16 GB
  - HDD 100 GB

For experiment, we used two scenarios. The first scenario is consisted of homogeneous workers and second scenario is consisted of heterogeneous workers. We use three different tasks for experiment that shown in Table 1.

Table 1: Task for Experiment

Task Name	Descriptions
write_high	Write 20000000 row
read_high	Read 65000000 row of file
read_moderate	Read 5000000 row of file

First scenario uses homogeneous workers with specification as following:

Table 2: Homogenous Workers Scenario

Step	Worker			Task	
	Name	Memory	CPU core	Name	number
1	Wk-1	1024 MB	1	write_high	5
	Wk-2	1024 MB	1	read_high	5
	Wk-3	1024 MB	1	read_moderate	5
2	Wk-1	1024 MB	1	write_high	10
	Wk-2	1024 MB	1	read_high	10
	Wk-3	1024 MB	1	read_moderate	10
3	Wk-1	1024 MB	1	write_high	15
	Wk-2	1024 MB	1	read_high	15
	Wk-3	1024 MB	1	read_moderate	15
4	Wk-1	1024 MB	1	write_high	30
	Wk-2	1024 MB	1	read_high	30
	Wk-3	1024 MB	1	read_moderate	30
5	Wk-1	1024 MB	1	write_high	45
	Wk-2	1024 MB	1	read_high	45
	Wk-3	1024 MB	1	read_moderate	45

Second scenario uses heterogenous workers with specification as following:

Table 3: Heterogenous Workers Scenario

Step	Worker			Task	
	Name	Memory	CPU core	Name	number
1	Wk-4	512 MB	4	write_high	5
	Wk-5	2048 MB	1	read_high	5

	Wk-6	2048 MB	4	read_moderate	5
2	Wk-4	512 MB	4	write_high	10
	Wk-5	2048 MB	1	read_high	10
	Wk-6	2048 MB	4	read_moderate	10
3	Wk-4	512 MB	4	write_high	15
	Wk-5	2048 MB	1	read_high	15
	Wk-6	2048 MB	4	read_moderate	15
4	Wk-4	512 MB	4	write_high	30
	Wk-5	2048 MB	1	read_high	30
	Wk-6	2048 MB	4	read_moderate	30
5	Wk-4	512 MB	4	write_high	45
	Wk-5	2048 MB	1	read_high	45
	Wk-6	2048 MB	4	read_moderate	45

V. EXPERIMENT RESULT

After running profiling, classification of tasks are shown in Table 4:

Table 4: Task Classification

Task Name	Classification
write_high	HIGH_CPU
read_high	HIGH_MEMORY
read_moderate	LIGHT

Performance of four scheduling algorithm evaluated by completion time and success rate. The lower completion time and higher success rate the better performance of the scheduling algorithm.

In homogenous workers scenario, completion time present in Figure 7 and success rate in Figure 8.

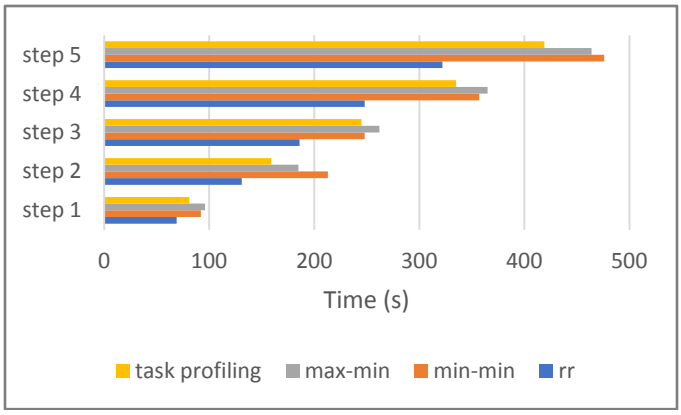

Figure 7: Completion Time Scenario Homogeneous Workers

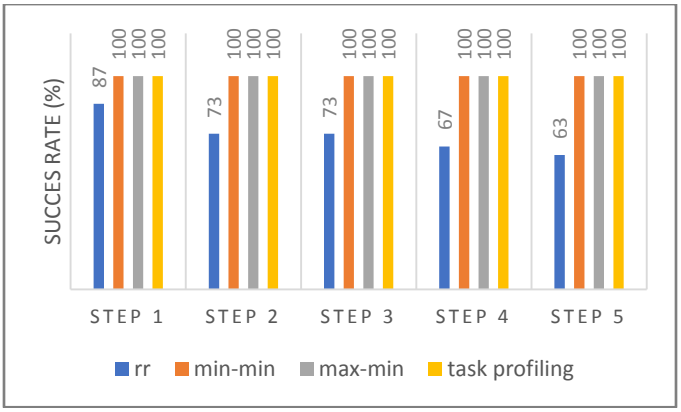

Figure 8: Success Rate Scenario Homogeneous Workers

In Figure 7, round robin algorithm has a lowest completion time because there is so many tasks failed. In Figure 8, round robin algorithm has lower success rate comparing to three others algorithm. In other side, task profiling algorithm lower the completion time and keep success rate high.

In heterogenous workers scenario, completion time present in Figure 9 and success rate in Figure 10.

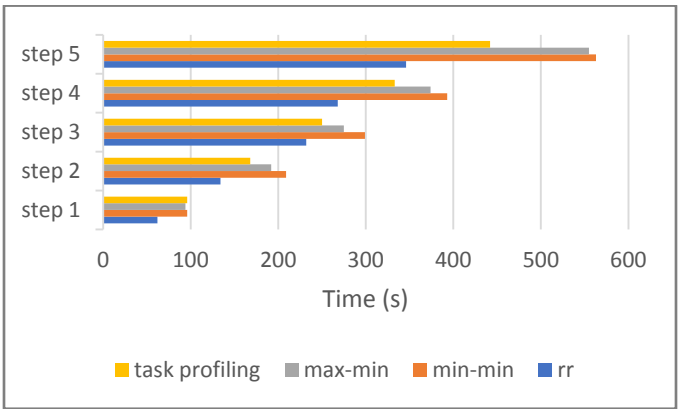

Figure 9: Completion Time Scenario Heterogenous Workers

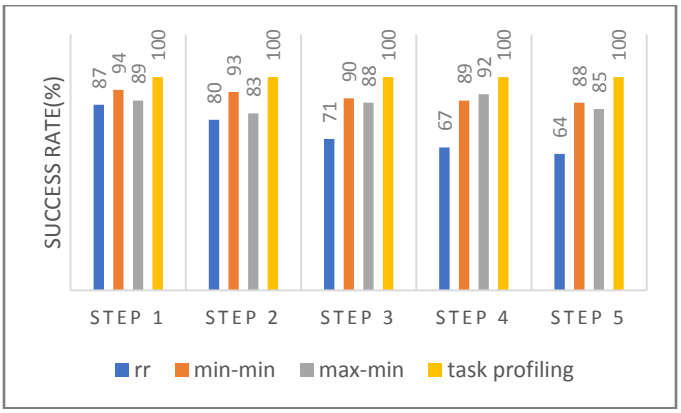

Figure 10: Success Rate Scenario Heterogenous Workers

Same as in homogeneous worker scenario, in heterogenous scenario task profiling algorithm show best performance comparing to another algorithm. In Figure 10, task profiling algorithm can maintain success rate at the highest value when the others algorithm failed.

VI. CONCLUSION AND FUTURE WORK

In high heterogeneity task environments, scheduling algorithm can't distribute the load in the usual way. Evenly task distribution may overload a worker and lead to a failure. To many failed tasks make a lower success rate and make completion time higher. The proposed task profiling algorithm prove that it suits on high heterogeneity task environments. Task profiling algorithm profiled the task, and distribute task based on task classification to the right resource with available resource. Task profiling algorithm can decrease completion time and maintain success rate.

References

[1] L. M.Mustafa, M. K. Elmahy and M. H.Haggag, "Improve Scheduling Task based Task Grouping in Cloud Computing System," *International Journal of Computer Applications*, vol. 93, no. 8, pp. 1-5, 2014.

[2] A. Al-Moayed and B. Hollunder, "Quality of Service Attributes in Web Services," in *International Conference on Software Engineering Advances*, Nice, France, 2010.

[3] D. Ardagna, G. Casale, M. Ciavotta, J. F. Pérez and W. Wang, "Quality-of-service in cloud computing: modeling techniques and their applications," *Journal of Internet Services and Applications*, vol. 5, no. 11, pp. 1-17, 2014.

[4] H. A. Akpan and B. R. Vadhanam, "A Survey on Quality of Service in Cloud Computing," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 27, no. 1, pp. 58-63, 2015.

[5] H. H. Ramadan and D. D. Kashyap, "Quality of Service (QoS) in Cloud Computing," */ (IJCSIT) International Journal of Computer Science and Information Technologies*, vol. 8, no. 3, pp. 318-320, 2017.

[6] M. Ahmed, "PHYSICAL SERVER AND VIRTUAL SERVER: THE PERFORMANCE TRADE-OFFS," *European Scientific Journal*, vol. 9, no. 12, pp. 222-232, 2013.

[7] Y. Jin and M. Tomoishi, "Web server performance enhancement by suppressing network traffic for high performance client," in *Network Operations and Management Symposium (APNOMS)*, Busan, South Korea, 2015.

[8] F. Nah, "A Study on Tolerable Waiting Time: How Long Are Web Users Willing to Wait?," in *Conference: 9th Americas Conference on Information Systems*, Tampa, FL, USA, 2003.

[9] H. Chen, F. Wang, N. Helian and G. Akanmu, "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing," in *National Conference on Parallel Computing Technologies (PARCOMPTECH)*, Bangalore, India, 2013.

[10] P. Kumari and M. Saxena, "A Round-Robin based Load balancing approach for Scalable demands and maximized Resource availability," *International Journal Of Engineering And Computer Science*, vol. 5, no. 8, pp. 17375-17380, 2016.

[11] R. Kaur and P. Luthra, "Load Balancing in Cloud System using Max Min and Min," *International Journal of Computer Applications*, pp. 31-34, 2014.

AUTHORS

First Author
Name: Thiar hasbiya Ditanaya
Qualification: student
Associated Institute: Department of Informatics, Institut Teknologi Sepuluh Nopember
Email Address: thiar.hasbiya@gmail.com

Second Author
Name: Royyana M Ijtihadie
Qualification: Lecturer
Associated Institute: Department of Informatics, Institut Teknologi Sepuluh Nopember
Email Address: roy@if.its.ac.id

Third Author
Name: Bagus Jati Santoso
Qualification: L
ecturer
Associated Institute: Department of Informatics, Institut Teknologi Sepuluh Nopember
Email Address: bagus@if.its.ac.id