Load Balancing Algorithm Based on Estimating Finish Time of Services in Cloud Computing

Nguyen Khac Chien*, Nguyen Hong Son**, Ho Dac Loc***
* University of the People's Police, Ho Chi Minh city, Viet Nam
** Post and Telecommunication Institute of Technology, Ho Chi Minh city, Viet Nam
*** Ho Chi Minh City University of Technology, Ho Chi Minh city, Viet Nam
nkchienster@gmail.com, ngson@ptithcm.edu.vn, hdloc@hcmhutech.edu.vn

Abstract— **Cloud computing is an emerging trend in the field of Information Technology and includes a large of distributed resources. The main goal of cloud computing service providers is to provide resources to workloads efficiently. Load balancing is an important technique for the goal. The technique is responsible for optimizing resource utilization, maximizing throughput, minimizing response time, and avoiding overloading of any single resources. So far, many load balancing algorithms have been proposed but their performance has to be still desired. In this paper, we propose a novel load balancing algorithm that is based on the method of estimating the end of service time. The simulation results show that our proposed algorithm improves response time and processing time.**

Keywords— **Cloud computing, Load balancing, Virtual machine Migration, Datacenter, Scheduling policy.**

I. INTRODUCTION

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management efforts or service provider interaction [11]. However, there are various issues in cloud computing usage that need to be addressed by varieties of solutions. It cannot provide high performance without load balancing. Workloads must be submitted to proper hosts in order to achieve maximum resource utilization with higher availability at minimized cost. Thus, there are many load balancing algorithms proposed for cloud computing and improved continuously. Load balancing algorithms are broadly categorized into static load balancing and dynamic load balancing. In case of static load balancing, load distribution depends on the load at the time of selection of node whereas dynamic load balancing performs load distribution at run time. It uses current load information for making distribution decisions [12]. Some dynamic algorithms are adaptive; for example, the algorithms can be modified as the system state changes.

In this paper, we analyze the dynamic load balancing algorithm in [3]-[10] and propose a load balancing algorithm based on the method of estimating the end of service time with the aim to improve the performance of cloud computing in terms of response time and processing time.

The rest of this paper is organized as follows: The related works are reviewed in Section II. In Section III, we propose a load balancing algorithm which is based on estimating the point of time to complete services in the heterogeneous cloud environment. Computer simulations are described in Section IV. Finally, we conclude our study in Section V.

II. RELATED WORK

Load balancing is the process of distributing the load among various nodes of a distributed system to improve resource utilization and job response time, also avoiding a situation of filling up a certain node with heavy load. Load balancing schemes ensure that all processors in the system or every node in the network execute approximately an equal amount of workload at any instant of time [6]-[9]. There are many load balancing schemes developed in [1]-[4]-[5]-[8]. However, no scheme is suitable for all applications and all distributed computing systems. The choice of load balancing mechanisms depends on applications and hardware specifications.

Load balancing can be applied to two levels: First, at the host level – It is possible to specify how much of the overall processing power of each core will be assigned to each virtual machine (VM), known as VM scheduling policy. Second, at the VM level – the VM assigns a fixed amount of the available processing power to the individual application services (task units) that are hosted within its execution engine, known as task scheduling policy. There are two types of scheduling policy: time-shared (TS) scheduling and space-shared (SS) scheduling. At each level, TS of SS can be used and the choice depends on specific system designs [3].

Research in [3]-[10]-[13] proposed an Active Monitoring Load Balancer (AMLB) algorithm. The algorithm keeps information about each VMs and the number of requests currently allocated to which VMs. When a request arriving, it identifies the least loaded VM and allocates the incoming request to the VM. Thereby, VMs with powerful processing capability is assigned more workloads than weaker VMs. This mitigates bottleneck situations and improves significantly response time.

Since the AMLB algorithm in [3]-[10] just based on known processing power of VMs and current number of assigned jobs, actual instant power of VMs and job size are not

considered. This may result assigning jobs to improper VMs. Because a VM with the strongest processing power early given and with the smallest number of assigned jobs may not the current strongest VM. As mentioned above, instant processing power of VMs depends on scheduling policy at two scheduling levels. Moreover, a big size job will take more processing power than small one. Thus, the number of jobs currently allocated to VM does not reflect the actual instant power of the VM.

III. PROPOSED LOAD BALANCING ALGORITHM

In our proposed algorithm, we take account of both actual instant processing power of VM and size of assigned jobs. We include two factors in a method of estimating the end of service time in VMs. The criterion of VM selection for the next job is VM can soonest finish it.

On next allocation request, the load balancing algorithm must estimate the time that all queuing jobs and the next job (of current request) are completely done in every VM. The VM that corresponds with the earliest will be chosen to distribute the job.

It is complicated to determine actual instant processing power of VMs. The processing power varies depending on scheduling policy at two scheduling levels in cloud computing. We use the way of calculating finish time in [7] and complete it with proposed formulas of calculating the processing power of virtual core depending on various scheduling cases.

Measuring the effectiveness of load balancing depends on several factors of which: load and capacity. Load is the queue index CPU and CPU utilization. The capacity has average response time of a user request. The objective of our algorithm is to improve the response time and processing time in four scheduling cases:

(1) SS in host level and SS in VM level;
(2) SS in host level and TS in VM level;
(3) TS in host level and SS in VM level;
(4) TS in host level and TS in VM level.

The expected response time can be determined by the following formula [3]-[10]-[7]:

$$Response\ Time = Fin_t - Arr_t + TDelay \quad (1)$$

Where, Arr_t is the arrival time of user request; Fin_t is the finish time of user request and *TDelay* is the transmission delay.

+ If the host scheduling policy is SS-SS or TS-SS, Fin_t can be determined by the formula:

$$Fin_t = est(p) + \frac{rl}{Capacity \times cores(p)} \quad (2)$$

+ If the host scheduling policy is SS-TS or TS-TS, Fin_t can be determined by the formula:

$$Fin_t = ct + \frac{rl}{Capacity \times cores(p)} \quad (3)$$

Execution time of a Job can be determined by the following formula:

$$Execution\ Time = \frac{rl}{Capacity \times cores(p)} \quad (4)$$

Or the processing time is the time of core processing, calculation speed is MIPS.

Where, *est(p)* is the time that job p is started; *ct* is the current simulation time; *rl* is the total number instruction of Job; *core(p)* is the number of cores, or processing elements required by Job; *Capacity* is the average processing capacity (in MIPS) of a core for job.

The *Capacity* parameter determines real performance for processing job on each VM. Obviously, *Capacity* depends on the scheduling policy on virtualized systems. Total processing capacity on a physical host is constant and depending on the number of physical cores and processing power of each core. However, when these resources are shared for many jobs simultaneously, each job requires some certain cores. If total number of the core is greater than the total number of physical core, there appears the concept of virtual core, and processing power of each virtual core may be smaller than of physical core. Therefore, *Capacity* is intrinsically the average processing power of a virtual core. From this analysis, we develop formulas of *Capacity* calculation in four scheduling cases as mentioned above. We have two levels of scheduling: scheduling VMs to share physical host resources and scheduling job to share VMs resources. Here, we propose formulas to calculate *Capacity* in each case and it will be used in our proposed load balancing algorithm:

(1) VM scheduling policy is SS, task scheduling policy is SS:

$$Capacity = \sum_{i=1}^{np} \frac{Cap(i)}{np} \quad (5)$$

Where, *Cap(i)* is the processing power of the core i, *np* is the number of real core of the considered host.

(2) VM scheduling policy is SS, task scheduling policy is TS:

$$Capacity = \frac{\sum_{i=1}^{np} Cap(i)}{Max(\sum_{j=1}^{\alpha} cores(j), np)} \quad (6)$$

Where, *cores (j)* is number of cores that job *j* needs. α is total job in VM which contains the job.

(3) VM scheduling policy is TS, task scheduling policy is SS:

$$Capacity = \frac{\sum_{i=1}^{np} Cap(i)}{Max(\sum_{k=1}^{\beta} \sum_{j=1}^{\gamma} cores(j), np)} \quad (7)$$

Where, β is number of VMs in the current host. γ is number of jobs running simultaneously in VM_k.

The reason why we write such expression is because VM can also have multiple jobs running simultaneously in SS scheduling. This is the case that number of cores in VM is greater than the number of cores required by a job. For example, VM has 4 cores, each job just needs 2 cores, there are two jobs running simultaneously).

(4) VM scheduling policy is TS, task scheduling policy is TS:

$$Capacity = \frac{\sum_{i=1}^{np} Cap(i)}{Max(\sum_{j=1}^{\delta} cores(j), np)} \quad (8)$$

Where, δ is total job of the considered host.

Since load balance algorithms perform in the DatacenterBroker, the parameter of transmission delay can be ignored (*TDelay = 0*).

Our load balancing algorithm based on the estimated time to complete service is described in figure 1.

Step 1: *Create a DatacenterBroker and maintaining a status index table of the VM and current Job allocated to any VM and determine whether its processing status have completed or not. At the same time of creating DatacenterBroker, no VM is allocated Job.*
Step 2: *When there is a request to allocate a VM, DatacenterBroker will analyse status index table, estimate the time of completion of the job on each VM based on the formulas proposed above. The calculation also includes the existing jobs in the queue of each VM. The VM with the earliest completion time that will be selected for the job. If there are more than one, the first one will be is selected.*
Step 3: *The algorithm return Id of the selected VM to DatacenterBroker.*
Step 4: *DatacenterBroker posts job to VM that are identified by Id.*
Step 5: *DatacenterBroker notify the algorithm about the new allocation.*
Step 6: *The algorithm will update the status index table of VM and of job.*
Step 7: *When the VM finishs processing requirements and DatacenterBroker is responded about job, it will update that job is completed in the status index table and decrease 1 job in the index table.*
Step 8: *Continue to step 2*

Figure 1. Description of the proposed algorithm

IV. EXPERIMENT

A. Configuration of Simulated Cloud Computing Model

In this section, we conducted experiments to compare the proposed algorithm with AMLB algorithm [3]-[10] in terms of the average response time and average processing time. Algorithms are programmed in the Java language using the CloudSim library [2]-[7].

Our simulations implemented in the paper include one Datacenter and three VMs running on the physical host. System parameters are given in Table 1 and we change the scenario by submitting the number of jobs increasing from 10 to 50 jobs to clarify the variation of response time and data processing time of the two algorithms. We also simulate in each of the four scheduling cases.

TABLE 1. SETTING CLOUD PARAMETER

Type	Parameter	Value
Datacenter	Number of Datacenter	1
	Number of Host	3
Host	Number of PE on Host	1-4
	MIPS of PE	10000 – 40000 MIPS
	Memory of Host	3072-12288
	Storage	1048576 - 4194304
	Bandwidth (BW)	10000 MB
Virtual machine (VM)	Number of VM	3
	RAM	1024–4096
	Bandwidth	1024 MB
Task/job	Number of job	10 - 50
	Length of job	1246-9660MI
	No. PE requirement	1-3

TABLE 2. PARAMETERS CONFIGURE VMS

Id	Memory (Mb)	Bandwidth (Mb)	No. PE/Core	Rate PE (MIPS)
0	4096	1024	4	32400
1	2048	1024	2	16200
2	1024	1024	2	8100

B. Experimental Results

1. Scenario 1:

In this case, SS policy applies to both the VM and the task. Using formula (2) to calculate the estimated completion time of the VM and the formula (5) for the calculation of the total performance of a host.

Figure 2 and Figure 3 show the performance of the two algorithms based on the average response time (ms) and average processing time (ms).

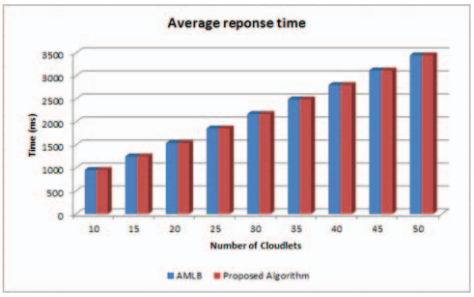

Figure 2. Average response time in scenario1

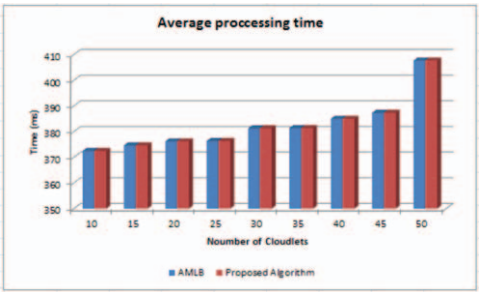

Figure 3. Average processing time in scenario1

Thus, if using SS scheduling policy for both VMs and tasks, the average processing time and average response time is equal for both algorithms.

2. Scenario 2:

In this case, TS policy is applied to allocate VMs to the host, while the task provided is based on SS policy. Estimated completion time of a job p is managed by the VM *i* and it is calculated using the formula (2) and the total capacity of a host with *np* processing elements, is calculated using the formula (7).

Figure 4 and Figure 5 describe the result of two algorithms implemented based on the average response time (ms) and average processing time (ms).

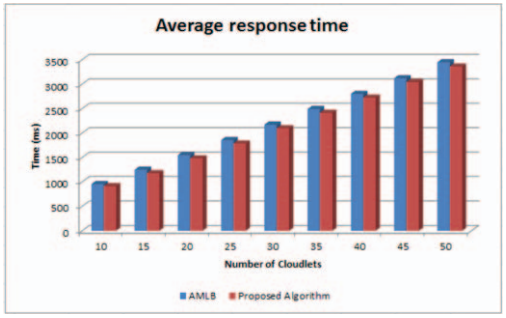

Figure 4. Average response time in scenario2

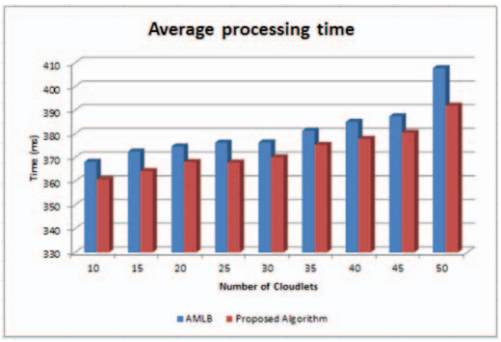

Figure 5. Average processing time in scenario2

Figure 4 and Figure 5 show that the response time and processing time of the proposed algorithm is improved much more than that of the AMLB algorithm.

3. Scenario 3:

In this case, SS policy is applied to allocate VM to the host and TS policies is formed on the basis of allocating tasks to processing core within a VM. Therefore, during the lifetime of the VM, all the tasks assigned to it are removed in the dynamic context. By using TS policy, estimated completion time of a job is managed by a VM and is calculated using the formula (3). In the TS policy, many job can have multi-tasks inside a VM. In this case, the total processing capacity of host is calculated by the formula (6).

Figure 6 and Figure 7 describe the result of two algorithms implemented based on the average response time (ms) and average processing time (ms).

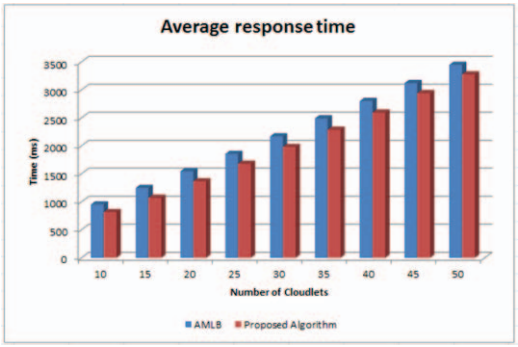

Figure 6. Average response time in scenario3

Figure 6 and Figure 7 show that the response time and processing time of the proposed algorithm is improved much more than that of the AMLB algorithm.

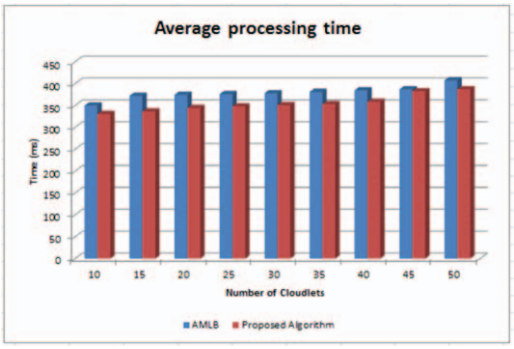

Figure 7. Average processing time in scenario3

4. Scenario 4:

In this case, TS schedule policy is applied to both the VM and task. Therefore, the processing power is shared simultaneously by VMs and each VM is shared by among its tasks. In this case, there is no queue latency related to the task. In the TS policy, estimated completion time of a job is calculated using the formula (3). In this case, the total processing capacity of host is formula (8).

Figure 8 and Figure 9 describe the result of two algorithms implemented based on the average response time (ms) and average processing time (ms).

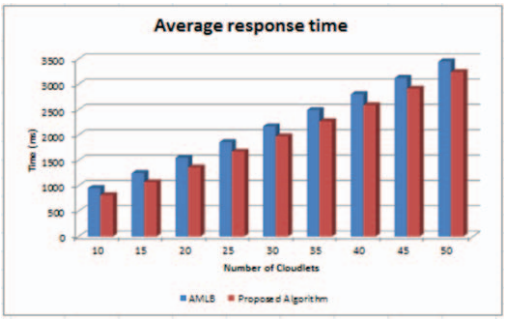

Figure 8. Average response time in scenario4

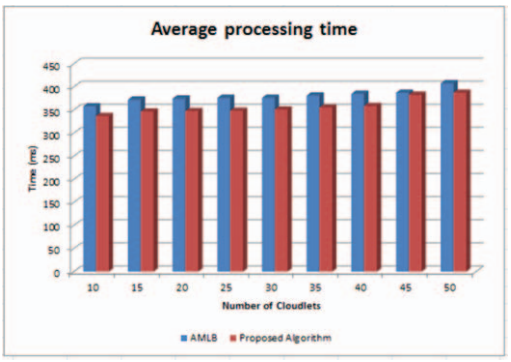

Figure 9. Average processing time in scenario4

Figure 8 and Figure 9 show that the response time and processing time of the proposed algorithm is improved than the AMLB algorithm.

The experimental results show that the most powerful VM with ID-0 in the proposed algorithm has been allocated much more jobs than the strongest VM with ID-0 in AMLB algorithm. The weakest VM with ID-2 in the proposed algorithm has always been allocated less job than the weakest

VM with ID-2 in the AMLB algorithm. So, in all cases, the average response time and average data processing time of proposed algorithm in three scheduling cases of SS-TS, TS-SS, and TS-TS are always smaller than the AMLB algorithm, only in the scheduling policy of SS-SS results in equal response time and equal data processing time.

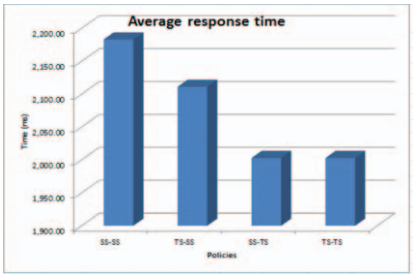

Figure 10. Average response time of various scheduling cases

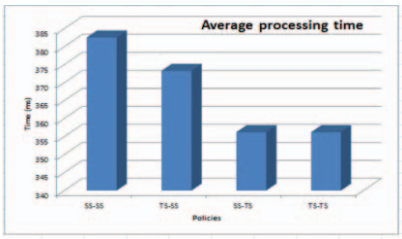

Figure 11. Average processing time of various scheduling cases

Also, the average response time and the average processing time corresponding to four scheduling cases of the proposed algorithm are showed in Figure 10 and Figure 11. The average response time and the the average processing time of TS-TS and SS-TS scheduling result are the best, the next is TS-SS scheduling policy and the worst is SS-SS scheduling policy.

V. CONCLUSION

In this paper, we propose a load balancing algorithm based on the method of estimating the end of service time in heterogeneous cloud computing environments. Scheduling cases of different levels were taken into account when we propose formulas to calculate the average processing power of a virtual core. Simulation results showed that the proposed algorithm is more effective. The processing time and response time are improved in four scheduling cases. Especially, the cases of time-shared always give the best results.

Load balancing directly affects the issue of datacenter power consumption with variety of workloads in the cloud. Load balancing helps effectively utilize computational resources, improve efficiency performance but it also causes the problem of energy consumption and carbon dioxide emissions. The challenge that needs an appropriate solution for balancing between energy consumption and emissions with carbon dioxide in cloud computing environments.

REFERENCES

[1] Ajith, S.N., Hemalatha, M., "An Approach on Semi-Distributed Load Balancing Algorithm for Cloud Computing System", *International Journal of Computer Applications*, 2012.

[2] CloudSim 3.0 API (Application Programming Interface), The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, The University of Melbourne, available: http://www.cloudbus.org/cloudsim/doc/api/index.html.

[3] Jasmin, J., Bhupendra, V., "Efficient VM load balancing algorithm for a cloud computing environment", *International Journal on Computer Science and Engineering (IJCSE)*, 2012.

[4] Md, F. A., Rafiqul, Z. K., "The study on Load Balancing strategies in distributed computing system", *International Journal of Computer Science & Engineering Survey (IJCSES)*. Vol.3, No.2, 2012.

[5] Ram, P.P., Goutam, P. R., "Load Balancing In Cloud Computing Systems", *Department of Computer Science and Engineering National Institute of Technology*, Rourkela Rourkela-769 008, Orissa, India, 2011.

[6] Ratan, M. and Anant, J., "Ant colony Optimization: A Solution of Load balancing in Cloud", *International Journal of Web & Semantic Technology (IJWesT),* Vol.3, No.2, April 2012.

[7] Rodrigo, N. C., Rajiv, R., Anton, B., Cesar, A. F. D. R., and Rajkumar, B., CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, *Software: Practice and Experience,* Volume 41, Number 1, Pages: 23-50, ISSN: 0038-0644, Wiley Press, New York, USA, January 2011.

[8] Soumya, R. and Ajanta, D.S., "Execution Analysis of Load Balancing Algorithms in Cloud Computing Environment", *International Journal on Cloud Computing: Services and Architecture (IJCCSA),*Vol.2, No.5, October 2012.

[9] Kumar, Y. R., MadhuPriya, M., Chatrapati, K. S., "Effective Distributed Dynamic Load Balancing For The Clouds", *International Journal of Engineering Research & Technology (IJERT)*, Vol. 2 Issue 2, February-2013.

[10] Soumya, R. J., Zulfikhar, A., "Response Time Minimization of Different Load Balancing Algorithms in Cloud Computing Environment", *International Journal of Computer Applications (0975–8887),* Volume 69, No.17, May 2013.

[11] Peter, M., Timothy, G., The NIST Definition of Cloud Computing, NIST Special Publication 800-145, 2011.

[12] William, L., George, K., Vipin, K., "Load balancing across near-homogeneous multi-resource servers", *0-7695-0556-2/00, IEEE*, 2000

[13] Bhathiya Wickremasinghe, Rodrigo N. Calheiros, Rajkumar Buyya, "CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications", 20-23, April 2010, pp. 446-452.

Nguyen Khac Chien was born in Vietnam on October 6, 1980. He received the master degree in Computer Science from the University of Natural Sciences - Vietnam National University, Ho Chi Minh City in 2008. He is currently a lecturer at the University of the People's Police, and is doing a PhD candidate Computer Engineering at the Posts and Telecommunication Institute of Technology. His research interests include Auto-Scaling, VM Migration and Load balancing in cloud computing.

Nguyen Hong Son, received his B.Sc. in Computer Engineering from Ho Chi Minh City University of Technology, his M.Sc. and PhD in Communication Engineering from the Post and Telecommunication Institute of Technology Hanoi. His current research interests include information security, computer engineering and cloud computing.

Ho Dac Loc was born in Vietnam on August 18, 1965**.** He graduated from university in Kharkiv Polytechnic Institute, USSR in 1991. His PhD in Kyiv Polytechnic Institute , Ukraine in 1994 and his Doctor of Science in Moscow Power Engineering Institute (MPEI), Russian Federation.