JACET Journal of Advances in Computer Engineering and Technology, 2(3) 2016

Improving the Palbimm Scheduling Algorithm for Fault Tolerance in Cloud Computing

Minoo Soltanshahi[1]

Received (2016-04-02)
Accepted (2016-11-08)

***Abstract* — Cloud computing is the latest technology that involves distributed computation over the Internet. It meets the needs of users through sharing resources and using virtual technology. The workflow user applications refer to a set of tasks to be processed within the cloud environment. Scheduling algorithms have a lot to do with the efficiency of cloud computing environments through selection of suitable resources and assignment of workflows to them. Given the factors affecting their efficiency, these algorithms try to use resources optimally and increase the efficiency of this environment. The palbimm algorithm provides a scheduling method that meets the majority of the requirements of this environment and its users. In this article, we improved the efficiency of the algorithm by adding fault tolerance capability to it. Since this capability is used in parallel with task scheduling, it has no negative impact on the makespan. This is supported by simulation results in CloudSim environment .**

***Index Terms* — scheduling algorithm, fault tolerance, cloudsim, palbimm algorithm, makespan.**

1- Computer Engineering Department, Kerman Branch, Islamic Azad University, IRAN (minoo.soltanshahi@gmail.com)

I. INTRODUCTION

Today, the use of information and communication technologies has substantially increased in human life. One of the latest technologies in this field is cloud computing with a variety of applications in the majority of organizations, firms, commercial centers, companies, etc. Cloud computing refers to distributed and dynamic resources shared to be deployed by users, even non-specialists, in any place and at any time [1],[2]. When a user presents a job to be processed to the cloud, this job is processed as a workflow in this environment. The workflows are usually sent to the scheduling programs in form of a balanced or unbalanced oriented non-circular graph, in which nodes and edges represent the tasks and relationship between them, respectively. In this stage, task scheduling is done through the selection of suitable resources [3],[4],[5]. Scheduling operation significantly affects system's efficiency, since appropriate scheduling allows optimal use of computing resources, increases the speed, and decreases costs. Therefore, the scheduling algorithms should be designed to respond properly to the requirements of this environment and its users [3],[4],[5],[6],[7]. There have been numerous studies conducted on this area, each proposing an algorithm meeting certain requirements such as cost, time, fault tolerance, load balancing, power consumption, etc. Palbimm is an algorithm that has met several of such requirements including: prioritization of tasks, balanced loading, and runtime reduction [7]. To improve the efficiency of this algorithm, we have added the important capability of fault tolerance to it. We called this algorithm "ftpalbimm" (fault tolerance in palbimm algorithm). Sometimes, workflows with pre-specified conditions are not completed properly. For example, a workflow may not finish within the specified timeline and fail for any reason. Therefore, the scheduling algorithms should be equipped with mechanisms capable of preventing this problem through the estimation of failure factor, or in case of failure, these mechanisms should be capable of completing the workflow properly by compensating for this failure within a minimum time period and with minimum costs. The proposed method tries to prevent failure through estimation of time required for the completion of each job over a certain resource. The remaining parts of this article have been structured as follows: the second section deals with the literature, the third section addresses the proposed method, the fourth section concerns with simulation results, and finally the conclusion is drawn in the fifth section.

II. RELATED WORKS

Scheduling is one the most important problems in cloud computing. It also has a significant effect on the efficiency of cloud computing. The min-min algorithm offers an effective scheduling for time reduction. The Min-Min algorithm first finds the minimum execution time of all tasks. Then it chooses the task with the least execution time among all the tasks. The algorithm proceeds by assigning the task to the resource that produces the minimum completion time. The same procedure is repeated by Min-Min until all tasks are scheduled. Figure1 illustrates Min-Min algorithm [8],[9]. The improved min-min algorithm was developed by Rajwinder Kaur et al. in 2013, executed through the following stages. At first, min-min algorithm is executed as task T is assigned to resource R with the shortest execution time. Then, resources are arranged based on execution time, calculating the makespan and selecting a resource capable of responding to makespan. At the next stage, executed tasks will find the resources producing makespan. Then, it will find the minimum completion time of those tasks and the resources capable of responding. It will apply the settings on every single task. If the next completion time of task is shorter than makespan and the new completion time of machine is shorter than makespan, it will schedule the task on the responding resource. Finally, it will update the ready time of both resources [9]. Later, the lbimm was developed to add load balancing to the imm algorithm, where the tasks are first mapped on the resources, and then the smallest tasks are selected from resources with heavy loading, calculating their completion times on the rest of resources. It then compares the shortest completion time against the output time of min-min. If it is shorter than completion time of min-min, the task will be mapped on that particular resource, updating the ready time of other resources. This process is repeated until other resources cannot produce a completion time shorter than that of the task on the heavy resource [10],[7]. In 2013, Haun kai chen et al. developed palbimm algorithm by adding task prioritization capability to lbimm algorithm. This algorithm attempts to fulfill the requirements of load balancing, prioritizing tasks and minimization of execution time. Figure2 illustrates palbimm algorithm [7]. In 2015, Rajeev Mangla et al. added recovery capability to palbimm. When a resource fails, this algorithm tries to continue the execution of tasks through the selection of feasible resources. The pseudo code for RPA-LBIMM scheduling algorithm is given in Figure 1 [11].

Step 1: **For** tasks in set S: Ti
Step 2: **For** all respources: Rj
Step 3: $Ct_{ij}=Et_{ij}+Rt_j$: End **For**: End **For**;
Step 4: **Do** while tasks set is not empty
Step 5: Apply PA-LBIMM
Step 6: **If** resource failure occurs
Step 7: **For** tasks in set S_f: T_k
Step 8: **For** all respources: Rf_i
Step 9: $FCt_{ij}=Ct_{ij}+rtf_j$;
Step 10: End **For**: End **For**;
Step 11: Find the minimum completion time of Tk produced by resource R_k
Step 14: **If** minimum completion time (R_k) < completion time of (Rf_i)
Step 15: Reassign Task T_k to Resource R_k
Step 16: Update ready time of both Rf_j and R_k
Step 17: End **If**;
Step 18: End **If**;
Step 19: End **Do**;

Fig 1. The Rpalbimm scheduling algorithm [11]

III. PROPOSED METHOD

In cloud computing systems, a service may for any reason after scheduling fail to properly complete a particular task within the deadline assigned by the user. Specifically within workflows, it may last for several months and ultimately fail. In order to improve the efficiency of palbimm, the ftlbimm algorithm was developed by adding the fault tolerance capability to palbimm. In ftlbimm algorithm, tasks are first scheduled over the resources by palbimm; then, the fault tolerance capability works together with the scheduling algorithm over the running resources. Due to the parallel execution of this operation, the execution time does not change much. To minimize and/or prevent failure, we first estimate completion time of each running task, and then compare it with the allowed completion time. If the completion time exceeds the deadline, the min-min algorithm will run over all existing services; then, task completion time over the output service of min-min algorithm is compared to the time required for the completion of the task over the current service; otherwise, the task is mapped on it and the stored results from the execution over the previous service are transmitted to the new service. In this way, the task execution continues over the new service. Otherwise, the task continues on the current service. the min-min algorithm helps to find the best resource which can offer the shortest run time for execute tasks on the failed resource in the less time. Therefore we used this algorithm in the proposed method. Although, the 3th step condition is not correct means that the run time which is chosen by min-min algorithm takes time more than the current resource. So the best resource to continue execution is the current resource. In the case of established condition which is mentioned, the rest of execution should be transmitted to the new resource. Therefore the use of free cloud storage resources can be suggest to the users. This suggestion prevents financial loss. Actually it can be one of the parameters in service level agreement to more satisfaction of users. In our new algorithm, we considered this method along with scheduling technique differently from the palbimm algorithm. To improve the efficiency of palbimm, the execution of this algorithm in parallel with the execution scheduling operation is recommended. The fault tolerance parallel code is presented in Figure 2.

```
1- estimate need time for task
2- if ( task(time_need) > task(time_deadline)
        1. rm =run min-min algorithm
           in s(n)
3- if (rm (time_execution) < task(time_need))
        1. store current state in
           storage cloud and return
           resource
        2. reassign task to resource
           and get current state from
           cloud storage
4- else
        1. continue run on current
           resource
5- end if
6- end if
```

Fig 2. The fault tolerance parallel code

IV. RESULTS OF SIMULATION

CloudSim is an extensible simulation toolkit that enables modeling and simulation of Cloud computing systems and application provisioning environments. The CloudSim toolkit supports both system and behavior modeling of cloud system components such as data centers, virtual machines (VMs) and resource provisioning policies [12]. It implements application provisioning techniques that can be extended with ease and limited effort. In addition, CloudSim provides custom interfaces for the implementation of resource provision policies and techniques for allocation of virtual machines under inter-networked cloud computing scenarios. CloudSim is a free open-source library for simulation of cloud computing scenarios. It has been designed in Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne [12]. We simulated palbimm and ftpalbimm with CloudSim. Results showed a slight time difference between these two algorithms at run time, which is due to parallel execution of the fault tolerance code. For better evaluation, three different scenarios, almost similar to the simulation details in [7], have been used in the execution of the algorithms. Tables 1-3 and Tables 4-6 represent the specifications of tasks and specifications of resources used in the scenarios.

TABLE I:
Tasks specification in senario a

Task-id	Size(MIPS)	Type
Task 1	160	VIP
Task 2	170	VIP
Task 3	320	VIP
Task 4	40	VIP
Task 5	360	VIP
Task 6	80	VIP
Task 7	800	VIP
Task 8	150	VIP
Task 9	120	Ordinary
Task10	630	Ordinary

TABLE II
Tasks specification in senario b

Task-id	Size(MIPS)	Type
Task 1	950	VIP
Task 2	240	VIP
Task 3	610	Ordinary
Task 4	490	Ordinary
Task 5	890	Ordinary
Task 6	460	Ordinary
Task 7	460	Ordinary
Task 8	30	Ordinary
Task 9	820	Ordinary
Task 10	450	Ordinary

TABLE III
Tasks specification in senario c

Task-id	Size(MIPS)	Type
Task 1	670	VIP
Task 2	700	VIP
Task 3	410	Ordinary
Task 4	230	Ordinary
Task 5	800	Ordinary
Task 6	650	VIP
Task 7	420	VIP
Task 8	300	Ordinary
Task 9	560	Ordinary
Task 10	380	Ordinary

TABLE IV
Resources specification in senario a

Resource-id	Cpu (MIPS)	Ram (MB)
Resource 1	750	256
Resource 2	800	256
Resource 3	550	256
Resource 4	900	256
Resource 5	600	256

TABLE V
Resources specification in senario b

Resource-id	Cpu (MIPS)	Ram (MB)
Resource 1	10	256
Resource 2	800	256
Resource 3	90	256
Resource 4	50	256
Resource 5	30	256

TABLE VI
Resources specification in senario c

Resource-id	Cpu (MIPS)	Ram (MB)
Resource 1	15	256
Resource 2	70	256
Resource 3	50	256
Resource 4	60	256
Resource 5	80	256

Diagram 1 shows the execution time of the three scenarios in above algorithms.

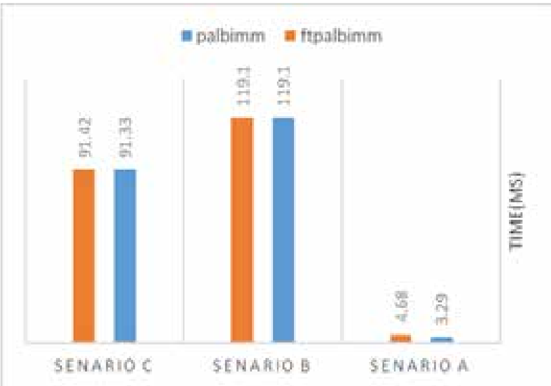

Diagram 1. Makespan results

Given the results from simulation of algorithms, we discovered that ftpalbimm responded to a larger number of requirements of the environment and user with a very little time difference as compared with palbimm.

V. CONCLUSION

Given the dramatic growth of computing and communication technologies in human life, cloud computing, as one the latest and most advanced type of such technologies, has become very important and efforts have been made by many researchers to increase efficiency and to optimize resource usage in this technology. One of the most significant factors that affect the efficiency of cloud computing is scheduling algorithms, as they have direct effect on a system's efficiency through careful selection of resources. Therefore, we added the fault tolerance capability to palbimm, which is executed in parallel with scheduling process, to improve its efficiency. This new algorithm was called "ftpalbimm." Then, both algorithms were assessed in CloudSim using three different scenarios. Results showed a little difference in execution time of the algorithms. In addition, the proposed method reduced the execution time in large workflows, where the occurrence of failure is more probable. Therefore, adding the proposed method to palbimm algorithm increases its efficiency in cloud computing environment.

REFERENCE

[1] Y. Luo and S. Zhou ," Power Consumption Optimization Strategy of Cloud Workflow Scheduling Based on SLA" wseas transactions on systems, E-ISSN: 2224-2678 , Volume 13, 2014.

[2] S. Qaisar and K. Khawaja, "cloud computing: network/security threats and countermeasures", interdisciplinary journal of contemporary research in business - january 2012 vol 3, no 9.

[3] B. Rashidi and M. Sharifi and T. Jafari , "A Survey on Interoperability in the Cloud Computing Environments" Published Online July 2013 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijmecs.2013.06.03.

[4] P. Sareen, " Cloud Computing: Types, Architecture, Applications, Concerns, Virtualization and Role of IT Governance in Cloud" , International Journal of Advanced Research in Computer Science and Software Engineering ,Volume 3, Issue 3, March 2013.

[5] M. soltanshahi and A. niknafs, " A Study on Factors Contributing to Efficiency of Scheduling Algorithms in a Cloud Computing Environment; Overview of Several Algorithms", Ciência e Natura, v. 37 Part 2, p. 427−433, december 2015, DOI: http://dx.doi.org/105902/2179460X.

[6] D. Tracy, Braun, "A comparison of eleven static heuristics for mapping a class ofindependent tasks onto heterogeneous distributed computing systems", Journal of Parallel and Distributed computing , Volume 61, Issue 6, Pages 810 – 837, 2001.

[7] H. Chen, F. Wang, N. Helian and G. Akanmu, "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing," Parallel Computing Technologies (PARCOMPTECH), 2013 National Conference on, Bangalore, 2013, pp. 1-8. doi:10.1109/ParCompTech.2013.6621389,publisher:IEEE URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6621389&isnumber=6621385

[8] S. Meraji and M. Salehnamadi , "A Batch Mode Scheduling Algorithm for Grid Computing" , J. Basic. Appl. Sci. Res., 3(4)173-181, 2013 © 2013, textroad Publication , ISSN 2090-4304 ,Journal of Basic and Applied Scientific Research ,www.textroad.com.

[9] T. Kokilavani, D.I. George Amalarethinam, "Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing", International Journal of Computer Applications (0975 – 8887) , Volume 20– No.2, April 2011.

[10] P. Warstein and H. Situ and Z. Huang, "Load balancing in a cluster computer", In proceeding of the seventh International Conference on Parallel and Distributed Computing, Applications and Technologies IEEE 2010.

[11] Er. Rajeev Mangla , Er. Harpreet Singh , "recovery and user priority based load balancing in cloud computing" , international journal of engineering sciences & research technology , ISSN: 2277-9655 Scientific Journal Impact Factor: 3.449(ISRA), Impact Factor: 2.114 , Mangla, 4(2): February, 2015.

[12] R.N.Calheiros, R.Ranjan, A.Beloglazov,C.A.F.De

Rose and R. Buyya.”Cloudsim:a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms.”,Software: Practice and Experience, 41(1):23–50,2011.