Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 46 (2015) 913 – 920

Procedia Computer Science

International Conference on Information and Communication Technologies (ICICT 2014)

Credit Based Scheduling Algorithm in Cloud Computing Environment

Antony Thomas[a,*] , Krishnalal G[a], Jagathy Raj V P[b]

[a]*Amal Jyothi College of Engineering,Mahatma Gandhi University,Kottayam,India*
[b]*School of Management Studies,Cochin University of Science And Technology,Cochin,India*

Abstract

Cloud computing in today's world has become synonymous with good service policies. In order to achieve good services from a cloud, the need for a number of resources arose. But cloud providers are limited by the amount of resources they have, and are thus compelled to strive to maximum utilization. Min-Min algorithm is used to reduce the make span of tasks by considering the task length. Keeping this in mind, cloud providers should achie*ve user satisfaction. Thus research favors scheduling algorithms that consider both user satisfaction and resources availability. In this paper an improved scheduling algorithm is introduced after analyzing the traditional algorithms which are based on user priority and task length. High prioritized tasks are not given any special importance when they arrive. The proposed approach considers all of these factors. The experimental results show a considerable improvement in the utilization of resources.

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)

*Keywords:*Cloud computing;Task priority;Task length

1. Introduction

The cloud computing is a blend of technologies where a large number of systems are connected in private or public networks. This technology provide dynamically scalable infrastructure for data, file storage, and application.

* Antony Thomas Tel.: +91 8281489771.
*E-mail address:*antonythomas.info@gmail.com

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)
doi:10.1016/j.procs.2015.02.162

The cost of content storage, application hosting, computation and delivery can be reduced significantly by this technology. The real strength of the cloud lies in the ability to distribute the workload over multiple nodes employing multiple technologies to create a high performance, highly scalable and available platform at a highly affordable price[1]. Scheduling means the set of policies for controlling the order of work to be performed by a computing system.

Scheduling is a major task in a cloud computing environment. In cloud computing environment datacenters take care of this task. A simple cloud architecture is shown in Fig 1. The datacenters receive tasks from the datacenter brokers which arrived from different users. In some cases these tasks may be associated with priorities. If so, a broker should consider these priorities and it is responsible for assigning the task. The algorithms like Min-Min will not consider user priority. A better scheduling algorithm is needed to achieve full utilization of resources.

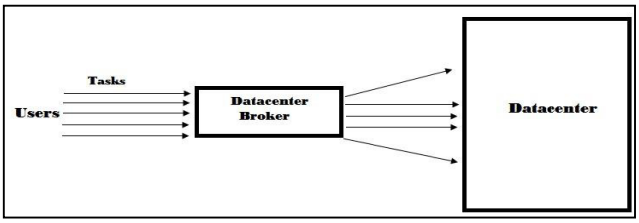

Fig 1 . A Simple cloud architecture

Min Min algorithm consider the task length as the granule for executing the tasks. An optimum condition occurs when task with minimum length is executed first. In order to achieve this optimum condition datacenter broker should consider this factor. But in the other side while considering priority of the task, the execution order of tasks may change which may be different from the order when the task length is considered. The main idea behind the proposed approach is considering both the task size and user priority. Based on these two criteria order of execution of the tasks will be decided by the datacenter broker.

Three questions are to be considered when applying parallel processing in executing tasks: 1) how to allocate resources to tasks; 2) in what order is the task executed in the cloud; and 3) how to schedule overheads when VMs prepare, terminate or switch tasks. Task scheduling and resource allocation can solve these three problems[2,18].

2. Related Work

A lot of scheduling algorithms are available today for cloud computing. But their performance is questionable. Different parameters are to be considered for scheduling. Two major parameters are task size and priority considered in the proposed approach. But these are not the only ones. There are some other parameters which influences the scheduling of tasks and utilization of resources. Lot of studies are taking place in this area. Some of the studies are Cost based scheduling[4], Energy efficient optimization methods[5], Activity based costing[6,20], Reliability Factor Based[7], Context aware scheduling[8], Dynamic slot based scheduling[9,19]. Among these parameters task length and priority are quite important.

2.1. Task Size (Min Min Algorithm)

The effective utilization of resources can be increased by using a load balancing algorithm. This is achieved by making use of resources of an unused(idle) processor while release the resources of processors having heavy load. The load balancing algorithm distribute the load among all the resources which are available. This type of algorithm also minimize the makespan with the valuable use of resources[3,16,17]. Min Min algorithm starts with a set of tasks. There is a task set T. Along with the task set there are some resources too. After implementing the algorithm, task set will be allocated or mapped to resource set. The algorithm chooses the task having minimum size.

The resource will be allocated to the task having minimum completion time. On completion of the assignment, task will be removed from the task set. The algorithm continues until task set becomes empty. Let the task set be

$T_1,T_2,T_3,$...etc and resource set be $R_1,R_2,R_3,$..etc. Expected completion time of task i on resource j is denoted as Ct_{ij}. It is calculated by using the equation 1.

$$Ct_{ij} = Et_{ij} + rt_j \quad (1)$$

rt_j represents the ready time of resource R_j. Et_{ij} stands for execution time of task i. Pseudo Code of Min Min algorithm is represented below.

```
1.    For all submitted tasks in the set; Ti
2.      For all resources; Rj
3.        Ctij=Etij +rtj; End For;End For;
4.    Do while tasks set is not empty
5.      Find task Tk that cost minimum execution time
6.      Assign Tk to the resource Rj while gives minimum expected
          complete time
7.      Remove Tk from the task set
8.      Update ready time rtj for select Rj
9.      Update Ctij for all Ti
10.   End Do
```

Fig. 2. The Min Min Scheduling Algorithm

The main issue of this algorithm is its consideration is only to task length and not to user priority. There may be a situation where high priority tasks exist. Priority has its own importance so this also should be considered for scheduling.

2.2. Priority Based

This technique works based on the user's priority of tasks. But in such a case, other problems may arise. In optimal conditions, the cloud system gets good results when shortest job is executed first. But this won't always be the case in this technique. Sometimes task with larger length is associated with a highest priority. This is the big issue in this algorithm.

2.3. Cost Based Resource Scheduling

This paper proposes a scheduling scheme based on the availability of resources and cost of each resource. Each resource is associated with a cost. The resource with lowest cost will be assigned to the task. The practical difficulty of this concept is that when a user needs a resource having a highest price[12]. In this situation user will not get the resource. So scheduling based on the resource cost has limitations.

2.4. Resource Scheduling Based on Energy Efficient Methods

The paper describes a method to schedule the resources such that energy consumption is minimum. The cloud computing architecture used today has number of hardware components to accomplish the user needs. Cloud providers are trying to reduce the energy consumption[13]. There is no direct relationship between energy consumption and task scheduling. The paper doesn't tell how to do actual scheduling of tasks.

2.5. Scheduling Based on Reliability Factor

Proper scheduling eliminates overloaded conditions during the utilization of resources. This paper tells how to tackle overloaded conditions by scheduling the jobs. It also explains how to handle a failure[15]. This factor(Failure) is not relevant because failure may happen to any system. The parameters that are more important need more thought.

2.6. Scheduling Based on Activity Based Costing

The paper describes a task scheduling scheme based on the cost of the resources. In order to execute a task, different resources have to be used. Cloud service provider charges a cost based on the usage of resources. The paper doesn't tell about the parameters that are essential for scheduling[14]. Proper scheduling reduces the cost on each resources.

2.7. Context Aware Scheduling

The paper describes how to utilize the resources fully in order to avoid the resource wastage. This situation can be avoided by utilizing the resources effectively[10]. The method used to solve the situation is to execute the client request in the client side itself without moving into the server.

2.8. Dynamic Slot Based Scheduling

Hadoop is a concept that deals with big data. MapReduce function is used for handling this big data. In this paper each task gets a time slot[11]. During that period task gets executed. This approach checks the utilisation of resources. This is a complex approach because it needs migration of tasks. Task migration is difficult when we deal with big data.

There are plenty of algorithms present for the scheduling of the tasks and utilisation of resources. The big question comes when we think about how to do scheduling. From the literature survey it is evident that various parameters can be considered for the scheduling of tasks such as resource cost, priority, etc. From the study of various approaches for scheduling, the task length and priority are considered as the parameters for detailed investigation in the proposed approach.

3. Proposed Approach

The proposed approach considers two parameters : (1) Task Length and (2)User Priority. The algorithm is based on credit system. Each task is assigned a credit based on their task length and priority. In the actual scheduling of the task, these credits will be considered.

3.1. Task length credit

The cloud system execute tasks having different length. If the tasks are arranged based on the increasing order of length, tasks having shorter length will reside at the beginning of the array and the task having highest length will be present at the last.

For the scheduling purpose, the algorithm should take tasks from both front and back, giving it a bit more stability. The credit system based on task length will work as follows: The first step is involved in finding the length of each task($Tlen_i$). The next step is calculating the average of tasks length. Lets the value is len_{avg}. The third step begins with the calculation of difference in length with respect to len_{avg}. Let the task set be $T_1,T_2,T_3,$...etc. Here equation 2 is used for finding the difference in length with the average length. This data is useful when tasks are arranged in an array in an increasing order of task length. The proposed algorithm neither takes task with larger length nor task with lower length. It takes each tasks from the middle.

$$\left| TLD_i = len_{avg} - Tlen_i \right| \quad (2)$$

where TLD_i is the task length difference of task i. It is computed by taking the absolute difference of task length for the i^{th} task and average value. After finding the difference in task lengths of each task, credits are assigned to each task. In this algorithm there are 5 credits and these credits are given to each tasks for different conditions. Before these steps, 4 different values are found from the length array. These 4 values forms the condition for assigning the credits. We can't simply choose 4 values. These values should be in a range of task length. The computations are given below.

$$value\_1 = high\_len/5 \quad (3)$$

$$value\_2 = high\_len/4 \quad (4)$$

$$value\_3 = value\_2 + value\_1 \quad (5)$$

$$value\_4 = value\_3 + value\_2 \quad (6)$$

where high_len is the highest value of task length. This can be found by Pseudocode is mentioned below.

For all submitted tasks in the set ;Ti
  | $\mathbf{TLD_i = len_{avg} - Tlen_i}$ |
  If $TLD_i \leq value\_1$
    then credit =5
  else if $value\_1 < TLD_i \leq value\_2$
    then credit =4
  else if $value\_2 < TLD_i \leq value\_3$
    then credit =3
  else if $value\_3 < TLD_i \leq value\_4$
    then credit =2
  else $value\_4 > TLD_i$
    then credit =1
End **For**

Fig. 3. The credit system based on task length

This algorithm adds credits based on the task length. After this step each task will associated with a credit ($Credit\_Length_i$)

3.2. Task priority credit

Task priority is also important for scheduling tasks. Each task may have different priorities, which are represented as values assigned to each task and the value can be the same for more than one task. The scheduling algorithm based on task priority has the problem of treating tasks with similar priority. In the proposed approach this does not arise as a problem because even though we are giving credits to each tasks based on their priority, the final scheduling will be based on total credit which is based on task length and its priority.

Here in this algorithm different tasks are assigned different credits. Credit value is generated based on the priority which is assigned to the each task. In the proposed approach priority average is not important. Suppose there are 10 tasks, then there will be 10 different credits . There will be 20 credits when dealing with 20 tasks. The fact is that these credits are not set by default. The credit value will change based on the priority that is assigned by the user. The Pseudocode is shown below.

```
For all submitted tasks in the set ;Ti
        Find out task with highest priority(Priority Number)
        Choose division_part
 For each task with priority Tpri
         find Pri_frac(i)=Tpri /division_factor
set credit as Pri_frac
End For
```

Fig. 4.The credit system based on task Priority

The primary step in the above algorithm is fnding the highest priority number. Second step in the algorithm is choosing the division factor for fnding Pri_frac for each task. For example if highest value of priority is a two digit number then choose division_part as 100. If it is 3 digit then division_part is 1000. Third step in the algorithm is calculating the Pri_frac for each task. This can be calculated by dividing priority value of each task with division factor of corresponding task. Finally this value(Pri_frac) will be assigned to each task as priority credit.

$$Total\_Credit_i = Credit\_Length_i * Credit\_Priority_i \quad (7)$$

The two credits are calculated separately. The final step in the algorithm is to find out the total credit based on task length and task priority. The Total credit is calculated by using the equation shown above. In the above equation $Credit\_Length_i$ is the credit based on task length. $Credit\_Priority_i$ is the credit based on priority. Finally task having highest credit will be scheduled first.

4. Results and Discussions

The algorithm mentioned above is simulated in cloudsim 3.0.3 simulator. After comparing the makespan of cloudlets in the two algorithms where one is based on task length and the other is based on user priority, we can observe that proposed algorithm shows better results. The simulation is done under the following conditions. Following three tables describes the simulation conditions.

Table 1. Basic Configuration

Number of Datacenters	2
Number of cloudlets	10
Number of Brokers	1
Number of hosts under each Datacenter	2

Each data center consists of several hosts. Each host has its own configuration. Here same configuration is applied for each hosts. Host configuration is mentioned below.

Table 2. Host Configuration

RAM(MB)	16384
MIPS (Lines of Codes)	1000
Storage(MB)	1000000
Bandwidth(MB/sec)	10000
Number of Virtual Machines	2

Host in the datacenter consists of several virtual machines. Each virtual machine has its own configuration. Here same configuration is applied for each VMs. Virtual machine configuration is mentioned below.

Table 3. Virtual Machine Configuration

Number cores	2
MIPS (Lines of Codes)	1000
Size(MB)	10000
RAM(MB)	512
Bandwidth(MB/sec)	1000

Fig 5 shows the comparison of makespan among algorithms.

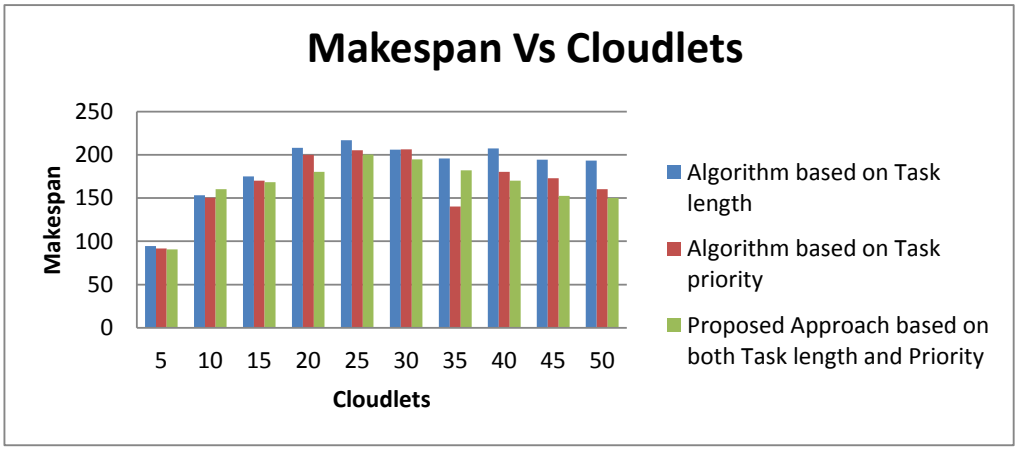

Fig. 5. Makespan Vs Cloudlets

Cloudlet number is represented in the X-axis. In the Y-axis makespan of cloudlets is represented. The result shows that makespan is increasing with the increase in cloudlet number. But results also shows that there is a slight decrease in the makespan after reaching its highest value. The result shows that makespan of proposed algorithm is better when compared with the other two algorithms. Fig 6 describes the relation with length and priorities of cloudlets. Here number of cloudlet chosen is 10.

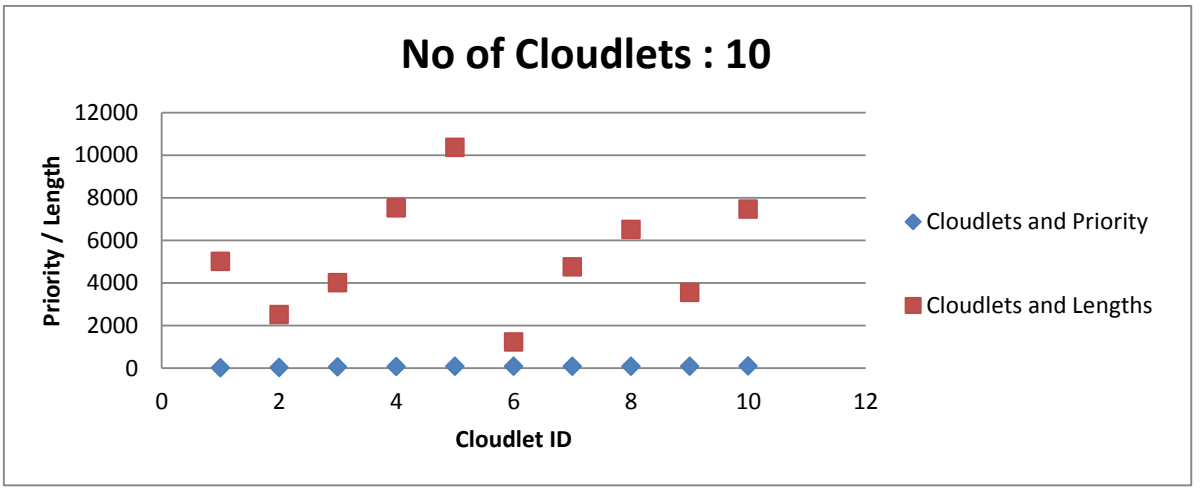

Fig. 6. Cloudlet length vs Priority

In the above graph, x- axis represent cloudlet id and y-axis represent cloudlet priority or cloudlet length.

5. Conclusion

In this paper, three cloud scheduling scenarios are presented. First scenario is based on the length of tasks. The second scenario is based on task priority. The third and the proposed approach works on both cloudlet priority and cloudlet length. From the simulation results it is concluded that, the proposed algorithm works efficiently than the other two methods. Makespan of the task is lesser when compared with the other two algorithms. It is also observed that makespan of task is decreasing after a certain value in the number of tasks. In future, the proposed scheme can be enhanced so as to consider other parameter like deadline. Task dead line has its own importance in scheduling certain tasks for the realtime systems.

References

1. M. Armbrust et aI. Above the Clouds: A Berkeley View of Cloud Computing, technical report. Univ. of California, Berkeley; Feb 2009.
2. Chandrashekhar S. Pawar, Rajnikant B. Wagh. Priority Based Dynamic Resource Allocation in Cloud Computing with Modified Waiting Queue; 2013. International Conference on Intelligent Systems and Signal Processing (ISSP).
3. T Kokilavani, GA DI. Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing. International Journal of Computer Applications. Number 2 - Article 7; 2011.
4. Zhi Yang, Changqin Yin, Yan Liu. A Cost-based Resource Scheduling Paradigm in Cloud Computing. 2011-12th International Conference on Parallel and Distributed Computing. Applications and Technologies.
5. Liang Luo, Wenjun Wu,Dichen Di, Fei Zhang, Yizhou Yan, Yaokuan Mao. A Resource Scheduling Algorithm of Cloud Computing based on Energy Efficient Optimization Methods.
6. Qi cao,Zhi-bo Wei,Wen-mao Gong. An Optimized Algorithm for Task Scheduling Based On Activity Based Costing in Cloud Computing.
7. Bo Yang, Xiaofei Xu, Feng Tan, Dong Ho Park. An Utility-Based Job Scheduling Algorithm for Cloud Computing Considering Reliability Factor; 2011 International Conference on Cloud and Service Computing.
8. Marcos D. Assuncao, Marco A. S. Netto, Fernando Koch, Silvia Bianchi. Context-aware Job Scheduling for Cloud Computing Environments. 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing.
9. Hsin-Yu Shih, Jhih-Jia Huang, Jenq-Shiou Leu. Dynamic Slot-based Task Scheduling Based on Node Workload in a MapReduce Computation Model.
10. H. P. Borges, J. N de Souza, B. Schulze and A. R. Mury. Automatic generation of platforms in cloud computing in Proceedings of the IEEE Network Operations and Management Symposium (NOMS 12) 2012.pp. 1311–1318.
11. J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters, in USENIX Symposium on Operating Systems Design and Implementation, San Francisco: CA; Dec 2004. pp. 137–150.
12. Lizhe Wang, Gregor von Laszewski, Andrew Younge, Xi He, Marcel Kunze, Jie Tao and Cheng Fu. Cloud Computing: a Perspective Study. New Generation Computing. Volume 28, Number 2; 137-146.
13. Yong Dong .Power Measurements and Analyses of Massive Object Storage System. Computer and Information Technology (CIT); 2010 IEEE 10th International Conference pp. 1317 – 1322 ; 2010.
14. Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal. Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. GRIDS Lab Technical Report; August 2008.
15. J. Wang. Soft Real-Time Switched Ethernet: Best-Effort Packet Scheduling Algorithm, Implementation, and Feasibility Analysis. Master's thesis. Virginia Tech; 2002.
16. Braun T.D, Siegel H.J, Beck N, Boloni L.L, Maheswaran M, Reuther A.I, Robertson J.P.et al. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed Computing. Vol. 61. No. 6, pp.810–837; 2001.
17. Dantong Yu and Thomas G. Robertazzi . Divisible Load Scheduling for Grid Computing. PDCS; 2003, 15th Int'l Conf. Parallel and Distributed Computing and Systems. IASTED. pp. 1 – 9; 2003.
18. S. K. Garg, R. Buyya, and H. J. Siegel. Time and cost trade off management for scheduling parallel applications on utility grids. Future Generation Computer System. 26(8):1344–1355; 2010.
19. K. Kambatla, A. Pathak, and H. Pucha. Towards optimizing Hadoop provisioning in the cloud in USENIX Workshop on Hot Topics in Cloud Computing (HotCloud09); 2009.
20. J Blythe, S Jain, E Deelman, Y Gil, K Vahi. Task scheduling strategies for workflow-based applications in grids. Cluster Computing and the Grid; 2005.