International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 4 60 - 64

An Improved Min-Min Task Scheduling Algorithm for Load Balancing in Cloud Computing

Prerit Chawda
M. Tech. (CSE), Dept. of CSE
SRM University, NCR Campus
Ghaziabad, Utter Pradesh, India
prerit.chawda@gmail.com

Partha Sarathi Chakraborty
Assistant Professor, Dept. of CSE
SRM University, NCR Campus
Ghaziabad, Utter Pradesh, India
parthasarathi@live.com

***Abstract*—** Cloud computing is a model that uses computing resources (CPU, memory, processor etc.) which are delivered as a service over a network internet. It is a new field for large scale distributed computing. In cloud computing, one of the major task is to execute large number of tasks with given available resources to achieve high performance, minimal total time for completion, Minimum response time, effective utilization of resources etc. Keeping all these perspective, we need to design, develop and propose a scheduling algorithm to develop an allocation map to represent the set of tasks on appropriate resources. Though traditional Min-Min algorithm is a simple, yet efficient algorithm that provides a better scheduling to minimize the total time for completion of tasks compared to the other algorithms. However the drawback of this algorithm is the load imbalanced, which is one of the major issue for cloud provider. In this paper, we introduce an improved load balancing algorithm keeping Min-Min algorithm as base in order to reduce the makespan and increase the resource utilization (ILBMM). The proposed method has two different phases, in the first phase traditional min-min algorithm is executed and in second phase it selects the task with maximum completion time and assigns it to appropriate resource to produce better makespan and utilize resource effectively

Keywords*-Cloud Computing; Load Balance; Min-Min Algorithm; Task Scheduling*

*****

I. INTRODUCTION

Currently Cloud computing has evolved as great potential technology that is known as a provider of dynamic services using very large scalable and virtualized resources over the Internet [5]. Cloud is subject to User Requirement, Load Balance and other constraints that have direct effect on user consumption of resources controlled by cloud provider [10].

With increasing number of users on cloud, assuring the optimum use of cloud resources are difficult task for cloud service providers and there are many issues associated with the task scheduling algorithm and so the load balancing can be used for cloud for balancing the load.

Load balancing [1] is a method that distributes the workload among diverse nodes in the given environment such that it ensures no node in the system is over loaded or sits idle for any instant of time. An efficient load balancing algorithm will make sure that every node in the system does more or less same volume of work.

The responsibility of load balancing algorithm is that to map the jobs which are set forth to the cloud domain to the unoccupied resources so that the overall available response time is improved as well as it provides efficient resource utilization. Balancing the load became one of the crucial concerns in cloud computing since we cannot predict the number of requests that are issued at each second in cloud environment. The unpredictability is due to the ever changing behavior of the cloud. The main focus of load balancing in the cloud domain is in allocating the load dynamically among the nodes in order to satisfy the user requirements and to provide maximum resource utilization by assorting the overall available load to distinct node

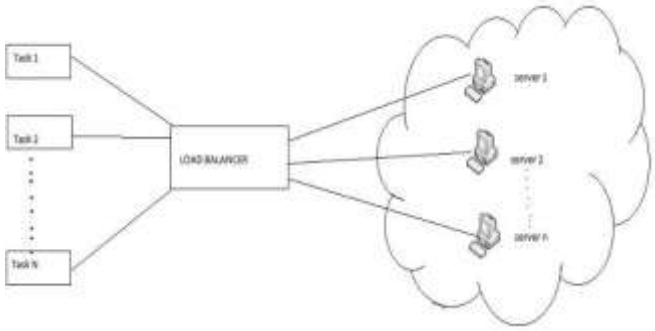

Fig 1 – Diagram for load balancing

A. Task scheduling in cloud computing

In cloud, many tasks have to be executed by the available resources to achieve distinct intensions. There is need for a scheduling algorithm that is used by task scheduler to outperform appropriate allocation map of task on resources [11].

There are different types of scheduling are based on different criteria's, such as Static and Dynamic scheduling, Centralize and Distributed scheduling, Preemptive and Non-Preemptive scheduling, Cooperative scheduling, Immediate and Batch mode scheduling [12][14][15].

Cloud task scheduling is an NP-complete problem in general [7]. In the typical cloud scenario, cloud users submit their tasks to cloud scheduler. The scheduler on cloud will first query the Cloud Information Service to gather the information regarding the availability and properties of resources, and then they schedule the tasks on the resources which satisfy the tasks requirements. Once task is executed, the result is sent to the users.

Scheduling of tasks in such cloud environment is the challenging task because of its high heterogeneity in operating systems, architecture, cloud providers and resource consumers [3].

The main purpose of a cloud task scheduling algorithm is to shorten overall the completion time of all tasks submitted by users, enhance the utilization of cloud resources and satisfy requirements of different users [2].In this paper a new task scheduling algorithm is proposed to

decrease job's completion time and improve the load balance in the cloud

II. RELATED WORK

A load balance scheduling algorithm aims to increase the utilization of resources with light load or idle resources thereby freeing the resources with heavy load. Main goal of this algorithm is to distribute load among all available resources and also to minimize the makespan which effectively utilizes the resources.

In the literature, large numbers of task scheduling algorithm were proposed in the past. Braun et al [7] have studied the relative performance of eleven heuristic algorithms for task scheduling such as Opportunistic Load Balancing (OLB), Minimum Execution Time (MET), Minimum Completion Time (MCT), Min-Min, Max-Min, Duplex, Genetic Algorithm (GA), etc. They have also provided a simulation basis for researchers to test the algorithms. Their results show that the simple Min-Min algorithm produces a better schedule that minimizes the makespan than the other algorithms and performs next to GA which the rate of improvement is also very small in most of the scenarios. In this section different task scheduling techniques are reviewed briefly

OLB allocates task to the resource that are to be ready next, without any execution time of task consideration on that resource [8]. If more than one resource becomes ready than, the resources are arbitrarily chosen. The intention behind OLB is to keep all machines as busy as possible one advantage of OLB is its simplicity, but because OLB does not consider task execution times, so it will generate poor makespan.

MET assigns each task to the resource that performs it in the least amount of execution time, no matter whether this resource is available or not at thattime[10]. This heuristic can cause a severe load imbalance across the resources. However, this is one of the heuristics that is implemented in SmartNet. The aim behind MET is to give each task to its best machine

MCT assigns each task to the resource which obtains earliest completion time for that task, which causes some of the tasks to be assigned to the resources whose execution time is not minimum for them [9]. The intension of MCT is to combine the benefits of OLB and MET while avoiding the drawbacks of these two algorithms.

Min-Min algorithm starts with the set of all the unmapped tasks. Machine with the least or minimum completion time for all jobs will be selected. Next the job whose overall completion time is minimum will be selected and will be mapped to the respective resource. Ready time of the resource will then be updated. Repeat this process until all the unmapped tasks are assigned. This algorithm considers all jobs at any time compared to the MCT, which shows that it produces a better makespan.

Max-Min is similar to Min-Min algorithm. The machine that has the minimum completion time for all jobs is selected. Then the job with the overall maximum completion time is selected and will be mapped to the resource. Next the ready time of the resource is updated. Repeat this process until all the unmapped tasks are alocated. The basic idea of this algorithm is to reduce the overall waiting time of the large jobs

SaeedParsa et al.proposed a new task scheduling algorithm called RASA [4]. It takes advantage of both Max-min and Min-min algorithm. RASA uses the Min-min strategy to execute small tasks before large ones and applies the Max-min strategy to avoid delays in the execution of the large tasks and to support concurrency in the execution of large and small tasks

Each of Max-min and Min-min algorithms have running time complexity of O(mn2), where m is the number of resources currently in the system and n is the number of submitted tasks which should be scheduled [4], [6]

III. TRADITIONAL MIN-MIN SCHEDULING ALGORITHM

The Min-Min algorithm is simple and still basis of present cloud scheduling algorithm [13]. It starts with a set U of all unmapped tasks. Then the resource R which has the minimum completion time for all tasks is found. Next, the task T with the minimum size is selected and assigned to the corresponding resource R (hence the name Min-Min). Last, the task T is removed from set U and the same procedure is repeated by Min-Min until all tasks are assigned (i.e., set U is empty).

The algorithm works better when the situation is like where the small tasks are greater in number of than the large tasks. The algorithm has a disadvantage that it leads to starvation. .Min-Min is a simple and fast algorithm capable of providing improved performance. Min-Min schedules the ideal tasks at first which results in best schedules and improve the overall make span. Assigning small task first is its drawback. Thus, smaller tasks will get executed first, while the larger tasks keeps on in the waiting stage ,which will finally results in poor machine use.

The pseudo code of Min-Min algorithm is represented in Fig 2 assuming we have a set of n tasks (T1, T2, T3 … Tn) need to be scheduled onto m available resources (R1, R2, R3 … Rm). We denotes the Expected Completion Time for task i on resources j as Ct_{ij} that is calculated as Ctij = Et_{ij} + rt_j, Where rt_j represents the Ready Time of resource R_j and Et_{ij} represents the Execution Time of task T_i on resource R_j.

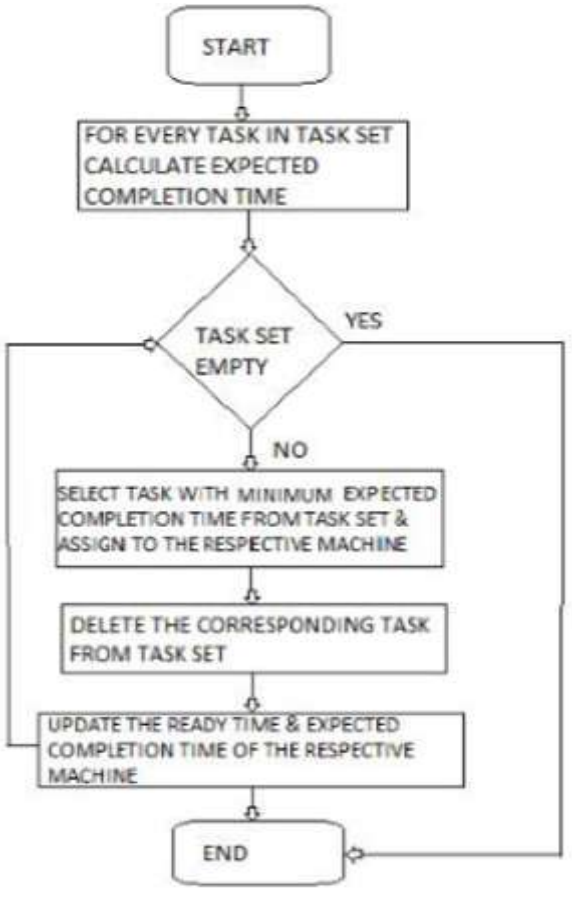

Fig 2 – Flow chart of Min-Min algorithm.

A. *Algorithm*

//Calculate the completion time matrix for all task on each resource

1. For all submitted tasks in task set; T_i
   1.1. For all resources; R_j
      1.1.1. Ct_{ij} = Et_{ij} + rt_j // Ct_{ij} means expected completion time of task 'i' on resource 'j'
2. Do while tasks set is not empty
3. Find task $T_k$that cost minimum execution time
4. Assign task T_k to resource R_j which gives minimum Expected completion time
5. Remove task T_k from tasks set.
   5.1. Update rt_j for selected R_j.
   5.2. Update C_{ij} for all T_i
6. End do

IV. IMPROVED LOAD BALANCE MIN-MIN ALGORITHM

In this paper, the Load Balance Improved Min-Min (ILBMM) scheduling algorithm is proposed that takes the characteristic of the Min-Min scheduling algorithm as foundation. The performance of Min-Min scheduling algorithm is considered to minimize the completion time of all works. However, the biggest weakness of Min-Min algorithm is it does not considers the work load of each resource. Therefore, some resources maybe always get busy but some nodes maybe still, as shown in Figure 2. The proposed ILBMM will improve the load unbalance of the Min-Min and reduce the execution time of each resource effectively

The pseudo code of ILBMM algorithm is represented in Fig 3. It starts by executing Min-Min algorithm. After execution of the concerned algorithm, it chooses the smallest size task from the most heavy load resource and calculates the completion time for that task on all other resources. Afterwards, the minimum completion time of the task is compared with the makespan produced by the algorithm. If it is less than makespan then the task is reassigned to the resource that produce it, and the ready time of both resources are updated. The process repeats until no other resources can produce less completion time for the smallest task on the heavy load resource than the makespan. Thus the heavy load resources are freed and the light load or idle resources are more utilized. This makes ILBMM to produce a schedule which improves load balancing and also reduces the overall completion time.

A. *Algorithm*

//Calculate the completion time matrix for all task on each resource

1. For all submitted tasks in the set; T_i
   1.1. For all resources; R_j
      1.1.1. $Ct_{ij}=Et_{ij}+rt_j$; //C_{ij} means expected completion time of task i on resource j
2. Do while tasks set is not empty
3. Find task T_k that cost minimum execution time.
4. Assign T_k to the resource Rj which gives minimum expected complete time
5. Remove T_k from the tasks set
   5.1. Update ready time rt_j for select R_j
   5.2. Update C_{ij} for all T_i
6. End Do

//Rescheduling to balance the load sort resources in order of completion time

7. Do while the most heavy load resource is considered no need for rescheduling
8. Find task T_i that cost minimum execution time on the heavy load resource R_j
9. Find the minimum completion time of T_i produced by resource R_k
10. If such minimum completion time <makespan
11. Reassign Task T_i to Resource R_k
12. Update the ready time of both R_j and R_k
13. End If
14. End Do

//where Makespan represents maximum completion time of all tasks which equals to the completion time of the most heavy load resource

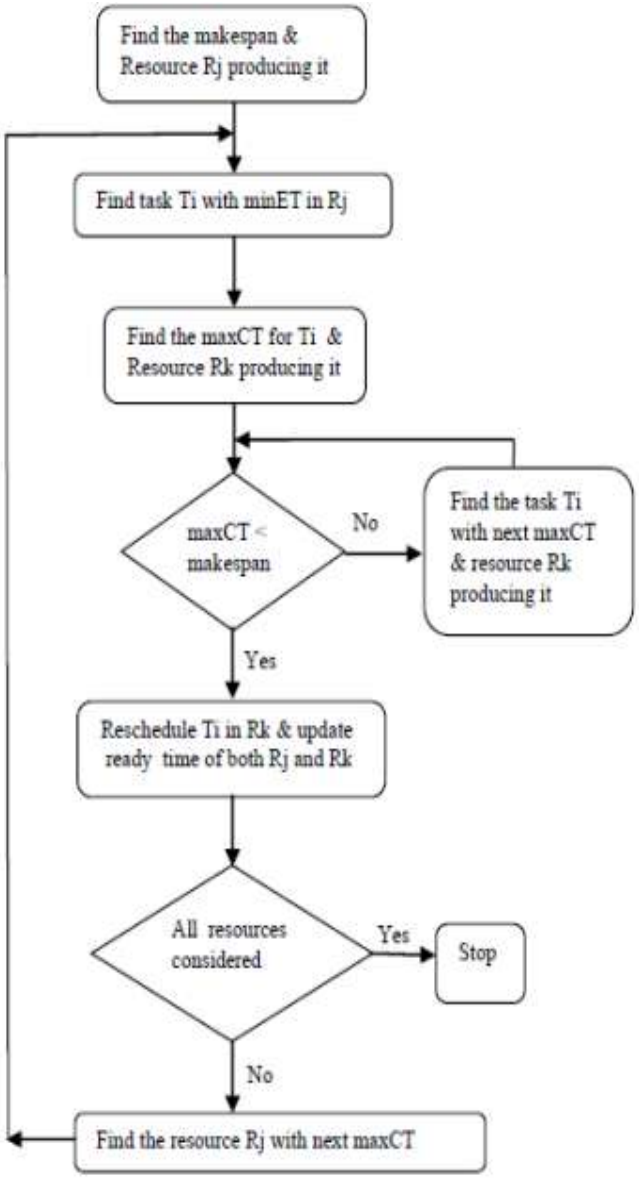

Fig 3 – Flow Chart of ILBMM

V. THEORETICAL ANALYSIS

Assume that task scheduler has meta-tasks and resource as given below. Table 1 represents the volume of instruction and data of task $T_1$to T4. Instruction volume is specified in MI (Millions instructions) Unit and data volume is specified in MB.

Table 1 – Task specification

Task	Instruction volume (MI)	Data volume (MB)
T1	8178	137
T2	11295	258
T3	12109	182
T4	6107	137

Table 2 represents processing speed and bandwidth of communication links for all resources where processing speed is specified in MIPS and bandwidth is specified in Mbps.

Table 2 – Resource specification

Resource	Processing speed (MIPS)	Bandwidth (Mbps)
R1	100	70
R2	350	60

Using Data given in table 1 and 2 to calculate the expected execution time of the task on each of the resource

Table 3 – Execution time of tasks on each resource.

Task	Resources R1	R2
T1	81.78	23.36
T2	112.95	32.27
T3	121.09	34.59
T4	61.07	17.45

Table 3 demonstrates calculated execution time of various task on various resources data in table 3 will be updated after all tasks are allocated.

In above scenario, Improved Min-Min achieves total makespan equal to 84.31 seconds and Min-Min achieves makespan equals 90.22 seconds

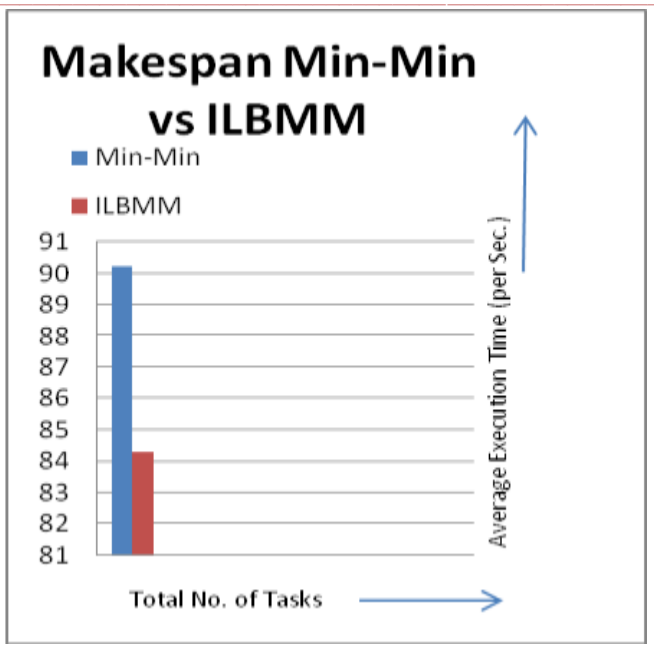

Fig 4 – Makespan (Total completion time of all task)

Figure above represent that the ILBMM gives better load balance and makespan than Min-Min algorithm

VI. CONCLUSION AND FUTURE WORK

To achieve high computing throughput in a cloud environment, anew scheduling algorithms, ILBMM were proposed in this paper. The Theoretical analysis show that under all possible situations the ILBMM is capable of decreasing completion time of tasks, improving load balance of resources and gain better overall performance than Min-Min algorithm.

This paper is only concerned with the makespan and load balancing for task scheduling based on Min-Min algorithm in Cloud environment. In future for simulation, we will use CloudSim which is java based simulation toolkit that enables modelling, simulation and experimenting on designing cloud computing infrastructures

REFERENCES

[1] N. Ajith Singh, M. Hemalatha, "An approach on semi distributed load balancing algorithm for cloud computing systems" International Journal of Computer Applications Vol-56 No.12 2012

[2] RS Chang, CY Lin, and CF Lin. "An Adaptive Scoring Job Scheduling algorithm for grid computing", Information Sciences – Elsevier, Volume 207, Pages 79–89, 2012

[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] SaeedParsa and Reza Entezari-Maleki , "RASA: A New Grid Task Scheduling Algorithm", International Journal of Digital Content Technology and its Applications,Vol. 3, pp. 91-99, 2009

[5] SalimBitam, "Bees life algorithms for job scheduling in cloud computing", International Conference on Computing and Information Technology, 2012

[6] R. F. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, J. D. Lima, F. Mirabile, L. Moore, B. Rust and H. J. Siegel, "Scheduling Resource in Multi-User, Heterogeneous, Computing Environment with SmartNet,"In the Proceeding of the Seventh Heterogeneous Computing Workshop, 1998

[7] Braun, Tracy D, 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 , Volume 61, Issue 6, Pages 810 – 837, 2001
[8] AnjuBala, Dr.InderverChana, "A Survey of Various Workflow Scheduling Algorithm in Cloud Environment",2nd national conference on information and communication technology(NCICT)2011
[9] George Amalarethinam. D.I, VaaheedhaKfatheen .S, "Max-min Average Algorithm for Scheduling Tasks in Grid Computing Systems", International Journal of Computer Science and Information Technologies, Vol. 3 (2) , 2012 ,3659-3663
[10] Elzeki, O. M., M. Z. Reshad, and M. A. Elsoud. "Improved Max-Min Algorithm in Cloud Computing", International Journal of Computer Applications, Volume 50, Issue 12, Pages 22-27, 2012
[11] S.-C.Wang, K.-Q. Yan, S.-S.Wang, C.-W. Chen, "A three-phases scheduling in a hierarchical cloud computing network", in: Communications and Mobile Computing (CMC), 2011 Third International Conference on,IEEE, 2011, pp. 114–117
[12] D.I. George Amalarethinam and P. Muthulakshmi, "An Overview of the scheduling policies and algorithms in Grid Computing ", International Journal of Research and Reviews in Computer Science, Vol. 2, No. 2, pp. 280294, 2011.
[13] Yu, Xiaogao, and Xiaopeng Yu. "A new grid computation-based Min-Min algorithm", Fuzzy Systems and Knowledge Discovery, FSKD'09 Sixth International Conference on, Volume 1, Pages 43 – 45, IEEE, 2009
[14] FatosXhafa, Ajith Abraham, "Computational models and heuristics methods for grid scheduling problems", Future Generation Computer systems, Vol. 26, pp. 608-621, 2010.
[15] T. Casavant and J. Kuhl, "A Taxonomy of scheduling in General purpose distributed computing systems", IEEE Trans on Software Engineering, Vol. 14, No. 2, pp. 141154, 198.