Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 57 (2015) 545 – 553

Procedia Computer Science

3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)

Enhanced Load Balanced Min-Min algorithm for Static Meta Task Scheduling in Cloud Computing

Gaurang Patel[a], Rutvik Mehta[b], Upendra Bhoi[c]

Computer Science & Engineering Department[a], Assistant Professor Information Technlogy Department[b],

Parul Institute of Engineering & Technology,Vadodara, Guajarat, India.

Assistant Professor Computer Science & Engineering Department[c],

Parul Institute of Technology,Vadodara, Guajarat, India.

Abstract

For today's most demanding service of cloud computing, there are many tasks required to be executed by the available resources to achieve best performance, reduce response time and utilize resources. There is a need of designing a new task scheduling algorithm that outperform appropriate allocation map of task. to achieve these challenges. As Load Balanced Min-Min Algorithm selects the task with minimum completion time and assigns it to appropriate resource, it sometimes does not produce better makespan and does not utilize resources effectively. This paper represents study of variety of task scheduling algorithms and modification of Load balanced Min-Min (ELBMM) algorithm for Static Meta-Task Scheduling. The modified algorithm is built based on comprehensive study of the impact of Load balanced Min-Min algorithm for Static Meta-Task Scheduling in grid computing. Enhanced Load balanced Min-Min algorithm (ELBMM) is based on Min-Min strategy and tasks rescheduling to use the unutilized resources effectively. It selects the task with maximum completion time and assigns it to appropriate resource to produce better makespan and utilize resource effectively.

Peer-review under responsibility of organizing committee of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)

Keywords: Cloud Computing, Meta Task Scheduling, ELBMM, Min-Min algorithm.

1. Introduction

Cloud computing provide a digital service delivery over the internet by various applications that are accomplished by computer systems in distributed datacenters [1][3]. Cloud Computing is getting more advanced now days. Cloud service providers are aimed to provide services using large scale cloud environment with cost effectiveness. Also,

Peer-review under responsibility of organizing committee of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)
doi:10.1016/j.procs.2015.07.385

there are few popular large scaled applications like social-networking-commerce etc. These applications can benefit to minimize the costs using cloud computing. It provides infrastructure, platform, and software which are made available as subscription-based services in a pay-as-you go model to consumers. Those services are known as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) in industries [4].Cloud computing is known as internet based computing service provided by various infrastructure providers on an on-demand basis, so that cloud is subject to Quality of Service (QoS), Load Balance (LB) and other constraints which have direct effect on user consumption of resources controlled by cloud infrastructure.

As the number of cloud users are increasing rapidly, assuring the optimum use of cloud resources are difficult task for cloud service providers and there are many issues associated with the task scheduling algorithms and so the load balancing for static meta-task described in [11] can be used for cloud for balancing the load. The set of assumptions here are that tasks have no deadlines and priorities should be assigned. The mapping process is to be executed statically in a batch mode fashion. The size of meta-task should be well known in advance. Security and others issues are assumed to be satisfied. The work here is limited to set of assumption. This can be implemented as a separate in itself. This paper only focuses on the issue of load balancing and poor makespan.

1.1 Task scheduling in cloud computing

In Cloud, many tasks have to be executed by the available resources to achieve distinct intensions. There is a need for a scheduling algorithm that is used by task scheduler to outperform appropriate allocation map of tasks on resources. Distinct types of scheduling are based on different criteria, such as Static/Dynamic scheduling, Centralized/Distributed scheduling, Preemptive/Non-Preemptive scheduling, Cooperative scheduling, Immediate/Online mode scheduling, Batch/Offline mode scheduling etc.

1.2 Role of task scheduler & terminologies associated to tasks scheduling

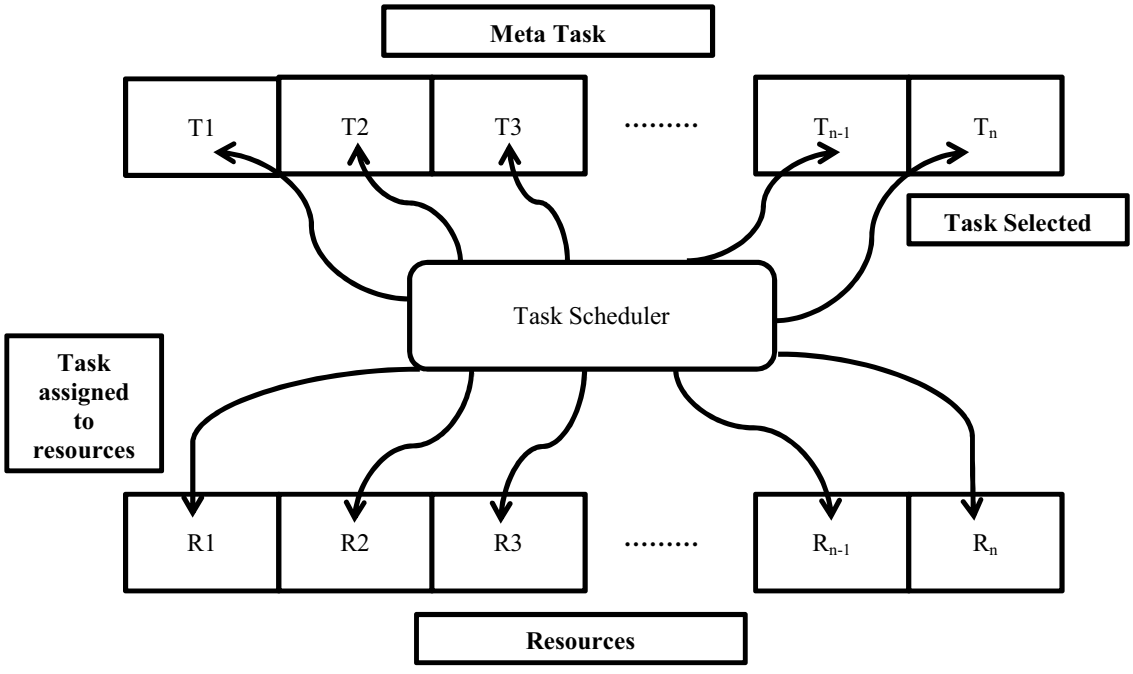


Figure 1 - Role of task scheduler

Terminologies related to Task scheduling are mainly concerned with calculating completion time and make-span. Here Completion Time = Waiting Time + Execution Time and the Meta tasks = set of tasks to be executed. The main objective is to calculate makespan. Makespan is overall completion time of Meta tasks.

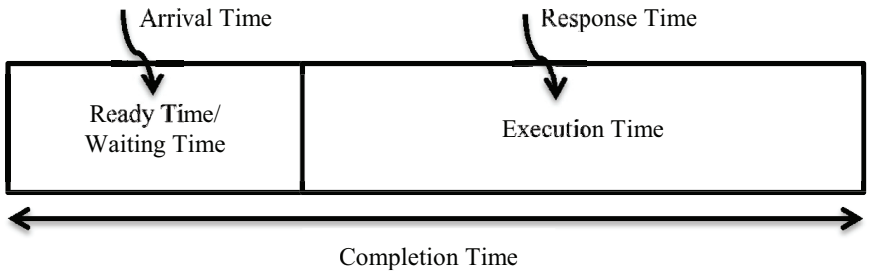


Figure 2 – Terminologies related to task scheduler

2. Related Task Scheduling Techniques

There are different task scheduling algorithms. In this section, different task scheduling techniques are reviewed briefly.

2.1 Opportunistic load balancing (OLB)

OLB schedules each task, in random order, to the next expected to be available machine, regardless of the task's expected execution time on that machine [8].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 expected task execution times, so it will generate poor makespan.

2.2 *Minimum Execution Time (MET)*

In contrast to OLB, Minimum Execution Time (MET) assigns each task, in random order, to the machine with the minimum expected execution time for that task, regardless of that machine's availability[9].The aim behind MET is to give each task to its best machine. This can cause a severe load imbalance across machines.

2.3 Minimum Completion Time (MCT)

Minimum Completion Time (MCT) assigns each task, in random order, to the machine that has the minimum expected completion time for that task [10]. The intension of MCT is to combine the benefits of OLB and MET, while avoiding the drawbacks of these two algorithms. In MCT, some tasks are needed to be assigned to machines that do not have the minimum execution time for them but it should consider total completion time on all machines.

2.4 *Min-min Task Scheduling Algorithm*

The Min-min heuristic begins with the set of all unmapped tasks (Set U). Then, the set of minimum completion times, M, for each task $t_i \in U_j$ is found. Moreover the task with the overall minimum completion time from M is selected and assigned to the corresponding machine (Thus named Min-min). At Last, the newly mapped task is deleted from U, and the process repeats until all tasks are mapped (i.e., U is empty). However, Min-min considers all unmapped tasks during each mapping decision and MCT only considers one task at a time. Min-min maps the tasks in the order that changes the machine availability status by the least amount that any assignment could. Let t_i be the first task mapped by Min-min on to an empty system. The machine that finishes t_i the earliest, say m_j, is also the machine that executes t_i the fastest. Therefore, the percentage of tasks assigned to their first choice (on the basis of execution time) is likely to be higher for Min-min than for Max-min (defined next). The expectation is that a smaller makespan can be obtained if more tasks are assigned to the machines that complete them the earliest and also execute them the fastest. Hear the main problem is that Min-min gives higher priority to small tasks and it increases response time for large tasks.

2.5 Max-min Task Scheduling Algorithm

In reference to the heuristic i.e. Min-min Algorithm, the Max-min heuristic also focus on reduced makespan. The Max-min heuristic begins with the set of unmapped tasks. Let U be the set of all unmapped tasks. Then, the set of minimum completion times, M, is found. The heuristic then selects overall maximum completion time from M and assigns to the resembling machine so it is named as Max-min Algorithm. Finally, the new mapped task is removed from U, and the process keeps on repeating until all tasks are mapped which implies till U is empty. Intuitively, Max-min tries to minimize the penalties incurred from performing tasks with longer execution times. For instance, let the meta-task being mapped has many tasks. Assume that one task has short execution time and one has very long execution time. Mapping the task with the longer execution time to its best machine first allows this task to be executed concurrently with the remaining tasks (with shorter execution times). For this case, this would be a better mapping technique than a Min-min mapping. Because in Min-min mapping all of the shortest jobs (tasks) would execute first, and then the longer running task would execute while several machines sit idle. Thus, the Max-min heuristic improves makespan. It may give a mapping with a more balanced load across machines.

2.6 Load Balanced Min-min Algorithm

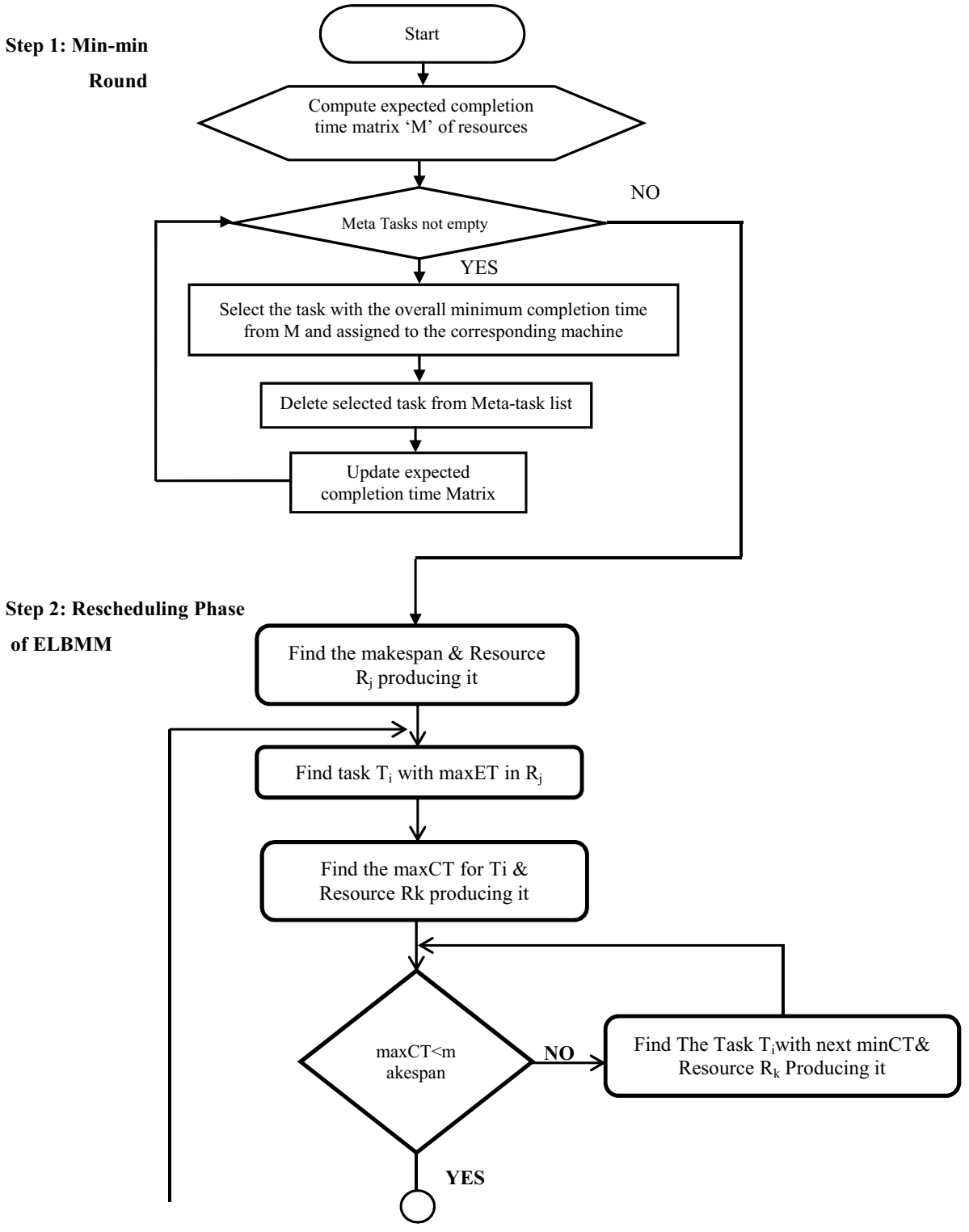
The traditional Min-Min algorithm is a simple algorithm that produces a schedule that minimizes the makespan than the other existing algorithms in the literature but this algorithm does not utilize resources effectively. Load Balanced Min-Min (LBMM) algorithm reduces the makespan and increases the resource utilization. This method has two-phases. In the first phase the traditional Min-Min algorithm is executed and in the second phase the tasks are rescheduled to use the unutilized resources effectively[12].

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. The limitation of Min-Min algorithm is that it chooses smaller tasks first which makes use of resource with high computational power. As a result, the schedule produced by Min-Min is not optimal when number of smaller tasks exceeds the large ones. To overcome this difficulty, Max-min algorithm schedules larger tasks first [7]. But in some cases, the makespan may increase due to the execution of larger tasks first. The waiting time of smaller tasks is also increased in Max-min. The LBMM algorithm outperforms all those algorithms both in terms of makespan and load balancing. Thus a better load balancing is achieved and the total response time of the system is improved. The proposed algorithm applies the Min-Min strategy in the first phase and then reschedules by considering the maximum execution time which is less than the makespan obtained from the first phase.

So LBMM executes Min-Min in the first round. In the second round it chooses the resources with heavy load and reassigns them to the resources with light load. LBMM identifies the resources with heavy load by choosing the resource with high makespan in the schedule produced by Min-Min. It then considers the tasks assigned in that resource and chooses the task with minimum execution time on that resource. Here in this schema problem is as Max-min gives highest priority to long tasks, it increases average response time for small tasks.

3. Enhanced Load Balanced Min-Min Approach for Task Scheduling

A unique modification of Load balanced Min-Min algorithm for Static Meta-Task Scheduling is considered here. The algorithm is built based on comprehensive study of the impact of Load balanced Min-Min algorithm for Static Meta-Task Scheduling in cloud computing. LBMM algorithm is based on two steps. First Min-Min Strategy is applied and then tasks are rescheduled to use the unutilized resources effectively and improve overall makespan. Flowchart given in figure 3 explains step-by-step process of enhanced load balanced Min-Min algorithm. Step 1 shows the process for Min-Min and step 2 shows the process for ELBMM.


Step 1: Min-min Round
Start
Compute expected completion time matrix 'M' of resources
Meta Tasks not empty
NO
YES
Select the task with the overall minimum completion time from M and assigned to the corresponding machine
Delete selected task from Meta-task list
Update expected completion time Matrix
Step 2: Rescheduling Phase of ELBMM
Find the makespan & Resource R_j producing it
Find task T_i with maxET in R_j
Find the maxCT for Ti & Resource Rk producing it
maxCT<m akespan
NO
Find The Task $T_i$with next minCT& Resource R_k Producing it
YES

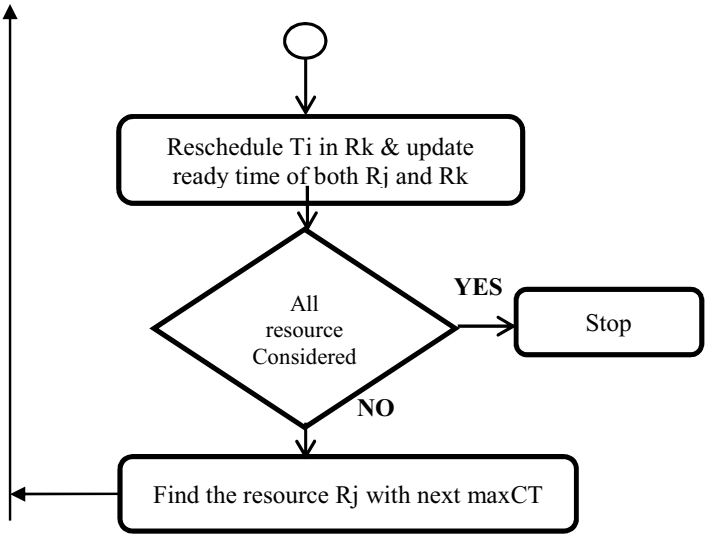


Figure 3 - Flowchart of Enhanced Load Balanced Min-Min

Algorithm

// calculate Completion Time Matrix for all tasks on each resource

1. for all submitted tasks in Meta-task; Ti

For all resources; Rj

Cij = Eij + rj // Cij means Expected Completion time of task 'i' on resource 'j'

//find the earliest completion time and the resource that obtains it

2. Do until all tasks are mapped

For each task find the earliest completion time and the resource that obtains it

Find the task Tk with the minimum earliest completion time assign task Tk to

The resource Rl that gives the earliest completion time

Delete task Tk from list

Update ready time of resource Rl

Update Cij for all i

End do

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

3. for all resources R

Compute makespan = max (CT(R))

End For

4. for all resources

For all tasks

Find the task Ti that has maximum ET in Rj

Find the MCT of task Ti

If MCT < makespan

Reschedule the task Ti to the resource that produces it

Update the ready time of both resources

End if

End for

End for

4. Theoretical Comparison & Analysis

Assume that Task scheduler has meta-tasks and resources as given below. Table 3.1 represents the volume of instructions and data of tasks T_1 to T_4. Instruction volume is specified in MI (Million instructions) unit and Data volume is specified in Mb.

Table 1 - Tasks Specification

Task	Instruction Volume (MI)	Data Volume (Mb)
T_1	8178	137
T_2	11295	258
T_3	12109	182
T_4	6107	137

Table 3.2 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 - Resources Specification

Resource	Processing Speed (MIPS)	Bandwidth (Mbps)
R_1	100	70
R_2	350	60

Using data given in Table 3.1 and Table 3.2, to calculate the expected execution time of the tasks on each of the resource.

Table 3 - Execution time of the tasks on each resource

Task	Resources	
	R1	**R2**
T_1	81.78	23.36
T_2	112.95	32.27
T_3	121.09	34.59
T_4	61.07	17.45

Table 3 demonstrates calculated execution time of various tasks on various resources. Data in Table 3 will be

updated after all tasks are allocated. Load Balanced Min-min strategy on meta-tasks achieves makespan equals to **90.22** seconds and Enhanced Load Balanced Min-min which achieves makespan equals **84.31** seconds.

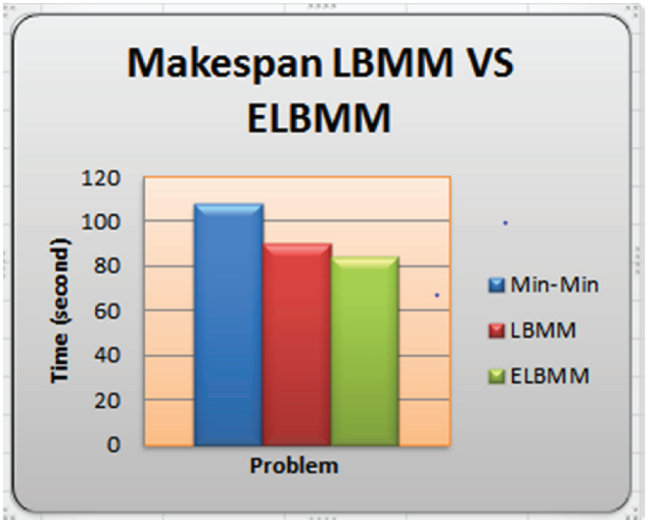


Figure 4 - Makespan (Total Completion Time of all tasks in Meta-tasks)

Figure above represents that the ELBMM algorithm gives better Load Balance and Markesan than LBMM algorithm and Min-min.

5. Conclusion

Load Balanced Min-min performs scheduling in two phase as it uses the advantages of two algorithms (Max-min and Min-min) and covers their disadvantages. As Load Balanced Min-Min Algorithm selects the task with minimum completion time and assigns it to appropriate resource, it sometimes doesn't produce better makespan and doesn't utilize resources effectively while Enhanced Load Balanced Min-min selects the task with maximum completion time and assigns it to appropriate resource. Theoretical Analysis and Result Analysis of LBMM and ELBMM shows that ELBMM produces better makespan and utilize resource as compared to LBMM.

6. References

[1] Salim Bitam, "Bees Life algorithms for job scheduling in cloud computing", International Conference on computing and Information Technology, 2012.

[2] The NIST Definition of Cloud Computing, Peter Mell Timothy Grance NIST Special Publication 800-145.

[3] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia "Above the Clouds: A Berkeley View of Cloud Computing", EECS Department,University of California, Berkeley ,Technical Report No. UCB/EECS-2009-28, February 10, 2009.

[4] Mythry Vuyyuru, Pulipati Annapurna, K.Ganapathi Babu, A.S.K Ratnam, "An Overview of Cloud Computing Technology", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-3, July 2012

[5] David Burford, CFA, MBA, MCP, "Cloud Computing: A Brief Introduction", LAB Enterprises Inc, February 20, 2010.

[6] L. Wang, G. Laszewski, M. Kunze and J. Tao, "Cloud computing: a perspective study, J New Generation

Computing", 2010, pp 1-11

[7] Sun Microsystems, "Introduction to cloud computing architecture". White Paper, Sun Microsystems, June 2009.

[8] Anju Bala, Dr.Inderveer Chana, "A Survey of Various Workflow Scheduling Algorithms in Cloud Environment", 2nd National Conference on Information and Communication Technology (NCICT) 2011

[9] O. M. Elzeki, M. Z. Reshad, M. A. Elsoud,"Improved Max-min algorithm in Cloud Computing", International Journal of Computer Applications (0975-8887) Volume 50 – No. 12, July 2012

[10] T.Casavant and J.Kuhl, "A Taxonomy of Scheduling in General Purpose Distributed Computing Systems", IEEE Trans. On Software Engineering, vol.14, no.3, February 1988, pp.141-154.

[11] Shu-Ching Wang, Kuo-Qin Yan *(Corresponding author), Wen-Pin Liao and Shun-Sheng Wang, "Towards a Load Balancing in a Three-level Cloud Computing Network", Institute of Electrical and Electronics Engineers - 2010.

[12] T. Kokilavani, Dr. 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