JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-4, Issue-4 E-ISSN: 2347-2693

Enhanced Load Balanced Min-Min Algorithm in Cloud Computing

RiddhiVarude[1], Ishita Shah[2], Mukesh Bhandari[3*]

[1,2] Department of Computer Engineering, Vadodara Institute of Engineering and Research, Gujarat Technological University, India
[3] Vadodara Institute of Engineering And Research, Gujarat Technological University, India

Available online at: www.ijcseonline.org

Received: Mar/23/2016 Revised: Apr /03/2016 Accepted: Apr/19/2016 Published: Apr/30/2016

***Abstract*— Cloud computing provides the applications and services presented over the Internet. These services are offered from the data-center all over the world. By using the environments of cloud computing many tasks are requires to be executed by available resources to achieve best performance, to reduce minimum response time, minimum completion time and utilization of resources etc. This paper focuses on the task scheduling and load balancing based on the different kinds of services and results .Using the environments of cloud computing the major problems are task scheduling and load balancing. This paper relates to benefits improved algorithms under the environment of Static & Dynamic cloud computing. According to the different types of scheduling, we define here the priority, efficiency and balances between the tasks respectively. Here proposed algorithm increases the resource utilization and reduces the makespan. In this paper, the experimental results shows the better algorithm from previous and fulfill the requirements of users.**

Keywords- Cloud Computing, Load Balancing, Min-Min Algorithm, Meta Task Scheduling.

I.INTRODUCTION

Cloud computing can be defined as a digital service delivery over the internet by different applications that are concluded by computer systems in distributed datacenters and it provides a high performance computing based on protocols that allow shared storage and computation over long distances [1]. Cloud computing is measured as internet based computing service as long as 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 expending of resources controlled by cloud infrastructure. Cloud computing as measured now a days to be a very popular because of the many benefits provided by the Cloud infrastructure. Hardware, Software and other services are accessible to users as a utility under an on-demand basis that is charged correspondingly to the amount of resources consumed by them. In some cases, Cloud providers use a part of their datacenter infrastructure for private resolutions and provide the rest unused ability as a cloud Service to public clients. Such setting enables cloud to increase the complexity of its resources capably and makes providers get money from such distributions. On the other side of service providing, the users come to be more comfortable and valuable as cloud allows them to enjoy performing their application/service and make them not worry about the infrastructure necessary and its difficulties assassination for their services [1,2].

In Fig 1, Cloud computing architecture is presented as layered model. Cloud layers are logically divided into three layers, Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) separately from top to bottom. From Fig 1, physical cloud resources (System Level) and middleware abilities form the basis provider of providing IaaS and PaaS in the form of a group of clearly datacenters and runtime environment and structure tools which ease the creation, deployment and execution process of application in the cloud. Finally, Cloud Application contains the applications available openly to the end users consuming SaaS services based on subscription model or pay-per-use basis [3].

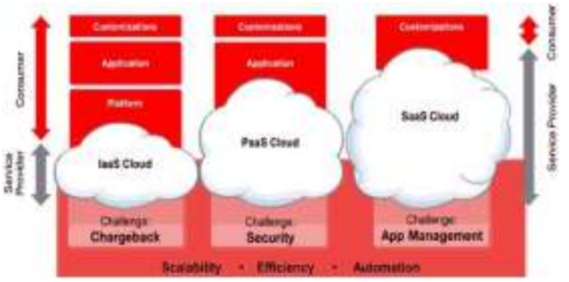


Figure1. Cloud Layered Organization [22].

A Cloud can be public, private, community or hybrid cloud. For public cloud, the infrastructure of cloud is open for common public or a large industry group. Public cloud always is held by cloud services seller. Where, private operates for a single organization. However, Community Cloud is shared by various organizations and supports a exact community. It may be managed by other (third party) organization. Last type, Hybrid, is a cloud whose infrastructure is a mixture of two or more clouds (i.e. private, community, or public). Hybrid computing is bound together by identical technology which allows data and application transportability [9].

II.RELATEDWORKS

Following Job scheduling techniques are currently established in clouds

A. *Opportunistic Load Balancing*

OLB allocates each task, in random order, to the next machine that is expected to be available, regardless of the task's expected execution time on that machine [4]. The intuition after OLB is to keep all machines as busy as possible. One benefit of OLB is its easiness, but because OLB does not consider normal task execution times, the mappings it finds can result in very poor makespans.

B. *Minimum Execution Time*

In compare to OLB, Minimum Execution Time (MET) allocates each task, inrandom order, to the machine with the best expected execution time for that task, unrelatedly of that machine's availability [4]. The motivation behind MET is to give each task to its great machine. This can reason a severe load imbalance through machines.

C. *Minimum Completion Time*

Minimum Completion Time (MCT) assigned each task, in random order, to the machine with the minimum expected completion time for that task [4]. This causes some tasks to be assigned to machines that do not have the minimum execution time for them. The intuition behind MCT is to combine the profits of OLB and MET, while escaping the situations in which OLB and MET perform poorly.

D. *Min-min task scheduling algorithm*

The Min-min experimental creates with the set U of all unmapped tasks. Then, the set of minimum completion times, M, for each $t_i \epsilon$ U, is found. Next, the task with the whole minimum completion time from M is selected and assigned to the consistent machine (hence the name Minmin). Last, the newly mapped task is separate from U, and the process repeats till all tasks are mapped (i.e., U is empty) [8]. Min-min is based on the minimum completion time, as is MCT. However, Min-min considers all unmapped tasks throughout each mapping choice and MCT only considers one task at a time. Min-min maps the tasks in the order that changes the machine accessibility status by the smallest quantity that any assignment could. Let $t_i$be the first task mapped by Min-min onto an empty system. The machine that completes $t_i$the earliest, say m_j, is also the machine that executes $t_i$the fastest. For every task that Min-min maps after t_i, the Min-min experimental changes the availability status of $m_j$by the least possible amount for every assignment. Therefore, the percentage of tasks allocated to their first choice (on the basis of execution time) is likely to be basic for Min-min than for Max-min (defined next). The probability is that a smaller makespan can be achieved if more tasks are allocated to the machines that complete them the earliest and also execute them the fastest.

E. *Max-min task scheduling algorithm*

The Max-min experimental is very similar to Min-min. The Max-min experimental also starts with the set U of all unmapped tasks. Then, the set of minimum completion times, M, is establish. Next, the task with the overall maximum completion time from M is selected and assigned to the reliable machine (hence the name Maxmin). Last, the recently mapped task is detached from U, and the process repeats til all tasks are mapped (i.e., U is empty) [8]. Spontaneously, Max-min attempts to minimize the penalties incurred from performing tasks with extended execution times. Assume, for example, that the metatask being mapped has many tasks with very short execution times and one task with a very long execution time. Mapping the task with the longer execution time to its best machine first permits this task to be executed concurrently with the remaining tasks (with shorter execution times). For this case, this would be a better mapping than a minmin mapping, where all of the shorter tasks would execute first, and then the extended running task would execute while different machines sit idle. Thus, in cases similar to this example, the Max-min experimental may give a mapping with a more balanced load through machines and a better makespan.

F. *Resource Aware Scheduling Algorithm*

The algorithm, RASA (Resource Aware Scheduling Algorithm), applies the Max-min and Min-min schemes alternatively to assign tasks to the resources. The algorithm creates a matrix C where C_{ij}denotes the completion time of the task $T_i$on the resource R_j. If the number of present resources is odd, the Min-min strategy is applied to assign the first task, otherwise the Max-min strategy is applied. The remaining tasks are allocated to their appropriate resources by one of the two schemes. For instance, if the first task is assigned to a resource by the Min-min strategy, the next task will be assigned by the Max-min strategy. In the next round the task assignment starts with a strategy different from the last round. For example if the first round starts with the Max-min strategy, the second round will starts with the Min-min strategy [2]. Experimental results displays that if the number of existing resources is odd it is preferred to apply the Min-min strategy the first in the first round otherwise is better to apply the maxmin strategy the first. Substitute exchange of the Min-min and Max-min strategies results in succeeding execution of a small and a large task on different resources and hereby, the waiting time of the small tasks in Max-min algorithm and the waiting time of the large tasks in Min-min algorithm are ignored. As RASA is contain of the Max-Min and Min-Min algorithms and have no time consuming instruction, the time complexity of RASA is O(mn^2) where m is the number of resources and n is the number of tasks (similar to Max-min and Min-min algorithms).

G .Improved Max-min Algorithm in Cloud Computing

Max-min algorithm allocates task Ti on the resource Rjwhere large tasks have maximum priority rather than smaller tasks. For example, if we have one long task, the Max-min could execute many short tasks concurrently while executing large one. The total makespan, in this case is determined by the execution of long task. But if metatasks contains tasks have relatively different completion time and execution time, the makespan is not determined by one of submitted tasks. We try to minimize waiting time of short jobs through assigning large tasks to be executed by slower resources. In additional, execute small tasks concurrently on fastest resource to finish large number of tasks during confirming at least one large task on slower resource. Based on these cases, where meta-tasks have standardized tasks of their completion and execution time, they suggested a considerable development of Max-min algorithm that indicates to improve of Max-min efficiency. Proposed improvement increases the chance of simultaneous execution of tasks on resources.

The algorithm computes the estimated completion time of the submitted tasks on every resource. Then the task with the overall maximum expected execution time is assigned to a resource that has the minimum whole completion time. Finally, this scheduled task is removed from meta-tasks and all calculated times are updated and the processing is repetitive til all submitted tasks are executed. The algorithm minimizing the total makespan which is the total complete time in large distributed environment. The proposed algorithm produces mapping scheme similar to RASA in such concurrency executing tasks and minimization of total completion time necessary to finish all tasks. Selecting task with maximum execution time points to select largest task should be executed. While selecting resource consuming minimum completion time means selecting slowest resource in the current resources. Thus distribution of the slowest resource to longest task allows to access of high speed resources for complete other small tasks concurrently. Also, we get shortest makespan of submitted tasks on current resources nearby concurrently. Not as original Max-min which proposed to be used if and only if submitted tasks is heterogeneous in their completion time and execution time, by means, there are clearly large tasks and small tasks [9].

"Select task with maximum execution time then assign to be executed by resource with minimum completion time" would be changed to “Select task with maximum completion time then assign to be executed by resource with minimum execution time".

H. LBMM

In this algorithm starts by executing the steps in Min-Min strategy first. It first identifies the task having minimum execution time and the resource producing it. Thus the task with minimum execution time is scheduled first in MinMin. After that it considers the minimum completion time since some resources are scheduled with some tasks. Since Min-Min chooses the smallest tasks first it loads the fast executing resource more which leaves the other resources idle. 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. The completion time for that task is calculated for all resources in the current schedule. Then the maximum completion time of that task is compared with the makespan produced by Min-Min. if it is less than makespan then the task is rescheduled in the resource that produces it, and the ready time of both resources are updated. Otherwise the next maximum completion time of that task is selected and the steps are repeated again. The process stops if all resources and all tasks assigned in them have been considered for rescheduling. Thus the possible resources are rescheduled in the resources which are idle or have minimum load.

III.PROBLEM DEFINITION

Due to the NP-completeness nature of the mapping problem, the developed approaches try to find acceptable solutions with reasonable cost considering many trade-offs and special cases. In this study, the proposed algorithms have been developed under a set of assumptions:

- The applications to be executed are composed of a collection of indivisible tasks that have no dependency among each other, usually referred to as *metatask*.
- Tasks have no deadlines or priorities associated with them.
- Estimates of expected task execution times on each machine in the HC suite are known. These estimates can be supplied before a task is submitted for execution, or at the time it is submitted.
- The mapping process is to be performed statically in a batch mode fashion.
- The mapper runs on a separate machine and controls the execution of all jobs on all machines in the suite.
- Each machine executes a single task at a time in the order in which the tasks are assigned (First Come First Served - FCFS).
- The size of the meta-tasks and the number of machines in the heterogeneous computing environment is known.

In static heuristics, the accurate estimate of the expected execution time for each task on each machine is known a priori to execution and is contained within an ETC (expected time to compute) matrix where *ETC (ti ,mj)* is the estimated execution time of task *i*on machine *j*.

The main aim of the scheduling algorithm is to minimize the makespan. Using the ETC matrix model, the scheduling problem can be defined as follows: Let task set T = t1, t2, t3, …. ,tn be the group of tasks submitted toscheduler and

Let Resource set R = m1, m2, m3, …. , mk

Be the set of resources available at the time of task arrival

Makespan produced by any algorithm for a schedule can be calculated as follows: **makespan = max (*CT (ti, mj)*)**

$$CT_{ij} = R_j + ET_{ij}$$

Where CT --> completion time of machine

ETij--> expected execution time of job i on resources

Rj--> resources

ready time or availability time of resource j after completing the previously assigned jobs.

The Enhanced Load Balanced Min-Min algorithm is developed to work for the above stated problem.

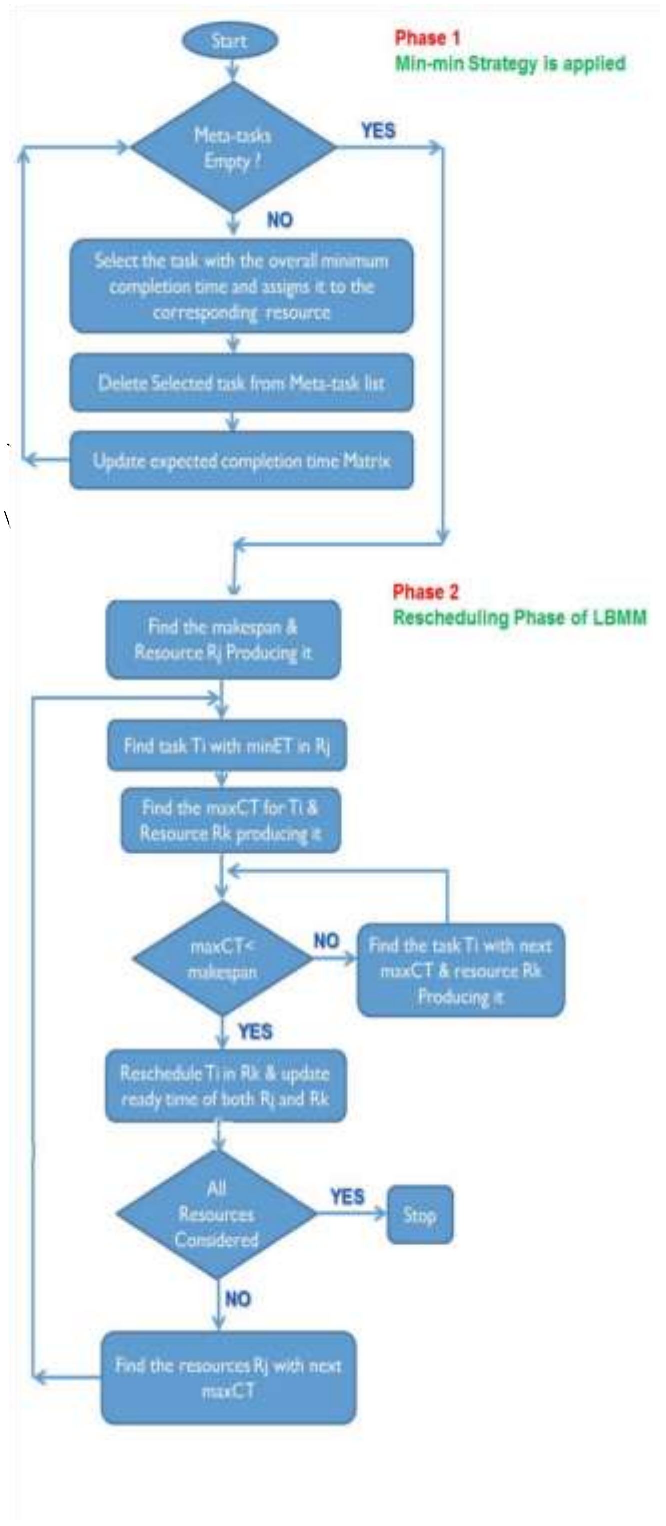
IV.ELBMM

A unique modification of Load Balanced Min-min algorithm is proposed.

In the Second Phase, Load Balanced Min-Min Algorithm selects the task with minimum completion time and assigns it to the corresponding resource, it sometimes doesn't produce better makespan and doesn't utilize resources effectively. So the idea is to select the task with maximum completion time and assign it to the corresponding resource to produce better makespan and utilize resource effectively.

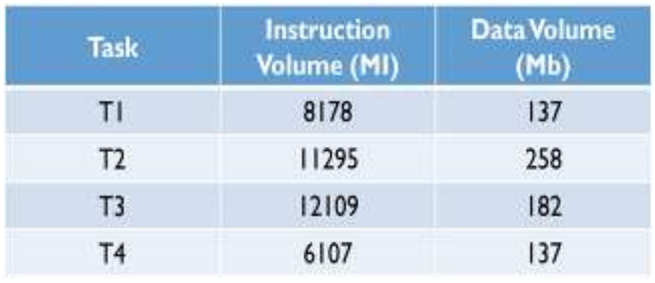
```
Phase 1: Applying Min-min Strategy

    For all tasks Ti
        For all resources
            $_{ij}$+ $r_j$// Finding Completion time of Task 'i' on Resource 'j' do

    until all tasks are mapped

        for each task find the earliest completion time and the
            resource that obtains it
        the task $T_k$with the minimum earliest completion time
        n task $T_k$to the resource $R_l$that gives the earliest completion
            time delete task $T_k$from list
        update ready time of resource $R_l$
        update $C_{il}$for all I

    end do
Phases 2: Rescheduling Phase of ELBMM

    for all resources R
        Compute makespan = max(CT(R)) End for

    for all resources for all tasks
        find the task Ti that has maximum ET in Rj
        d the MCT of task Ti // MCT=Maximum completion time
        if MCT <Makespan
            Reschedule the task Ti to the resources that produces it
            update the ready time of both resources
        End if
    End for
End for
```

Flowchart:


Start
Phase 1
Min-min Strategy is applied
Meta-tasks Empty ?
YES
NO
Select the task with the overall minimum completion time and assigns it to the corresponding resource
Delete Selected task from Meta-task list
Update expected completion time Matrix
Phase 2
Rescheduling Phase of LBMM
Find the makespan & Resource Rj Producing it
Find task Ti with minET in Rj
Find the maxCT for Ti & Resource Rk producing it
maxCT< makespan
NO
Find the task Ti with next maxCT & resource Rk Producing it
YES
Reschedule Ti in Rk & update ready time of both Rj and Rk
All Resources Considered
YES
Stop
NO
Find the resources Rj with next maxCT

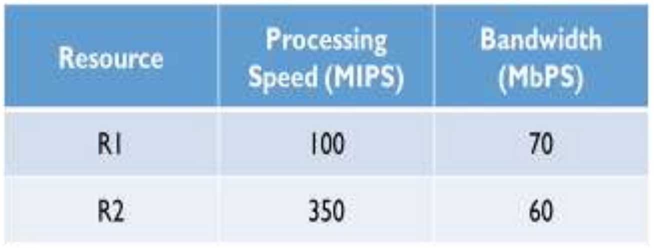
V. AN ILLUSTRATIVEEXAMPLE

Example of ELBMM, here tasks are 4 respectively T1, T2, T3 & T4. Resources are 2 respectively R1 & R2.



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

Table1. List of Tasks



Resource	Processing Speed (MIPS)	Bandwidth (MbPS)
R1	100	70
R2	350	60

Table2. List of Resources

Calculate Expected Execution Time of all tasks on each Resource

Task	R_1	R_2
T_1	81.78	23.36
T_2	112.95	32.27
T_3	121.09	34.59
T_4	61.07	17.45

Table3. Expected Execution Time of Tasks

Step 1: Applying Min-min Strategy

In min-min strategy first select the minimum task. All the tasks assign to the both resources using Minmin strategy. When task T4 assign to the R1 it takes comparatively more time than to R2 and same condition applied for all tasks. Thus, all tasks are carried out by R2 & R1 remained unused.

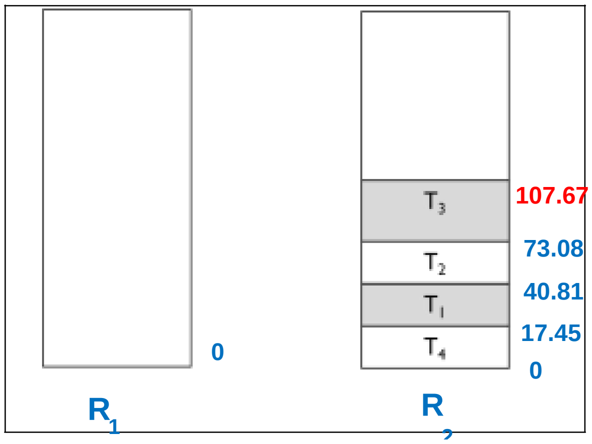
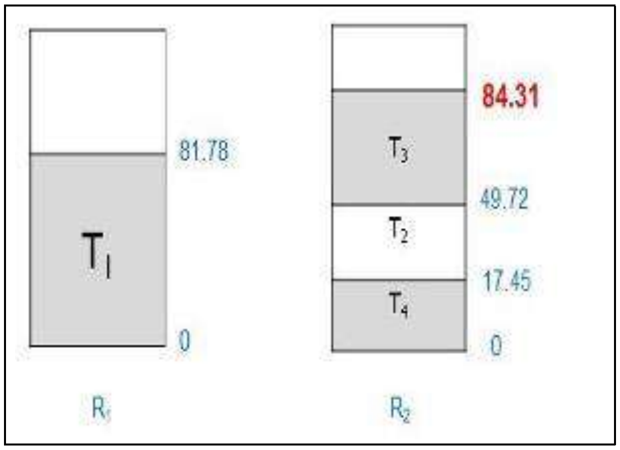


Figure2. Applying Min-Min strategy

Step 2: Applying ELBMM Strategy



Makespan by applying ELBMM - **84.31**second

Figure3. Applying ELBMM Strategy

For Load balancing strategy the tasks which are taking maximum time are assign to the other resources. Here task T1 are assign to resource R1.when assigning task from R2 to R1 the makespan of R1 should be equal to or less than R2.

VI. RESULTS AND DISCUSSION

Below Figure represents that the **Makespan (Total Completion Time of all tasks in Meta-tasks)** by applying ELBMM algorithm is less as compared to LBMM and Minmin.

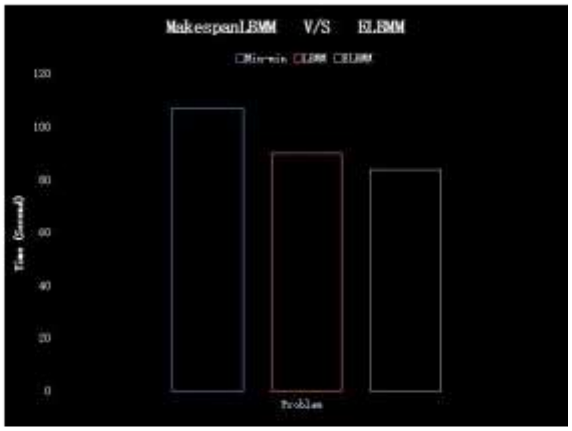


Figure4. Comparison between algorithms

VII.CONCLUSIONS AND FUTURE WORK

Load Balanced Min-Min Algorithm selects the task with minimum completion time and assigns it to the corresponding 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 the corresponding resource.

Theoretical Result Analysis of LBMM and ELBMM shows that ELBMM produces better makespan and utilization of resources as compared to LBMM.

This study can be further extended by implementing ELBMM algorithm in cloudSim which is java based simulation toolkit that enables modelling, simulation and experimenting on designing cloud computing infrastructures to prove this concept.

VIII.REFERENCES

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

[2] Saeed Parsa 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.

[3] Rajiv Ranjan, RajkumarBuyya, Cesar A.F.De Rose, Anton Beloglazov, Rodrigo N. Calheiros, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", unpublished.

[4] Tracy D. Braun, Howard Jay Siegel and Noah Beck , "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems",

Journal of Parallel and Distributed Computing 61, 810-837 (2001)

[5] Thomas A. Henzinger , Anmol V. Singh, Vasu Singh, Thomas Wies, “Static Scheduling in Clouds”.

[6] 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.

[7] M.Arora, S.K.Das, R.Biswas, “A Decentralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environments”.

[8] Henri Casanova, Arnauld Legrand, DmitriiJagorodnov and Francine berman, "Heuristics for scheduling parameter Sweep Applications in Grid Environments".

[9] O. M. Elzeki, M. Z. Reshad and M. A. Elsoud, "Improved Max-Min Algorithm in Cloud Computing", International Journal of Computer Applications (0975 – 8887).

[10] FatosXhafa, Ajith Abraham, “Computational models and heuristic methods for Grid scheduling problems”, “Future Generation Computer Systems 26”, 2010, pp.608-621.

[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] Hak Du Kim and Jin Suk Kim, “An On-line Scheduling Algorithm for Grid Computing Systems”, Electronics and Telecommunications Research Institute, Taejon, Korea, November 2003.

[13] D.Maruthanayagam and Dr.R.Umarani, “Enhanced Ant Colony Algorithm for grid scheduling”, International Journal Comp.Tech.Appl, Vol 1 (1) 43-53, November 2010.

[14] Saeed Parsa 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.

[15] T.Kokilavani, Dr. D.I. George Amalarethinam,”Load Balanced Min-min Algorithm for Static Meta-task Scheduling in Grid Computing", International Journal of Computer Application (0975-8887), Volume 20- No.2,April-2011.

[16] RajkumarBuyya, Rajiv Ranjan, Rodrigo N. Calheiros, “Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities”, International Conference on High Performance Computing and Simulation, HPCS2009, pp.1-11, 2009.

[17] Ghalem, B., Fatima Zohra, T., and Wieme, Z. “Approaches to Improve the Resources Management in the Simulator CloudSim” in ICICA 2010, LNCS 6377, DOI: 10.1007/978-3-642-16167-4_25, pp. 189–196, 2010.

[18] L. Wang, G. Laszewski, M. Kunze and J. Tao, “Cloud computing: a perspective study, J New Generation Computing”, 2010, pp. 1-11

[19] Sun Microsystems, “Introduction to cloud computing architecture”. White Paper, Sun Microsystems, June 2009.

[20] MythryVuyyuru, Pulipati Annapurna, K. Ganapathi Babu, A.S.K Ratnam, "An Overview of Cloud Computing Technology", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 22312307, Volume-2, Issue-3, July 2012.

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

[22] www.google.co.in