Various Load Balancing Algorithms in cloud Computing

Bhavisha Patel[1], Mr. Shreyas Patel[2]
[1]M.E Student, Department of CSE, Parul Institute of Engineering & Technology, Vadodara, India, bhavu2761991@gmail.com
[2]Assistant Professor, Department of CSE, Parul Institute of Engineering & Technology, Vadodara, India, Shreyaspatel_89@yahoo.com

ABSTRACT

Cloud Computing is the use of computing resources that are delivered as a service over a network. In cloud computing, there are many tasks requires to be executed by the available resources to achieve best performance, utilize the resources efficiently under the condition of heavy or light load in the network, minimize the response time and delay, for maintain system stability, to improve the performance, increase the throughput of the system, to decrease the communication overhead and to minimize the computation cost. This Paper Describe various load balancing algorithms that can be applied in cloud computing. These algorithms have different working and principles.

Keyword: - *Cloud Computing, Load Balancing, Load Balancing Algorithms, Round Rubin, Max-Min, Min-Min, ESCE, AMLB, Throttled Algorithm, Modified Throttled Algorithm, Weighted Active Monitoring Algorithm, MET, MCT, Improved Max-Min, OLB, Improved Cost Based Algorithm*

---

1. INTODUCTION

Cloud computing is an internet based technology in which internet can be represented as a cloud. Cloud is a platform provides pool of resources like applications, services and storage network etc to a computers and devices based on pay-per-use model [1]. Some characteristics of cloud computing is:

- Scalability and elasticity
- Reliability
- Device and location independence
- Security
- On-demand service
- Pay-per-use model

Cloud can be Public, Private or Hybrid [1].

Public cloud: A cloud is called a "public cloud" when the services are rendered over a network that is open for public use. Public cloud services may be free or offered on a pay-per-usage model.

Private cloud: Private cloud is cloud infrastructure operated for a single organization, whether managed internally or by a third-party, and hosted either internally or externally.

Hybrid cloud: Hybrid cloud is a combination of two or more private, community or public clouds that remain distinct entities offering the benefits of multiple deployment models.

Cloud computing providers offer their services according to several fundamental models:

Infrastructure as a service (IaaS): Cloud infrastructure services, known as Infrastructure as a Service (IaaS). In IaaS, the physical resources like storage and networking services can be split into a number of logical units called Virtual Machine (VM). These models manage virtualization, servers, hard drives, storage, and networking. Amazon EC2, Google Compute Engine (GCE).

Platform as a Service (PaaS):

Cloud platform services, or Platform as a Service (PaaS), are used for applications, and other development. PaaS is a framework they can build upon to develop or customize applications. Example of PaaS is Google app engine, AWS, Apache.

Software as a Service (SaaS):

Cloud application services, or Software as a Service (SaaS), use the web to deliver applications. SaaS eliminates the need to install and run applications on individual computers. Example of SaaS is Google Gmail, Microsoft 365.

2. LOAD BALANCING IN CLOUD COMPUTING [13]

Load balancing is the process of distribution of user request on available resources to maximize the throughput of the system and utilize the resources effectively. Load balancing required to improve the performance of the system by minimize the overall completion time and avoid the situation where some resources are heavily loaded or others remains under loaded in the system.

The goals of load balancing are:

- Improve the performance
- Maintain system stability
- Build fault tolerance system
- Accommodate future modification.
- Energy is saved in case of low load
- Cost of using resources is reduced
- Maximize throughput of the system
- Minimize communication overhead
- Resources are easily available on demand
- Resources are efficiently utilized under condition of high/low load
- Minimize overall completion time (makespan)

3. LOAD BALANCING ALGORITHMS IN CLOUD COMPUTING

Introduction related your research work Introduction related your research work Introduction related your research work Introduction related your research work Introduction related your research work Introduction related your research work Introduction related your research work Introduction related your research work Introduction related your research work Introduction related your research work

A. Round Rubin [2]

It is a task scheduling technique in which tasks are scheduled on the basis of time quantum. A small unit of time is called time slice which is defined in this algorithm. It is also called time sharing system in which each process will execute in particular time slot without any priority. These techniques divide the task request equally on the available service providers or resources. Round rubin is a static algorithm. The main advantage is that it is a starvation free. Every process will be executed by CPU for fixed time slice. So in this way no process left waiting for its turn to be executed by the CPU. All tasks will executed without any prioritization. It selects the load randomly and leads to

the situation where some nodes are heavily loaded and some are lightly loaded. Though the algorithm is very simple and easy to implement but there is an additional load on the scheduler to decide the time quantum. Round-rubin scheduling algorithm doesn't give special priority to more important tasks. This means an urgent request doesn't get handled any faster than other requests in queue. Throughput is low as the large process is holding up the Central processing unit for execution. In round rubin, equal amount of time will be assigned to each and every process for execution, but it can frustrate those with medium-length tasks. And it will take a more time to execute small task. For example in the supermarket its only allows shoppers to check out 11 items at a time. People with 10 or fewer items are unaffected. People with 100 items take longer, but there's a sense of fairness for loads that size. A customer with 10 items should go to the end of the line after checking out 10, making her wait in line seem longer than is normally appropriate. This isn't a problem for computers, but it can frustrate users and customers. Another disadvantage is that it will not consider the current load of the servers.

As shown in figure 3.1 there are three resources R1, R2, R3 and five tasks T1,T2, T3, T4 and T5. Here time slice/quantum is predefined. So tasks T1 and T4 are scheduled on resource R1 for a fixed time slot. And tasks T2 and T5 are scheduled on R2. Last one is scheduled on resource R3.

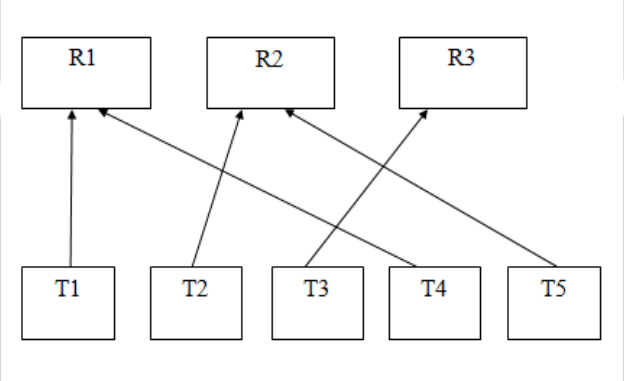

Fig 3.1 Round Rubin Algorithm [3]

B. OPPORTUNISTIC LOAD BALANCING ALGORITHM (OLB) [5]

This algorithm consider the allocated job request of each available virtual machine and assign the selected unexecuted task to the available virtual machine which has the lowest load among all the virtual machines. OLB assign the job in random order without considering the expectation execution time for the job on that virtual machine. As a result it provides better load balanced but it gives poor overall completion time (makespan). This algorithm is very simple and easy to implement and each virtual machine often keep busy. The main goal of OLB algorithm is to maintain load balance and makes each node in a working state.

C. Minimum Completion Time (MCT) [3]

The Minimum Completion Time job scheduling algorithm dispatches the selected job to the available VM that can provide the minimum completion time. Due to this reason there are some tasks to be assigned to machines that do not have the minimum execution time for them. The main

Idea of determining the minimum completion time in the scheduling algorithm is the processor speed and the current load on each VM. The algorithm first scans the available VMs in order to determine the most suitable virtual machine to perform the unexecuted job then dispatches that virtual machine and starts execution. Minimum Completion time is calculated according to this equation:

Ctij = Etij + rtj

Where, rtj = Ready Time of resource Rj

Etij = Execution Time of task Ti on resource Rj

Ctij = Completion Time of task Ti on resource Rj

Example:

Table 1 describes the list of tasks and list of available resources.

Resource	**MIPS**	**Task**	**MI**
R1	20	T1	10
R2	18	T2	5
R3	14	T3	22
		T4	15

Table 1 Resource and Task Specification

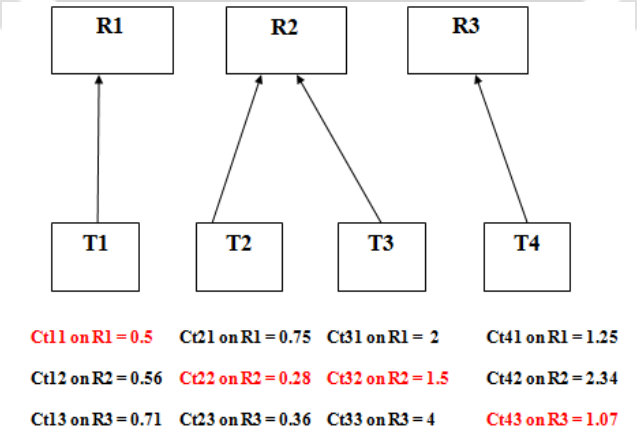

Fig 3.2 Minimum Completion Time (MCT)

As shown in figure 2.2, task T2 and T3 are scheduled on resource R2 and task T1 is executed by resource R1 and task t4 is scheduled on resource R3. [3]

D. Minimum Execution Time (MET) [8]

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. The aim behind MET is to give each task to its best machine. This can cause a severe load imbalance across machines. So some virtual machines are overloaded and some are underutilized in the system.

Expected Execution time is calculated according to this equation:

Etij = Task Length (MI) / Processing Speed (MIPS)

Where, Etij = Execution time of the task Ti on Rj

Example:

Resource	MIPS	Task	MI
R1	10	T1	10
R2	8	T2	15
R3	5	T3	20
		T4	25

Table 2 Resource and Task Specification

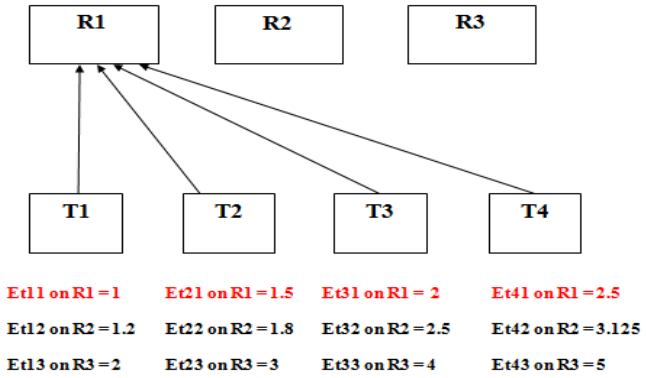

Fig 3.3 Minimum Execution Time (MET)

As shown in figure 3.3 Minimum Execution Time (MCT) considers only the processing power of the virtual machine. According to the ratio of processing speed and the task length current task is scheduled on that resource which gives minimum execution time. In given example tasks T1, T2, T3, T4 are scheduled on resource R1. And resource R2 and R3 are remaining idle.

E. Throttled Load Balancing Algorithm

(TLB) [6]

Throttled load balancing algorithm maintains the index list for all virtual machines. Index list maintain the information of the state (Busy/Ideal) of each virtual machine. When a new request arrives, throttled load balancer scan the indexed list from the top until the first free virtual machine is found. Throttled load balance return the ideal virtual machine id to the data center controller. Further, the data center controller allocate job to that identified virtual machine. If suitable virtual machine is not present to execute the task then throttled load balancer returns -1 to the data center controller and it will queued until the any one virtual machine becomes free. This algorithm is dynamic.

ALGORITHM [7]:

1) Throttled load balancer maintains index Table of virtual machines and state (BUSY/AVAILABLE) of virtual machines. At the starting all virtual machines are free.
2) Data Center Controller receives a job request.
3) Data Center Controller sends the request to the Throttled Load Balancer for the allocation of request.
4) Throttled Load Balancer starts with the available virtual machine list from the first index for the availability of free virtual machine.

   Case 1: if virtual machine found then
   a. the Throttled Load Balancer returns the Virtual Machine id to the Data Center Controller
   b. Data Center Controller sends the request to that identified virtual machine.
   c. Data Center Controller notifies the Throttled Load Balancer of the new allocation of job
   d. Throttled Load Balancer updates the allocation Table

   Case 2: if no Virtual machine is found

   a. The Throttled Load Balancer returns -1.

5) When the Virtual machine finishes execution of the request, and the Data Center Controller receives the response cloudlet, it notifies the Throttled Load Balancer of the Virtual machine deallocation.
6) If there are more requests, Data Center Controller repeats step 4 with first index again and process is repeated until size of index Table is reached.
7) Continues from step 2.

Throttled Load Balancing Algorithm parse the index from the beginning every time whenever the new request is arrive at the Data Center Controller. So resulting some virtual machines are overloaded and some are underutilized. Resulting load sharing is not uniform on available servers. And also resources utilization is not properly done by this algorithm.

Another major disadvantage is that this algorithm does not consider the advance load balancing requirements like execution time and completion time of each request on the available resources for the load balancing [2]

F. Equally Spread Current Execution Algorithm (ESCE) [2]

In Equally spread current execution technique load balancer spread equal load to all the available servers or virtual machines which connected with the data centre. Load balancer maintains an index Table which contains the record of Virtual machines and number of requests currently assigned to the Virtual Machine (VM). If the request comes from the data centre to allocate the new VM, it scans the index Table and fine the least loaded virtual

machine. The load balancer also returns the least loaded VM id to the data centre controller. In case there are more than one VM is found than first identified VM is selected for handling the request of the client/node. The data centre revises the index Table by increasing the allocation count of identified VM. When VM completes the assigned task, a request is communicated to data centre which is further notified by the load balancer. The load balancer again revises the index Table by decreasing the allocation count for identified VM by one but there is an additional computation overhead to scan the queue again and again. In this algorithm load is equally spread so whole system is load balanced and no virtual machines are under loaded or overloaded. And because of this data migration and virtual machine cost is also reduced. The main advantage is that there is a no overhead to assign the time slot for each process.

G. Min-Min [4]

Min-Min algorithm starts with a set of unmapped / unscheduled jobs. Min-Min Task Scheduling Algorithm is a static scheduling algorithm. This algorithm first identifies the jobs having minimum execution time and these tasks are scheduled first in this algorithm. Then it will calculate the expected completion time for each tasks according to available virtual machines and the resource that has the minimum completion time for selected task is scheduled on that resource. The ready time of that resource is updated and the process is repeated until all the unexecuted tasks are scheduled. Hence Min-Min algorithm chooses the smallest size tasks first and assigned these tasks to fastest resource. So it leaves the some resource overloaded and other remains underutilized or idle. In addition it does not provide load balanced in the system. It will provide a better makespan and resource utilization when the number of the large task is more than the number of the small task in meta-task. Main disadvantage of this algorithm is that it selects small tasks to be executed firstly, which in turn large task delays for long time. Min=Min algorithm is failed to utilize resources efficiently which lead to a load imbalance.

Expected completion time of each task to the available resources is calculated using this equation:

$$Ctij = Etij + rtj$$

Where, rtj = Ready Time of resource Rj

Etij = Execution Time of task Ti on resource Rj

Ctij = Completion Time of task Ti on resource Rj

Example:

Resource	**MIPS**	**Task**	**MI**
R1	10	T1	10
R2	8	T2	15
R3	5	T3	20
		T4	25

		T5	50

Table 3, represents the processing power (MIPS) of each resource and the task size (MI) of each task. Data given in Table III are used to calculate the expected completion time and execution time of the tasks on each of the resources.

Figure 3.4 demonstrates calculated expected complete time for each task. On next step of the algorithm iteration, data in figure will be updated until all tasks are allocated.

As shown in Figure 3.4, Tasks T1. T3, T5 are scheduled on resource R1 and tasks T2, T4 are executed by resource R2.

The Min-Min algorithm achieves total makespan (overall completion time) 8 seconds but uses only two resources R1, R2. And resource R3 is still remains idle.

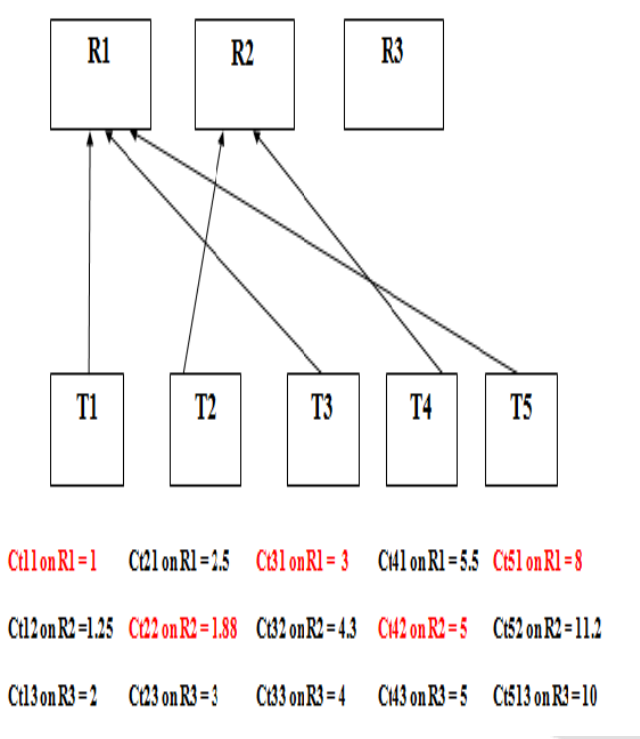

Fig 3.4 Min-Min Task Scheduling Algorithm

H. Max-Min

Max-min algorithm allocates task Ti on the resource Rj where large tasks have highest priority rather than smaller tasks in the meta-task. It is begin with a set of unmapped tasks. Then calculate expected execution time and expected completion time of each task on available resource. Then select the task with overall maximum completion time and assign this task to the resource which give overall minimum execution time. The ready time of that resource is updated. Finally, the new mapped task is removed from the meta-task, and the process keeps on repeating until all tasks are mapped which implies till meta-task is empty. The idea of Max-Min algorithm is to reduce the waiting time of large tasks. As Max-min gives highest priority to long tasks, it increases average response time for small tasks. This algorithm performs better when the number of small task is much more than the long one.

Example:

Table 4 describes the list of tasks and list of available resources.

Resource	**MIPS**	**Task**	**MI**
R1	10	T1	10
R2	8	T2	15
R3	5	T3	20
		T4	25
		T5	50

Table 4 Resource and Task Specification

Task/Resource	**R1**	**R2**	**R3**
T1	1	1.25	2
T2	1.5	1.875	3
T3	2	2.5	4
T4	2.5	3.12	5
T5	5	6.25	10

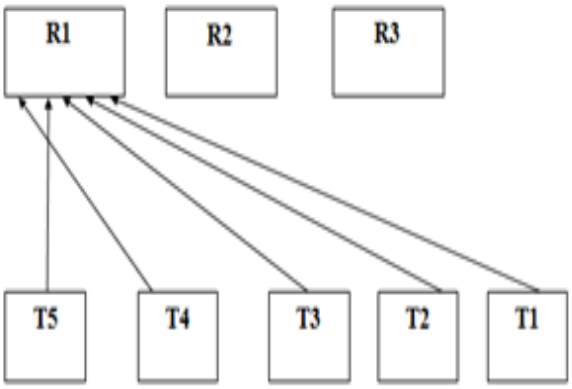

Fig 3.5 Max-Min Task Scheduling Algorithm

Table 5 demonstrates the calculated expected execution time for each task on available resources. Figure 3.5 shows the scheduled task on the resources with minimum execution time.

I. Improved Max-Min Task Scheduling Algorithm [9]

Improved max-min algorithm is a modified version of Max-Min algorithm. Improved Max-Min algorithm based on expected execution time instead of completion time as a selection basis of task and resources. this algorithm applied on set of unscheduled/unmapped tasks. First it will calculate the Expected Execution time and completion time of each task on available resources. Then it identifies the task with maximum execution time (largest task in the meta-tasks) and scheduled this task on that resource which gives the minimum completion time (slowest resource). After this remove the mapped task from the meta-tasks and update the ready time of the selected resource. While meta-tasks is not empty then apply the Max-Min algorithm on the remaining small tasks. This algorithm try to minimize the waiting time of small tasks by executing large task on slowest resource and concurrently assign the small task on another available resources by max-min strategy. It will improve the makespan and response time for the small tasks concurrently.

Example:

Resource	**MIPS**	**Task**	**MI**
R1	100	T1	500
R2	300	T2	780
		T3	800
		T4	1200

		T5	900

Table 6 Resource and Task Specification

Task (Maximum Expected Execution Time – Largest Task) = T4

Resource (Minimum Completion Time -- Slowest Resource) = R1

Task/Resource	**R1**	**R2**
T1	5	1.6
T2	7.8	2.6
T3	8	2.7
T4	12	4
T5	9	3

Table 7 Expected Execution Time of the tasks on each Resources

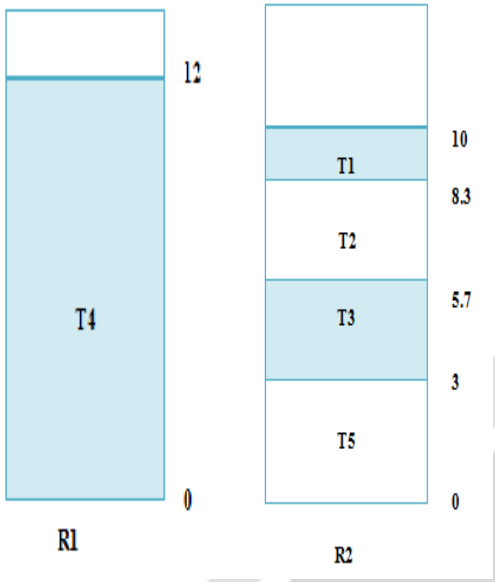

Fig 3.6 Gantt chart of Improved Max-min Algorithm

J. Improved Cost-Based Algorithm [10]

Traditional way for task scheduling in cloud computing use the direct task of users as the overhead application basis. The problem is that there are no relationship between the overhead application basis and the way that different task cause overhead cost of resources. For large number of simple tasks this increases the cost and the cost is decreased if we have a small number of complex tasks. To minimize the cost and minimize the makespan improved activity based cost algorithm is proposed. The objective of this algorithm is to schedule task groups to the available resources in the cloud computing environment, where all resources have different processing power and different computation cost. For reducing the communication overhead and computation time this algorithm is proposed. In this algorithm selection of resources is based on the cost of resource. Job grouping is done on the basis of the resource's processing capability.

In improves activity based costing algorithm job scheduling strategy consider processing requirement of each task/job and processing capabilities of resources. This algorithm focuses on scheduling of independent tasks. It will create coarse-gained jobs by grouping fine-gained jobs together to reduce job assignment overhead and also reduce the computation ratio. Scheduler receives number of tasks and calculates their priority level and put into appropriate list high, medium, low.

Now job grouping algorithm is applied to this lists to allocate the task-groups to different available resources. Then scheduler gathers characteristics of available resources. Select particular resource and multiplies MIPS with the granularity size. Grouping is done according to the capability of the resource. Granularity size is the time within which a job is processed at the resources. Granularity size is used to determine total amount of jobs that can be completed within a specified time in a particular resource. If the task-groups total length is less than the MIPS of resources then we can add another task from the list. And subtract the length the last task from the task group. Repeat this process until the all tasks in the list are grouped into task-groups.

The priority level of each task can be calculated as in Equation (1),

$$Lk = n\Sigma\, i = 0 Ri,k * Ci,k / Pk$$

Where,

Ri,k : The ith individual use of resources by the kth task.

Ci,k : The cost of the ith individual use of resources by the kth task.

Pk : The profit earned from the kth task.

Lk : The priority level of the kth task.

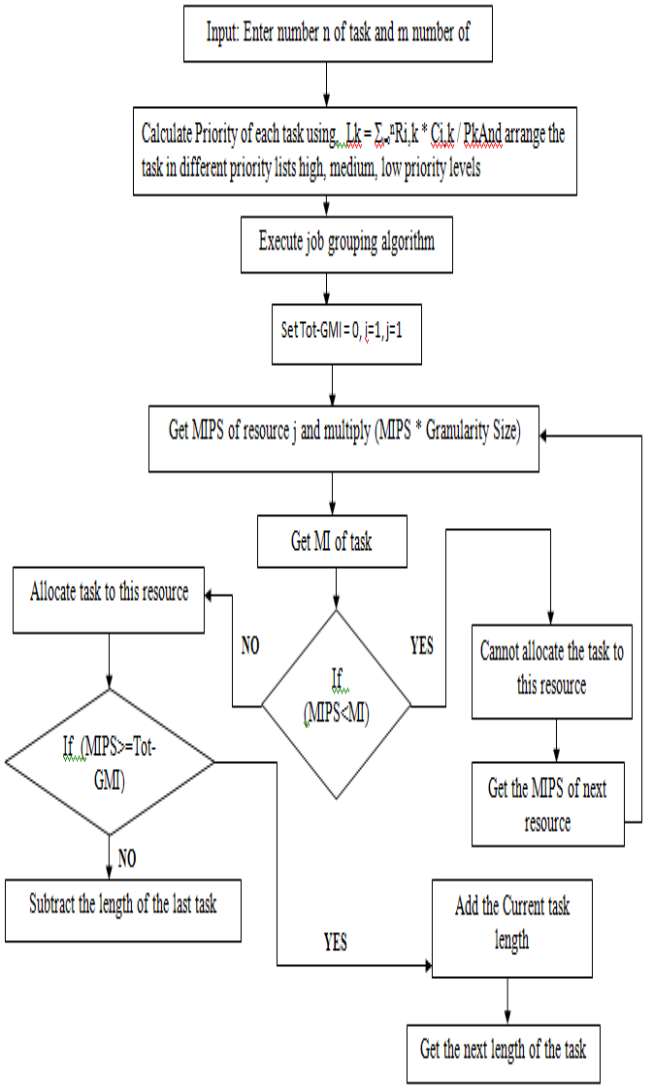

Fig 3.7 Flowchart of Improved ABC Algorithm

In figure 3.8 user tasks and available resource is given. According to given granularity size tasks 0,1,2,3 are grouped on the basis of resource capability and requirement of tasks and assigned this task to the available resource R1. Next tasks are scheduled on another capable resource.

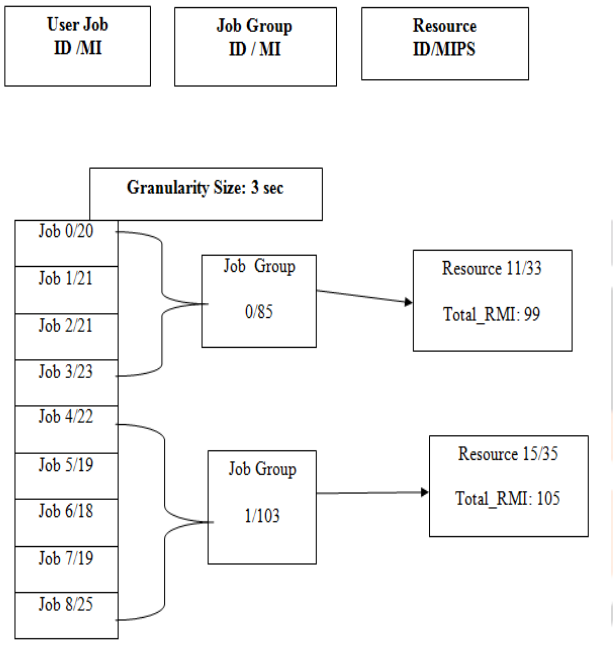

Fig 3.8 Example of Job Grouping in Improved ABC Algorithm

K. Weighted Active Monitoring Load Balancing Algorithm

The Weighted Active Monitoring Load Balancing Algorithm is proposed by modifying the Active Monitoring Load Balancer by assigning a weight to each Virtual Machine. In this proposed Load balancing algorithm all VM are assigned different amount of the available processing power. First create virtual machines of different computing power in terms of its core processor, processing speed (MIPS), memory, storage. Allocate weighted count according to computing power of virtual machines. If one VM is capable of having twice as much load as the other, then assigned weight is '2' or if it can take four times load then assigned weight is '4' and so on.

For example:

- Host server with single core processor, 1GB of memory, 1TB of Storage space, 1000000 bandwidth will have weighted count=1
- Host server with 2 core processor, 4GB of memory, 2TB of Storage space and 1000000 bandwidth will have weighted count=2
- Host server with quard core processor, 8GB of memory 4TB of Storage space and 1000000 bandwidth will have weighted count=4 and so on..

In this algorithm WeightedActiveVmLoadBalancer maintains an index Table of VMs, associated weighted count and the number of requests currently allocated to the VM. When a request will arrive, load balancer parse the index and find the least loaded virtual machine. It allocate request to the most powerful VM according to the weight assigned. If there are more than one, the first identified is selected. After finishing processing request on the virtual machine, load balancer dealloacate the virtual machine and update the resource allocation Table by decreasing allocation count by one.

L. Modified Throttled Algorithm [7]

Throttled algorithm is modified for balancing the load between the resources. In throttled algorithm whenever the new request is arrive at the cloudlet, the task is submitted to the throttled virtual machine balancer. If there are more than one request or more request group are present then it will store into the queue until the one of virtual machine becomes available. In throttled algorithm load balancer maintain record of state (Busy/Available) of virtual machine. When the request is come, load balancer parse the index from the beginning and return the free virtual machine id. In this algorithm each time it will parse the index from the beginning so initial virtual machines are heavily loaded and remaining are ideal or under loaded. Throttled algorithm give better response time but it is failed in assigning all incoming jobs uniformly. Modified throttled algorithm overcomes all disadvantages of the throttled algorithm. Difference between the modified throttled algorithm and throttled algorithm is that modified throttled algorithm will not parse the index from the beginning each time. When the next request arrives, the Virtual Machine at index next to already assigned VM is chosen depending on the state of VM. In proposed algorithm all the VMs are utilized completely and properly. Proposed algorithm distributes load nearly uniform among VMs, with improved response time compared to existing algorithms.

L. Active Monitoring Load Balancing Algorithm [12]

The Active Monitoring Load Balancer maintains information about each virtual machine's and the number of request currently allocated to which virtual machine when a request is allocate a new virtual machine arrives. If there are more than one Virtual machine, the first identified is selected ActiveMonitoringLoadBalancer returns the virtual machine id to the data centers controller. The data centers controller sends the request to the virtual machine identified by that id. The data center controller notifies the ActiveMonitoringLoadBalancer to new allocation and cloudlets is sent to it.

4. CONCLUSIONS

In this study, the behavior of this load balancing algorithms, namely: Round Rubin, Max-Min, Min-Min, ESCE, AMLB, Throttled Algorithm, Modified Throttled Algorithm, Weighted Active Monitoring Algorithm, MET, MCT, Improved Max-Min, OLB, Improved Cost Based Algorithms are examined in a cloud computing environment. This load balancing algorithms are evaluated by considering the major characteristics of the cloud computing environment. Some of the algorithms are beneficial in some cases like max-min is provide better performance if there are number of small tasks I more than the longer ones. There is not a single algorithm that provides maximum throughput, minimum response time and effective resource utilization.

REFERENCES

[1] Tushar Desai, Jignesh Prajapati, "A Survey Of Various Load Balancing Techniques And Challenges In Cloud Computing", International Journal Of Scientific & Technology Research Volume 2, Issue 11, November 2013, ISSN 2277-8616.

[2] Vishrut Majmudar, Harshal Trivedi and Jitendra Bhatia, "HTV Dynamic Load Balancing Algorithm for Virtual Machine Instances in Cloud" 978-0-7695-4931-6/12 $26.00 © 2012 IEEE.

[3] Isam Azawi Mohialdeen, "COMPARATIVE STUDY OF SCHEDULING AL-GORITHMS IN CLOUD COMPUTING ENVIRONMENT," Isam Azawi Mohialdeen / Journal of Computer Science 9 (2): 252-263, 2013, ISSN 1549-3636.

[4] Gbola akanmu, Professor Frank Wang, Huankai Chen, "User-Priority Guided Min-Min Scheduling Algorithm For Load Balancing in Cloud Computing", IEEE 2013.

[5] Shu-Ching Wang, Kuo-Qin Yan , Wen-Pin Liao and Shun-Sheng Wang, "Towards a Load Balancing in a Three-level Cloud Computing Network", 978-1-4244-5540-9/10/$26.00 ©2010 IEEE.

[6] Ms. G. Vidya, Mr. M. Ajit, "VM Level Load Balancing in Cloud Environment", 4th ICCCNT 2013 July 4-6, 2013, Tiruchengode, India Published, IEEE-31661.

[7] Shridhar G.Domanal and G. Ram Mohana Reddy, "Load Balancing in Cloud Computing Using Modified Throttled Algorithm", Department of Information Technology National Institute of Technology Karnataka Surathkal, Mangalore, India.

[8] O. M. Elzeki, M. Z. Rashad, M. A. Elsoud, "Overview of Scheduling Tasks in Distributed Computing Systems", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-3, July 2012.

[9] Upendra Bhoi1 and Purvi N. Ramanuj, "Enhanced Max-min Task Scheduling Algorithm in Cloud Computing", Volume 2, Issue 4, April 2013, ISSN 2319 - 4847.

[10] Mrs.S.Selvarani1 and Dr.G. Sudha Sadhasivam, "Improved Cost-Based Algorithm For Task Scheduling InCloud Computing", 978-1-4244-5967-4/10/$26.00 ©2010 IEEE.

[11] Jasmin James And Dr. Bhupendra Verma, "Efficient VM Load Balancing Algorithm for a Cloud Computing Environment", Jasmin James Et Al. / International Journal On Computer Science And Engineering (IJCSE), ISSN : 0975-3397.

[12] Md. S. Q. Zulkar Nine, Md. Abul Kalam Azad, Saad Abdullah, Rashedur M Rahman , "Fuzzy Logic Based Dynamic Load Balancing in Virtualized Data Centers", published in IEEE 2013.

[13] Chao Yin, Haozheng Ren, Yihua Lan, "The Load Balancing Algorithm in Cloud Computing Environment", 2012 2nd International Conference on Computer Science and Network Technology 978-1-4673-2964-4/12/$31.00 ©2012 IEEE.