An Alternative Approach to Resolve Load Balancing Problems in Cloud Computing

[1] Pooja Anjee, [2] Rishab Nagaraj, [3] Samiksha M[4] Nithya Ganesan
[1], [2], [3], Department of Computer Science and Engineering, [4] Faculty of Computer Science and Engineering,
RV College of Engineering, Bangalore-560085
[1] mspooja13@gmail.com [2] risnag1995@gmail.com [3] samiksha.manjunath@gmail.com,[4] nithyag@rvce.edu.in

***Abstract*: With the increase in adoption and deployment of cloud computing services in recent times, most users are moving from traditional computing to cloud computing as it gives various favourable features like on-demand access, broad network access, rapid elasticity, etc. All these services incorporate sharing of resources which leads to an increase in load on a single machine. This increase in load causes further overall degradation of system performance. Resource allocation and Task scheduling are some of the major problems associated with Load balancing. A simulator is used to model a cloud computing system where we can comprehend the problems identified with load balancing (effective resource utilization) and develop algorithms to solve the problem. The algorithm must reassign the total load to each resource node of the system as a whole. This paper presents answers for solving load related problems in distributed computing.**

***Index Terms*—cloud computing, load balancing, data transfer, load balancing algorithms.**

I. INTRODUCTION

In telecommunications, the cloud is a public space that is found between the endpoints of a transference. Information sent over a WAN enters the system from the start of the transmission line (by making use of a standard convention set) and after that reaches the cloud where the new information and the previously transmitted existing information share the memory space. The new information then leaves the cloud and is encapsulated, deciphered and transported by utilizing various strategies, while staying in the same form it was on entering the cloud. A system cloud is expected to guarantee that no two bundles follow the same path when information is sent over a packet switched network. The random area the information enters before it is reaches the end of the transmission line is known as the cloud.

The cloud empowers you to utilize stored information from anyplace on the planet at any time. A conventional PC setup requires you to be in the same area as your data storage device. Utilizing a cloud eliminates this prerequisite. The cloud supplier can possess the equipment and the product that is important to maintain their home or business applications and house them as well.

Cloud computing is characterized as conveyance of processing assets over the Internet. Instead of keeping information all on your hard drive, you can utilize a service over the Internet, at any area, to store your data. Cloud computing conveys infrastructure, software and platform, which are accessible as membership only services in a pay-as-you-go model to clients.

Load balancing is the procedure of enhancing execution of a parallel, conveyed framework by redistributing load among every one of the processors. In a distributed system of computing hosts, framework execution can critically rely on this compelling division of work.

II. METHODOLOGY

Depending on the cloud configuration, cloud computing has two environments - static and dynamic.

A. Static Environment

In a static environment, the resources are not flexible. The cloud provider has to install homogeneous resources. It requires prior knowledge of resources and user requirements such as maximum node size, processing capability, storage space and performance. The user requirements and allocated resources cannot change during run time. Algorithms that accomplish load balancing in static environment cannot adapt to the changes occurring during run time. Simulation in the static cloud is simple but unsuitable for heterogeneous cloud environment. Here are some load balancing algorithms that can be used in the static environment.

B.Static Algorithms

1. Token Routing Algorithm

This algorithm minimizes the system cost. This is accomplished by swapping the positions of tokens in the system [1]. However, in a scalable cloud system, agents cannot normally have enough data with respect to workload distribution because of tightening influences influencing correspondence. In this way, the workload distribution among specialists is arbitrary. This disadvantage can be dispensed with by the utilization of a heuristic methodology for token based load balancing. The algorithm gives quick and practical directing choices. It is redundant for the agents to have the complete learning of their worldwide state or of the neighbour's working burden. To choose where the token is to be passed, the agents construct their own knowledge base. This base is gotten from tokens obtained previously and there is no overhead in correspondence.

2. Round Robin Algorithm

In this algorithm, every process is split between all available processors [2]. This assignment is done by round robin ordering. The order for allocation of processes is maintained locally without having to depend on allocations by the other individual processors. In spite of the workload circulation among all processors being the same, the execution times for different procedures are distinctive. In this way, at any time, there might be some over-burdened nodes and still other unused nodes.

3. Randomized Algorithm

In this algorithm [2], we assign probability to each node. The order of allocating processes is maintained for all processors. This algorithm works better for processes that have the same load. Otherwise, performance problems may arise. The algorithm functions admirably when the Round Robin calculation requires high computation time for a process queue.

4. Opportunistic Load Balancing Algorithm

This algorithm [5] neglects workload of a VM at any moment. It attempts to keep each node busy. This algorithm deals with unexecuted tasks in random order for the currently available node quickly. Each task is assigned to a node randomly. It provides load balance schedule with poor results. The task processes slowly because it does not calculate the running time of the current node.

5. Min-Min Load Balancing Algorithm

The cloud manager identifies [6] the execution and completion time of unassigned tasks waiting in a queue. The parameters related to the job are known beforehand. In this type of algorithm, the cloud manager first deals with the jobs that have minimum execution time by assigning them to the processors as per their capability to complete the required job in specified completion time. The jobs that take maximum time to execute have to wait for an unspecified period of time. Once all the tasks in the processor have been assigned, the assigned tasks are updated in the processors and the assigned task is removed from the waiting queue. This algorithm gives a better outcome for jobs having low running time. The main drawback of the algorithm is that starvation can occur.

6. Max-Min Load Balancing Algorithm

Max-Min algorithm [7] works similar to Min-Min algorithm. However, after finding out the least time taken for running, the cloud manager deals with tasks executing in maximum time. The assigned task is expelled from the task list and the execution time for remaining errands is updated on the same processor. The algorithm follows a static approach so all the user requirements are known well in advance and the algorithm performs well. An enhanced version of max-min algorithm was proposed in [9]. Improvement in the efficiency of the algorithm is achieved by increasing the opportunity for concurrent execution of tasks on resources.

C. Dynamic Environment

In a dynamic environment, resources are flexible. The cloud provider has to install heterogeneous resources. Here, the cloud cannot depend on prior knowledge because it depends on run-time statistics. The user's needs can change at execution-time. Algorithms that achieve load balancing in cloud systems in dynamic environments must adapt to changes in load during run-time. Simulation in dynamic environments is difficult but very adjustable with cloud computing.

D. Dynamic Algorithms

1. Central Queuing Algorithm

This algorithm deals with the rule of dynamic distribution [4]. Every task, on arrival, is inserted into the queue by a queue manager. When a request is received by the manager, the first task is removed and sent to the requesting process. If there are just unavailable tasks in the queue, the request is buffered till another task is accessible. However, if another task enters the queue while unanswered requests are still in the queue, the first such request is expelled from the queue and another task is given to the requesting process. At the point when the load falls below the threshold value, then the local manager asks the central load manager to send another action. The central manager then processes the request if and only if a ready action is found. If not found, the request is queued until another action reaches the queue.

2. Connection Mechanism Algorithm

This is an algorithm [3] that is based on least connection mechanism. It dynamically counts the connections required such that each server can get an accurate estimation of the load. The load balancer records connections for every server. The quantity of the connections increments when new connections are dispatched to the server and reduces when the connections complete or timeout happens.

3. Biased Random Sampling Load Balancing Algorithm

In this algorithm [11], the system is delineated as a virtual chart. Each server is taken as a vertex of the node and the in degree represents the accessible free resources that the nodes have. On the premise of the in degree, the load balancer assigns the task to the node. In the event that the nodes have no less than one in degree, the load balancer gives the job to that node. At the point when the employment is given to the node, in degree is diminished by one, and is increased again when the task is executed. Random Sampling techniques are utilized as parts of expansion and cancellation of procedures. The procedures are centralized by their threshold values, which show maximum distance from one node to the destination node. This separation is known as walk length. The neighbour of the chosen node is selected for traversal. In the wake of getting the request, the load balancer randomly chooses a node and considers the walk length for the node with the threshold value. In the event that the threshold value is not exactly equivalent to the present walk length, the occupation is executed at that hub. Something else, the walk length of the employment is expanded by one and another neighbour is randomly chosen. The execution is inversely proportional to the quantity of servers.

4. Ant Colony Optimization Based Load Balancing Algorithm

Aim of the algorithm [6] is to search for the best path between the source of food and the colony of ants on the basis of their behaviour. The algorithm aims to efficiently distribute workload among all the nodes. When a request is initialized, the ant moves toward the food source from the head node. Regional Load Balancing Node (RLBN) is chosen in Cloud Computing Service Provider (CCSP) as a head node. Ants keep records of every node they visit. The ant records the data for future decision making. The ant deposits pheromones during their movement for other ants to select the next node. The intensity of the pheromones can vary on the basis of certain factors like distance of food, quality of food etc. When the job ends successfully, the pheromones are updated. Each ant builds its own result set and the set is later consolidated into a complete solution. The ant continuously updates a single set rather than updating the colony's result set. By the ant pheromone trails, the solution set is continuously updated.

5. Honeybee Foraging Load Balancing Algorithm

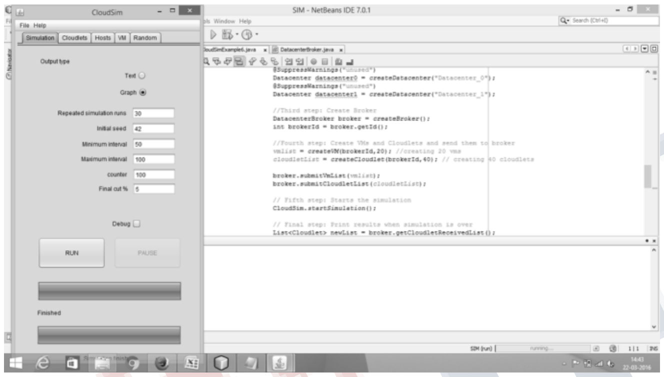
This algorithm [10] is a nature inspired decentralized load balancing technique across a heterogeneous virtual machine of cloud computing environment through a local server action and maximizes the throughput. The current workload of the VM is calculated and then it decides the condition of the VM states i.e. whether they are over loaded, under loaded or balanced. According to the current load of the VM, they are grouped. The priority of the task is taken into consideration after the task is removed from the overloaded VM for which tasks are waiting. Then the next task is scheduled to the lightly loaded VM. The earlier removed task is helpful for the finding the lightly loaded VM. These tasks are known as scout bee in the next step. Honey Bee Behaviour motivated Load balancing technique decreases the reaction time of the VM and also lessens the waiting time of the errand.

III. CLOUDSIM

Cloud-based applications have unmistakable structure, configuration, and circulation prerequisites. Assessing the execution of planning and assignment processes on the Cloud base for various purposes and servicing models under evolving load, vitality execution and system size is a problem. Test beds such as Amazon EC2 make the production of results hard to do, as it is hard to monitor resources and requirements simultaneously. The conditions prevalent in the internet environment are beyond the tester's control.

Another approach is to use simulation tools which allow for generating results of hypotheses before software development. In the cloud computing domain, access to the platform and utilization of resources incurs payment in real money. At the server side, a simulation environment permits assessment of various kinds of resources renting scenarios under varying loads. With such tools and studies it could assist the servers in providing cost optimization for that resource. When there are no such simulation platforms, clients and servers need to rely on theoretical assessments or on experimentation techniques that may result in poor performance [12]. To avoid the above problems, we use CloudSim. By using CloudSim, analysts and engineers can concentrate on configuration problems for the system that they need to research, on resources that can incur costs without bothering about low level elements identified with Cloud-based infrastructures and services.

IV. RESULTS AND OBSERVATIONS



Output type
Text ○
Graph ●
Repeated simulation runs 30
Initial seed 42
Minimum interval 50
Maximum interval 100
counter 100
Final cut % 5
Debug ☐

```
@SuppressWarnings("unused")
Datacenter datacenter0 = createDatacenter("Datacenter_0");
@SuppressWarnings("unused")
Datacenter datacenter1 = createDatacenter("Datacenter_1");

//Third step: Create Broker
DatacenterBroker broker = createBroker();
int brokerId = broker.getId();

//Fourth step: Create VMs and Cloudlets and send them to broker
vmlist = createVM(brokerId,20); //creating 20 vms
cloudletList = createCloudlet(brokerId,40); // creating 40 cloudlets

broker.submitVmList(vmlist);
broker.submitCloudletList(cloudletList);

// Fifth step: Starts the simulation
CloudSim.startSimulation();

// Final step: Print results when simulation is over
List<Cloudlet> newList = broker.getCloudletReceivedList();
```

Fig 1: Inputs given to Simulation

The inputs given to simulation, cloudlets, host, VMs and random parameters will be kept as the typical values set in CloudSim.

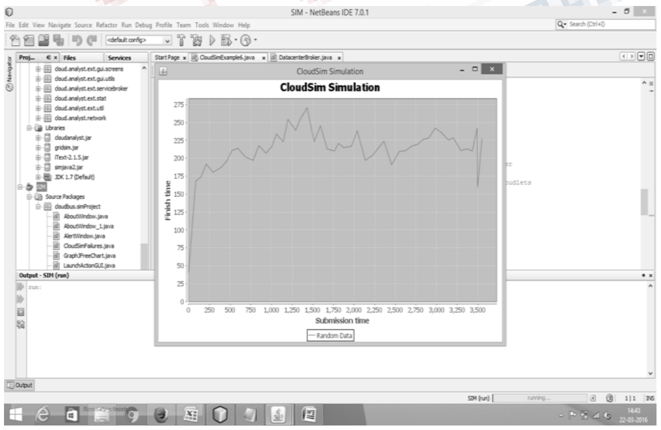
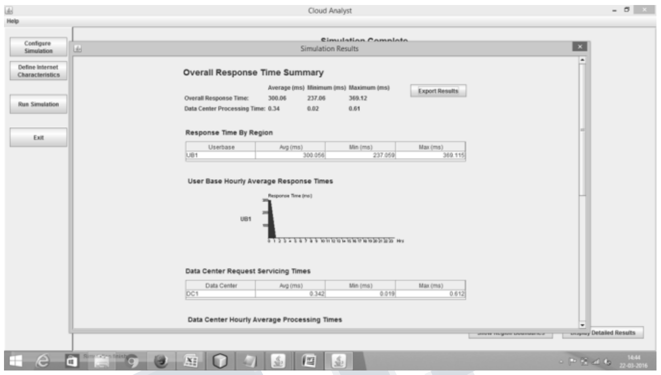


Fig 2: Graphical Representation of the Simulation

The output is depicted as a graph of Finish Time vs. Submission time. As submission time increases, finish time constantly stays within the same range, thereby causing the host to utilize the VM's execution potential to the best of its ability.



Overall Response Time Summary

	Average (ms)	Minimum (ms)	Maximum (ms)
Overall Response Time:	300.06	237.06	369.12
Data Center Processing Time:	0.34	0.02	0.61

Response Time By Region

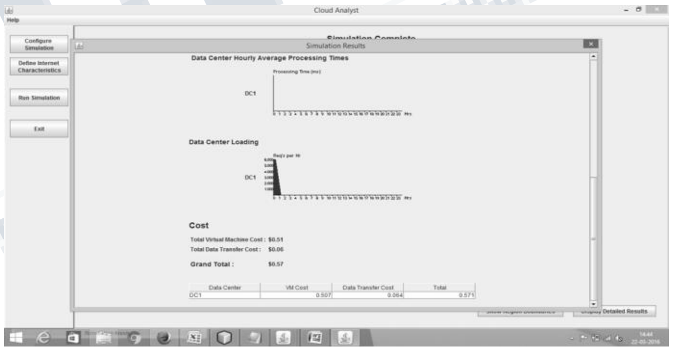
Userbase	Avg (ms)	Min (ms)	Max (ms)
UB1	300.056	237.059	369.115

Data Center Request Servicing Times

Data Center	Avg (ms)	Min (ms)	Max (ms)
DC1	0.342	0.019	0.612

Fig 3: Overall response times of data centres

We at first expect that the web application is deployed in one area, say Region 0. The overall response time and data processing times will be as appeared in the table i.e., response time by region.



Cost

Total Virtual Machine Cost : $0.51
Total Data Transfer Cost : $0.06
Grand Total : $0.57

Data Center	VM Cost	Data Transfer Cost	Total
DC1	0.507	0.064	0.571

Fig 4: Data Transfer and Virtual Machine cost

Practically, as the application develops, there is a need to deploy it in various regions. If we consider a second region, say, Region 1, we will observe that in spite of the fact that the total cost (virtual machine cost and data transfer cost) remains around same in both cases, the overall response time and data centre processing time turns out to be half when we utilize two data centres as opposed to one. Subsequently, to diminish the response time, it is ideal to utilize two data centres rather than one. Thus, we can exchange information faster.

REFERENCES

[1] Zenon Chaczko, Venkatesh Mahadevan, Shahrzad Aslanzadeh, Christopher Mcdermid, (2011), “Availability and Load Balancing in Cloud Computing”, International Conference on Computer and Software Modeling IPCSIT vol.14 IACSIT Press, Singapore 2011

[2] Zhong Xu, Rong Huang, (2009), “Performance Study of Load Balancing Algorithms in Distributed Web Server Systems”, CS213 Parallel and Distributed Processing Project Report.

[3] P. Warstein, H. Situ and Z. Huang, (2010), “Load balancing in a cluster computer”, In the proceedings of the seventh International Conference on Parallel and Distributed Computing, Applications and Technologies, IEEE

[4] M. Beltran, A. Guzman and J. L. Bosque, (2011), “Dealing with heterogeneity in clusters”, In the proceedings of the Fifth International Symposium on Parallel and Distributed Computing, ISPDC.

[5] Ratan Mishra and Anant Jaiswal, ― Ant Colony Optimization: A solution of Load Balancing in Cloud, International Journal of Web & Semantic Technology (IJWesT), April 2012

[6] Che-Lun Hung, Hsiao-hsi Wang and Yu-Chen Hu, ― Efficient Load Balancing Algorithm for Cloud Computing Network, IEEE Vol. 9, pp: 70-78, 2012

[7] 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 Vol- 20 No.2, 2011

[8] Karanpreet Kaur, Ashima Narang, Kuldeep Kaur, "Load Balancing Techniques of Cloud Computing", International Journal of Mathematics and Computer Research, April 2013.

[9] Upendra Bhoi, Purvi N. Ramanuj ― Enhanced Max-min Task Scheduling Algorithm in Cloud Computing, International Journal of Application or Innovation in Engineering & Management (IJAIEM), Volume 2, Issue 4, April 2013.

[10] Dhinesh B. L. D., P. V. Krishna ― Honey bee behaviour inspired load balancing of tasks in cloud computing environments, in proc. Applied Soft Computing, volume 13, Issue 5, May 2013, Pages 2292-2303.

[11] Rahmeh O. A., Johnson P., Taleb-Bendiab A., A Dynamic Biased Random Sampling Scheme for scalable and reliable Grid Networks‖, The INFOCOMP Journal of Computer Science, vol. 7, 1-10

[12] Paper on CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services Rodrigo N. Calheiros, Rajiv Ranjan1, César A. F. De Rose, and Rajkumar Buyya