Load Balancing in Cloud Computing Empowered with Dynamic Divisible Load Scheduling Method

Sohaib Ahmad[1], Maryam Nafees[2], Ayesha Atta[3]

[1,2,3] Department of Computer Science, GC University Lahore

Email: mughal.sohaib9014@gmail.com

(Received: 08 August 2021; Accepted:03 Sep 2021; Issue Published: 12 Sep 2021)

ABSTRACT

The need to process and dealing with a vast amount of data is increasing with the developing technology. One of the leading promising technology is Cloud Computing, enabling one to accomplish desired goals, leading to performance enhancement. Cloud Computing comes into play with the debate on the growing requirements of data capabilities and storage capacities. Not every organization has the financial resources, infrastructure & human capital, but Cloud Computing offers an affordable infrastructure based on availability, scalability, and cost-efficiency. The Cloud can provide services to clients on-demand, making it the most adapted system for virtual storage, but still, it has some issues not adequately addressed and resolved. One of those issues is that load balancing is a primary challenge, and it is required to balance the traffic on every peer adequately rather than overloading an individual node. This paper provides an intelligent workload management algorithm, which systematically balances traffic and homogeneously allocates the load on every node & prevents overloading, and increases the response time for maximum performance enhancement.

KEYWORDS: Load balancing, Load scheduling, Dynamic methods, Network topologies, Divisible Scheduling.

1. INTRODUCTION

Cloud computing is an on-demand service in which shared resources, information, software and other devices are provided according to the client's requirement at a specific time [6]. Cloud computing provides different services such as IAAS (Infrastructure as a Service), PAAS (Platform as a Service), SAAS (Software as a Service), for which users pay a different amount as per their requirement [13]. The demand for cloud computing and its services increased with the developing world. Cloud computing's primary purpose is to share resources and provide personal storage over the internet with minimum effort. The main advantages of cloud computing are low cost, improved performance, infinite storage space Etc [5].

The major challenge which we are going to discuss in this paper is "Load balancing." As the number of users increases day-by-day in cloud environments, load balancing has become a challenging problem for cloud service providers.

Load balancing means distributing the load on all the servers and nodes equally. This problem arises when the users' requests become high, and some servers become overload, and some underload. To overcome this problem and divide the requests equally on the servers according to the algorithms' capacity. This paper considers and reviews some of the algorithms defined by other authors in their research papers.

The main objective is to reduce this load, minimize the makes pan time and maximize resource utilization. No request has to wait for an extended period and be stuck until the server gets free.

In this paper, the algorithm we propose is "Divisible Load Scheduling" in this, we divide our tasks into subtasks and assign those subtasks to different nodes. In this way, we can reduce the processing time of the job. This algorithm works as a post-order traversal of a tree. The subtasks distributed network used different topologies (tree, star, ring, etc.).

All these topologies are connected through gateways and communicate through them. Nowadays, cloud computing is the heart favourite topic of many researchers, and It will become more prevalent in the coming years as the reach of the internet increases day by day [5].

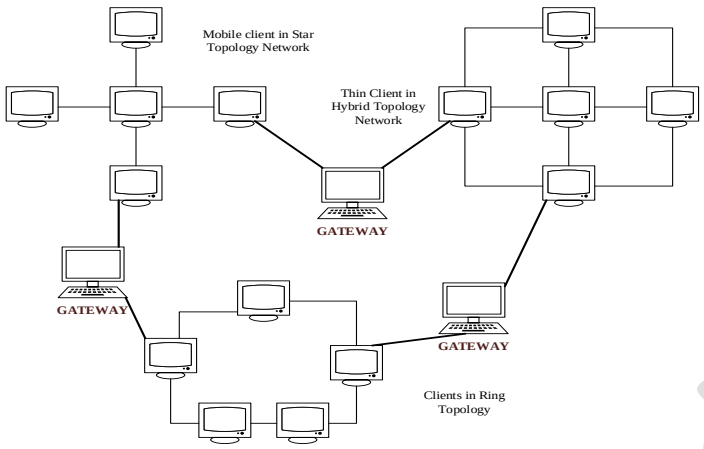

Figure1: Different topologies existing within a cloud

Nodes have different data centers that keep track of all the parameters and performance. The central computer server divides the task into subtasks and assigns them to the corresponding slave computers. We use different algorithms in these topologies to give the task according to the capacities and performances. After completing the tasks, the master computer collects the results and completion times of slaves and calculates them accordingly. In this way, we keep track of and reduce our load balancing issue.

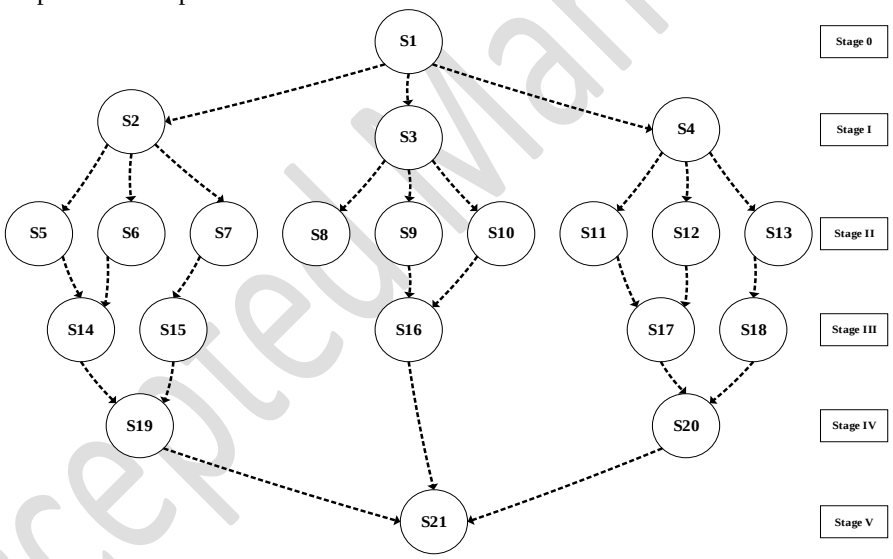

Figure 2: State Diagram of the proposed work

In the initial phase, the execution starts with S0 and parallel S2, S3 & S4, respectively. The task is alienated into segments. S14, S16 & S17 are reliant on S5, S6 & S9, S10 & S11, S12 respectively. S15 and S18 are dependent on S7 & S13, correspondingly. The final results are assessed and collected by the last node, S21.

2. LITERATURE REVIEW

The algorithms of load balancing problems discussed in some research papers are listed below. In [1], they discussed the load balancing problem. They proposed an algorithm whose fundamental concern is dividing the load between all servers equally, not overburden a single server with all requests. This algorithm put a counter at each node (server) to check the request's count and track which node has more requests. This algorithm works by checking all the counter values at each node, the node with the minimum number of submissions is selected and allot the application request to that server(node). Then allocate the client request to its data centre. In this way, the requests are assigned at nodes to do load balancing and increase work efficiency. In the last, we increment the counter

of the designated node (server) by 1. Furthermore, after completing the application request, decrement the counter of the server by 1. In this way, we keep the count and manages requests equally.

In [2], the author minimizes the tasks' makespan time and maximizes resource utilization. He proposed another algorithm, "Task migration," for the load balancing problem in cloud computing. This algorithm takes various virtual machines and different arrays for these VM's to calculate the underloaded, overloaded, and balanced VM's. Firstly, they arranged the number of tasks in descending order and VM's according to their power. They then assigned the functions to VM's on FCFS (first come, first serve) basis. Find the capacity and load of the VM's. Suppose the load is less than the ability. In that case, we apply the load balancing operation by checking which machine is overloaded, underloaded, or balanced and keeping a record of all the machine tasks. Arrange the overloaded machine tasks id in descending order and underloaded in ascending order. Then move the jobs from overloaded to underloaded until the underloaded ones reach the threshold value. In this way, we can maintain a balance between virtual machines.

In [3] for the load balancing problem, the author proposed an algorithm inspired by the "firefly algorithm." The characteristics of the firefly approach are. All fireflies are attracted towards each other no matter what their attributes are (unisex nature). Initially, they calculate The light intensities of all fireflies, and the firefly with less power moves towards the brightest one. Brightness has an inverse relation with the distance; an objective function also determines the brightness. The algorithm proposed by the author in the paper is similar to this. In his algorithm, first, the population is formed. The requests that come on a specific server are assigned to a free node. All these make a scheduling list, which is known as the population. Then find the scheduling index; as the firefly algorithm says, there is an attraction between fireflies.

Similarly, there is an attraction between the nodes and the requests, which depend on the proposed system's attributes. Find the interest by given formulas, find the scheduling index, and update the list, so the queue with a high scheduling index is on top. In the last, find the node with minimum load and transfer the task to that node.

In [4], the author uses the " Genetic algorithm " to reduce the load balance problem and decreases the makespan time using the "Genetic algorithm." This algorithm works on fixed lengths. Therefore, all the solutions are converted in binary form to initialize a population and find the paper's equation's fitness value. After this, the algorithm moves to the selection step, where we remove the highest fitness value chromosomes and deal with the lowest value to make a mating pool. Then do the crossover by randomly selecting a point to form an offspring. After crossover, mutates them using probability 0.5. Take these new offspring as a unique population and do iterations by taking them. Test that the obtained solution is optimal or not. If not, then continue doing iterations. This algorithm uses the CloudAnalyst tool, which is GUI-based and shows everything in a graph. At the end of this paper, the author does simulation analysis and gives us the result, which shows that this algorithm works better than other techniques like FCFS, RR, and SHC.

In [5], the author discusses the load balancing algorithm with an efficient version of the "throttled algorithm." The author tries to combine three algorithms into a single algorithm to increase efficiency and reduce load balancing. In his proposed algorithm, he used a hash map that keeps track of all the virtual machines. The hash map contains the current status and the expected response time. When the request comes to the data centre, it is handed to the throttled balancer, whose duty is to transfer it to a suitable VM. So, he checks the hashmap, and if it found the VM with less load and less response time, then throttled send the request to that VM, and updated the hashmap. If throttled does not find the VM, then he sends the message to the datacenter of unavailability. The proposal has to wait until the VM gets free. When VM becomes

free, the data centre sends a message to throttle, and then again, the balancer finds the suitable VM for the request. In this way, this algorithm works. This algorithm works far better than others.

In [6], the paper discusses load balancing algorithms & projects the idea of sub-division. Conventional tasks are divided into sub-tasks. Each sub-task is assigned an incredibly particular job, e.g., A task "T" is divided into T1, T2, T3…Tn (where n is a possible natural number of subsets).

Some of these tasks are sequentially completed rest are executed parallelly, which reduces the execution time of a job comparatively, as each subtask's implementation period is reduced. The system gets a massive performance boost.

In [7], the research paper debates the use of various algorithms in cloud computing. It provides the advantages and disadvantages of the algorithms. It also delivers a comparison of each one with others on immobile parameters and attributes. Each time Particle Swarm Optimization (PSO) algorithm outnumbers others. Bird & Fish flocks are the main inspiration for this algorithm. The swarm population works by the protocol of divide and conquer, and they form groups with a particular task and scatter around in search of their desired goal. They rejoin when they have achieved their target.

Similarly, In PSO, each task has several particles, and they move in random directions with vector velocity to find the particular space. Each particle adjusts its path based on Velocity, Pbest, Gbest. Performance is measured on the attributes of the fitness function.

In [8], A stochastic hill-climbing approach is used based on Round Robin Theorem and FCFS (First Come First Serve) to balance load distribution on the Cloud. The proposed algorithm is non-distributed (Centralized). With the following tactics, the performance boost is significant, and outcomes are reasonably encouraging. Still, the technique needs other software for improvements—an experiment conducted via Coanalytic with hypothetic generated configuration. The settings were created, keeping in mind random e-auction & social sites, e.g., Facebook, Google+, etc.

In [9], an algorithm is proposed that minimizes the server and request load concerning priorities. Cloud manager takes the request from the user and stores them in a stack, and then prioritizes. It develops a request table based on time allocation, task priority, job size & resources. The rest of the scheduling and resource allocation is executed with the Bee-Colony Algorithm. This algorithm helps to balance incoming traffic & QoS for Cloud Environment. The algorithm also results in a less execution period. The algorithm can further be improved with the addition of VM, specifically for comprehensive simulation.

In [10] this paper, a heuristic algorithm grounded Ant-based control system is used to resolve load management. Every node in the Cloud has a configuration dependent on Accommodation Capacities, Destination Probability & Pheromone table. Ant is thrown from a node to an arbitrary destination. Incoming ants apprise the pheromone table according to the entries. E.g., an ant will travel from source to destination and inform the corresponding entities. The model is a symmetric algorithm system for an asymmetric network that may differ. Thus, it is only efficient for symmetric routing networks. This algorithm can optimize the maximum performance and minimize declining parameters efficiency as CPU Circulation & Memory volume and network load for the Cloud. The paper does not discuss fault tolerance, which can be explored in the future.

In [11], The Author defines the New Scheduling Algorithm for Load-balancing "Max-Min Algorithm." This algorithm's primary purpose is to reduce the turnaround time of all the incoming requests and increase the VM's processing time by providing the best possible schedule for the tasks. In this algorithm, the tasks Ti are provided to Resources Rj. For all the tasks, Calculate the completion time concerning the Execution time of the task on Resource Rj. The tasks with the highest completion time are assigned to the slower available machine. This algorithm gives higher priority to the tasks with maximum execution time. Huge tasks have higher priority. The hugs tasks are assigned to slower machines, and smaller tasks are assigned to the fastest machines to increase the tasks' average execution time.

In [12], the author defines the "Fuzzy logic-based Load Balancing" technique. The paper focuses on the two main attributes, "processor speed" and "load of VM." The article presents the Fuzzy logic approach as an improved version of Round Robin to upgrade the utilization of the resources and accessibility of the cloud environment. The projected algorithmic rule starts with the request an association with Resource. It tests for accessibility of Resource. Calculate the association strength if the Resource is found. It then chooses the association employed to access The Resource as per processor speed and load in the virtual machine by applying mathematical logic (fuzzy logic).

In [13], the paper proposed the Load balancing algorithm to increase Cloud computing's productiveness using a priority queue. The purpose of the priority queue Is to prioritize their affluent users. If the user requests services from the server. Furthermore, its waiting queue is full. Later, it is transferred to the priority queue that the Request Manager handles. And from there, it is sent to the required and available server according to the percentages. This technique is beneficial for cloud providers who want to supply higher services to their affluent users.

In [14], the author proposed the "Enhanced Min-Min Algorithm". This algorithm uses the simple Min-Min algorithm along with the rescheduling technique. This

algorithm has two phases. It works as a simple Min-Min algorithm in one phase, where the completion time of tasks is calculated and minimum completion time tasks are assigned to the slowest machine. It does not provide the appropriate results sometimes. That is why the author defines the second phase, where the rescheduling of tasks occurs, and the tasks with maximum completion time are assigned to their appropriate resources. The paper results proved that this Enhanced version works better and speeds up the processing time and utilization of resources compared to the LBMM.

In [15], The Author defines the main objective of Cloud Computing to provide its services effectively to the clients. He proposed an algorithm for load balancing called "Migration of Virtual Machines." This algorithm migrates the VM if the resource utilization becomes maximum (above 90%) towards the Resource having minimum CPU utilization. There are chances of having minimum migrations and maximum utilization of resources by using this algorithm.

In [16], the author proposed a load balancing solution on multi-core processing, which prevents shared memory from using other multiprocessing load balancing solutions. The solution maintains a lock on the user session. The solution requires a modified Linux kernel. The solution improves multi-core environment performance while handling multi load-balancing processes in a single load balancer.

In [17] author introduced a load balancing policy for web servers. The proposed model was intended to be used all over the world. The model reduces the number of requests to the closest

remote server, which eventually helps reducing response time with overloading web servers. Middleware refers to implement these protocols. It is also a handy tool in enduring web-server overload.

In [18] Author investigates a self-aggregation algorithm that connects similar services by local re-wiring, which optimizes job scheduling. It accomplishes local balancing through mapping tasks throughout local server actions. The algorithm boosted the system performance significantly, but throughput decreased with the increase in the system size. With this policy, the system becomes more diverse. Thus the protocol is only suitable for diverse population servers.

In [19], the author proposed VectorDot, a load balancing algorithm that deals with the hierarhical complexity of the servers' data center & resource allocation, and storage that integrated servers and storage virtualization technologies. The principle of the dot product is the primary factor in determining item requirements and system workload on servers, storage and switches nodes.

In [20], the authors proposed a load balancing algorithm for Virtual Machines called Central Load Balancing Policy (CLBVM). It divides workload evenly across the distributed VMs or cloud environment. Except for the fault-tolerant system, this policy overall increases system performance.

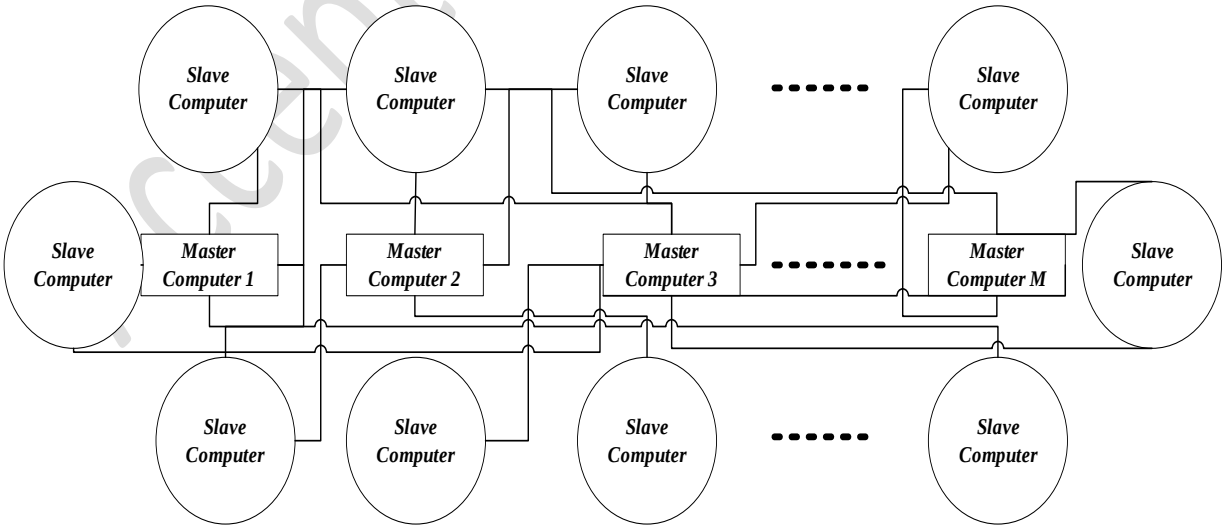

Figure 3: M is the number of total master computers on the system, each connected with Several slave computers in a star topology tree network

B_{m_i} Load assigned by master computer "m" to slave "i."

a_{m_i} We designed a constant contrarywise proportional to the measured speed of slave "i" of the Cloud.

b_{m_i} We designed a constant inversely proportional to the communication speed of connection "i" of the Cloud.

T_{ms} The time is taken by the"i^{th}"slave in the entire load measurement, momently when the constant speed of the slave is 1. It is commonly known as the Measurement of Intensity constant.

T_{cm} Time to transmit all the measured load over the connection when the load assigned to the slave is 1—generally known as Communication intensity of constant.

T_{m_i} The total time a slave takes from the beginning of the scheduling time, i.e., t = 0. The time in which the slave completes its task reports back. This time includes execution, waiting, reporting, and transmitting time.

T_{fm} is when the last slave completes the task and reports back to the master "m."

$$T_{fm} = max\ (T_{m_1}, T_{m_2}, T_{m_3}, \dots\dots\dots\dots\dots, T_{m_N})$$

T_f is the time at preceding master node receives the outcomes from the slaves.

$$T_f = max\ (T_{f_1}, T_{f_2}, T_{f_3}, \dots\dots\dots\dots\dots, T_{f_N})$$

4. PROPOSED ALGORITHM FOR MEASUREMENT AND REPORTING TIME

Initially, at the time, i.e. (t = 0), all the slaves were idle. Moreover, formerly master computers start to communicate with their first slave computer as the task arrives in the Cloud. By t = t_1 time, the master computer dispatches the instruction to corresponding slaves. Furthermore, the slave computers receive their instructions, as shown in fig 4. The assumption is that the calculations are made. Only one slave returns the call to the root master computer. The slaves will receive their load subsequently, and computation will begin when all the slave have acknowledged their load share.

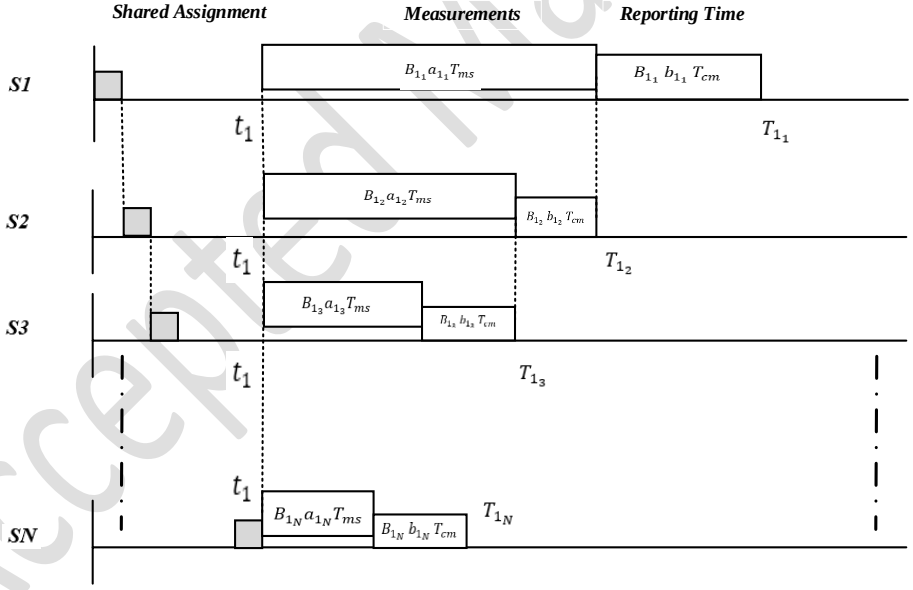

Figure 4: Reporting time graph for a single tree network with a master computer and "N" number of salves executing subsequentially

Considering the first root, master computer along with its slaves. By the definition of T_{m_i}, we can address as:

$$T_{1_1} = t_1 + B_{1_1}\, a_{1_1}\, T_{ms} + B_{1_1}\, b_{1_1}\, T_{cm} \quad (1)$$

$$T_{1_2} = t_1 + B_{1_2}\, a_{1_2}\, T_{ms} + B_{1_2}\, b_{1_2}\, T_{cm} \quad (2)$$

.

.

.

$$T_{1_N} = t_1 + B_{1_N}\, a_{1_N}\, T_{ms} + B_{1_N}\, b_{1_N}\, T_{cm} \quad (3)$$

By the equations, we can assume that the total measurement load originated at every master computer with the efficiency of normalizing to unit load. Thus, each master computer we handle the unit load as $\frac{1}{M}$ Load. So,

$$B_{1_1} + B_{1_2} + B_{1_3} + \cdots + B_{1_{N-1}} + B_{1_N} = \frac{1}{M} \quad (4)$$

As from the diagram, we can relate that:

$$B_{1_1}\, a_{1_1}\, T_{ms} = B_{1_2}\, a_{1_2}\, T_{ms} + B_{1_2}\, b_{1_2}\, T_{cm} \quad (5)$$

$$B_{1_2}\, a_{1_2}\, T_{ms} = B_{1_3}\, a_{1_3}\, T_{ms} + B_{1_3}\, b_{1_3}\, T_{cm} \quad (6)$$

.
.
.

$$B_{1_{N-2}}\, a_{1_{N-2}}\, T_{ms} = B_{1_{N-1}}\, a_{1_{N-1}}\, T_{ms} + B_{1_{N-1}}\, b_{1_{N-1}}\, T_{cm} \quad (7)$$

$$B_{1_{N-1}}\, a_{1_{N-1}}\, T_{ms} = B_{1_N}\, a_{1_N}\, T_{ms} + B_{1_N}\, b_{1_N}\, T_{cm} \quad (8)$$

By this evaluation, we get a general expression:

$$\boldsymbol{B_{1_i} = s_{1_i} B_{1_{i-1}}} \quad (9)$$

We can define"s_{1_i}"as:

$$s_{1_i} = \frac{a_{1_{i-1}} T_{ms}}{a_{1_i} T_{ms} + b_{1_i} T_{ms}}$$

And I can be related as, *i = 2, 3….N.*

the above recursive equation for B_{1_i} can be created as:

$$B_{1_i} = \prod_{j=2}^{i} s_{1_j}\, B_{1_1} \quad (10)$$

By using the above equation, we can relate to B_{1_1}, as

$$B_{1_1} + \sum_{i=2}^{N} \prod_{j=2}^{i} s_{1_j}\, B_{1_1} = \frac{1}{M} \quad (11)$$

In other ways:

$$B_{1_1} = \frac{1}{M(1+\sum_{i=2}^{N} \prod_{j=2}^{i} s_{1_j}} \quad (12)$$

By putting in the equation-(10)

$$B_{1_1} = \frac{\prod_{j=2}^{i} s_{1_j}}{M(1+\sum_{i=2}^{N} \prod_{j=2}^{i} s_{1_j})}$$

Where *i = 2,3,4,……N.*

Now the minimum measuring & reporting time on the cloud network shall be calculated by:

$$T_{f_1} = t1 + \frac{a_{1_1} T_{ms} + B_{1_1} b_{1_1} T_{cm}}{M(1+\sum_{i=2}^{N} \prod_{j=2}^{i} s_{1_j})} \quad (13)$$

Similarly, a generalized equation for master computer "r" can is derived as:

$$T_{f_r} = t1 + \frac{a_{r_1} T_{ms} + B_{1_1} b_{r_1} T_{cm}}{M(1+\sum_{i=2}^{N} \prod_{j=2}^{i} s_{r_j})} \quad (14)$$

The *case I*: if the network has the same measurement size & connection speed).

If this case, we can relate as:

$$s_{1_1} = s_{1_2} = s_{1_3} = \cdots = \cdots = s_1$$
$$a_{1_1} = a_{1_2} = a_{1_3} = \cdots = \cdots = a_1$$
$$b_{1_1} = b_{1_2} = b_{1_3} = \cdots = \cdots = b_1$$

By eq-(5):

$$B_{1_1}\,(1 + s_1 + s_1^2 + \cdots + s_1^{n-1}) = \frac{1}{M}$$

Here"s_1"is:

$$s_1 = \frac{a_1 T_{ms}}{a_1 T_{ms} + b_1 T_{cm}} \quad (15)$$

Abridging the overhead equation:

$$B_{1_1} = \frac{1-s_1}{M(1-s_1^N)} \quad (16)$$

By this equation, the first master computer will contract the measured amount of data from the slaves *N-1* by using the value of B_{1_1}.

$$B_{1_i} = B_{1_1} s_1^{i-1} \quad (17)$$

Here and now, the least measuring and reporting period of a homogenous system will be prearranged as:

$$T_{f_1} = t1 + \frac{(1-s_1)(a_1 T_{ms} + b_1 T_{cm})}{M(1-s_1^N)} \quad (18)$$

When "N" approaches infinity, the measurement & reporting period of the network method become $t1 + b_1 T_{cm}/M$. When the number of corresponding slaves of a master approach infinity, reporting time exceeds the measurement period.

5. PERFORMANCE ANALYSIS

In figure 5, measurement & report time are graphically articulated against the homogeneous slave's equivalent to their master, in the case of communication bandwidth "b" varying within the range of 0 and 1, with a variable interval & measurement speed is static at 1.5. for T_cm=1 & T_ms=1.

We can conclude from the graph that faster communication speed outcomes in smaller report period and report time catches up with

communication speed after a certain number of slaves. The number of master computers brings a neglectable effect on the Cloud's performance compared to a single master computer cloud.

In figure 6, the inverse measuring speed "a" varies in the range of 1 and 2 with a variable interval & the inverse speed link static at 0.2. The outcome authorizes the measurement time b_1 T_cm That is, in particular case 0.2, while "N" methods to infinity.

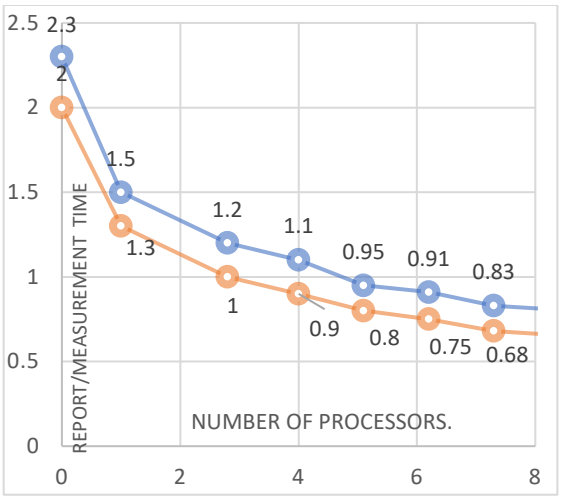

Figure 5: This graph indicated the report time with admiration to the particular slaves working corresponded by the master computer that acquires the task with variable link speed "b" and single tree network.

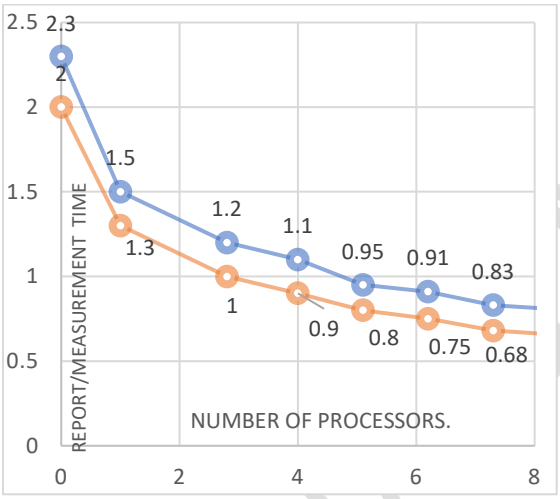

Figure 6: This graph indicated the report time with admiration to the particular slaves working corresponded by the master computer that acquires the task with variable link speed "a" and single tree network.

6. CONCLUSION

Till now, we have discussed cloud computing, its concepts and load balancing. Load balancing can become the bottleneck for performance. We have discussed several possible optimized algorithms for the solution of load balancing. This paper has described the complications, drawbacks & performance deteriorating factors of a cloud that suffers from a heavy workload. We have proposed a dynamic divisible load theory across nodes to balance the load and applied it to the Cloud for performance and productivity boost. This approach includes various topologies used across multiple thin and thick clients connected via network topology. Nodes have gateways, master computer, and their particular slave nodes. This paper gives various notation and measurement parameters used in the examination and the load balancing analysis of the Cloud. Our study shows that implementing our proposed work and the Cloud gets a significant amount of recital enhancement.

7. FUTURE WORK

Cloud Computing is a diverse field of concept and research. Load balancing plays a vital role in implementation of cloud computing. There is a tremendous extent of progress in this area. We have only discussed dynamic scheduling algorithms in cloud implementation. The given algorithm can also improve over time with the development of some parameters.

REFERENCES

[1] Haryani, N. and Jagli, D., 2014. Dynamic method for load balancing in cloud computing. *IOSR Journal of Computer Engineering*, vol 16, issue 4, pp.23-28, 2014.

[2] Kumar, Mohit, and S. C. Sharma. "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing." *Procedia computer science 115*, pp.322-329, 2017.

[3] P. Florence and V. Shanthi, " A Load Balancing Model Using Firefly Algorithm In Cloud Computing," *Journal of Computer Science*, vol. 10, no. 7, pp. 1156-1165, 2014.

[4] K. Dasgupta, B. Mandal, P. Dutta, J. Mandal and S. Dam, "A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing", *Procedia Technology*, vol. 10, pp. 340-347, 2013.

[5] Patel, D. and Rajawat, A.S.," Efficient throttled load balancing algorithm in cloud environment". *International Journal of Modern Trends in Engineering and Research*, Vol 2, Issue 03, pp.463-480, 2015.

[6] Ram Prasad Padhy (107CS046), P Goutam Prasad Rao (107CS039), "Load Balancing In Cloud Computing System", *Department of Computer Science and Engineering National Institute of Technology, Rourkela Rourkela-769 008*, Orissa, India May 2011.

[7] Bharti, Mohali, Punjab, India. "*International Journal of Computer Applications*" Volume 92 – No.9, pp. 0975 – 8887, 2014

[8] Mondal, Brototi, Kousik Dasgupta, and Paramartha Dutta. "Load balancing in cloud computing using stochastic hill climbing-a soft computing approach." *Procedia Technology 4*, pp. 783-789, 2012.

[9] Soni, Ashish & Vishwakarma, Gagan & Jain, Yogendra. "A Bee Colony based Multi-Objective Load Balancing Technique for Cloud Computing Environment". *International Journal of Computer Applications.114*.,pp. 19-25, 10.5120/19967-1825, 2015.

[10] Ratan Mishra1 and Anant Jaiswal, "Ant Colony Optimization: A Solution of Load balancing in Cloud.*" International Journal of Web & Semantic Technology (IJWesT)*, Vol.3, No.2, April 2012, DOI:10.5121/ijwest.2012.320333

[11] Mao Y., Chen X., Li X. |Max–Min Task Scheduling Algorithm for Load Balance in Cloud Computing". In: Patnaik S., Li X. (eds) Proceedings of International Conference on Computer Science and Information Technology. *Advances in Intelligent Systems and Computing*, vol 255. Springer, New Delhi, 2014.

[12] S. Sethi, "Efficient Load Balancing in Cloud Computing using Fuzzy Logic", *IOSR Journal of Engineering*, vol. 02, no. 07, pp. 65-71, 2012.

[13] Rahul Rathore, Bhumika Gupta, Vaibhav Sharma, Kamal Kumar Gola, "*A New Approach For Load Balancing In Cloud Computing*". ISSN 2277-3061, 2014.

[14] Patel, G., Mehta, R., & Bhoi, U. Enhanced Load Balanced Min-min Algorithm for Static Meta Task Scheduling in Cloud Computing. *Procedia Computer Science*, vol 57, pp. 545–553, 2015.

[15] Razali, Rabiatul & ab rahman, Ruhani & Zaini, Norliza & Samad, Mustaffa. Virtual machine migration implementation in load balancing for Cloud computing. pp. 1-4. 10.1109/ICIAS.2014.6869540, 2014.

[16] Liu Xi., Pan Lei., Wang Chong-Jun. and Xie Jun-Yuan. *3rd International Workshop on Intelligent Systems and Applications*, 2011.

[17] Nakai A.M., Madeira E. and Buzato L.E. *5th Latin- American Symposium on Dependable Computing*, pp. 156-165, 2011.

[18] Randles M., Lamb D. and Taleb-Bendiab A. 24th International Conference on Advanced *Information Networking and Applications Workshops*, pp.551-556, 2010.

[19] Singh, Aameek, Madhukar Korupolu, and Dushmanta Mohapatra. "Server-storage virtualization: integration and load balancing in data centers." In SC'08: *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pp. 1-12. IEEE, 2008.

[20] Bhadani, Abhay, and Sanjay Chaudhary. "Performance evaluation of web servers using central load balancing policy over virtual machines on cloud." *In Proceedings of the Third Annual ACM Bangalore Conference*, pp. 1-4. 2010.

.