2023 Annual International Conference on Emerging Research Areas: International Conference on Intelligent Systems (AICERA/ICIS) | 979-8-3503-0345-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/AICERA/ICIS59538.2023.10420318

An Efficient Virtual Decentralized Cloud Load Balancing(VDCLB)

1st Aditya Gandhi
Computer Science and Engineering
Graphic Era Deemed to be University
Dehradun, India
gandhi.aditya11@gmail.com

2nd Sonali Singh
Computer Science and Engineering
Graphic Era Deemed to be University
Dehradun, India
sonalisingh1192003@gmail.com

3rd Nripendra Verma
Computer Science and Engineering
Graphic Era Deemed to be University
Dehradun, India
nripendraverma.7224@gmail.com

4th Umang Garg
Computer Science and Engineering
Graphic Era Hill University
Dehradun, India
umangarg@gmail.com

5th Ashish Garg
Computer Science and Engineering
Graphic Era Deemed to be University
Dehradun, India
ashish.garg@geu.ac.in

Abstract— **Through *numerous data centers located all over the world, cloud computing offers a variety of services to customers throughout the globe, including data storage and data abstraction. The right technique to managing the incoming and outgoing data is necessary for this kind of large-scale data refining. There are two methods to balance the load: physically and virtually. We suggest a decentralized approach in this study for effective load balancing at the data centers. This work is organized by modules that discuss various cases. The idea of a load vector, which is responsible for storing all the addresses of nodes located there, is covered in module I. The procedure for controlling the workload at the nodes is explained in module II along with the algorithm. Module III assigns the incoming requests to the appropriate.***

Keywords—Cloud Computing, Virtual Decentralized Load Balancing.

I. INTRODUCTION

As is well known, a lot of people throughout the world are interested in cloud computing, which necessitates effective data management on a huge scale. Data management via networks has developed into a laborious task that takes a lot of time, manpower, and cost.

Load Balancers balance the load at the cloud data centers. To preserve server speed and usage while obtaining a better and more effective solution that meets user needs, load balancing is crucial. In order to ensure the appropriate operation and performance of the cloud and the pleasure of the user, load balancing is a method in which the load is uniformly distributed among the many nodes present at the data centers. There are thousands of requests at any given time that must be appropriately and promptly balanced. Also, a significant problem that may be solved by correctly managing the resources is cloud scalability.

The two different methods of load balancing are centralized and decentralized. With the aid of a central load balancer, centralized load balancing is carried out. The load balancer decides when and among which servers the load should be divided in order to maintain the right flow of data in and out. A decentralized load balancer, on the other hand, is a collection of many load balancers that internally distributes the workload to the server's nodes without the need for a centralized load balancer.

Various methods for decentralized load balancing are currently available to balance the load on cloud platforms. Some of the proposed algorithms are: Round Robin Algorithm, Virtual Machine Migration Approach, Opportunistic Algorithm and many more.

There are a lot of physical nodes in large-scale data centers. The above mentioned algorithms are the most common and widely used algorithms since they provide fast working and good effectiveness while dealing with large number of processes at cloud server center.

A revolutionary concept made possible by virtualization technology allows for the creation of unique virtual environments on top of real infrastructure. With the capacity to consolidate several virtual computers on a single physical node, virtualization techniques offer a lot of flexibility. This allowed for the resizing of the resource capacity allotted to various virtual machines as well as the on-demand migration of virtual computers between numerous physical nodes for a variety of goals. The idea of live or seamless transfer of virtual machines, which involves extremely brief downtimes ranging from tens of milliseconds to a second, has recently been implemented by most modern virtualization technologies products. As a result, the migration of virtual computers has become a promising method for resource management algorithms to use.

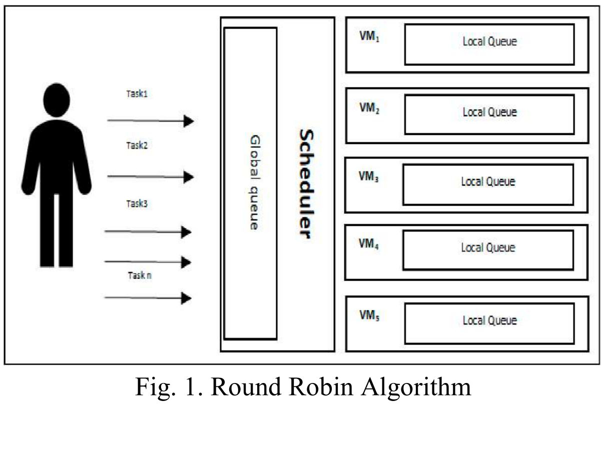


Fig. 1. Round Robin Algorithm

The [6] large-scale data center that serves as the focus of this paper's objective cloud environment often has a substantial number of physical nodes (PNs). Figure 1 depicts the high-level system architecture. Virtual machines (VMs) are frequently utilized to host numerous third-party applications on the real infrastructure. In response to incoming workloads, many VMs can be dynamically started or halted on a physical node, sharing the resources from the same physical devices. These VMs offer usability and flexibility for cloud end users by allowing applications based on many operating system environments to operate on a single physical node. At the top level, applications running on several virtual machines located on the underlying physical infrastructure provide services to end users.

Since the incoming workload changes greatly, each VM's resource requirements will also vary greatly. VMs can be dynamically relocated among various physical nodes in order to condense these workloads and free up some unused resources.

II. Related Work

Round Robin:

Although several researchers have conducted several studies to improve load distribution efficiency over data centers by proposing various techniques including Round Robin Algorithm, virtual machine migration approach, and many more. In some studies [10] researchers examine the features of mobile applications and find a linear connection between the control flow complexity and the mobile application's detection effectiveness. The main idea of their work was based on the their DRRHA approach, the tasks which are received from users in the ready queue based on the arrival order. After that each task arrives there and gets sorted on the basis of the SJF manner. Every round, leads to the calculation of average mean in the task queue, then later quantum time is calculated and on that basis the workload is distributed over data centers.[3] Some of them have also proposed method by improving the functionality of the Enhanced Active Monitoring Load Balancing (EAMLB) algorithm was created. It enhances reaction time more than Active Monitoring and Round Robin.

The performance analysis of a load balancing algorithm in a heterogeneous cloud computing environment was described in [4] by the authors. They looked at three distinct service broker policies along with the two algorithms Round Robin and Throttled. Through simulation using the Cloud Analyst tool, they were able to demonstrate that the Throttled algorithm with optimized reaction time provides the best performance in heterogeneous cloud computing. These algorithms have not been put into practice in a real-time setting. Researchers compared different cloud computing static and dynamic algorithms in [7]. They made their comparison on the difficulties they currently face with cloud computing. They discovered that Load Balancing Min-Min (LBMM) is more effective at utilizing resource parameters and requires less reaction time in many algorithms.

VM Migration:

Another [1] research has proposed a double-threshold migration approach, having two threshold values named as 'upper threshold' and 'lower threshold', respectively. The upper threshold setting is aimed to preserve the capacity of CPU for unpredicted workload rises and to prevent SLA violations. The lower threshold is settled as it is trying to switch more physical nodes which are not fully utilized by the processor into sleep mode.

Throttled load balancing:

An [3]index table with the ID of the virtual machine and its state whether it is AVAILABLE or BUSY -is used in this load-balancing strategy. It first determines the status of the VM before allocating any tasks to it. If VM state is allocated and given a VM id, which tells the balancer to update the index table: if the NM ids busy, then the task must wait in line.

In a study it was discovered [5] a method that enables the LEM operator to react to fast frequency response service procurement signals sent by the balancing authority even when the service resolution requires sub- second latency The suggested cloud edge architecture offers scalability and elasticity for different DR/DER configurations and has been evaluated using the IEEE Low Voltage Benchmark model.

III.METHODOLOGY

Load Vector:-

On each physical node, the load data of the nodes must first be gathered for our model. Every physical load maintains a load vector in accordance with the decentralized technique to receive addresses and indexes from other peer nodes.

A queue or an array may be used to implement LV_i's storage of the PNs (Physical Node) indices, as seen in the Figure 2. Information about the source ID is contained in each component of the load vector. The load vector of LV_i could be established once all the load indices for the current control interval have been received so that later it could give us the migration of threads for load balancing.

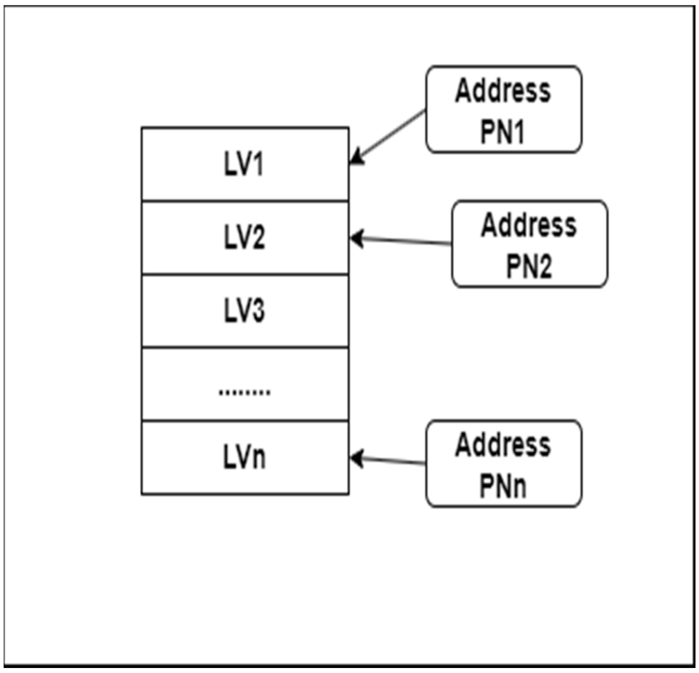


Fig. 2. Load Vector

Task Scheduling:-

For this module, we'll employ two threads(processes) that will operate concurrently in opposition to one another. PRO1 and PRO2 are two processes on which the processor will work. One thread (PRO1) will point to LV [0] at the start, and another thread (PRO2) will point to LV[n] (n being the last index). PRO1 will travel from index 0 to the final index (n), and PRO2 will move in the other manner, from the last index (n) to the first index (0).

The threads will initially determine whether the specified node is empty or busy. The thread will verify and work on the node if it has workload for a set amount of time, move on to the next node, and continue the process there. The threads will proceed in the direction that was assigned to them earlier until they reach the last node in that direction, which for PRO1 is the last node and for PRO2 is the first node. In case the two threads intersect at a particular node then at that moment PRO1 will work on that node and PRO2 will move to the next node. This is done to ensure that the work flow of the algorithm remains constant.

Algorithm for this approach is given in Figure 3, which depicts the two threads start working in the respective assigned direction and process each node one by one for a given amount of time. Also Figure 4 depicts the architectural model of the algorithm implemented in Figure 3.

Algorithm: -

```
1.  Start
2.  Assign PRO1 at LV [0] and PRO2 to LV[n]
3.  Start the threads to move in the respective direction.
4.  Check if index of PRO1 is same as the index of PRO2, if
    true then go to step 5 otherwise go to step 6.
5.  Then move PRO2 to next index and continue processing.
6.  Check whether index of PRO1 and PRO2 has workload,
    if true then go to step 7 else go to step 8.
7.  Process PRO1 and PRO2
8.  Move PRO1/PRO2 to the next index.
9.  If PRO1 ->reached to the end of the load vector (in the
    assigned direction), also PRO2-> reached to end of the
    load vector(in the assigned direction) then go to step 10
    otherwise repeat steps 11.
10. Reverse the directions of both the threads and go to step
    3.
11. PRO1/PRO2 keep on processing the last node and go to
    step 4.
12. Repeat the whole process by going by to step 3.
```

Fig. 3. Algorithm for the task scheduling approach

Only when both nodes have reached their respective ends will the thread indexes be switched and the procedure will begin once again. If one thread's process has been completed, i.e. it has reached the end of its load vector, then we will make this thread sleep for the time the other thread is in process. We will put the thread in work in a monitor so that only that thread keeps on working towards its direction and reaches the end of load vector, while the other thread stops working for that specific time. In order to stop the threads from operating in the same direction in the future, this is done.

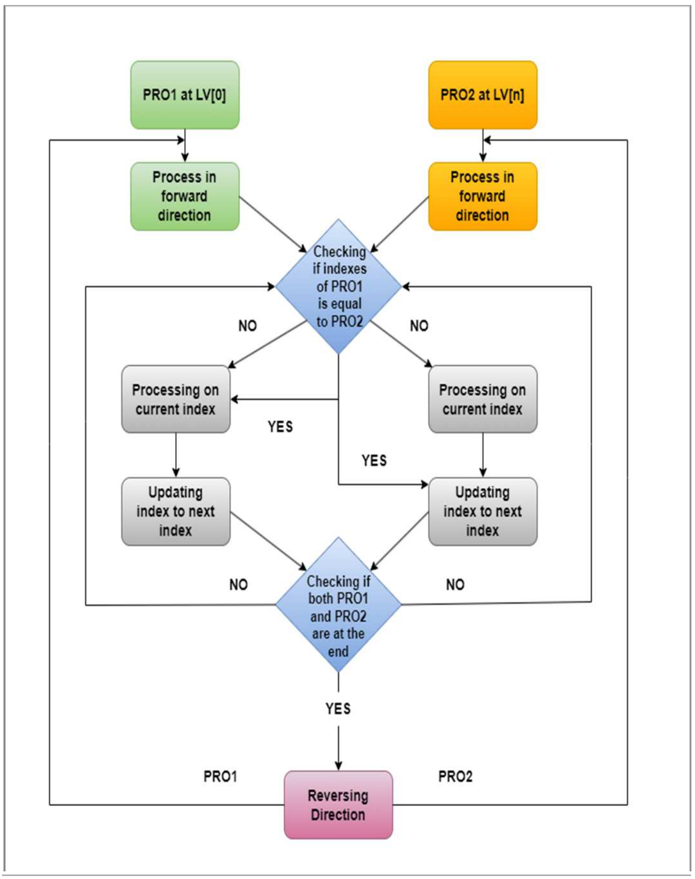

PRO1 at LV[0]
PRO2 at LV[n]
Process in forward direction
Process in forward direction
Checking if indexes of PRO1 is equal to PRO2
NO
NO
Processing on current index
YES
YES
Processing on current index
Updating index to next index
Updating index to next index
NO
NO
Checking if both PRO1 and PRO2 are at the end
YES
PRO1
PRO2
Reversing Direction

Fig. 4. Architecture for Task Scheduling

Freshly Received Request:-

In this module, we will balance the incoming workload by distributing it among the nodes in an alternate manner, so that the first request or workload will go in the direction of thread 1 (PRO1), which will then locate the next empty node in its general vicinity and receive the workload assignment. The second request or workload will now move in the direction of thread 2 (PRO2), discover the following vacant node, and assign the workload to it.

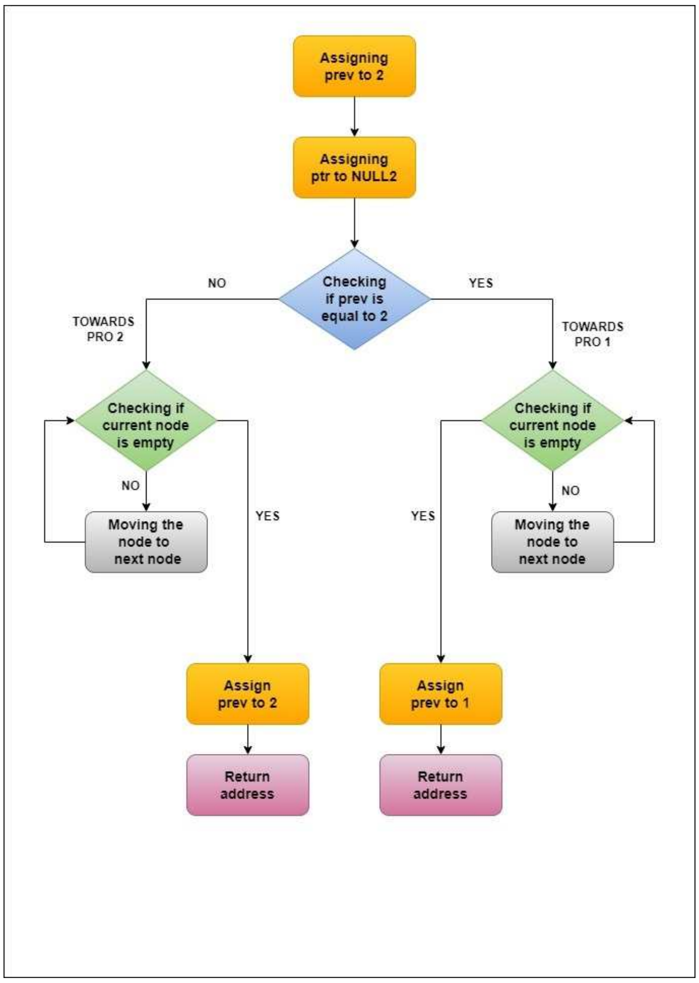
We have created a variable called prev that will hold the data from the previous thread's directory and will either be set to "1" (representing thread 1(PRO1)) or "2" (representing thread 2(PRO2)). The incoming workload will move to the thread 2's directory if the value at prior is "1.".

In order to assign the load node discovered, we have also taken a pointer (PTR) that will locate, store, and return the address of the empty node. If the thread has reached to the end of the load vector, then the incoming request shall wait for a short period of time till the threads are assigned new load vectors, i.e their direction is reversed. The algorithm is implemented in Figure 5 and the architectural depiction of the algorithm is shown in Figure 6.

Algorithm: -

```
1.  Assign 2 to prev
2.  Assign NULL to PTR
3.  Check if the value at prev is '2', if true go to step 3
    otherwise go to step 7
4.  Check if the node next to the index at which thread
    1 is pointing is empty, if true go to step 5 otherwise
    go to step 4
5.  Move to next node index and go to step 3
6.  Assign the address at the index found to PTR
7.  Assign 1 to prev and return
8.  Check if the node next to the index at which thread
    2 is pointing is empty, if true go to step 9 otherwise
    go to step 8
9.  Move to next node index and go to step 7
10. Assign the address at the index found to PTR
11. Assign 2 to prev and return
```

Fig. 5. Algorithm for the Freshly Received Request



Assigning prev to 2
Assigning ptr to NULL2
NO
Checking if prev is equal to 2
YES
TOWARDS PRO 2
TOWARDS PRO 1
Checking if current node is empty
Checking if current node is empty
NO
NO
Moving the node to next node
YES
YES
Moving the node to next node
Assign prev to 2
Assign prev to 1
Return address
Return address

Fig. 6. Architecture for Freshly Received Request

Discussion And Findings

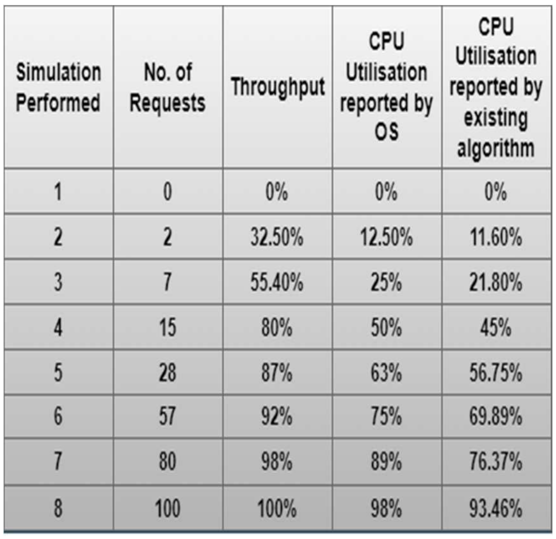
It becomes necessary to handle the workload with faster processing times as cloud computing expands. The model that is suggested in this article addresses these problems more successfully by speeding up node responses and balancing the volume of incoming requests. [14] It is extremely difficult to manage these data in an organized way in a cloud computing setting because cloud data centers and users are dispersed widely across the globe. We can efficiently distribute the task and make use of all the resources with the aid of this model.

The technique solves the issues of load imbalance and excessive migration costs caused by conventional scheduling algorithms by reaching the perfect load balancing. The experiment's findings demonstrate that this approach can more successfully achieve load balancing and efficient resource usage.

Test Simulated :-

As per the simulation performed, the result obtained is provided in table 1 also graph associated with the result is depicted in Figure 7. Table provides the information based on number of requests received by the processor and throughput. The simulation based on above parameters is carried out for a given time. Throughput is calculated on the basis of number of requests completed in the given time.

Table 1: Simulation results and parameters



Simulation Performed	No. of Requests	Throughput	CPU Utilisation reported by OS	CPU Utilisation reported by existing algorithm
1	0	0%	0%	0%
2	2	32.50%	12.50%	11.60%
3	7	55.40%	25%	21.80%
4	15	80%	50%	45%
5	28	87%	63%	56.75%
6	57	92%	75%	69.89%
7	80	98%	89%	76.37%
8	100	100%	98%	93.46%

Also a comparison graph is drawn between CPU utilization reported by operating system and CPU utilization observed by an existing algorithm. It can be clearly observed from the graph that the algorithm proposed is providing better results than the previous algorithm.

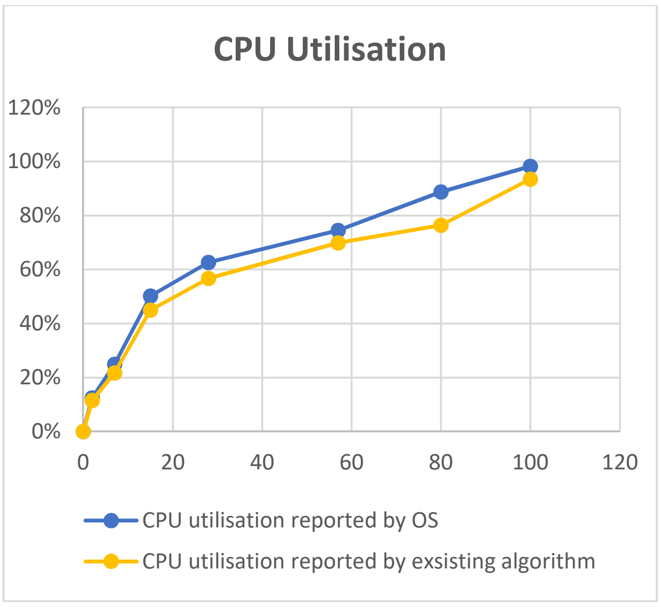


Fig. 7. CPU Utilization Representation

Conclusion

Finding new and improved ways to use cloud services is a field of research given the growing significance of cloud computing. Currently used and implemented algorithms for cloud load balancing offer a way for better utilizing cloud resources. Every dynamic and static algorithm has pros and cons that mostly deal with minimizing the time taken by nodes to execute commands, delays in operations caused by long queues, overloading of nodes, etc.

Highlights of this research paper involves an architecture of how the work can be balanced by distributing the load and processing it using the algorithm mentioned in above section. This paper involves an efficient algorithm in providing a better and faster way to balance the existing workload and managing the new requests. The proposed work in the field of cloud helps in reducing the processing time at the data centers.

References

[1] X. Wang, X. Liu, L. Fan, and X. Jia, “A Decentralized Virtual Machine Migration Approach of Data Centers for Cloud Computing,” Mathematical Problems in Engineering, vol. 2013. Hindawi Limited, pp. 1–10, 2013. doi: 10.1155/2013/878542.

[2] E. Jafarnejad Ghomi, A. Masoud Rahmani, and N. Nasih Qader, “Load-balancing algorithms in cloud computing: A survey,” Journal of Network and Computer Applications, vol. 88. Elsevier BV, pp. 50–71, Jun. 2017. doi: 10.1016/j.jnca.2017.04.007.

[3] S. Garg, Dr. D. V. Gupta, and R. K. Dwivedi, “Enhanced Active Monitoring Load Balancing algorithm for Virtual Machines in cloud computing,” 2016 International Conference System Modeling &amp; Advancement in Research Trends (SMART). IEEE, 2016. doi: 10.1109/sysmart.2016.7894546.

[4] V. Behal and A. Kumar, “Cloud computing: Performance analysis of load balancing algorithms in cloud heterogeneous environment,” 2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence). IEEE, Sep. 2014. doi: 10.1109/confluence.2014.6949291.

[5] A. Bachoumis et al., “Cloud-Edge Interoperability for Demand Response-Enabled Fast Frequency Response Service Provision,” IEEE Transactions on Cloud Computing, vol. 10, no. 1. Institute of Electrical and Electronics Engineers (IEEE), pp. 123–133, Jan. 01, 2022. doi: 10.1109/tcc.2021.3117717.

[6] S. Chhabra and A. K. Singh, “Optimal VM Placement Model for Load Balancing in Cloud Data Centers,” 2019 7th International Conference on Smart Computing &amp; Communications (ICSCC). IEEE, Jun. 2019. doi: 10.1109/icscc.2019.8843607.

[7] Fei Ma, Feng Liu, and Zhen Liu, “Distributed load balancing allocation of virtual machine in cloud data center,” 2012 IEEE International Conference on Computer Science and Automation Engineering. IEEE, Jun. 2012. doi: 10.1109/icsess.2012.6269396.

[8] Jinhua Hu, Jianhua Gu, Guofei Sun, and Tianhai Zhao, “A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment,” 2010 3rd International Symposium on Parallel Architectures, Algorithms and Programming. IEEE, Dec. 2010. doi: 10.1109/paap.2010.65.

[9] S. Mall and A. K. Sharma, “Analyzing Load on Cloud: A Review,” 2018 Second International Conference on

Computing Methodologies and Communication (ICCMC). IEEE, Feb. 2018. doi: 10.1109/iccmc.2018.8487230.

[10] V. N. Volkova, L. V. Chemenkaya, E. N. Desyatirikova, M. Hajali, A. Khodar, and A. Osama, “Load balancing in cloud computing,” 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). IEEE, Jan. 2018. doi: 10.1109/eiconrus.2018.8317113.

[11] Y. Guo, Y. He, and X. Liu, “Research and implementation of load balancing algorithm for mass mobile application detection task,” 2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS). IEEE, Aug. 2016. doi: 10.1109/ccis.2016.7790224.

[12] V. Sreenivas, M. Prathap, and M. Kemal, “Load balancing techniques: Major challenge in Cloud Computing - a systematic review,” 2014 International Conference on Electronics and Communication Systems (ICECS). IEEE, Feb. 2014. doi: 10.1109/ecs.2014.6892523.

[13] A. Singh, M. Korupolu, and D. Mohapatra, “Server-storage virtualization: Integration and load balancing in data centers,” 2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, Nov. 2008. doi: 10.1109/sc.2008.5222625.

[14] R. P. Goldberg, “Survey of virtual machine research,” Computer, vol. 7, no. 6. Institute of Electrical and Electronics Engineers (IEEE), pp. 34–45, Jun. 1974. doi: 10.1109/mc.1974.6323581.

[15] S. Swarnakar, Z. Raza, S. Bhattacharya, and C. Banerjee, “A Novel Improved Hybrid Model for Load Balancing in Cloud Environment,” 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN). IEEE, Nov. 2018. doi: 10.1109/icrcicn.2018.8718697.

[16] Y. Zhao and W. Huang, “Adaptive Distributed Load Balancing Algorithm Based on Live Migration of Virtual Machines in Cloud,” 2009 Fifth International Joint Conference on INC, IMS and IDC. IEEE, 2009. doi: 10.1109/ncm.2009.350.

[17] H. Mehta, P. Kanungo, and M. Chandwani, “Decentralized content aware load balancing algorithm for distributed computing environments,” Proceedings of the International Conference &amp; Workshop on Emerging Trends in Technology - ICWET ’11. ACM Press, 2011. doi: 10.1145/1980022.1980102.

[18] Ghutke Bhushan and Shrawankar Urmila, "Pros and Cons of Load Balancing Algorithms for Cloud Computing", Proceedings of International Conference on Information Systems and Computer Networks IEEE, pp. 123-127, 2014.