Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 4640443, 12 pages
https://doi.org/10.1155/2022/4640443

Research Article

Designing a Component-Based Throttled Load Balancing Algorithm for Cloud Data Centers

Dawit Mekonnen,[1] **Alemayehu Megersa**,[1] **Rakesh Kumar Sharma**,[2] **and Durga Prasad Sharma**[3]

[1]*Arba Minch Institute of Technology, Arba Minch University, Arba Minch, Ethiopia*
[2]*University of Maryland Eastern Shore, Princess Anne, MD, USA*
[3]*AMUIT, MOEFDRE Under UNDP and MAISM-RTU Kota, Jaipur, India*

Correspondence should be addressed to Dawit Mekonnen; dawit.mekonnen@amu.edu.et

Received 18 July 2022; Accepted 15 September 2022; Published 3 October 2022

Academic Editor: Vimal Shanmuganathan

Cloud services are accessed from different geographical locations where client migration or switching from one server to another based on the loads is a common phenomenon. One of the most critical challenges the cloud data centers face is managing the loads over geographically dispersed data centers and their virtual machines (VMs). VMs need to be balanced with the varied loads or dynamics of traffic. There are possibilities of the highest loads to be tolerated by the VMs over the cloud servers without crashing. Load balancing issues are managed by load balancing algorithms. Load balancing algorithms have varied issues of efficiency due to certain parameters like the capability of the lowest resource utilization, response time, higher overhead while checking the idle or normal nodes, and many others. Throttled load balancing algorithm manages loads of the virtual machines by dividing the virtual machines into two segments, that is, "available" and "free." To do this, the throttled algorithm uses a single component to assign the virtual machines and other tasks. The throttled algorithm utilizes only the first VMs available, the next, and so on. These strategic issues most often degrade the performance of the applied load balancing algorithm. Such issues create a curiosity to enhance this algorithm's performance for efficiently managing the dynamic loads of the cloud VMs. This research paper proposes a component-based throttled load balancing algorithm with VM reader, free VM holder, and free VM manager components. The VM reader component reads all available VMs. The free VM component holds free VMs temporarily until they are moved to the free VM manager component. For the performance test, the cloud analyst simulation tool was used. Based on the comparative analysis with the other five popularly used load balancing algorithms, the component-based algorithm's performance is significantly enhanced. The proposed algorithm resulted in 325.30-microsecond response time and 27.12-microsecond processing time by the closest data center service broker policy. The newly proposed "component-based throttled load balancing algorithm" is found to be better than the existing throttled algorithm and the other five selected algorithms in terms of response time, processing time, and resource utilization.

1. Introduction

Cloud computing is the use of computer system resources on-demand as a public utility for the users of the cloud. That means people suffering from different constraints, either hardware or software or power, can proceed to their work from the cloud itself without requiring extra facilities in which they can have a high amount of storage, and they can use their application software and others without any random access memory (RAM), central processing unit (CPU), and other constraints. They can use it simply from their device, which can access the Internet through the browser or preinstalled cloud application.

There are several definitions of cloud computing. The one which is adapted from the National Institute of Standards and Technology (NIST) which is commonly known as the NIST definition defines cloud computing as a model that lets and provisions its users to get ubiquitous, convenient,

on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) [1].

These days cloud computing is an excellent platform that is used to store users' data at low cost, being available entirely all the time. However, it has several critical issues that need serious consideration, like security, fault tolerance, and load balancing [2]. Due to its simplicity to use and other factors, cloud systems have multiple customers, increasing each year [3]. These cloud users are masked from their background operations. Even they cannot tell exactly from which server or virtual machine (VM) they are processing their data.

Cloud computing operates in a distributed computing manner [4, 5]; by using the cloud, people who are found in different geographical locations can access a single cloud system. The increased number of cloud users from different areas has made the requests (the load) that go to the servers too much. The cloud has several nodes or VM that receive the requests for processing. So these nodes or VMs need to respond effectively to the requests efficiently.

Load balancing is the sharing of the loads between different nodes to effectively give services to the cloud user without hampering the underlying infrastructure. It is the distribution of loads over the different nodes, which provides good resource utilization when nodes are overloaded with jobs [4]. Load balancing is a technique that is used to balance the load between the nodes or VM of the cloud. Balancing means that it makes the less loaded node take additional loads and makes the highly loaded node to be free from taking or accepting additional loads. The load balancing is done by using a cloud-based load balancing algorithm. Cloud-based load balancing algorithm is a method that is implemented by the load balancers to balance the load in the cloud VMs.

The utilization of web assets is broadly expanding, bringing about the increment of workload exponentially [6, 7], so identifying the efficient cloud-based load balancing algorithm to use is essential to manage all the loads.

This study is an effort to develop a component-based algorithm for the throttled cloud-based load balancing algorithm; that is, the study proposed a new component-based cloud-based load balancing algorithm by using the throttling concept. Further, the study has also done deep performance analysis experiments on the existing cloud-based load balancing algorithms with the developed component-based cloud-based load balancing algorithm to find the best performing algorithm for different scenarios.

In general, load balancing is a technique that ensures that VMs should not be kept idle while other VMs are being overutilized [8]. The main goal of the load balancing concept in a cloud environment is to ensure the optimum utilization of resources with minimal response and processing times over cloud data centers. Since there is exponential growth in cloud users, it is not constant; that is, it fluctuates from time to time [9]. It is desirable to serve the users as per their service level agreements (SLAs) and desired service specifications, that is, with a better or desired performance of the resources in minimal request and response times.

The two types of cloud-based load balancing algorithms are most often used in this context; that is, static algorithms like round-robin and randomized algorithms do not check the state of the VMs before assigning the loads to the VM. The main goal of such algorithms is to lower the response time [8]; therefore, the parameters like load status are ignored. In cloud environment's load balancing scenarios, the best-case situation occurs when the first or the random VM is free or idle for assigning the task. On the other hand, dynamic algorithms like throttled and active VM monitoring checks the state of the VM before assigning the load to the VM. At this point, repeatedly checking the node for load status creates another challenge of task overhead [10]. This situation needs a strong measurement to resolve these problems by providing an effective cloud-based load balancing algorithm for efficient allocation of VM.

The throttled load balancing algorithm is one of the dynamic load balancing algorithms, which implements the throttled adjustable mechanisms. In this algorithm, the data center controller directly forwards the user requests to the throttled VM load balancer. Now the load balancer checks for the availability of a VM and sends its ID to the data center controller. In general, the throttled load balancer algorithm manages different tasks starting from reading the current VM state to communicating to the data center controller for delivering the ID of the free or underloaded VM. All these load balancing tasks, along with other specific tasks, are being done only on a single component of the load balancer by creating a task overhead on the same load balancer. To deliver effective service, the tasks of the load balancer should not have to be constrained into a single part; that is, the tasks have to be shared with different components to lower the unnecessary overhead.

The throttled load balancer also maintains a table that manages the salient states of VMs. The load balancer provides the first available VM ID that it gets from the status table every time. In this scenario, whenever the data center controller forwards a request for a VM, it sends back only the foremost VM lists from the table of available lists of VMs. This implies that it does not use every VMs, which is found in the table of available VMs.

The aforementioned scenarios revealed a clear research gap towards enhancement in the throttled algorithm. More specifically, the existing ranking (underutilized, overutilized, or idle) status and performance of the throttled algorithm require a significant improvement in the existing state of design towards better performance. The scope of a performance enhancement can be a unique contribution to the throttled algorithm as the task overhead and the less resource utilization of the throttled need to be resolved categorically. The problems which need to be resolved in assigning the VMs using throttled algorithm are significantly important for the overall performance of the cloud environment.

1.1. Research Goal and Questions to Be Answered. In this research paper, efforts have been made to answer the three important questions: What are the deficiencies in the

existing load balancing algorithms used for load balancing over cloud data centers? Which parameters can be explored to enhance the performance of the throttled algorithm? How to design an enhanced component-based throttled algorithm for improved response time, processing time, and resource utilization with a comparison matrix?

The general objective of this research was to develop an enhanced component-based throttled load balancing algorithm for the better management of workloads over cloud VMs. To achieve this, four specific tasks were performed; that is, parameter-based performance deficiencies are enlisted in a matrix by comparing the existing cloud-based load balancing algorithms. A component-based throttled algorithm for improved response time, processing time, and resource utilization is developed. For performance validation of the results, the two different test scenarios are created in a simulated environment using the cloud analyst tool. The first scenario was focused on the response time and processing time; however, the second scenario was created to showcase resource utilization. The results demonstrated in the simulated environment showcased a clear enhancement in the algorithmic performance of the throttled algorithm.

In developing the enhanced component-based throttled algorithm, the study was primarily confined to the load balancing issue of cloud computing, that is, more specifically, on VMs load balancing. The other issues of the cloud, that is, security and fault tolerance, are not included in the study. The research analyzed only the VM's load balancing by comparing numerous static and dynamic algorithms for both conventional and nonconventional algorithms. In addition, the study only compared the algorithms that are used for managing the VM loads only in a simulated environment to imitate the real cloud environment. Only three parameters, that is, response time, processing time, and resource utilization, are considered to enhance the algorithmic performance.

To justify the research dimension, a rigorous literature survey was done to identify and analysis of the research gaps based on the state of art research contributions by numerous researchers. Balancing the VM's load is an essential parameter to ensure effective service delivery. A study [11] states that cloud computing can be efficient using the load balancing method and showed that load balancing significantly increases user satisfaction for cloud services. This study does not suggest any performance enhancement in the existing algorithm. Further, the study does not mention how to implement the proposed partitioning mechanism as an effective solution. Another survey study [12] presented eight cloud load balancing algorithms from both the static and dynamic algorithms currently used in the cloud computing environment. However, the study shows only the basic operations of the algorithms but does not include any experimental setups to test the presented algorithms. Different kinds of nonconventional hybrid algorithms are presented as a survey in the study [13]. The study clarifies that the cloud analyst tool is the most frequent tool that is used by different studies in the load balancing domain. The analysis presented in the survey does not mention whether they used a homogeneous VM or heterogeneous VM environment.

Another effort was made in [14] on load balancing algorithms. In this research, two different categories of load balancing algorithms, namely, the dynamic and the static, are used. The study recommended dynamic load balancing algorithms for larger loads and better performance [14]. This study explained centralized load balancing algorithms and different parameters used to compare load balancing algorithms by describing the basic load balancing algorithms like the round-robin algorithm, randomized algorithm, and others with their respective adverts and demerits.

Different studies have made efforts to compare and analyze the cloud VM load balancing algorithms. A study [6] compared ten algorithms and suggested optimization-based algorithms to overcome resource utilization and response time problems. The study did not mention any experimental environments with respective scenarios. The suggestion needs to be supported with numerical figures of experiment results.

In a study [15], round-robin, equally spread current execution, and throttled cloud load balancing algorithms were compared, and it was found that throttled load balancing algorithm performs better than the two by using three different cases using a cloud analyst simulation tool. The throttled algorithm showed better results in response time and data center processing time. This study [15] neglected the analysis that can be obtained on resource utilization. Resource utilization is a typical parameter in comparing algorithms, and it needs to be examined. Another performance comparison was made in a study [10]. In this study, the same algorithms with cloud analysts yielded that the round-robin algorithm is the best performing amongst the other throttled and equally spread current execution algorithms in a homogeneous environment.

The study [16] proposed a balanced throttled algorithm. The cloud analyst tool was used to check the performance of the proposed algorithm with the round-robin algorithm, throttled algorithm, and active VM monitoring algorithm. This study has enhanced the response time of the throttled load balancing algorithm. However, the study [16] does not mention the experimental setups used for undertaking the experiment.

1.2. Research Methodology. This paper used a mixed version of descriptive and experimental research designs. As seen in Figure 1, the study undergoes experimental analysis of the different load balancing algorithms to identify the best or the most efficient algorithm in different scenarios and cases by experimenting in simulated environments. The study tries to explore the different scenarios using both qualitative and quantitative data for analysis approaches.

2. The Component-Based Cloud VM Load Balancing Algorithm

The throttled load balancing algorithm consists of only one component inside its load balancer implementation [17, 18]. The proposed component-based throttled load balancing algorithm consists of three kinds of different components. These components are described as follows:

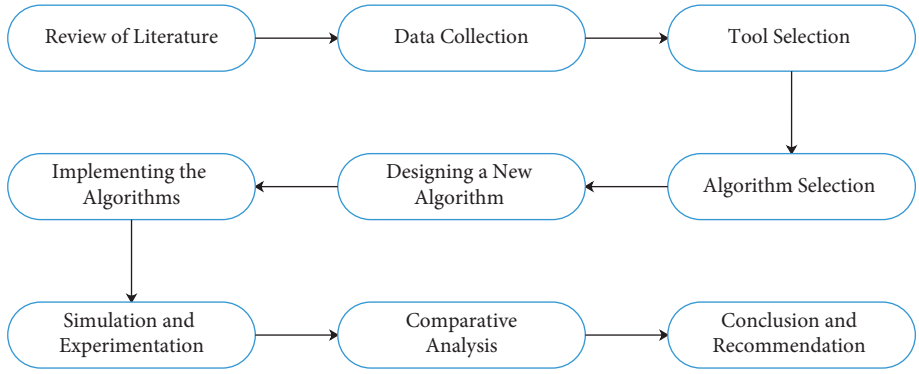

Figure 1: Research process flow.

VM Reader. The primary task of this component is to read and hold the existing VMs inside its allocation table (Table 1). The VM state (available or busy) is to be known before assigning them to the request. Now the VM reader component reads all the VMs that are found in that data center with their respective ID and state. The state of the VM that is read by this VM reader is either Busy or Available.

Free VM Holder. The free VM holder component is used as a temporary free VMs holder. It communicates with the VM reader and collects the information about the free VMs from the allocation table of the VM reader component. All the VMs which are found in the free VM holder are considered free VMs that are ready for transfer to the free VM manager (Table 2).

Free VM Manager. Primarily, the free VM manager component takes the VMs that are found in the free VM holder. This component is responsible for managing the VMs. When the data center controller requests for specific VM, the free VM manager sends the ID of a single VM from its table (Table 3) by selecting randomly.

2.1. Component-Based Throttled Load Balancing Algorithm. In the proposed algorithm, Figure 2 describes the basic operations of the proposed algorithm, that is, the component-based throttled load balancing algorithm. In general, sharing multiple tasks for different entities lowers the burden on a single component and results in better performance.

The newly proposed component-based algorithm tries to enhance the performance of the throttled algorithm by mitigating the task overhead. This task overhead is found on the single component of the throttled algorithm. As is seen in Figure 2, initial requests from different user bases are received by the data center controllers. Then, the data center controller requests the load balancer for the free VM to use. Now the proposed load balancer automatically responds to the request directly by sending the status of the free VMs getting from the free VM manager. Since the free VM manager component only contains the available VMs list, it is very easy to send a respective VM ID to the request of the data center controller. After getting the ID from the free VM manager, the data center controller communicates to the VM identified by the ID.

Thus, the existing throttled algorithm, the newly proposed component-based algorithm, presents a better allocation table with an instant updating mechanism for the VMs status. In the existing algorithm, there was a single component for reading the VM and sending the free VM, and therefore it was focused only on a replacement mechanism to update.

On the other hand, in the proposed component-based throttled algorithm, when a given VM is allocated to a specific task, the VM is removed from both the free VM holder and free VM manager component. Also, in the VM reader, the state for that VM is updated to show it is a busy state. In the case of deallocation of a VM, only the state in the VM reader component is updated to the available state.

3. Design of the Component-Based Algorithm

The design of the proposed algorithm, that is, the "component-based throttled algorithm," is presented in Figures 3 and 4. As seen in the algorithm design, it starts by reading the VMs and ends up by updating the components of the algorithm upon allocation and deallocation of a VM.

4. Simulated Experimentation and Discussion

To show the implementation, the experimentations were done in a simulated environment using the cloud analyst simulation tool. Initially, the simulation setup was considered for large-scale data of the cloud applications such as social network data. The study used Facebook's statistical data to configure the user bases of client numbers (cloud service consumer numbers). Facebook was selected because of its highest number of social clients than the other cloud-based social networks that are accessed globally during all off-peak hours and on-peak hours. According to the *Miniwatts Marketing Group* report, Facebook has more than 2.22 billion users globally [19].

Table 1: VM reader.

VM	VM_1	VM_2	VM_3	VM_4	VM…	VM…	$VM_{…N}$
VM ID	VM_1ID	VM_2ID	VM_3ID	VM_4D	VM_5ID	VM…ID	$VM_{…N}$ ID
VM state						…….	$VM_{…N}$ State

Available VMs
Busy VMs

Table 2: Free VM holder.

VM	VM_1	VM_2	VM_3	VM_4	VM…	VM…	$VM_{…N}$
VM ID	VM_1ID	VM_2ID	VM_3ID	VM_4D	VM_5ID	VM…ID	$VM_{…N}$ ID
VM state						…….	$VM_{…N}$ State

VM	VM_1	VM_3	VM…	VM…	$VM_{…N}$
VM ID	VM_1ID	VM_3ID	VM_5ID	VM…ID	$VM_{…N}$ ID
VM state				…….	$VM_{…N}$ State

Available VMs
Busy VMs

Table 3: Free VM manager.

VM	VM_1	VM_3	VM…	VM…	$VM_{…N}$
VM ID	VM_1ID	VM_3ID	VM_5ID	VM…ID	$VM_{…N}$ ID
VM state				…….	$VM_{…N}$ State

Available VMs

4.1. Experimentations in Simulated Environment. The experimentations were done mainly in two parts, each of which is based on two different scenarios of the service broker policy of the cloud load balancing algorithm. The service broker policy was used to determine which data center should receive the incoming requests upon implementing the different policies. This simulated experiment was done using these two policies.

The first service broker policy was the closest data center service broker policy. This service broker policy was used for pointing the requests to the nearest data center with an active VM in which their request was processed. The second service broker policy in the cloud load balancing algorithm was the maximum response time service broker policy. This policy mainly maximizes the response time for a particular request by executing the request in different data centers to get the shortest time of response.

The first part experiment was done to analyze the response time and data center processing time at constant execution time by altering data center and VM numbers. A total of 36 experiments, as shown in Table 4, were done for the selected five algorithms, that is, round-robin (RR) algorithm, throttled algorithm, equally spread current execution algorithm (ESCE), threshold algorithm, honey bee algorithm, and the proposed component-based algorithm. The second experiment was done to analyze resource utilization by setting the VM and the data center constant and varying the execution time. In this, a total of 12 experiments were done for the selected five algorithms and the proposed

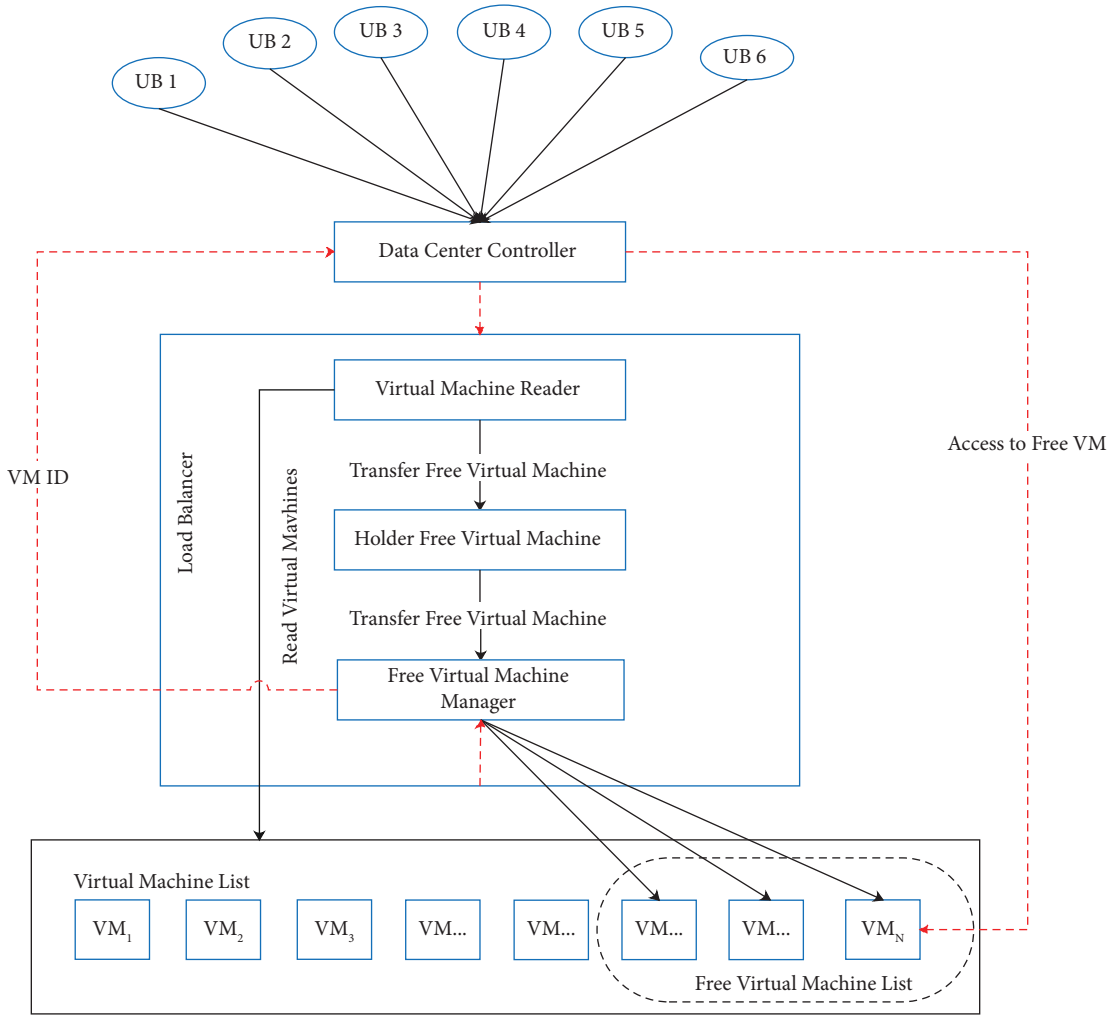

Figure 2: Proposed component-based throttled load balancing algorithm.

component-based algorithm. Undertaking experiments with different kinds of experimental setups is essential for critically evaluating the performance of the algorithm in different scenarios in contrast with miscellaneous algorithms.

4.2. Experiment Part One

4.2.1. Sample Experimentation. Scenario 1 Case 1 (S1C1): 1 Data Center with 25 VMs. In this experiment, one data center was used to manage all the requests. The data center was placed in Region 2, which was assumed to be the center for all the other regions on average. In this data center, 25 VMs were installed.

S1C1_Experiment 6: Component-Based Load Balancing Algorithm. As shown in Table 5, in this experimental scenario, the new component-based algorithm was simulated for 24 hrs of execution time over the cloud.

4.2.2. Scenario 1 Experimental Results (Using the Closest Data Center Service Broker Policy). By using the closest data center service broker policy, it was observed that, with minimum data center number and VMs in Case 1, the component-based load balancing algorithm achieved the highest performance in terms of response time and data center processing time than the other algorithms. In addition, from the experiment, it is observed that the proposed algorithm found higher performance than the other algorithms with an increased number of data centers and VMs. In this experimental scenario, the number of cloudlets from the users was kept constant. In this scenario, Facebook's statistical data with a large number of cloud users at on and off-peak hours were used by changing the number of the data centers and VMs in each experiment. From these experiments, it was observed that the component-based algorithm was found to be more efficient than the other algorithms when cloud systems consist of a limited number of data centers with limited VM distribution. The results are depicted in Figures 5 and 6 where the average response time and the average data center processing time are also presented by using the closest service broker policy.

4.2.3. Scenario 2 Experimentations Results (Using the Maximum Response Time Service Broker Policy). By using the maximum response time service broker policy in the experiment, it was observed that the performance, specifically

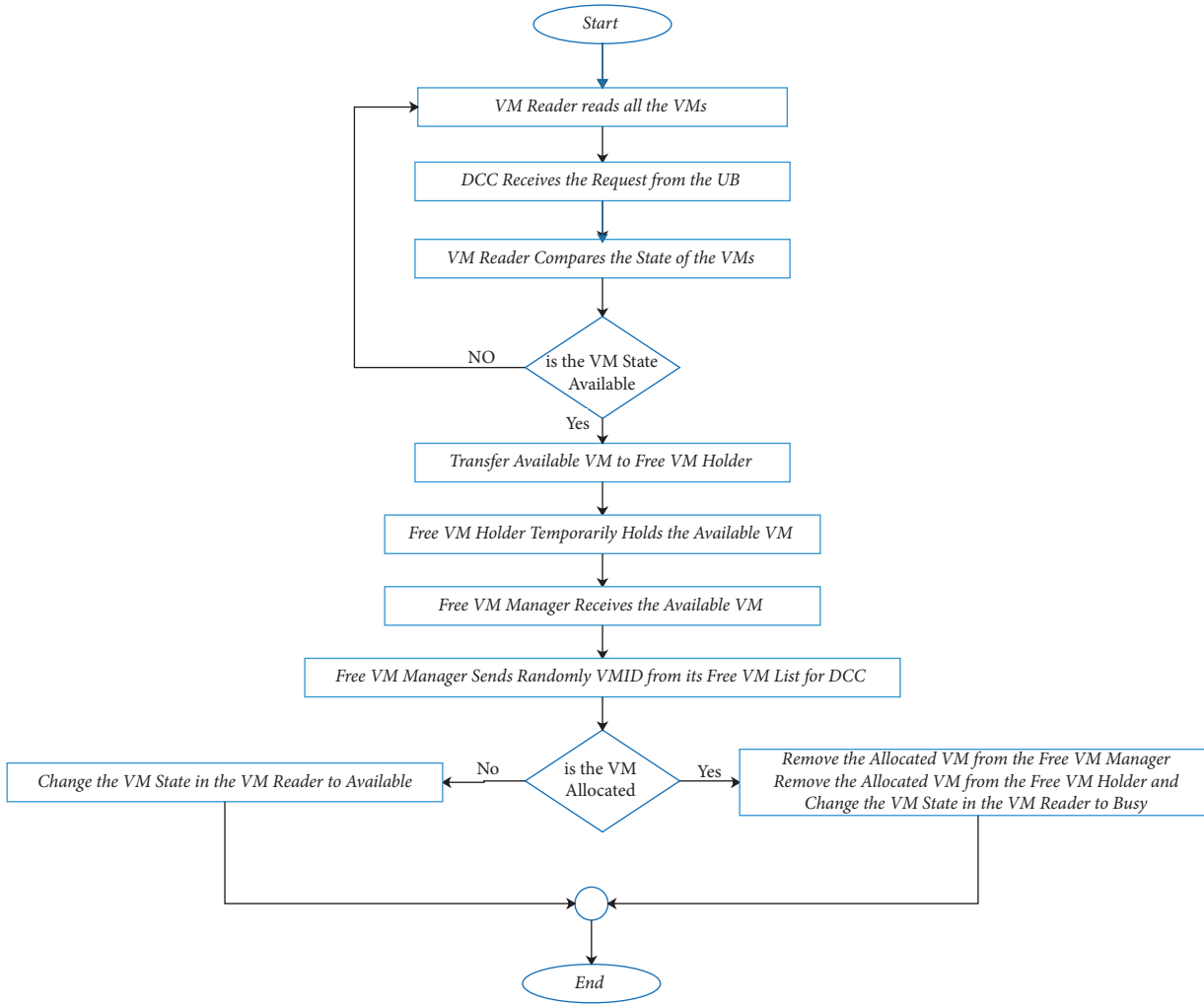

Figure 3: Design of the component-based algorithm.

the response time for the component-based algorithm, was found to be higher than the algorithms in the foremost two cases. The whole second experimental scenario is depicted in Figures 7 and 8. Also, the average data center processing time was found to be higher than the others in the third case. This clearly shows that the component-based algorithm over cloud-based systems using a limited number of data centers and VMs can minimize the response time on the service broker policy of maximizing response time.

4.3. Experiment Part 2. The first experiment was conducted to test the resource utilization of the algorithms within the 1-hour execution time. The second one was done for a 24-hour execution time.

4.3.1. Sample Experiment: Experiment 1: Resource Utilization for 5 VM for 1 hr. As presented in Figure 9, by using the closest data center service broker policy and maximizing response time service broker policy for all the six algorithms, the resource utilization was measured based on the VM allocation count.

4.3.2. First and Second Experimental Results on Resource Utilization. From the experiments as depicted in Figure 9, it was observed that the component-based algorithm for each VM performed more uniformly than the throttled algorithm. The experimental results for the resource utilization showed that no VM is underutilized and no VM is overutilized in contrast with the throttled. Also, the uniformity of the allocation of the VMs is depicted in Figure 10 by using the component-based algorithm in the case of the second experiment, which was conducted for 24 hours of execution time.

4.4. Comparative Analysis. A comparative analysis was also performed to identify the efficiency of the algorithms based on the selected parameters, that is, response time, data center processing time, and resource utilization. In each experiment and case, the algorithms were ranked based on their performance results on the selected parameters. The first algorithm, which showed the best performance, was given a weight of 1, and the least performing algorithm was given a weight of 6. Table 6 presents the overall rank based on the comparison results of the performance of the algorithms.

```
Input: Different requests from different user bases
Output: Allocated VM for the requests
1. Create HashMap FreeVirtualMachineManager<Integer,VirtualMachineState>
2. Create HashMap VirtualMachineReader<integer,VirtualMachineState>
3. Create HashMap FreeVirtualMachineHolder<integer,VirtualMachineState>
4. Create DCControler DCB
5. VirtualMachineReader Reads All the VMs
6. DCControler DCB Receives New requests from Userbases
7. PUBLIC PROCEDURE getNextAvailableVm
8.    Set  VMID <- -1
9.      set  SvmID<- -1
10.     if VirtualMachineReader.Size() >  0 Then
11.          Create Iterator Sitr
12.            For Each Sitr in VirtualMachineReader.keySet().iterator()
and Sitr.hasNext()
13.                Set SvmID<-Sitr.next();
14.                Create VirtualMachineState state
15.                  Set State<- VirtualMachineReader.get(SvmID)
16.                  If state.equals(VirtualMachineState.AVAILABLE Then
17.                      FreeVirtualMachineHolder.put(SvmID,
VirtualMachineState.AVAILABLE)
18.                  End If
19.            End For
20.                  If FreeVirtualMachineHolder.size() > 0
21.            FreeVirtualMachineManager.putAll(FreeVirtualMachineHolder)
22.                            Create ArrayList
valuesList<FreeVirtualMachineManager.keySet()>
23.                            Set rand <-
Random().nextInt(valuesList.size())
24.                            Set randomValue <- valuesList.get(rand);
25.                            Set vmId<-randomValue
26.                  End If
27.       End If
28.    allocatedVm(vmId)
29.    return vmId
30. End PROCEDURE
31. Public PROCEDURE cloudSimEventFired(CloudSimEvent e)
32. Print "******This Algorithm Is Developed By Dawit Mekonnen*********"
33. Print newline()
34. print "------Please Wait Until the simulation Finishes------"
35. Print newline()
36.      Event e<- get Id
37.      if Event e_Id == ALLOCATED_TO_VM  Then
38.          Set vmIdb<-Event.getId()
39.          FreeVirtualMachineHolder.remove(vmIdb);
40.          VirtualMachineReader.put(vmIdb, VirtualMachineState.BUSY);
41.            FreeVirtualMachineManager.remove(vmIdb);
42.      Else if Event e_Id ==VM_FINISHED Then
43.            Set vmIda<-Event.getId()
44.    VirtualMachineReader.put(vmIda,VirtualMachineState.AVAILABLE);
45.     End if
46. End PROCEDURE
```

Figure 4: Pseudocode of the component-based algorithm.

Table 4: Experimentation scenarios and cases for part 1 experiment.

Scenarios	*Scenario 1 (S1): closest data center service broker*	*Scenario 2 (S2): optimize response time service broker*
Case 1 (C1)	1 data center with 25 VMs	1 data center with 25 VMs
Case 2 (C2)	3 data centers with 50, 70, and 100 VMs, respectively	3 data centers with 50, 70, and 100 VMs, respectively
Case 3 (C3)	6 data centers with 60, 70, 80, 90, 100, and 110 VMs, respectively	6 data centers and 60, 70, 80, 90, 100, and 110 VMs, respectively

Table 5: Sample experiment one result for S1C1: Experiment 6.

	Avg. (ms)	Min. (ms)	Max. (ms)
Overall response time	325.30	42.17	698.59
DC processing time	27.12	0.06	185.91

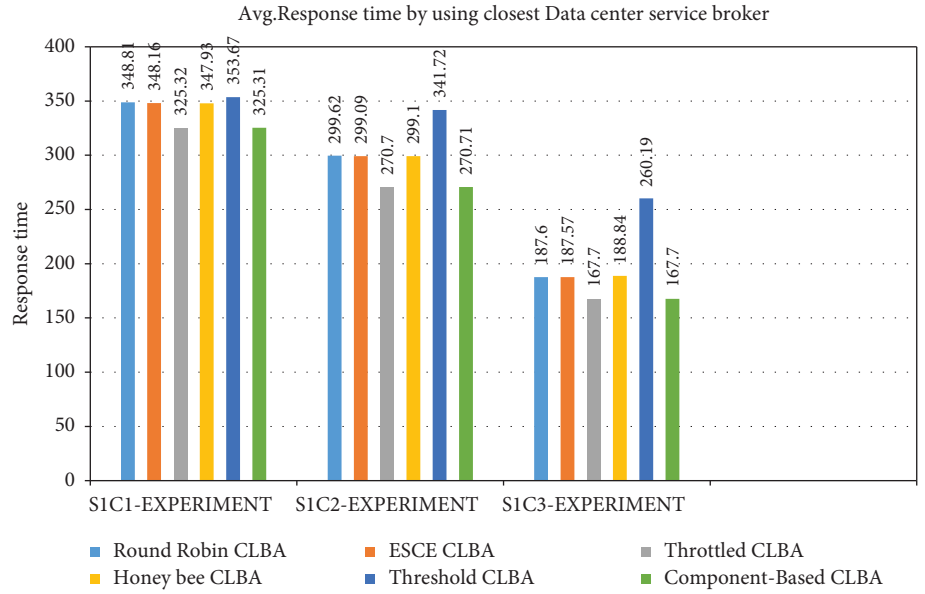

Figure 5: Avg. response time by using the closest data center service broker.

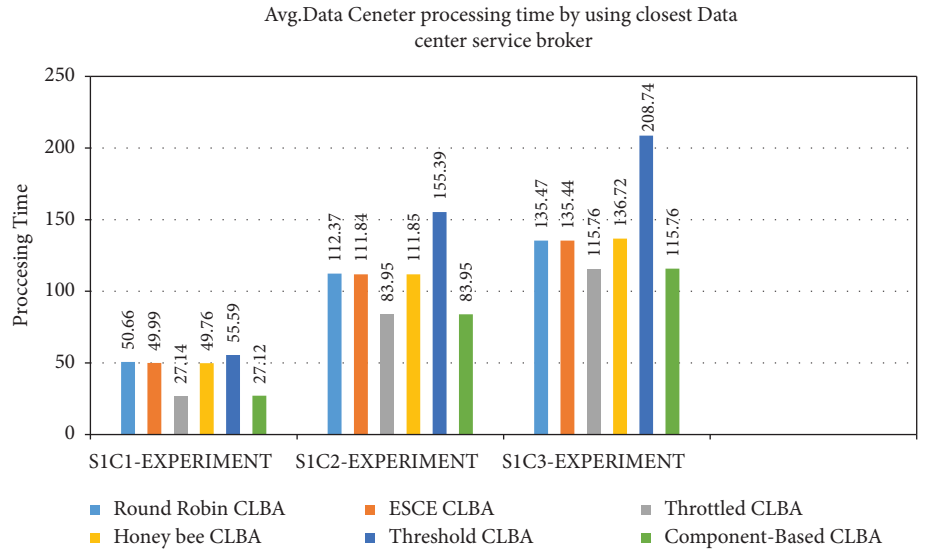

Figure 6: Average data center processing time by using the closest data center service broker.

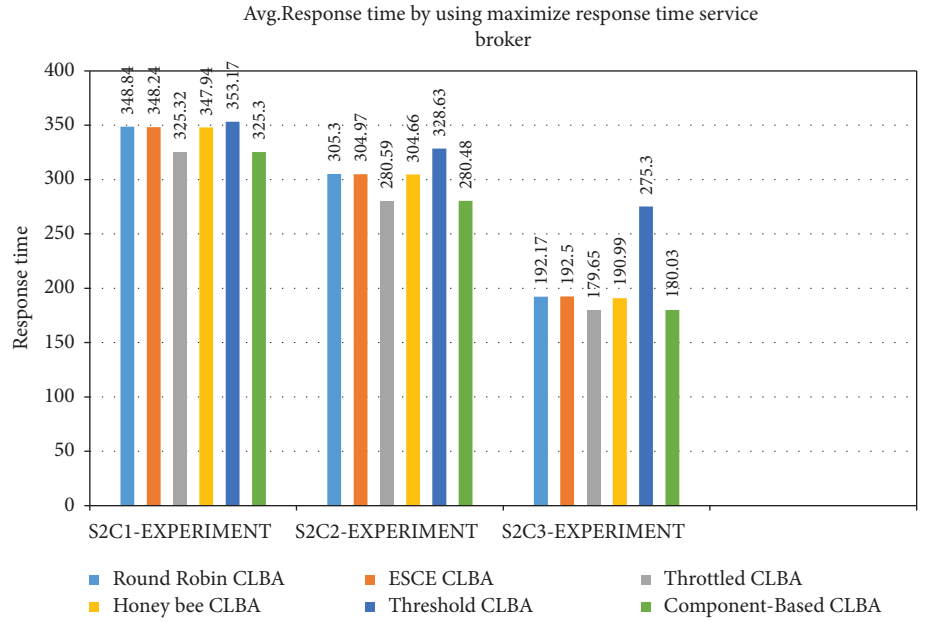

Figure 7: Avg. response time by using maximize response time service broker.

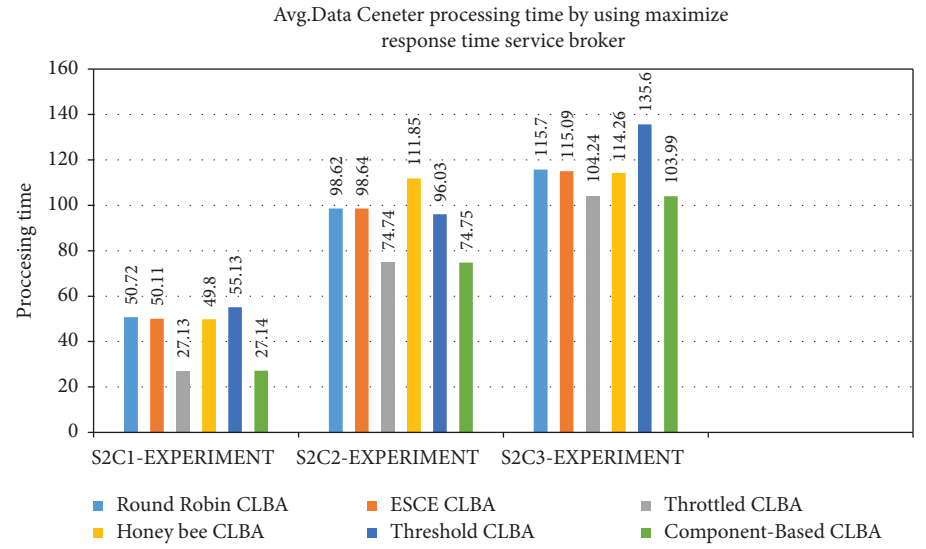

Figure 8: Avg. data center processing time by using maximize response time service broker.

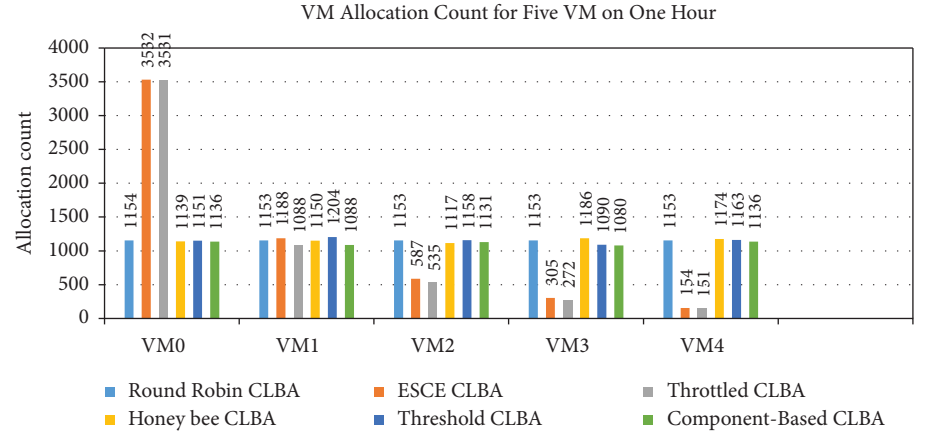

Figure 9: VM allocation count for 5 VMs for 1 hr.

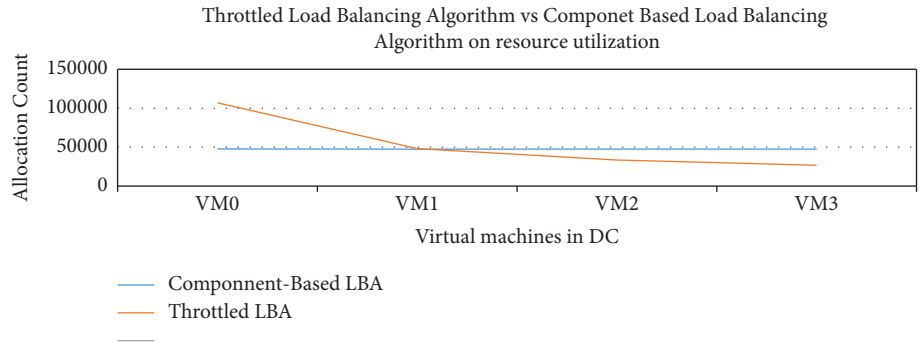

Figure 10: Throttled versus component-based on resource utilization second experiment.

Table 6: Overall comparative performance analysis of the algorithms.

Algorithms	Algorithms' relative rank based on			The overall sum of the relative ranks
	Response time	Dc processing time	Resource utilization	
Round-robin CLBA	5	5	1	11
ESCE CLBA	4	4	5	13
Throttled CLBA	2	1	5	8
Honey bee CLBA	3	3	3	9
Threshold CLBA	6	6	3	15
Component-based CLBA	1	1	2	4

The rank-based comparative analysis of the selected algorithms concludes that the component-based load balancing algorithm was found as the most efficient algorithm on the selected parameters, that is, response time, data center processing time, and resource utilization.

5. Conclusion

Cloud computing systems are extensively used by numerous business houses and private and public sector organizations. In addition to the issues and concerns such as security, fault tolerance, and scalability, load balancing is becoming the most crucial matter for the efficiency of computing over cloud-based systems. This paper focuses on resolving the issues in the load balancing environments of cloud-based VMs. In general, load balancing techniques were evolved to ensure the optimum utilization of cloud resources. This implies that none of the VMs are allowed to be kept idle, underutilized, or overloaded while operating. In this paper, the efforts are made to compare the performance of the various load balancing algorithms using selected parameters, that is, response time, data center processing time, and resource utilization. These algorithms are generalized as static and dynamic. In this paper, we proposed a newly evolved "component-based throttled load balancing algorithm" for solving the problem of the "task overhead" in the existing algorithms. The paper compared the various load balancing algorithms in different simulated experimental setups using the cloud analyst simulation tool. In the first part of the experiment, the response time and data center's processing time are experimentally measured and compared for the proposed component-based algorithm and the other aforementioned five algorithms. The first part experiment was done by varying the number of data centers and the VMs (i.e., in the heterogeneous environment) by using them in two different scenarios of the service broker policy. The second part experiment was done to find the best performing algorithm for the best resource utilization by allocating the VMs to the process efficiently. Unlike the first part, this experiment was done by varying the execution time in the simulation to 1.0 hr and 24 hr in a homogeneous environment with a fixed number of VMs.

In the first part experiment, the component-based algorithm showed a better performance in terms of the response time and the data center processing time than the throttled and other algorithms. But in the second part experiment, the round-robin algorithm performed better in allocating the VMs effectively. However, the newly proposed component-based algorithm was found to be very efficient in contrast with the existing throttled algorithm. This showed that the proposed component-based throttled algorithm is found to be more efficient than the existing throttled algorithm in allocating the VMs effectively, that is, resource utilization.

Finally, the selected algorithms were ranked based on their performance in each of the experiments. The weight-based relative ranking was used to compare and determine the most efficient algorithm. Upon parameter-based comparative analysis, the component-based throttled load balancing algorithm was identified as the most efficient algorithm based on the selected parameters, that is, average response time, average data center processing, and resource utilization. Finding and considering other parameters that could help to check the performance of the cloud load

balancing algorithms experimentally is the future work that needs to be done after this study. Moreover, enhancing this component-based throttled algorithm with optimization techniques is additional future work that needs to be done after this study.

Data Availability

The data were collected from Facebook statistics data for configuring the user base of the cloud analyst https://www.internetworldstats.com/facebook.htm (accessed October 23, 2020). The data for undergoing the comparison were obtained from the results of the simulated experiment.

Disclosure

This academic research is part of the M.S. final research thesis at Arba Minch University under the Federal Ministry of Education of Ethiopia, Arbaminch University, Ministry of Education of Ethiopia.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] P. Mell and T. Grance, "The NIST-national institute of standars and technology- definition of cloud computing," *NIST Special Publication*, p. 7, 2011.

[2] K. E. Narayana, S. Kumar, and K. Jayashree, "A review on different types of deployment models in cloud computing," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 4, no. 2, pp. 1475–1481, 2011.

[3] T. Diaby and B. B. Rad, "Cloud Computing: A Review of the Concepts and Deployment Models," *International Journal of Information Technology and Computer Science*, vol. 9, no. 6, pp. 50–58, 2017.

[4] S. M. Lanjewar, S. S. Surwade, S. P. Patil, P. S. Ghumatkar, and Y. B. Gurav, "Load Balancing in Public Cloud," 2014, https://www.iosrjournals.org.

[5] D. Yadav, D. P. Sharma, and B. Keshwani, "A study of intranet over cloud," *International Journal of New Innovations in Engineering and Technology*, vol. 7, no. 2, pp. 1–6, 2017.

[6] R. Rajeshkannan and M. Aramudhan, "Comparative study of load balancing algorithms in cloud computing environment," *Indian Journal of Science and Technology*, vol. 9, no 20, 2016.

[7] J. Muda, S. Tumsa, A. Tuni, and D. P. Sharma, "Cloud-enabled E-governance framework for citizen centric services," *Journal of Computer and Communications*, vol. 8, no. 7, pp. 63–78, 2020.

[8] M. T. Student, M. S. Siva Skandha, and M. N. Sandeep Chaitanya, "Load balancing model for cloud services based on cloud partitioning using RR algorithm pooja," *International Journal of Electronics, Communications and Computer Engineering*, vol. 6, no. 5, pp. 102–106, 2015.

[9] M. T. Student, M. S. Siva Skandha, and M. N. Sandeep Chaitanya, "Load Balancing Model for Cloud Services Based on Cloud Partitioning Using RR Algorithm Pooja," 2015, https://www.ijecce.org.

[10] M. S. Shakir and E. A. Razzaque, "Performance Comparison of Load Balancing Algorithms Using Cloud Analyst in Cloud Computing," in *Proceedings of the 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, New York, NY, USA, October 2017.

[11] D. Thazhathethil, N. Katre, J. Mane-Deshmukh, M. Kshirsagar, and A. Nadaph, "A model for load balancing by partitioning the public cloud," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 32971 page, 2007.

[12] M. O. Ahmad and R. Z. Khan, "A survey on load balancing algorithms in cloud computing," *International Journal of Autonomic Computing*, vol. 2, no. 4, p. 366, 2017.

[13] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: a big picture," *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 2, pp. 149–158, 2020.

[14] S. Gupta, A. Dixit, and H. Dev, "A study on various load balancing algorithms for response time reduction in cloud," *International Journal of Current Engineering and Scientific Research (IJCESR)*, vol. 4, no. 10, 2017.

[15] T. Adityasaisrinivas, K. Govinda, S. S. Manivannan, and E. Swetha, "Analysis of load balancing algorithms using cloud analyst," *International Journal of Recent Technology and Engineering*, vol. 6, pp. 684–687, 2019.

[16] S. Y. Mohamed, M. H. N. Taha, H. N. Elmahdy, and H. Harb, "A proposed load balancing algorithm over cloud computing (balanced throttled)," *International Journal of Recent Technology and Engineering*, vol. 10, no. 2, pp. 28–33, 2021.

[17] B. Patel and S. Patel, "Various load balancing algorithms in cloud computing," *IJARIIE*, vol. 1, no. 2, pp. 187–202, 2015.

[18] H. Yeh, D. Ph, A. Chassiakos, and D. Ph, "Comparative Analysis of Load Balancing Algorithms in Cloud Computing," 2017, https://arxiv.org/ftp/arxiv/papers/1403/1403.6918.pdf.

[19] Facebook, "Facebook World Stats and Penetration in the World - Facebook Statistics," https://www.internetworldstats.com/facebook.htm.