Threshold Based Load Balancing Algorithm in Cloud Computing

Shusmoy Chowdhury
Department of Computer Science
Missouri State University
Springfiield, USA
sc26s@missouristate.edu

Ajay Katangur
Department of Computer Science
Missouri State University
Springfiield, USA
ajaykatangur@missouristate.edu

2022 IEEE International Conference on Joint Cloud Computing (JCC) | 978-1-6654-6285-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/JCC56315.2022.00011

***Abstract*—Cloud computing has become an emerging trend for the software industry with the requirement of large infrastructure and resources. The future success of cloud computing depends on the effectiveness of instantiation of the infrastructure and utilization of available resources. Load Balancing ensures the fulfillment of these conditions to improve the cloud environment for the users. Load Balancing dynamically distributes the workload among the nodes in such a way that no single resource is either overwhelmed with tasks or underutilized. In this paper we propose a threshold based load balancing algorithm to ensure the equal distribution of the workload among the nodes. The main objective of the algorithms is to stop the VMs in the cloud being overloaded with tasks or being idle for lack allocation of tasks, when there are active tasks. We have simulated our proposed algorithm in the Cloudanalyst simulator with real world data scenarios. Simulation results shows that our proposed threshold based algorithm can provide a better response time for the task/requests and data processing time for the datacenters compared to the existing algorithms such as First Come First Serve (FCFS), Round Robin(RR) and Equally Spread Current Execution Load Balancing algorithm(ESCELB).**

***Index Terms*—Cloudanalyst, Cloud computing, Load balancing, Round Robin, Threshold policy**

I. Introduction

Cloud computing has substituted traditional computing technologies and provides services to clients at any time and any location on a pay-per-use basis [1], [2]. Many cloud computing providers like Amazon Web Services (AWS), Microsoft Azure, Google, IBM cloud, Rackspace, Red Hat, Verizon cloud, VMware [3] provide facilities for the user to access various configurable computing resources (servers, storage, networks). Users can easily access information using various computing devices such as desktops, laptops, cell phones, and tablets. The efficiency and effectiveness of cloud computing service depends on how effectively load balancing is managed.

Load Balancing is the process of assigning the tasks in an effective and efficient way using proper resource allocation and minimizing the response time of the machine. An efficient load balancing algorithm or policy tries to ensure equal work distribution among the resources. The objective of load balancing is enhancing the cloud performance by balancing the load among various resources like network links, central processing units, disk drives etc. to achieve optimal resource utilization, maximum throughput, minimum response time, and avoiding overloading of resources. Load Balancing is used in the data centers to distribute and execute the tasks among the VM's and provide faster and efficient services to clients [4].

Load Balancing algorithms play a key role in achieving the objective of cloud computing. A number of factors have to be considered when designing an efficient load balancing algorithm. The nodes in the data center as well as the data centers are geographically distributed. So, there can be factors like network delay, communication delay, distance between the distributed computing nodes, distance between user and resources, and so on. Load Balancing algorithms are often designed in such a way that a single node makes all the decisions. The biggest drawback of this approach is failure of this node. If a physical machine gets overloaded, some Virtual Machines (VMs) need to transfer to a distant location using a VM migration load balancing approach. User requirements also can change dynamically which requires executing them on heterogeneous nodes for effective resource utilization and to minimize response time. Storage management is also important for effectively utilizing the resources. On-demand availability and scalability of cloud services allows users to access these services any time and scale up or scale down quickly. Finally, the load balancing algorithm cannot be overly complex using a large number of resources.

Although we listed several challenges in designing a load balancing algorithm, most load balancing algorithms focus mostly on response time, proper resource utilization and throughput of the datacenter. User satisfaction depends on these features. The load balancing algorithms are designed in a way so that the load is distributed among the processors and maximizes the resource utilization at the same time minimizing the execution time of the tasks [5]–[7].

Utilizing fully all the available resources of parallel and distributed systems are one of the main prerequisites of load balancing. Load Balancing algorithms distribute the workload across the servers, datacenters(DCs), hard drives other computer resources and provide the cloud service providers (CSP) a mechanism to distribute application requests across multiple applications deployed in different DCs. Load Balancing algorithms can be classified as centralized or decentralized,

This material is based upon work supported by the National Science Foundation under Grant OAC-1828380

978-1-6654-6285-3/22/$31.00 ©2022 IEEE
DOI 10.1109/JCC56315.2022.00011

dynamic or static, and periodic or non-periodic.

In this paper we have used a modified threshold based load balancing algorithm as detailed in Section III to achieve better performance in cloud computing. This algorithm will mark the Virtual machines as Underloaded or Overloaded and assign tasks according to the current load on the VMs. Cloudanalyst - A CloudSim based Visual Modeler is used to simulate the real-world scenarios and to analyse the performance of the algorithm. The performance of the algorithm is compared with the most commonly used scheduling algorithms, First Come First Serve (FCFS), Round Robin (RR) and Equally Spread Current Execution Load Balancing (ESCELB). The rest of the paper is organized as follows. Section II provides the summary of related work performed using several other algorithms for load balancing in the cloud. Section III provides all the details regarding the proposed threshold based load balancing algorithm. Section IV details the system design of the proposed algorithm. The system implementation details are provided in Section V. Section VI provides the details on the experimental setup using Cloudanalyst, which is used to simluate several scenarios to evaluate the correctness and accuracy of the proposed algorithm. Section VII provides a detailed analysis of comparing the proposed threshold based load balnacing algorithm against other algorithm like FCFS, RR, ESCELB. Finally, Section VIII concludes this paper with a summary of our work and future extension of this research.

II. Related Work

Load Balancing on cloud computing has graved the attention of the researchers around the world to provide an effective and efficient cloud environment for the users. Load Balancing algorithms play an essential role in increasing the performance of the cloud datacenter.

Amandeep Kaur Sidhu et al. [8] gave a detailed analysis of distinct kinds of load balancing techniques and the challenges of these load balancing techniques in cloud computing. According to their paper load balancing algorithms follow two different classifications based on system load and system topology. Based on the system load there are three types of approach employed; static, dynamic and mixed. The system topology is classified into two types, 1)load balancing with defined static rules, 2) dynamic load balancing which adapts which adapts the load distribution to system status changes, by changing their parameters and algorithms dynamically. The authors mentioned the metrics of an efficient load balancing algorithm in the cloud such as scalability of a system with any finite number of nodes, optimized resource utilization, improved performance at a reasonable cost, minimized response time and overhead associated with each node. To design a good load balancing algorithm, one needs to consider some major goals like cost effectiveness of the technique, scalability and flexibility of the system and prioritization of the resources or tasks. The authors did not analyse any algorithm and just provided advantages and the problems associated with a number of load balancing algorithms currently used such as Round Robin, Connection Mechanism, Randomized, Equally Spread Current Execution Load Balancing Algorithm, Throttled Load Balancing Algorithm, Biased Random Sampling, Min-Min Algorithm, Max-Min Algorithm, Token Routing.

Syed Hamid Hussain Madni et al. [9] has analyzed the resource allocation methods in the cloud. The authors provided the parameters to increase the performance in a cloud. This paper gave a detailed analysis about the importance of allocating resources in the cloud, requiring resource allocation policies, strategies, and algorithms to distribute and migrate resources to best support both suppliers and user.

Katyal Mayank et al. [10] provided an excellent comparative study on distinct kinds of load balancing algorithms depending on the various aspects of the cloud computing environment. This paper provides a brief introduction to diverse types of load balancing algorithm schemes and comparison with their advantages and drawbacks. The authors here provide four interesting case scenarios, 1) one host and VM both in space sharing manner, 2) both in time sharing manner, 3) host in time sharing and VM in space sharing, and 4) VM in time sharing and host in space sharing. The authors classified the load balancing algorithms based on cloud environment, spatial distribution of nodes and hierarchy of tasks.

Shahbaz Afzal et al [11] provided a detailed review of the load balancing techniques and highlighted the crucial challenges to develop efficient load balancing algorithms.

Armstrong et. al. [12] uses Minimum Execution Time (MET) of a task to randomly assign it to the nodes on which it is expected to be executed the fastest, regardless of the current load on that node. Use of some existing scheduling techniques like Min-Min, RR and FCFS for load balancing also are presented in their paper.

Yang Xu et. al. [13] discussed an intelligent method for load balancing. It is designed with a novel model to balance data distribution to improve cloud computing performance in data-intensive applications, such as distributed data mining.

Einollah Jafarnejad Ghomi et al [14] provides background on task scheduling and load-balancing algorithms and proposes new classification based on seven categories which are 1) Hadoop MapReduce load balancing, 2) natural phenomena-based load balancing, 3) agent-based load balancing, 4) general load balancing, 5) application-oriented, 6) network-aware, and 7) workflow specific. The authors focus on studying the existing load balancing mechanisms, providing a new classification, and clarifying the advantages and disadvantages of the load-balancing algorithms in each class and finally discussing the aspects to improve load balancing algorithms.

III. Proposed Algorithm

The main goal of our algorithms is to distribute the task load equally among all the Virtual machines in the datacenter. Often, we can see that some of the Virtual machines are overcrowded with an enormous number of tasks, on the other hand some machines remain idle. As a result, the overall processing time of the DCs is increased although there are enough resources available to process the tasks. We try to

address the issue and balance the load among the nodes. We will used two kinds of threshold values in our algorithm.

- **Overload Threshold:** Overload threshold value as shown in "Fig. 1" is the no. of tasks used to define a VM as overloaded. When the VM reaches or exceeds the overload threshold we try not to assign tasks to it.
- **Underload Threshold:** The underload threshold value as in "Fig. 1" is used to track the underloaded VMs. When the no. of tasks in a VM decreases down to the underload threshold value we start assigning tasks to that VM.

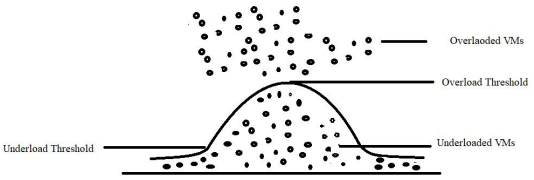

Fig. 1. Distinct kinds of VM's in Threshold algorithm

We divide the VMs into two distinct categories.

- **Underloaded VM's:** If the no. of tasks in a VM is under the underload threshold, then it is called an underloaded VM. In our algorithm we will focus more on tracking the underloaded nodes so that the task load can be distributed equally thereby not overcrowding the VMs.
- **Overloaded VM's:** Overloaded VMs are those that are crowded by enormous no. of tasks. When the no. of tasks is more than the overload threshold, we declare it as an overloaded VM. Our algorithm focuses on moving the workload of the overloaded VMs to underloaded VMs.

Although we can use single threshold value to find the underloaded and overloaded VMs, there is a motivation to use two values. With the two threshold approach we can have a safe zone where tasks are executed without constantly going through the hassle of being designated as an overloaded or underloaded VM. When the VM's exceed the overload threshold we will stop assigning tasks to that VM and find another underloaded VM. But as soon as the overloaded VM finishes one task it would become an underloaded VM if we used a single threshold value. Now as we are using two threshold values, the overloaded node will not become underloaded VM. It has to finish a certain number of assigned tasks and go down to the underload threshold value and then will be designated as an underloaded VM. It will take some time to finish the tasks and then become an underloaded VM. The overload and underload threshold values for this paper are determined by trial error process. A better way to determine the thresholds is with machine learning and adaptive algorithms which will be our future work.

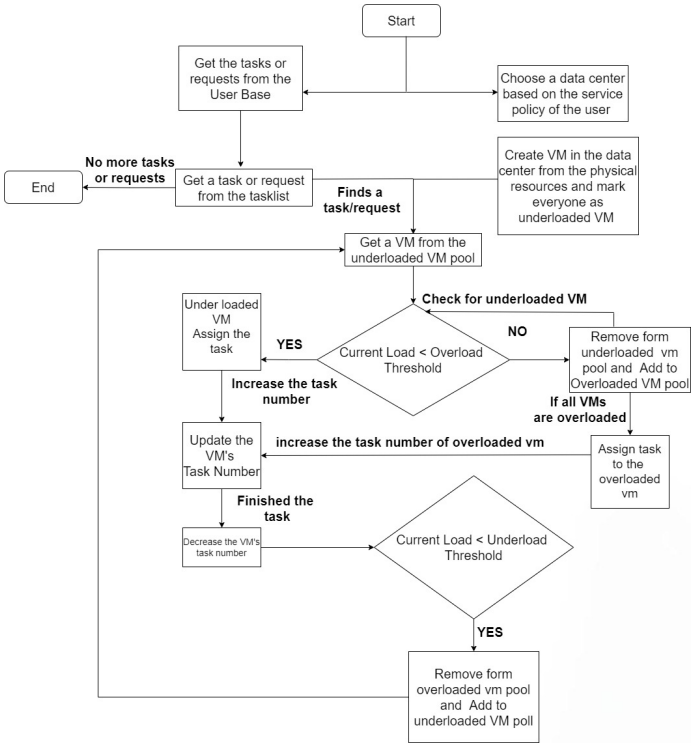

Fig. 2. Flow chart of Threshold algorithm

IV. System Design

Our proposed algorithm as shown in "Fig. 2" starts with creating the user base and the VMs in the Cloudanalyst. When the DCs are created, we mark all VMs in the datacenter as underloaded VMs as there are no tasks assigned on these VMs. When a task arrives in the datacenter, we choose a VM from the underloaded VMs. Then we check if the current VMs load is equal to or greater than the overload threshold. If the current load of the VM is less than the overload threshold, then it is an underloaded VM and we can allocate the task to it. But if the current load of the VM is equal to or greater than the overload threshold then it is an overloaded VM. If it is an overloaded VM, then we will remove it from the underloaded VM list and add it in overloaded VMs list. We then look again in the underloaded VM list for another underloaded VM to assign the task. But if there are no underloaded VMs then we will choose the overloaded VM with less task count to assign the new task. Whenever the VMs finish a task, we will decrease the task count and then check whether it is an overloaded VM. If it is an overloaded VM, we then check if the current task load is under the underload threshold value. If it is less than the underload threshold value, then we remove the VM from the overloaded VM and add it in the underloaded VMs list. The gap between the underloaded threshold value and the overload threshold value ensures the availability of the resources in the VMs and blocks the overcrowding of tasks in the VMs.

V. System Implementation

We need an appropriate simulation environment to measure the efficiency and effectiveness of a load balancing algorithm. CloudSim [15] is the one of the first tools that can be used to model cloud simulation. CloudSim allows VMs to be managed by hosts which in turn are managed by DCs. But CloudSim does not provide any user interface to create the userbases or configuration to design the datacenter. Also, it is a console-based application, so we cannot generate or store the reports of our simulations for future analysis.

Cloudanalyst [16] addresses these problems of CloudSim and therefore is used to simulate the cloud environment for testing the efficiency and effectiveness of our proposed algorithm. Cloudanalyst is built on top of CloudSim extending its features in a GUI format. It provides an opportunity to set the parameters for setting a simulation environment for research problems. So Cloudanalyst can provide a real-world simulation environment to check the effectiveness of our algorithms. Cloudanalyst also provides us with GUI based results which makes it easier for us to compare and analyse the results.

VI. Experiment Setup

A. User Base:

In Cloudanalyst the world is divided into six regions which represent the six main continents in the world as shown in Table I. The six regions are numbered from 0 to 5. We have created six userbases from each region. We also defined the peak hours for each userbase. Peak hours means the time of the day when a maximum number of users of that userbase will try to access the cloud. Our algorithm needs to be efficient while handling the enormous number of requests during that time. In Table I the number of requests at peak hour and off-peak hour are shown. The number of requests at off-peak hour will be less compared to that at peak hour.

TABLE I
User Base Dataset for the Experiment

User Base Name	Region	User Requests per Hour	Data Size per Request (Bytes)	Peak Hour Start (GMT)	Peak Hour End (GMT)	No. of Average Peak Hour Requests	No. of Average Off-Peak Hour Requests
UB1	0	12	100	13	15	400000	40000
UB2	1	12	100	15	17	100000	10000
UB3	2	12	100	20	22	300000	30000
UB4	3	12	100	1	3	150000	15000
UB5	4	12	100	21	23	50000	5000
UB6	5	12	100	9	11	80000	8000

B. Datacenter Configuration

Datacenters are an especially important part of cloud computing. The VMs in the DCs are responsible for performing tasks in the cloud environment. In Cloudanalyst, for this paper we have configured the DCs to align with real world scenarios. Datacenter is nothing but a combination of physical servers. In cloud computing the datacenter creates an abstraction over the physical hardware. It is called Virtual Machine (VM). The VM's configuration can either be the same as the physical servers or can be created with less resources to create more VM's over the physical hardware. We cannot create more VM's than the available resources or physical servers. Tasks are scheduled onto the VM's using the load balancing algorithms. The datacenter for this paper has been configured with the following specifications:

- Region (The region where the datacenter is located): 0-5
- Architecture (Architecture of the servers used in the data center): X86
- Operating System: Linux
- Virtual Machine Monitor (VMM): XEN hypervisor
- Cost per VM Hour: $ 0.1
- Cost per 1Mb Memory Hour: $ 0.05
- Storage cost per 1 GB Hour: $ 0.1
- Data Transfer cost per Gb: $ 0.1
- Number of Physical servers: 20

We will use twenty physical servers in every datacenter we use. Each physical server is configured as defined below:

- Memory: 2048 MB (2 GB)
- Storage: 100000 MB (100 GB)
- Available Bandwidth: 10000 Mbps (10 Gbps)
- Number of Processors: 4
- Processor Speed: 100 Mhz
- Virtual Machine Policy: Time Shared/ Space Shared

We will use the same configuration for all the physical servers. The VMs are created on these physical servers.They will all have the same memory 1024 MB (1 GB), 10000 MB (10 GB) image size and 1000 Mbps (1 Gbps) bandwidth. The load balancer selects an appropriate VM based on the task load for optimal cloud computing performance.

Amazon Web Services [17] is a widely used cloud platform across the world. Many companies [18] as well as the developers use the Amazon Elastic Compute Cloud (AWS EC2) for deploying their applications. Most of them [19] use the free tire instances of the AWS EC2. The overall datacenter in our experiments reflects the free tier configuration of AWS EC2. Many users all round the world choose this configuration to deploy small applications in cloud. So, our datacenter configuration chosen is very similar to real world datacenter configuration.

C. Service Broker Policy

Service broker policy defines how a connection is established between the userbase and the DCs. Service broker policies defines the rules for the userbase to chose a datacenter. There are two service broker policies available in the Cloudanalyst platform. In this paper we use the closest datacenter service broker policy. In this service broker policy, the user base will choose the nearest datacenter available. If there is a datacenter available in the same region of the userbase, it will choose the datacenter irrespective off the task load or the response time of the datacenter. In other cases, the userbase will calculate the distance and choose the nearest datacenter.

TABLE II
AVERAGE RESPONSE TIME (RT) IN MILLISECONDS USING ONE DATACENTER (CLOSEST DATACENTER SERVICE BROKER POLICY).

Configuration Name	Datacenter Configuration	RT using Threshold	RT using RR	RT using FCFS	RT using ESCELB
CC1	25 VMs	12733.38	16533.76	108241212.21	16531.72
CC2	50 VMs	6024.58	7965.21	10824125.67	7965.50
CC3	75 VMs	3939.70	5242.87	10824120.43	5242.87

TABLE III
AVERAGE DATA PROCESSING TIME (PT) IN MILLISECONDS USING ONE DATACENTER (CLOSEST DATACENTER SERVICE BROKER POLICY).

Configuration Name	Datacenter Configuration	PT using Threshold	PT using RR	PT using FCFS	PT using ESCELB
CC1	25 VMs	12439.64	16221.21	108241212.21	16219.15
CC2	50 VMs	5762.75	7692.23	10824125.67	7692.53
CC3	75 VMs	3688.10	4983.51	10824120.43	4983.53

VII. RESULTS AND EVALUATION

Table II and Table III shows the comparison of average response time of the tasks and average data processing time of one datacenter for the proposed threshold based algorithm and the other algorithms. We have used the user base configuration mentioned in the experiment setup. In this case we used a single datacenter containing 25, 50 and 75 VMs to process the tasks or requests of the userbases. The datacenter is in region 0. From Tables II and III we can see that if we increase the number of VMs the response time as well as the data processing time decreases. In this scenario the response time and data processing using 50 VMs is almost half compared to the time using 25 VMs. So, with more VMs we can get more improved performance in the cloud. From Table II and Table II it is clear that our proposed threshold based load balancing algorithm outperformed the existing load balancing algorithms such as RR, FCFS and ESCELB. From the results, we can see that FCFS response time and data processing times compared to threshold based load balancing algorithm as well as the other load balancing algorithms are very high. For this reason, in this paper we have decided to not consider FCFS load balancing algorithm for the rest of our evaluation scenarios. Table IV and Table V shows the results of average

TABLE IV
AVERAGE RESPONSE TIME (RT) IN MILLISECONDS USING TWO DATACENTERS (CLOSEST DATACENTER SERVICE BROKER POLICY).

Configuration Name	Datacenter configuration	RT using Threshold	RT using RR	RT using ESCELB
CC1	DCs with 25 VMs	12811.41	15277.95	15278.45
CC2	DCs with 50 VMs	6030.81	7628.07	7627.89
CC3	DCs with 75 VMs	3901.94	5066.38	5066.43
CC4	DCs with 25 and 50 VMs	10195.31	12215.88	12216.30
CC5	DCs with 50 and 75 VMs	5235.28	6589.05	6589.18
CC6	DCs with 25 and 75 VMs	5217.67	11177.46	11177.28

response time and data processing time of the tasks using three DCs. From the results, it is obvious that using two DCs does not greatly improve the performance of the load balancing algorithms. This is due to the service broker policy. The service broker policy provides the rules and norms for choosing a datacenter by the userbases. We have used the

TABLE V
AVERAGE DATA PROCESSING TIME (PT) IN MILLISECONDS USING TWO DATACENTERS (CLOSEST DATACENTER SERVICE BROKER POLICY).

Configuration Name	Datacenter configuration	PT using Threshold	PT using RR	PT using ESCELB
CC1	DCs with 25 VMs	12612.21	15068.82	15069.32
CC2	DCs with 50 VMs	5872.29	7461.07	7460.89
CC3	DCs with 75 VMs	3756.34	4914.23	4914.26
CC4	DCs with 25 and 50 VMs	10007.24	12018.99	12019.40
CC5	DCs with 50 and 75 VMs	5080.01	6426.66	6426.79
CC6	DCs with 25 and 75 VMs	9157.97	10985.16	10985.15

closest datacenter service broker policy where the userbase looks for the nearest DCs and assigns the tasks there. So, the total number of tasks is divided between these DCs. But as the userbase peek time does not overlap with one another, the performance is almost same as that of the single datacenter. But as stated earlier, the greater number of VMs increases the performance of the load balancing algorithm. From the results in Table IV and Table V it is clear that two DCs with 75 VMS on each produces the best response time as well as the best data processing time. From Table IV and Table V it is clear that the proposed threshold based load balancing algorithm outperforms all other algorithms.

TABLE VI
AVERAGE RESPONSE TIME (RT) IN MILLISECONDS USING THREE DATACENTERS (CLOSEST DATACENTER SERVICE BROKER POLICY).

Configuration Name	Datacenter Configuration	RT using Threshold	RT using RR	RT using ESCELB
CC1	DCs with 25 VMs	11323.38	13703.41	13702.71
CC2	DCs with 50 VMs	5361.71	6645.13	6645.18
CC3	DCs with 75 VMs	3461.46	4451.34	4451.24
CC4	DCs with 25, 25 and 50 VMs	11321.18	13697.68	13698.39
CC5	DCs with 25, 50 and 50 VMs	9367.58	11463.96	11464.07
CC6	DCs with 25, 25 and 75VMs	11338.31	13696.80	13696.72
CC7	DCs with 25, 75 and 75 VMs	8737.38	10734.06	10734.31
CC8	DCs with 50, 50 and 75 VMs	5371.60	6643.52	6643.61
CC9	DCs with 75, 75 and 50 VMs	4274.63	5914.52	5914.63
CC10	DCs with 25, 50 and 75 VMs	9294.84	11463.17	11463.14

TABLE VII
AVERAGE DATA PROCESSING TIME (PT) IN MILLISECONDS USING THREE DATACENTERS (CLOSEST DATACENTER SERVICE BROKER POLICY).

Configuration Name	Datacenter Configuration	PT using Threshold	PT using RR	PT using ESCELB
CC1	DCs with 25 VMs	11176.48	13545.48	13544.78
CC2	DCs with 50 VMs	5244.05	6521.03	6521.09
CC3	DCs with 75 VMs	3353.24	4337.64	4337.54
CC4	DCs with 25, 25 and 50 VMs	11174.34	13539.75	13540.45
CC5	DCs with 25, 50 and 50 VMs	9230.16	11316.41	11316.51
CC6	DCs with 25, 25 and 75VMs	11191.21	13538.85	13538.78
CC7	DCs with 25, 75 and 75 VMs	8602.83	10589.94	11315.57
CC8	DCs with 50, 50 and 75 VMs	5253.79	6519.41	6519.50
CC9	DCs with 75, 75 and 50 VMs	4610.00	5793.88	5793.98
CC10	DCs with 25, 50 and 75 VMs	9157.74	11315.59	11315.57

Table VI and Table VII shows the results of average response time and data processing time of the tasks using three DCs. We can see that the results are almost identical with the results of the two DCs. Increasing the VM numbers in DCs increases the performance of the DCs.

Table VIII provides a comparison of the Response Time and Data Processing Time for various datacenter configurations. The response times for one data center and two data centers are

TABLE VIII
COMPARISON OF RESPONSE TIME AND DATA PROCESSING TIME IN THRESHOLD BASED ALGORITHM FOR DIFFERENT DATACENTER WITH 75 VMS.

	One Datacenter	Two Datacenter	Three Datacenter
Response Time	3939.7	3901.94	3461.46
Data Processing time	3688.10	3756.34	3353.24

identical, but the response time is greatly improved with three data centers. "Fig. 3" gives us the graphical representation of our hypothesis. The graph curves downward for three DCs. All the previous experiments are conducted by choosing the closest datacenter service broker policy.

Fig. 3. Comparison of response time in Threshold based algorithm for different Datacenter with 75 VMs

VIII. CONCLUSION

In this paper a threshold based load balancing algorithm is designed and implemented for cloud computing for achieving a promising tendency towards solving high demanding applications and many kinds of problems. The main objective of the threshold based load balancing algorithm is to achieve high performance in cloud computing using real-world scenarios. In the proposed threshold based load balancing algorithm, we used two pre-determined threshold values, overload threshold and underload threshold to determine the overloaded and underloaded VMs. The uniqueness of using two threshold values is to ensure efficient utilization of the VMs. An overloaded VM has to finish a certain number of assigned tasks and then go down to the underloaded threshold value for it to be designated as an underloaded VM. Analysis of the results shows that our proposed algorithm has outperformed the most commonly used load balancing algorithms like round robin, first come first serve and equally spread load balancing algorithms. The threshold based load balancing algorithm performs better when there are a large number of tasks, where other load balancing algorithms fail because of the large number of tasks on the VMs.

In future extension of the algorithm, we can use machine learning techniques to find out the optimal threshold values. We can analyze the total assigned tasks and use machine learning techniques like super position, neural networks, deep learning etc. to find out the overload and underload threshold values. We can dynamically change the threshold values for the overloaded and underloaded VMs with the progress of the simulations. We will also try to combine multiple service broker policies with our algorithm to obtain even better results. We will also implement fault tolerance policy on the algorithm so that the algorithm can automatically recover if it fails for huge workloads.

REFERENCES

[1] E. Brown, "Different types of cloud computing service models," 2015. [Online]. Available: https://www.bluepiit.com/blog/different-types-of-cloud-computing-service-models/.

[2] R. Buyya, J. Broberg, and A. M. Goscinski, *Cloud computing: Principles and paradigms*. John Wiley & Sons, 2010.

[3] "Top 10 cloud computing service providers in 2017," 2017. [Online]. Available: https://www.technavio. com/blog/top-10-cloud-computing-service-providers-2017

[4] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, and S. Dam, "A genetic algorithm (ga) based load balancing strategy for cloud computing," *Procedia Technology*, vol. 10, pp. 340–347, 2013.

[5] S. H. Bokhari, "On the mapping problem," *IEEE Trans. Computers*, vol. 30, no. 3, pp. 207–214, 1981.

[6] S. Salleh and A. Y. Zomaya, *Scheduling in parallel computing systems: fuzzy and annealing techniques*. Springer Science & Business Media, 2012, vol. 510.

[7] A. Y. Zomaya, "Parallel and distributed computing: The scene, the props, the players," *Parallel and Distributed Computing Handbook*, vol. 1, no. 1, pp. 5–23, 1996.

[8] A. K. Sidhu and S. Kinger, "Analysis of load balancing techniques in cloud computing," *International Journal of computers & technology*, vol. 4, no. 2, pp. 737–741, 2013.

[9] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, "Recent advancements in resource allocation techniques for cloud computing environment: a systematic review," *Cluster Computing*, vol. 20, no. 3, pp. 2489–2533, 2017.

[10] M. Katyal and A. Mishra, "A comparative study of load balancing algorithms in cloud computing environment," *arXiv preprint arXiv:1403.6918*, 2014.

[11] S. Afzal and G. Kavitha, "Load balancing in cloud computing–a hierarchical taxonomical classification. j. cloud comput. 8 (1), 1–24 (2019)."

[12] R. Armstrong, D. Hensgen, and T. Kidd, "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions," in *Proceedings Seventh Heterogeneous Computing Workshop (HCW'98)*. IEEE, 1998, pp. 79–87.

[13] Y. Xu, L. Wu, L. Guo, Z. Chen, L. Yang, and Z. Shi, "An intelligent load balancing algorithm towards efficient cloud computing," in *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[14] E. J. Ghomi, A. M. Rahmani, and N. N. Qader, "Load-balancing algorithms in cloud computing: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 50–71, 2017.

[15] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[16] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *2010 24th IEEE international conference on advanced information networking and applications*. IEEE, 2010, pp. 446–452.

[17] S. Mathew and J. Varia, "Overview of amazon web services," *Amazon Whitepapers*, 2014.

[18] M. Cunha, N. Mendonca, and A. Sampaio, "Investigating the impact of deployment configuration and user demand on a social network application in the amazon ec2 cloud," in *2011 IEEE Third International Conference on Cloud Computing Technology and Science*. IEEE, 2011, pp. 746–751.

[19] J. Xiong, S.-H. Shi, and S. Zhang, "Build and evaluate a free virtual cluster on amazon elastic compute cloud for scientific computing." *International Journal of Online Engineering*, vol. 13, no. 8, 2017.