


A Multi-Agent-Based VM Migration for Dynamic Load Balancing in Cloud Computing Cloud Environment

Soumen Swarnakar, Netaji Subhash Engineering College, India*

 <https://orcid.org/0000-0001-6699-347X>

Chandan Banerjee, Netaji Subhash Engineering College, India

Joydeep Basu, Netaji Subhash Engineering College, India

Debanjana Saha, Netaji Subhash Engineering College, India

ABSTRACT

Cloud computing is the use of remote servers on the internet to store, manage, and process data. It is a demand-based service where users need to pay only for what they use. Cloud computing users are extensively distributed throughout the globe, so it is a big challenge to keep track of this huge data. Load balancing is the distribution of workloads in a smart way among multiple compute resources, like virtual servers. Compute resources can be added or removed from the load balancer according to the needs of the user. A load balancer is primarily used to optimize the use of resources, costs, and VMs, as well as to maximize throughput, reduce response time, and prevent overloading in various VMs. In this paper, multi-agent-based virtual machine migration has been proposed for dynamic load balancing in a cloud computing environment. The proposed algorithm shows better results in terms of makespan time, average response time, and data center processing time than other conventional cloud load balancing algorithms.

KEYWORDS

Cloud Computing, Dynamic Load Balancing, Efficient Load Balancing, Multi Agent, Resource Allocation, Virtual Machines, VM Migration

1. INTRODUCTION

Distributed computation is gaining popularity in this era where IT is growing rapidly. We need both the piling of massive data and efficient management, and here comes the role of cloud computing. A few characteristics of cloud computing are on-demand service, scalability, resource pooling, rapid elasticity, etc. Cloud computing is used everywhere, from small firms to big enterprises. Jobs

DOI: 10.4018/IJAC.320479

*Corresponding Author

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

are mainly assigned based on algorithms like FCFS, Weighted least connection, Round Robin etc. Though these algorithms work well, a few problems are still there for which it takes more time to perform the desired work. Many papers have already come up with solutions for doing the task faster. Many researchers have experimented with different algorithms for the optimization of load balancing systems. In this paper, we came up with a dynamic approach that takes multiple agents by using which tasks are dynamically distributed to different VMs of different configurations in different data centers. Some time it is seen that when tasks are coming to VMs of different configurations, some of VMs are overloaded or some are under loaded. When tasks are trying to enter in an overloaded VM, it will take more time to get the resource; as a result delay in execution happens. Therefore there is a necessity for migrating tasks from one VM to another underloaded VM dynamically. In this research paper, an improved algorithm has been proposed with a dynamic approach for cloud load balancing taking multiple agents' like remaining time, task load, and capacity of VM into consideration. The remaining section of our research work is divided into four different sections. Section II describes the summary of Related Works. In Section III, the proposed architecture for the load balancing algorithm is discussed. Section IV describes the simulation results of proposed algorithm and its comparison with some existing algorithms. Finally, a conclusion has been drawn in section V.

2. RELATED WORK

Many research-based works have been done on the cloud. In this section, a few research papers among them will be discussed, which were designed previously. In the algorithm, incoming tasks were allocated in different VMs of different processing speed at different data centers by using dynamic tables. The simulation was done using a cloud analyst tool.

Hung et al. (2021) presented a two-phase genetic method for the migration-based load balancing of virtual machine hosts in cloud computing. Gene expression programming has been applied for describing virtual machines' performance and has been used for load prediction of virtual machine hosts after balancing load. A genetic algorithm was used to compare current and future loads on virtual machine hosts in order to migrate VMs for load balancing. The simulation has been done on the Jnet cloud environment. The simulation results showed better results than the previously described methods in that paper.

Sefati et al. (2021) proposed a new algorithm for load balancing in a cloud environment based on the capability of resource reliability. Here, at first, the algorithm attempts to find the idle or busy nodes, and after finding this node, each node's threshold value and fitness function are calculated for task allocation. The simulation results using CloudSim revealed that the projected method's response time and costs are lower than the other described methods.

Swarnakar et al. (2020) proposed an algorithm for calculating average response time and makespan time for all incoming tasks and was compared with load balancing algorithms which were designed previously. In the algorithm, incoming tasks were allocated to different VMs of different processing speeds at different data centers by using dynamic tables. It helped us by giving us the concept of dynamic tables, and by using that concept, we proposed an algorithm which is more efficient than other algorithms.

Shahapure and Jayarekha (2018) proposed an algorithm for resource management with load balancing using virtual machines. The algorithm helps to select VMs for load migration considering the minimum migration. This helps in reducing execution time and increasing scalability.

Zhang et al. (2016) proposed a distributed load balancing concept depending on multi-agents. The proposed concept reduces the quantity of workload data transmission and the load of the VM scheduler.

Mishra et al. (2012) stated that one way to maintain the computing as a service vision in cloud-based solutions is through virtualization. Live VM migration helps in allocating the current state of one VM from one PM to another. It also detects overloaded states and enables continuous maintenance activities. Virtualization fulfils the requirements of a cloud-based solution. It gives a virtualized representation of resources used to create VMs.

Panwar and Mallick (2015) proposed a method for allocating arriving jobs to different VMs that is based on the dynamic load management technique. Jobs are assigned on the basis of the load assigned in the VM. At first, the VM with the minimum load is assigned the job. One drawback of that algorithm is that every time while allocating a new job, all the VMs are considered and checked, which increases the response time. Our proposed algorithm solves this difficulty and provides a better response time by providing dynamic tables in the load balancer for updated conditions of different virtual machines in data centers.

Liaquat et al. (2016) discussed various VM migration techniques. A queue-based migration model was proposed and discussed to migrate VM memory pages in an efficient way. The VM process helped to reach various management goals, like load balancing, fault tolerance, and green cloud computing. VM migration is not free and it consumes a certain amount of sender and receiver resources for a successful migration process.

Afzal and Kavitha (2019) suggested an algorithm for load balancing in cloud environments, keeping in mind the future challenges. They reviewed and compared various existing algorithms and guided future researchers to deal with unbalanced load problems like ML, nature-inspired algorithms, and derived mathematical algorithms.

Beri and Behal (2015) surveyed about cloud computing and explained various service models in cloud computing. It helped us to analyse various categories of service provided. The study provided a brief explanation of the layered architecture of cloud and helped us to evaluate that how cloud services work.

Zongyu and Xingxuan (2015) discussed a modified Round Robin algorithm in web servers based on clusters. Their proposed algorithm showed better results compared to the conventional Round Robin algorithm, as the proposed algorithm has decreased the load range significantly and also reduced the load variance in web servers.

3. PROPOSED WORK

The aim of the proposed procedure is to minimize the execution time and increase the scalability of the VMs. In this proposed architecture, we are using VM migration based on multi-agents to consider the available constraints and without degrading the performance. VMs change their resources dynamically, so it leads to discrepancies in the resource utilisation of the PMs. So some of the VMs get overloaded and some stay underloaded. Therefore, the proposed system will rectify that and will maximize the resource utilisation and minimize the turnaround time. This proposed algorithm for balancing load will involve migration of jobs from a highly loaded VM to an underloaded or less loaded VM and also considers if the migration is required based on the constraints.

In this proposed system, two types of VM are taken into consideration: one type of VM with a higher configuration and another type of VM with a lower configuration. The higher configurations VMs have more processing speed than the lower ones. The higher configuration VMs are VM_{Hi} where $i = 1, 2, 3, \dots, n$ and the lower configuration VMs are VM_{Lj} where $j = 1, 2, 3, \dots, n$. The load balancer consists of three dynamic tables, which hold the status of all the VMs. Now the first dynamic table will hold the details and availability of specific types of VMs in the second and third dynamic tables. The second dynamic table holds the list of available higher configuration VMs along with their data centre IDs. The third table holds the details of lower configuration VMs along with their data centre IDs. The objective of the proposed algorithm will be to minimize the response time and get a better average makespan time. The algorithm is designed in such a way that it will reduce unnecessary time consumption if a job is processed in a lower configuration VM and eventually a higher configuration VM gets underloaded. That is why the resources will be managed by the VM migration technique if necessary. This mechanism will make the mechanism scalable and also reduce the makespan time and average response time as the most efficient VMs will be utilized optimally. This paper has described a VM migration based dynamic algorithm and it basically consisting of three sections. The sections are described as below:

- a. **Load Monitoring Section (LMS):** The details of every VM, irrespective of their configuration, will be stored and indexed here. It will predict whether a VM is overloaded or underloaded based on the threshold value calculated by mathematical formulae shown in section 3.1, section 3.2. If a VM is above the threshold, then it will be marked as overloaded, and if it is below the threshold value, then it will mark that specific VM as underloaded.
- b. **Load balancing module:** This load balancer has a important role for distribution the jobs to different virtual machines of different configurations connected with different data centers. A migration-based algorithm is maintained here to distribute and migrate jobs from overloaded VM to under loaded VMs. Existing availability of different configuration VMs are maintained in different dynamic tables as described in section 3.3. The load balancer will distribute the loads to different VMs or migrating the loads for shifting jobs from lower configuration VM to higher configuration VM according to the rules set in the algorithm as described in section 3.4 with the help of different dynamic tables shown in section 3.3 showing current availability of VMs of different configurations connected with different data centers. It will distribute the load evenly according to the architecture. A load balancer will run in the Physical machine. It calculates the data transmission rate, processing speed, and RAM usage of the VMs.
- c. **VM Migration Section:** Its job is to migrate the extra load from overloaded VMs to underloaded ones or whenever there is an unutilized higher configuration VM and the job execution is going on at a lower level. It checks the list of higher-configured VMs and calculates the time taken for migration. If the migration time is longer than the calculated threshold time, then the migration will affect the overall performance of the mechanism. Then this module will not execute the migration. The mathematical explanation for checking a VM is overloaded or under-loaded is shown in section 3.1 and mathematical explanation for the condition to migrate from lower configuration VM to higher configuration VM is shown in section 3.2.

A load balancing algorithm using VM migration needs to decide which VMs are overloaded and which ones are under loaded. Some of the important formulas are as follows:

$$PMload = \sum_{i=1}^n \frac{VMcpu}{n} + \frac{VMbw}{n} + \frac{VMmem}{n}$$

$$VMcpu = \frac{total\ required\ mips}{total\ mips\ of\ PM}$$

Mathematical formulation used in this paper is described as below

Symbols	Meaning
CPU	Millions of instructions processed per seconds
Mem	Memory
Bw	Bandwidth
PM _{load}	Load of physical machine
Max _{th}	Maximum threshold
Min _{th}	Minimum threshold
T _{mig}	Time required for migration
T _{res}	Time for copying the resources
Mig _{data}	Migration data
T _{ef}	Effective time
VM _{util}	Total VM utilized
T _{th}	Time threshold

$$VMbw = \frac{bw \text{ used by } VM}{total \text{ bw of } PM}$$

$$VMmem = \frac{RAM \text{ used by } VM}{total \text{ RAM of } PM}$$

The resource utilization r of a VM i is indicated by U_i^r . So, $U_i = VM_{cpu} + VM_{bw} + VM_{mem}$
The migration will be decided by the CPU utilization, the bandwidth, the memory.

$$T_{cpu} = \frac{\sum_{i=1}^n VM_{cpu}}{total \text{ mips of } PM}$$

$$T_{bw} = \frac{\sum_{i=1}^n VM_{bw}}{total \text{ bw of } PM}$$

$$T_{ef} = \frac{T_{cpu} + T_{bw} + T_{ram}}{3}$$

Maximum Threshold $\Rightarrow \text{Max}_{th} = U_i^r - T_{ef}$

Minimum Threshold $\Rightarrow \text{Min}_{th} = U_i^r - 1$

Now let C_i^r be the capacity currently utilized by a VM So, $VM_{util} = U_i^r - C_i^r$

3.1 Mathematical Explanation to Decide if a VM is Overloaded or Underloaded

If $VM_{util} > \text{Max}_{th}$, the VM is overloaded. The next job is to be distributed among the other VMs.

If $VM_{util} < \text{Min}_{th}$, then the VM is underloaded then no need of shifting instead it can execute the incoming job.

3.2 Mathematical Explanation to Decide for Migration

Each VM records a T_{mig} , which means the last time the VM was migrated to a physical machine. (If no migration occurred, consider 0).

Now, if a specific VM is migrated frequently, then the service quality will be affected. Also, if a job which is residing already in a lower configuration VM and meanwhile a higher configuration VM gets under loaded, then load balancer will check if it is profitable to migrate the VM to higher one. If not, then the migration process will not take place. To decide whether migration will take place or not, we have to set a time threshold, T_{th} .

$$T_{mig} = T_{res} + Mig_{data}$$

$$Mig_{data} = \text{Max}_{th} - PM_{load}$$

$$T_{res} = T_{ef} + PM_{load}$$

So, let T_{now} be the current time which is used for the processing of the job under the specific VM with lower configuration. Now, even if there is a higher configuration VMs available the load balancer will check the following formula:

$$T_{now} + T_{mig} + T_{des} < T_{scr}$$

$$\Rightarrow T_{now} + T_{mig} < T_{scr} - T_{des}$$

$$\Rightarrow T_{now} + T_{mig} < T_{th} \text{ (Considering } T_{scr} - T_{des} \text{ as } T_{th})$$

Where T_{dest} is the time taken for the job completion in the destination VM (i.e migration happened) and T_{src} is the total time the VM (where the job is currently residing) will take to finish the job without migration. The $T_{src} - T_{dest}$ is actually the difference in performance between the higher configuration VMs and lower configuration VMs. This can be decided by taking the average of the execution time of the specific type VMs. Thus, we can generalise a time threshold for all the processes.

So, the condition is the load balancer will check if $T_{now} + T_{mig}$ is greater or lesser than T_{th} .

If $T_{now} + T_{mig} < T_{th}$, then migration will take place and it will make the procedure more efficient.

If $T_{now} + T_{mig} > T_{th}$, then the migration will cause unnecessary delay so migration will be avoided.

3.3 Dynamic Tables Used in Proposed Work

In this proposed algorithm, the jobs let $p=1,2,3$, n will come to different virtual machines where these VMs are of different efficiencies. Among them, some are categorized as higher configuration VMs and some as lower configuration VMs connected in different data centers. A list will be created which will keep the records of the higher configuration VMs in a sorted fashion according to their current workload.

Suppose the snapshot of details of available VMs is shown in Table 1, details of higher configuration VMs are shown in Table 2 and details of lower configuration VMs are shown in Table 3. These tables will be dynamically changed in respect of time to fulfill the allocation of incoming jobs in different VMs of different configuration in different data centers. The use of these dynamic tables have been shown in section of proposed algorithm in section 3.4.

Table 1.
Details of available VMs

Table number	Available VMs
Dynamic table II	3
Dynamic table III	6

Table 2
Details of higher configuration VMs

Current available virtual Machine ID	Data Center ID
VM_{H1}	1
VM_{H4}	2
VM_{H6}	3

Table 3.
Details of lower Configuration VMs

Current available virtual Machine ID	Data Center ID
VM_{L1}	1
VM_{L4}	1
VM_{L6}	2
VM_{L11}	3
VM_{L12}	6
VM_{L20}	8

3.4 Proposed Algorithm: Multi Agent Based Dynamic Load Balancing (MADLB)

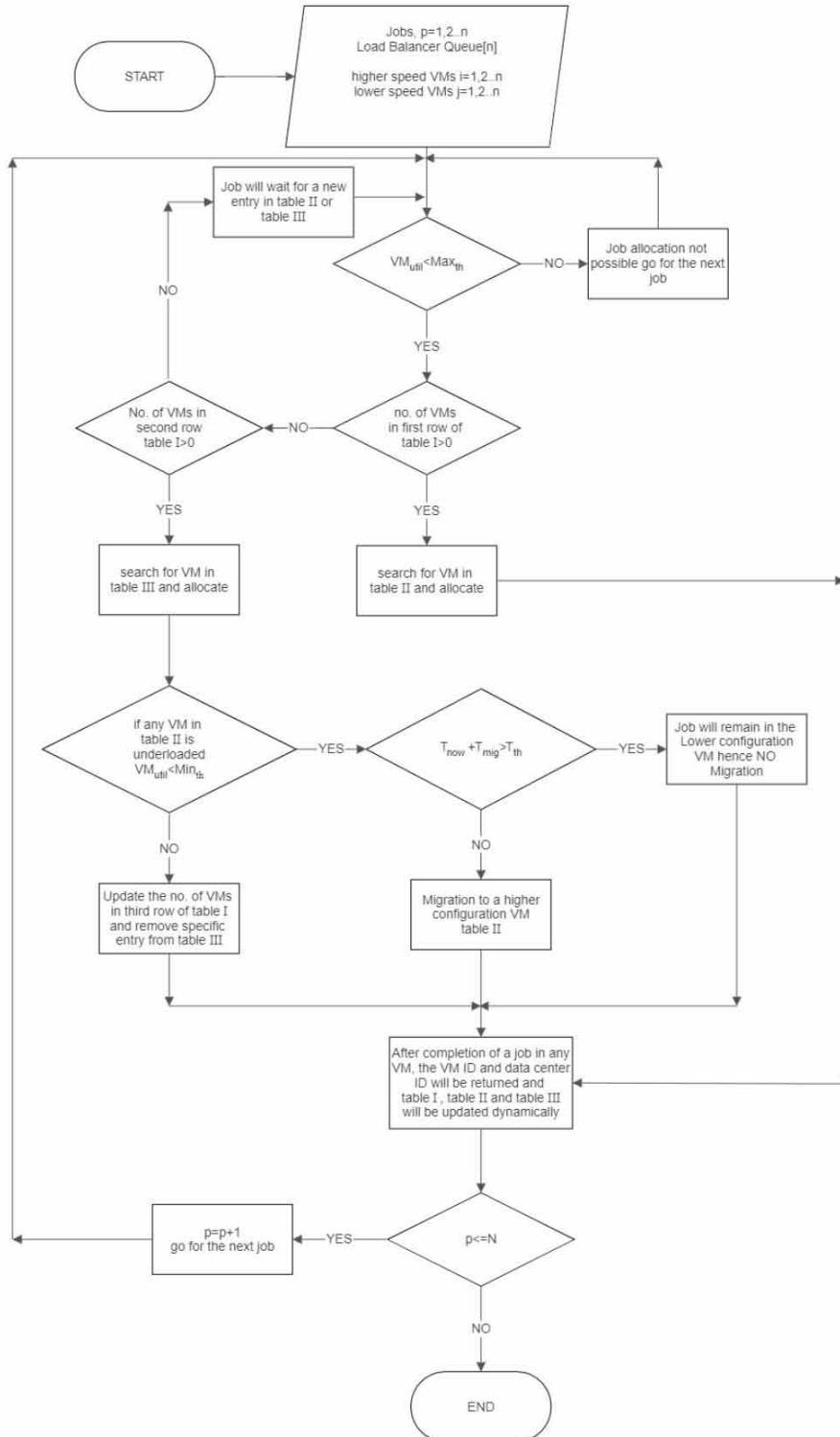
```
a.  START
b.  Jobs will come to load balancer queue.
c.  Load balancer will distribute the jobs to different data
    centers connected with different virtual machines of different
    configurations with the help of Dynamic
    Table 1 which is basically consisting of total available VMs in
    different data centers. Now, the incoming jobs will be disbursed
    to higher or lower configuration VMs as described below.
d.  At first the load balancer will check for availability of VMs
    in Dynamic Table 1.
    While (VMutil < Maxth) is true for every VM, do the
    following:
        i) if (number of VMs in first row of table 1 >0):
            The load balancer will search for available VM ID in Dynamic
            Table 2, will allocate the job through available data center ID
            and will remove the specific entry from table 2.
        ii) else:
            The load balancer will search if the number of VMs in second row
            of Table 1 >0:
            The load balancer will search for an available VM ID and Data
            Center ID in Dynamic Table 3.
            if (in case of the job already residing in a VM):
                The load balancer will check if any VM in Table 2 is under
                loaded according to VMutil < Minth is false.
            else:
                if (Tnow + Tmig >Tth):
                    Then the job will remain in the lower configuration VM and
                    migration will not take place.
                else:
                    The workload will be migrated to a higher configuration VM in
                    Table 2, the load balancer will search for an ideal higher
                    configuration VM from the sorted list of higher configuration VMs
                    and will allocate it to the most idle one.
e.  After completion of execution of the job, the corresponding
    VM ID and data center ID will be fetched by the data center and
    will add or remove rows dynamically in corresponding tables.
f.  If incoming jobs are still available in the load balancing
    queue, then repeat the process 4.
g.  END
```

4. EXPERIMENTAL RESULT

The simulation is carried out using Cloud Analyst tool. It is a graphical user interface based on Cloud Sim Architecture.

The load balancing algorithms are written in Java. The Cloud Sim architecture is used by cloud analyst to control the data centers. Cloud Analyst is a GUI-based tool for configuring any geographically spreaded system. Using Cloud Analyst simulation, results can be generated in different form like charts, tables, and execution times etc.

Figure 1.
Flow diagram of proposed algorithm



Now the simulation of our proposed algorithm is done in Cloud Analyst tools and is compared with the different existing algorithms like Round Robin, Equally spread current execution, and Throttled Load Balancing algorithm. Then the cloud analyst is configured accordingly. Six user bases situated at six different regions of the world are considered, and the duration of the simulation is kept to 60 minutes. The User base configurations are shown in figure 2. Three Data centers are selected in different regions of the world to process the request of the users. These Data Centers offer services to the User Bases. In each of the data centers, 10 virtual machines are used with 2048 MB of memory in each. The user base configuration and configuration of the data centers are shown in figure 2 and figure 3. Thereafter, the proposed load balancing policy is selected from advanced configuration settings and executed, which is shown in figure 4. The simulation with selected user bases and the datacenters are shown in the world map figure 5.

After the above mentioned configuration is simulated according to the new proposed load balancing algorithm, the data is collected from the simulation results and is compared against the existing load balancing policies. With the help of the CloudAnalyst tool, execution has been done. The duration of the simulation is 60 minutes, with a different number of tasks of different lengths taken randomly. The simulation is conducted with 200, 300, and 500 randomly chosen tasks of different lengths for the existing algorithms and our proposed algorithm. The simulation results are drawn based on average overall response time, average data center response time, and average makespan time to plot different graphs comparing the performance of the proposed algorithm with the existing ones. The average response time, average data center response time, and makespan time plotted graphs are shown in figures 6, 7, and 8 respectively.

After getting the simulation results and comparing the results with other existing algorithms, it can be decided that our proposed algorithm performs comparatively better than the existing algorithms in every aspect of average response time, data center processing time, and makespan time.

Figure 2.
User base configurations

Configure Simulation

Main Configuration
Data Center Configuration
Advanced

Simulation Duration: min ▼

User bases:

Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
UB1	0	60	100	17	18	1000	100
UB2	1	60	100	17	18	1000	100
UB3	2	60	100	17	18	1000	100
UB4	3	60	100	17	18	1000	100
UB5	4	60	100	17	18	1000	100

Add New
Remove

Application Deployment Configuration:

Service Broker Policy: Closest Data Center ▼

Data Center	# VMs	Image Size	Memory	BW
DC1	10	10000	2048	1000
DC2	10	10000	2048	1000
DC3	10	10000	2048	1000

Add New
Remove

Cancel
Load Configuration
Save Configuration
Done

Figure 3.
Data center configurations

Configure Simulation

Main ConfigurationData Center ConfigurationAdvanced

Data Centers:

Name	Region	Arch	OS	VMM	Cost per VM \$/Hr	Memory Cost \$/s	Storage Cost \$/s	Data Transfer Cost \$/Gb	Physical HW Units
DC1		0 x86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC2		2 x86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC3		4 x86	Linux	Xen	0.1	0.05	0.1	0.1	1

Add New

Remove

Physical Hardware Details of Data Center : DC1

Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	204800	100000000	1000000	4	10000	TIME_SHARED

Add New

Copy

Remove

Cancel

Load Configuration

Save Configuration

Done

Fig. 4.
Load Balancing Policy

Cloud Analyst

Help

Configure Simulation

Define Internet Characteristics

Run Simulation

Exit

Configure Simulation

Main ConfigurationData Center ConfigurationAdvanced

User grouping factor in User Bases:
(Equivalent to number of simultaneous users from a single user base)

10

Request grouping factor in Data Centers:
(Equivalent to number of simultaneous requests a single application server instance can support.)

10

Executable instruction length per request:
(bytes)

100

Load balancing policy across VM's in a single Data Center:

DynaMig Load Balancer

Cancel

Load Configuration

Save Configuration

Done

Figure 5.
User base and data Ccenter distribution

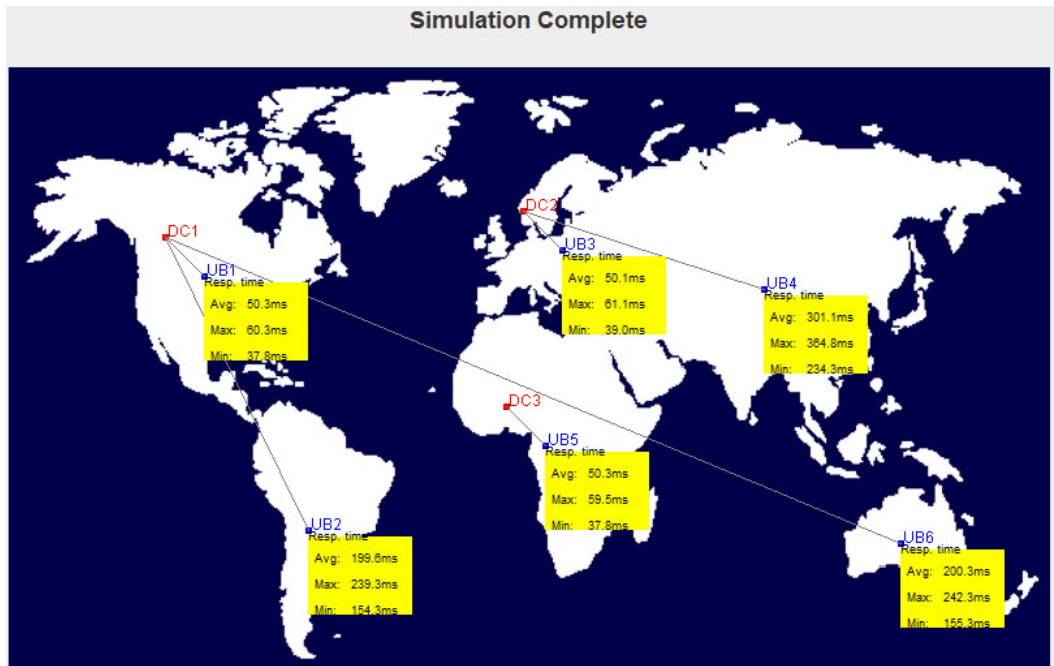


Figure 7.
Comparison of data center processing time of different algorithms

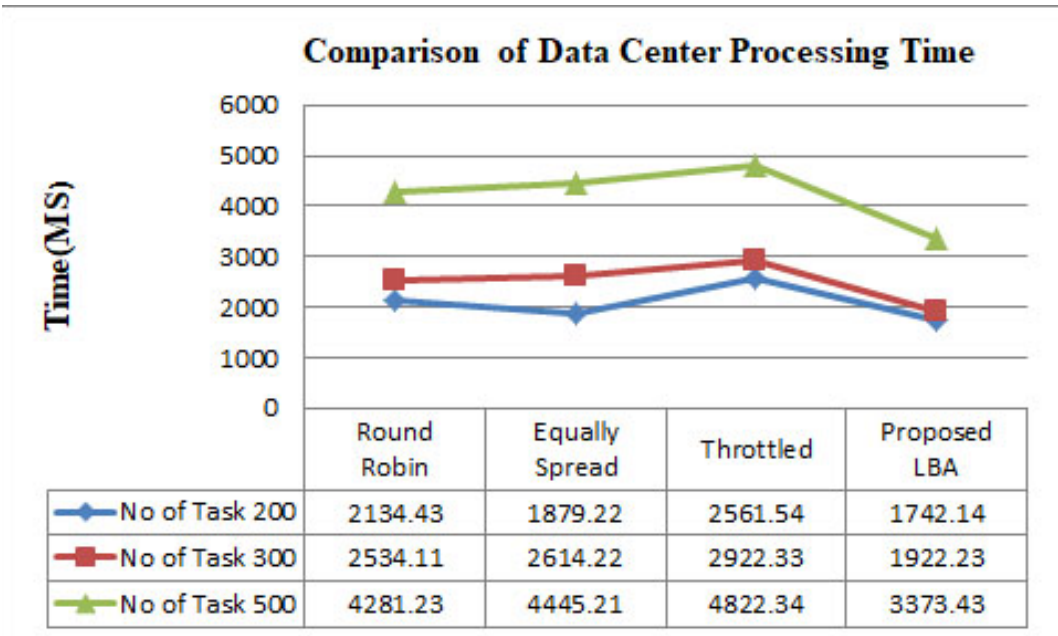


Figure 6.
Comparison of overall response time of different algorithms

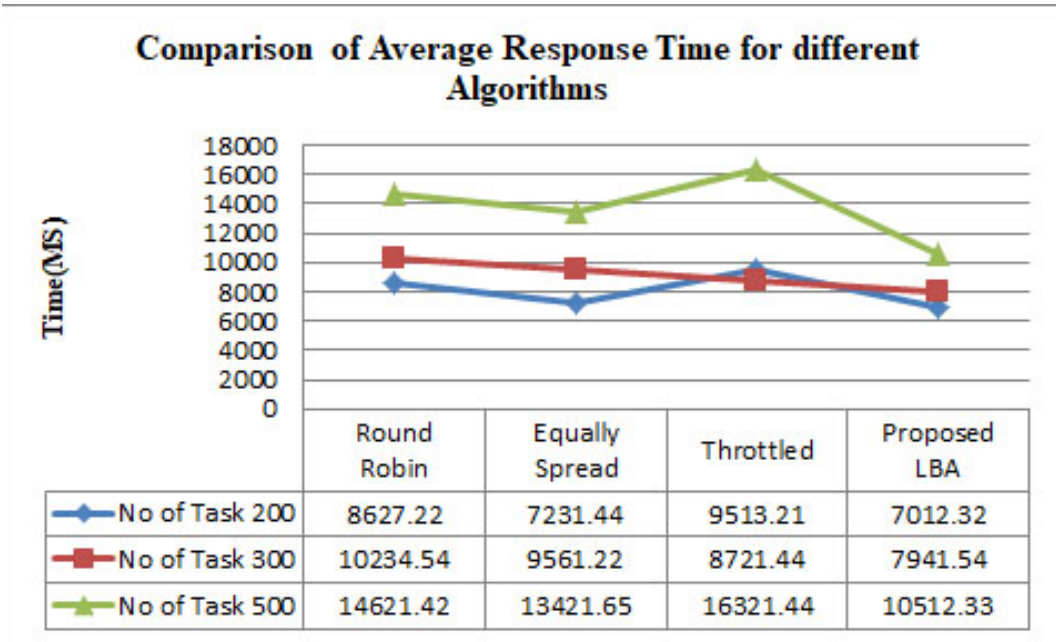
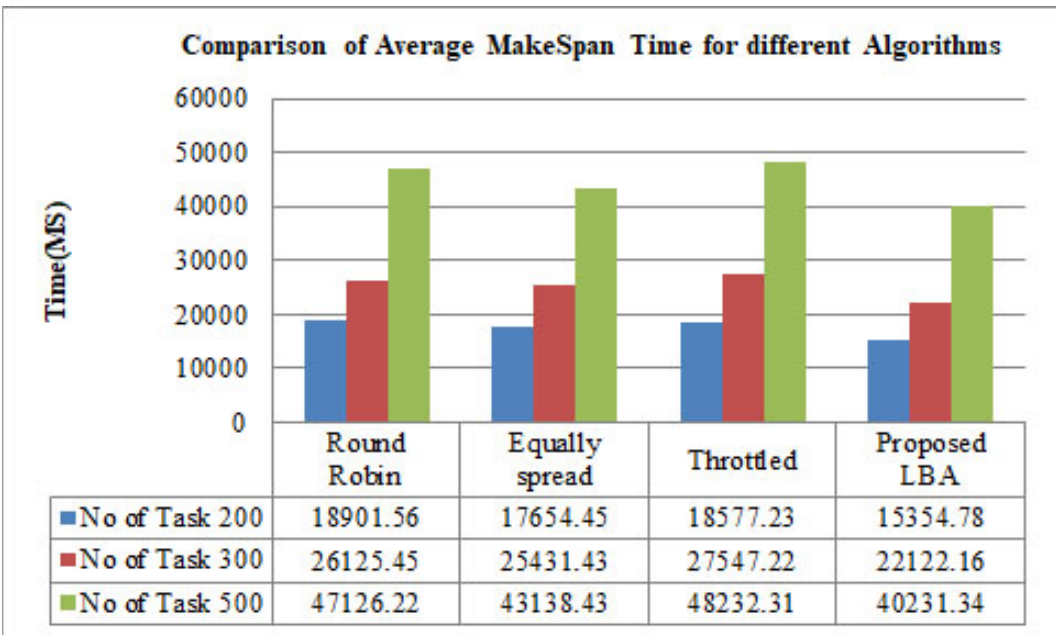


Figure 8.
Comparison of makespan time of different algorithms



5. CONCLUSION

In this paper, our proposed algorithm has been compared with three different existing scheduling algorithms for executing incoming requests in a cloud environment. Each algorithm is observed and their scheduling criteria like average response time, resource utilization, data center processing time, throughput, and performance are found. The main problems with the existing algorithms are that they didn't consider the configuration or capacity of the VMs and allocated incoming jobs again and again to the overloaded VMs while other VMs of same data center remained idle in the meantime. Our proposed algorithm checks the availability of VMs and also assigns jobs according to their threshold value. While processing a job, it also checks if any higher configuration VMs are idle or not so that if migration is profitable, it performs the migration. That is why the total response time and the makespan time have been reduced. According to our study, it can be concluded that our proposed algorithm has reduced execution time, reduced waiting time and showing proactive performance with a large number of tasks and doing more efficiently VMs utilization of different configuration to fulfill the objective of our work.

REFERENCES

- Afzal & Kavitha. (2019). Load balancing in cloud computing – A hierarchical taxonomical classification. *Journal of Cloud Computing Advances, Systems and Applications*, 8(22).
- Beri, R., & Behal, V. (2015). Cloud Computing: A Survey on Cloud Computing. *International Journal of Computers and Applications*, 3(16).
- Hung, L. H., Wu, C. H., Tsai, C. H., & Huang, H. C. (2021). Migration-Based Load Balance of Virtual Machine Servers in Cloud Computing by Load Prediction Using Genetic-Based Methods. *IEEE Access: Practical Innovations, Open Solutions*, 9, 49760–49773. doi:10.1109/ACCESS.2021.3065170
- Liaqat, M., Ninoriya, S., Shuja, J., Wasim Ahmad, R., & Gani, A. (2016). *Virtual machine migration enabled cloud resource management: a challenging task*. arXiv:1601.03854
- Mishra, M., Das, A., Kulkarni, P., & Sahoo, A. (2012). Dynamic Resource Management Using Virtual Machine Migrations. *IEEE Communications Magazine*, 50(9), 34–40. doi:10.1109/MCOM.2012.6295709
- Panwar, R., & Mallick, B. (2015). Load Balancing in Cloud Computing Using Dynamic Load Management Algorithm. *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, 778-778. doi:10.1109/ICGCIoT.2015.7380567
- Sefati, S., Mousavinasab, M., & Zareh Farkhady, R. (2022, January). Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: Performance evaluation. *The Journal of Supercomputing*, 78(1), 18–42. doi:10.1007/s11227-021-03810-8
- Shahapure & Jayarekha. (2018). Virtual machine migration-based load balancing for resource management and scalability in cloud environment. *International Journal of Information Technology*. DOI: 10.1007/s41870-018-0216-y
- Swarnakar, S., Kumar, R., Krishn, S., & Banerjee, C. (2020). Improved Dynamic Load Balancing Approach in Cloud Computing. *2020 IEEE 1st International Conference for Convergence in Engineering (ICCE)*, 195-199.
- Zhang, Liu, & Chen. (2016). A Multi-Agent based load balancing framework in Cloud Environment. *IEEE 9th International A Multi-Agents based load balancing framework in Cloud Environment 2016*, 278-281.
- Zongyu, X., & Xingxuan, W. (2015). A predictive modified round-robin scheduling algorithm for web server clusters. *2015 34th Chinese Control Conference (CCC)*, 5804-5808.

Soumen Swarnakar completed his M.Tech. degree from IEST, Shibpur, Howrah, India (formally known as Bengal Engineering and Science University, Shibpur) and currently working as an Assistant Professor in the department of Information Technology in Netaji Subhash Engineering College, Kolkata, India since 2006. His research Interests are Cloud Computing, Data Mining.

Chandan Banerjee has completed his PhD from Jadavpur University, Kolkata, India. He is currently employed as a Professor and Head of the Department of Information Technology at Netaji Subhash Engineering College in Kolkata, India. His research interests are in Cloud Computing and Machine Learning. He has more than 30 research publications.

Joydeep Basu is B.Tech. (Information Technology) passout student from Netaji Subhash Engineering College, Kolkata, India in the year 2022. His research interests are in Cloud Computing and Machine Learning.

Debanjana Saha is B.Tech (Information Technology) passout student from Netaji Subhash Engineering College, Kolkata, India in the year 2022. Her research interests are in Cloud Computing and Machine Learning.