Priority Based VM Migration with Dynamic Load Balancing in Cloud Environment

Abstract. Cloud computing involves using internet-based remote servers for data storage, management, and processing. This on-demand service allows users to make payments for simply that which they utilize. Since users of cloud computing are widespread globally, managing this vast amount of data presents a significant challenge. Load balancing efficiently allocates workloads across multiple computing resources, such as online servers. The load balancer's resources can be changed or added to based on user requirements. The primary goals of load balancing are to optimize resource usage, costs, to enhance throughput, and to prevent VMs from being overloaded. This research work introduces a migration of jobs in virtual machines using the priority of tasks, remaining execution time as well as migration time for the cloud dynamic load balancing. The proposed method shows better result in comparison with some existing cloud load balancing algorithms in respect of average makespan time, average response time.

Keywords: Cloud Computing, Dynamic Load Balancing, Efficient Load Balancing, Multi-Agent, Resource Allocation, Virtual Machines, VM Migration, Priority based

1 Introduction

In the current era of fast-growing IT, distributed computing is becoming increasingly popular. Cloud computing plays a very important role in the accumulation and effective handling of massive data. It offers features like rapid elasticity, services on-demand, pooling of resources, and scalability. Cloud computing is prevalent in all types of businesses, from small-scale to large enterprises. Job allocation typically relies on algorithms such as Weighted Least Connection, FCFS and Round Robin. When tho, these algorithms perform good, they can sometimes take longer to complete tasks. Numerous papers have proposed ways to speed up task execution, and researchers have tested various algorithms to enhance load-balancing systems for optimization. This paper introduces a dynamic method that employs multiple agents to distribute tasks across various virtual machines (VMs) in different configurations and data centers. Often, tasks directed to VMs with varying setups may lead to some VMs becoming overloaded while others remain underutilized. When tasks attempt to enter an overloaded VM, acquiring resources can take longer, resulting in delays in execution. Consequently, it becomes essential to dynamically migrate tasks from an overburdened VM to an underutilized VM. But the crucial point is while shifting jobs from one overloaded VM to an underloaded VM, migration is executed according to the priority of the tasks that are present in the Priority Queue associated with a VM. Total Migration of tasks doesn’t happen in this case but happens based on the Task's priority and their completion time starting from the waiting time in the queue to its completion time. This research paper presents an enhanced algorithm with an Agile technique for cloud load distribution, incorporating Several contributors such as capability of VM, time left, task load, and a priority queue instead of a regular queue. The remaining parts of this study are split into 4 distinct parts. Section II outlines an overview of related works. In Section III, The structural design of the proposed load-balancing algorithm is explained. Section IV describes the simulation results of the suggested algorithm and comparison with existing cloud load balancing algorithms. Lastly, conclusion has been drawn in Section V.

2 Related Work

Some of the research work that has already been done on the cloud shall be discussed in this section.

S. Swarnakar et al. (2023) [1] have created an algorithm (MAVMM) in their research article that entails moving jobs from virtual machines with a lot of load to ones with less load while also considering whether the migration is dependent on a lot of constraints. But in this algorithm priority of tasks have not been considered. In this proposed algorithm, it has been considered to execute the tasks.

Sefati et al. (2022) [2] suggested a new load-balancing technique for cloud environments based on the reliability of different resources. In this case, the method looks for nodes that are busy or idle before calculating the fitness function and threshold value for each node in order to assign tasks. The CloudSim simulation results showed that the anticipated solution has lower costs and a lower response time than the other approaches.

Hung et al. (2021) [3] created a two-phase evolutionary algorithm for load-balancing virtual machine hosts in a cloud computing system based on migration. Gene programming expressions have been utilized to describe virtual machine performance and to anticipate the load on virtual machine hosts following load balancing. To move virtual machines (VMs) for load balancing, a genetic algorithm was employed to compare the loads on virtual machine hosts, both present and prospective. The Jnet cloud environment was used to conduct the simulation. The simulation has shown much better results than the previously described methods in that particular paper.

S. Afzal et al. (2019)[4] proposed a load-balancing technique for cloud computing systems that considers future challenges. To address imbalanced load issues, they examined and contrasted several current algorithms and those that will be developed in the future.

C.K. Rath et al. (2018) [5] described how to use a genetic algorithm in a cloud environment to schedule jobs dynamically with a balancing load. To identify a new fitness function, standard deviation (SD) is utilized. Their suggested algorithm's primary goal was to minimize makespan time. By distributing the load and making efficient use of the processor, the suggested scheduling policy has been used to demonstrate the outcome, which can reduce both idle and makespan time.

S. Swarnakar et al. (2018) [6] have developed a novel hybrid model for load balancing in cloud computing systems through their research. Through the combination of the two classic algorithms—the Weighted Least Connection Algorithm and the Weighted Round Robin Algorithm—they have created a more intelligent and effective method of load balancing. Using CloudSim, their suggested algorithm was compared to the previously listed algorithms, and the results were better.

M. Kumar et al. (2017)[7]suggested a dynamic-based load-balancing method that reduces the makespan time of projects with greater flexibility in the cloud environment while also effectively utilizing cloud resources. The simulation's output outperformed other standard load balancing methods in terms of both resource utilization and load balancing performance.

R. J. Khan et al. (2017)[8], conducted an extensive examination of the efficacy of different load-balancing algorithms. The performance of algorithms, including energy economy and incorrect tolerance, was examined using a variety of methods.

O. Kaneria et al. (2016) [9] suggested changing the load balancing algorithms to give a job to a host with the most processors to enhance access speed and make optimum use of cloud-based resources. Cloudlets were assigned to virtual machines (VMs) in a data center by a resource allocation strategy. This simulation has been run using Cloud-Sim.

A. Dave et al. (2016) [10] performed a survey on several optimization strategies for load balancing. Swarm-based optimization and evolutionary techniques have been discussed for load-balancing scenarios.

Certain types of methodologies and investigations were suggested by A. Ragmani et al. (2016) [11] and were employed for various load-balancing practices. They also suggested a novel approach to cloud computing that results in faster reaction times, as well as an improved load-balancing strategy.

3 Proposed Work

The proposed approach aims to reduce the duration of execution and improve the VMs' scalability. We are utilizing multi-agent VM migration in this suggested architecture to take into account the available limits without sacrificing speed. The dynamic resource changes made by virtual machines result in disparities in the PMs' resource usage. As a result, while some VMs become overwhelmed, others remain underloaded. As a result, the suggested solution will address that, maximize resource use, and reduce turnaround time. Jobs will be moved from a heavily loaded lower configuration virtual machine (VM) to an underloaded or less laden higher configuration VM using the suggested load balancing technique, which also determines whether the migration is necessary based on the priority queues.

Within our proposed arrangement, we are considering two types of virtual machines (VMs): a type with a higher configuration and a type with a lower configuration. Processing speed is higher in virtual machines with higher configurations than with lower ones. Lower configuration Virtual Machine (VMLn), where n = 1,2,3...k, are the lower configuration VMs, and Higher configuration Virtual Machine (VMHm), where m = 1,2,3...k, are the higher configuration virtual machines. The three dynamic tables that make up the load balancer are used to store the status of each virtual machine. The availability and details of particular VM types in the second and third dynamic tables are now stored in the first dynamic table. Together with their data center IDs, the list of VMs with higher configurations that are available is stored in the second dynamic table. The details of virtual machines with lesser configurations and their data center IDs are contained in the third table. The suggested algorithm's goals are, reducing response time and improving average completion time. The approach is engineered to minimize superfluous time consumption in the event that a job is executed in a virtual machine with a lower configuration, thus preventing a higher configuration VM from becoming underloaded. For this reason, if needed, the VM migration approach will be used to manage the resources. Because the most productive VMs will be used to their full potential, this technique will minimize average response time and makespan time while also making the mechanism scalable. This study is divided into four main sections that present a dynamic algorithm based on virtual machine migration. Below is a description of each section:

- **a) Load Monitoring Section** (**LMS**): All VM details, regardless of configuration, will be indexed and kept here. By using mathematical formulas to establish the threshold value, it may determine if a virtual machine (VM) is overloaded or underloaded. Virtual machines (VMs) will be classified as overloaded or underloaded depending on whether they are above or below the threshold.
- **b) Load Balancing Module**: This load balancer is crucial for allocating tasks to

various virtual machines with various setups that are connected to various data centers. In this case, an approach based on migration is employed to allocate and move certain tasks from virtual machines that are overcrowded to those that are underloaded according to priority. The current availability of virtual machines with varying configurations is kept up to date in several dynamic tables. With the aid of various dynamic tables displaying the VMs that are currently available of various configurations connected to varied data centers, Following the guidelines established by the algorithm, the load balancer will be dividing the loads among various virtual machines (VMs) or migrate the loads for moving jobs from lower configuration VM to higher configuration VM. Following that architecture, the load will be distributed equally. The physical machine will be running a load balancer. It computes the virtual machines' processing speed, RAM consumption, and data transfer rate.

c) **VM Migration Section**: The purpose of the VM Migration Section is to move an excess workload from overloaded virtual machines (VMs) to the ones that are not overloaded, or whenever a VM with a higher configuration is idle and jobs are running in lower configuration machines. It computes the migration time and verifies the list of virtual machines with higher configurations. The migration of the jobs can have an impact on the overall performance of the whole process if it takes longer than the estimated threshold period. This module will hence not carry out the migration. Below, in section 3.1, is a mathematical description of how to determine whether a virtual machine (VM) is overloaded or underloaded, as well as how to migrate jobs from the lower configuration machines to machines with higher configuration.

d) **Priority Queue:** Moving high-priority tasks from VMs that are overcrowded to those that are underloaded or optimally laden is crucial for maximizing resource utilization and guaranteeing the timely completion of crucial processes. This procedure entails keeping a close eye on task priorities and VM resource consumption, spotting overloaded VMs, choosing appropriate destination VMs with balanced or lower resource utilization, carrying out the migration, and then redistributing resources amongst VMs as needed. This technique maximizes overall efficiency, lowers resource contention, and improves system performance by reallocating jobs to more appropriate VMs.

Abbreviations	Definition
Mips	Millions of instructions per second
$Load_{PM}$	Load on a physical machine
CPU_{VM}	CPU capability of a virtual machine
Bw	Bandwidth
Mem	Memory
Max_{th}	Maximum threshold
Min_{th}	Minimum threshold
$Util_i$	Resource utilisation of VM
$Util_i^r$	Resource utilisation ‘r’ of VM ‘i’
CPU_T	The time needed by CPU
BW_T	The time needed for bandwidth
RAM_T	The time needed in ram
Tef	Efficiency
VM_{util}	Capacity of VM currently in use
$Migration_{Time}$	Duration of migration
T_{res}	The amount of time needed to duplicate resources
Mig_{data}	Data to be migrated
T_{EXE}^{LC}	Execution Period for Lower Configuration on vm
T_{EXE}^{HC}	Execution Period for Higher Configuration vm
T_{th}	Time Threshold

3.1 Mathematical Explanation to decide if a VM is Over or Underloaded

A load-balancing algorithm of VM migration is required to determine which virtual machines (VMs) are underloaded and which are overloaded. Here are a few of the crucial formulas:

$$Load_{PM} = \sum_{i=1}^{k} \frac{CPU_{VM}}{n} + \frac{VM_{bw}}{n} + \frac{VM_{mem}}{n}$$

$$CPU_{VM} = \frac{totalreqUtiliredmips}{totalmipsofPM}$$

$$VM_{bw} = \frac{BWusedbyVM}{totalBWofPM}$$

$$VM_{mem} = \frac{RAMusedbyVM}{totalRAMofPM}$$

A virtual machine's (VM) resource usage r is shown by $Util^r_i$. Thus, $Util_i = CPU_{VM} + VM_{bw} + VM_{mem}$, so processor consumption, bandwidth, and memory will determine the migration will happen or not.

$$CPU_T = \frac{\sum_{i=1}^{n} \quad CPU_{VM}}{totalmipsofPM}$$

$$BW_T = \frac{\sum_{i=1}^{n} \quad VM_{bw}}{totalbwofPM}$$

$$Tef = \frac{CPU_T + BW_T + RAM_T}{3}$$

Maximum Threshold $Max_{th} = U^r_i - T_{ef}$

Minimum Threshold $Min_{th} = U^r_i - 1$

Let's now C^r_i be a virtual machine's current capacity. So, $VM_{util} = U^r_i - C^r_i$

If $VM_{util} > Max_{th}$, there is too much load on the VM. Next, divide the job among the remaining virtual machines.

If $VM_{util} < Min_{th}$, then job shifting is not required since the VM is not overloaded enough to migrate. Rather, it can carry out the incoming job.

3.2 Mathematical Explanation to Decide for Migration

Every virtual machine captures Migration Time, this indicates when the virtual machine was last moved to a physical computer. (If there was no migration, assign 0 to it). Now, the quality of service will be impacted if a certain job is moved around a lot. Also, the load balancer will decide if it makes sense to relocate a job to a virtual machine with a higher configuration if it is currently executed in a lower configuration virtual machine and that VM becomes underloaded in the interim. The migration process won't happen if it doesn't. To determine if migration will occur or not, a time threshold must be established, Tth.

$Migration_{Time} = T_{res} + Mig_{data}$

$Mig_{data} = Max_{th} - PM_{load}$

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

So, let T_{now} be the current time that the work under the particular virtual machine (VM) with a lower configuration is being processed. Now, the load balancer will verify the following formula even if VMs with a higher configuration are available.:

$T_{now} + Migration_{Time} + T_{dest} < T_{scr}$

$\Rightarrow T_{now} + Migration_{Time} < T_{scr} - T_{dest}$

$\Rightarrow T_{now} + Migration_{Time} < T_{th}$ (Considering $T_{scr} - T_{dest}$ as T_{th})

Where T_{dest} is the amount of time needed to finish the task in the target virtual machine (i.e. migration happened) and T_{src} is the amount of time the virtual machine (VM) ,where the job is now running, will need to complete it without migrating. The T_{scr} - T_{dest} is actually the performance gap between VMs with higher configurations and those with lesser configurations. The average execution time of the particular type of virtual machines (VMs) can be used to determine this. As a result, a time threshold can be applied to all processes. So, the circumstances that the load balancer will check if $Tnow+Migration_{Time}$ is greater or lesser than Tth.

If T_{now} + $Migration_{Time}$ < T_{th}, then migration will occur, improving the process's efficiency.

If T_{now} + $Migration_{Time}$ > T_{th}, then the migration will be prevented since it will create needless delays.

3.3 Explanation to Decide for Migration Based on Priority

Priority based migration is decided if the priority of the jobs are over the average. Average Priority is found by:

***Priority**$_{Avg}$* = $\frac{P1+P2+.....Pn}{N}$, where P_n is the priority of each job in the queue.

Table 1. Snapshot of Priority Table

Virtual Machine ID	Job no.	Priority
VM_{H1}	1	10
VM_{H1}	2	9
VM_{H1}	3	4
VM_{H1}	4	1

3.3 Dynamic Tables Used in Proposed Work

Dynamic Table I. Snapshot of Available Virtual Machines in Dynamic Table II and Dynamic Table III

Table Number	Available Virtual Machines

Dynamic Table I	4
Dynamic Table II	8

Dynamic Table II. Snapshot of Available High Configuration Virtual Machines and Corresponding Data Center

Current available virtual machine ID	**Data Center ID**
VM_{H4}	4
VM_{H12}	7
VM_{H16}	9

Dynamic Table III. Snapshot of Available Low Configuration Virtual Machines and Corresponding Data Center

Current available virtual machine ID	**Data Center ID**
VM_{L21}	9
VM_{L25}	9
VM_{L27}	11
VM_{L31}	13
VM_{L33}	16
VM_{L37}	18

3.4 Proposed Algorithm:

1. Start
2. Jobs will come to the load balancer in a queue. The jobs will be assigned with a priority (represented by a number) showing the priority of the job. Jobs with lower numerical value assigned are jobs with higher priority. The priority range is 1 to 10. The default value is 5. The lower the value, the higher the priority of the client.
3. The availability of VMs with higher or lower configurations is checked using Dynamic Table I. The first row of Dynamic Table I represents VMs with higher configuration and number of such VMs available, whereas the second row represents VMs with lower configuration and number of such VMs available.
4. Now for every data center:

   a) Tasks are kept in a priority queue of each data center. Now, the Dynamic Tables I, II, and III are consulted for the distribution of jobs.

i) Tasks with higher priority go to the higher configuration virtual machine.

ii) Tasks with lower priority go to the lower configuration virtual machine.

5. After the tasks have been allocated to the virtual machines, the tasks wait in a queue and are executed one by one.

   a) If a virtual machine is of low configuration and is overloaded, then the Dynamic Table II is checked in search for higher configuration virtual machines.

   b) If higher configuration machines are available, then the total remaining execution time of all waiting jobs in that queue is checked.

   i) If $\mathbf{T_{EXE}^{LC} < Migration_{Time} + T_{EXE}^{HC}}$, then the job doesn't migrate. New higher configuration machine is searched.

   ii) If $\mathbf{T_{EXE}^{LC} > Migration_{Time} + T_{EXE}^{HC}}$, then the job migrates to the higher configuration virtual machine following the rules below:

   Rule 1: In case, when jobs are migrated, then the average priority of the queue is calculated.

   Rule 2: Jobs with higher priority than the average priority of the queue get migrated to the higher configuration VMs.

6. If more jobs keep coming, go to step 4 and repeat the same procedure.
7. End

The working of the proposed algorithm has been represented in Figure 1.

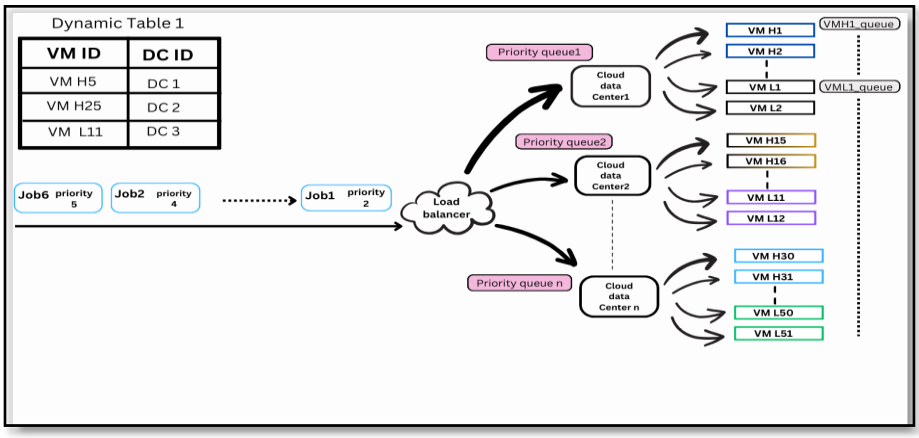


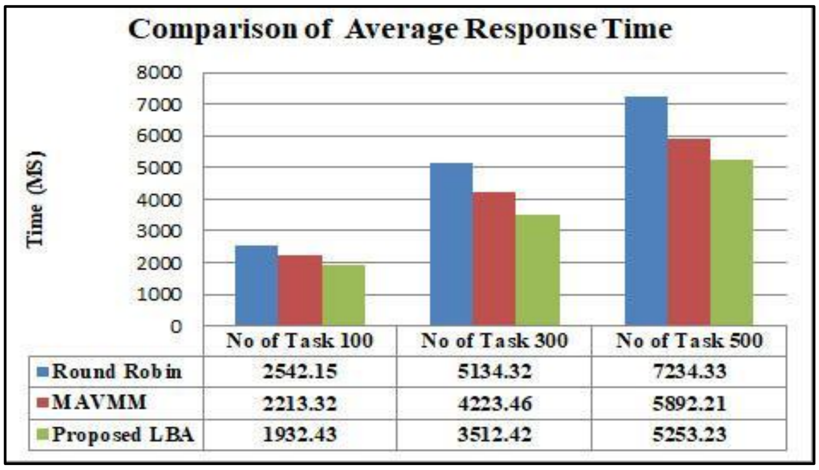
Fig1. Proposed algorithm of priority based VM Migration

4 Experimental Result

Cloud Analyst tool, which is a graphical user interface of Cloud Sim Architecture has been used for the execution of the experiment. Java programming language is used for proposed load balancing algorithms.

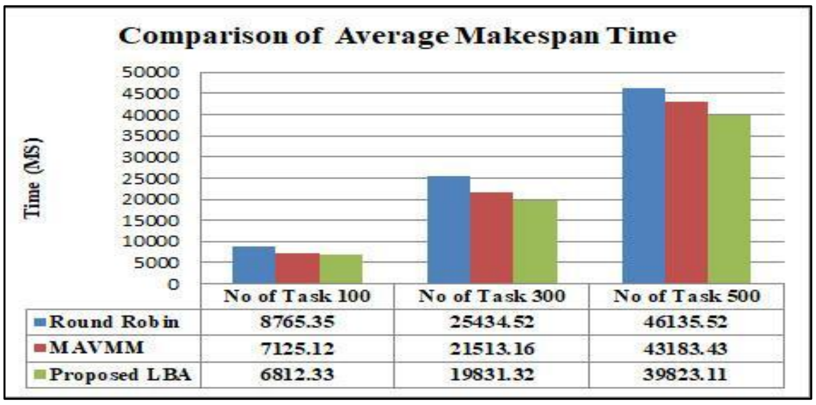
Simulation of Proposed algorithm is compared with MAVMM [1] and Round Robin Algorithm. Six user bases located at six different regions of the world has been considered, and the duration of the simulation is kept to twenty minutes. Three Data Centers are considered for the proposed experiment where each data center is considered to have ten virtual machines having 2048 MB of memory in each.

100, 300, and 500 tasks with different lengths have been chosen randomly for the simulation of the experiments. The comparative study of simulation for proposed algorithm with MAVMM [1] and Round Robin Algorithm, based on average overall response time and average makespan time have been shown in figure 2 and figure 3.



	No of Task 100	No of Task 300	No of Task 500
🟦 Round Robin	2542.15	5134.32	7234.33
🟥 MAVMM	2213.32	4223.46	5892.21
🟩 Proposed LBA	1932.43	3512.42	5253.23

Fig 2. Comparison of Avg. Response Time with different no. of Tasks



	No of Task 100	No of Task 300	No of Task 500
🟦 Round Robin	8765.35	25434.52	46135.52
🟥 MAVMM	7125.12	21513.16	43183.43
🟩 Proposed LBA	6812.33	19831.32	39823.11

Fig 3. Comparison of Avg. Response Time with different no. of Tasks

In the above two figures , it is seen that the proposed algorithm is performing better than Round Robin and MAVMM [1] in respect of average response time and average makespan time.

5 Conclusion

The algorithm that we have proposed has been compared with the already existing algorithm like Round Robin as well as the Multi-Agent-Based VM Migration (MAVMM) [1] algorithm. The problem with the algorithms with which the proposed algorithm has been compared, did not consider factors such as priority of tasks. The lower numerical value of priority implies higher priority. Whether any job in the waiting queue of Data Center need to be shifted from lower configuration VMs to the higher configuration VMs has been decided based on migration time as well as priority of tasks. Not only does our proposed algorithm assign jobs to the corresponding available VMs with lower or higher configuration but also based on their priority value. It ensures that not all the jobs in the waiting queue of lower configuration VMs are shifted to higher configuration VMs but only with higher priority tasks will be migrated if and only if remaining execution time is higher than migration time. This ensures maximum and efficient resource utilization, executing all the tasks at an optimal time, thereby also reducing the Makespan time as well as response time.

References

1. Swarnakar, S., Banerjee, C., Basu, J., Saha , D.: A Multi-Agent Based VM Migration for Dynamic Load Balancing in Cloud Computing Cloud Environment. International Journal of Cloud Applications and Computing, Vol. 13, Issue 1 (2023)
2. Sefati, S., Mousavinasab, M., & Zareh Farkhady, R.: 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 (2022)
3. Hung, L. H., Wu, C. H., Tsai, C. H., & Huang, H. C.: 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 (2021)
4. Afzal, S., and Kavitha, G.: Load balancing in cloud computing - A hierarchical taxonomical classification. Journal of Cloud Computing Advances, Systems and Applications, Vol 8, Article No 22 (2019)
5. Rath, C.K., Biswal, P., Suar, S.S.: Dynamic Task Scheduling with Load Balancing using Genetic Algorithm. In: International Conference on Information Technology (ICIT), Bhubaneswar, pp. 91-95 (2018)
6. Swarnakar, S., Raza, Z., Bhattacharya, S., Banerjee, S.: A Novel Improved Hybrid Model for Load Balancing in Cloud Environment. In: Fourth International Conference on Research in Computational Intelligence and Communication Networks, doi:10.1109/ICRCICN.2018.8718697 (2018)
7. Kumar, M., Sharma, S. C.: Dynamic load balancing algorithm to minimize the makespan time and utilize the resources effectively in cloud environment. International Journal of Computers and Applications, Vol. 42, Issue 1, pp. 108-117,(2017)
8. Khan, R. J., Md. Ahmad, O.: A survey on load balancing algorithms in cloud computing. International Journal of Autonomic Computing (IJAC), Inderscience Pub., Vol. 2, No. 4, (2017)
9. Kaneria, O., Banyal, R. K.: Analysis and improvement of load balancing in Cloud Computing. In: International Conference on ICT in Business Industry & Government (ICTBIG), Indore, pp. 1-5, DOI: 10.1109/ICTBIG.2016.7892711(2016)
10. Dave, A., Patel, B., Bhatt, G.: Load balancing in cloud computing using optimization techniques: A study, In: International Conference on Communication and Electronics Systems (ICCES), Coimbatore, pp. 1-6. 10.1109/CESYS.2016.7889883(2016)
11. Ragmani, A., Omri, A.E., Abghour, N., Moussaid, K., Rida, M.: A Performed Load Balancing Algorithm for Public Cloud Computing Using Ant Colony Optimization. In: 2nd International Conference on Cloud Computing Technologies and Applications, Morocco, pp. 221-228 (2016)