Mutative ACO based Load Balancing in Cloud Computing

Saurabh Singhal *Member, IAENG,* Ashish Sharma

***Abstract*—With the increment in figuring advancements, Cloud computing has permitted the clients to get to assets from anyplace whenever and to pay for the assets based on pay-per-use. Cloud computing allocates the submitted tasks to virtual machine (VM) by dynamic allocation. With the bandwidth and network connectivity getting better, more number of users and organizations are moving to Cloud. All things considered, there are different difficulties that are looked by Cloud service provider. One of the most important challenges is of load balancing. With the number of users gradually increasing, load balancing has become more important. Numerous calculations have been proposed till now for load adjusting, where every one of it centres around the various boundaries. However, these techniques suffer from various issues like - stuck in native optima. None of the calculations has demonstrated to be totally proficient. To deal with the issues identified with existing meta-heuristic procedures, in this paper, a mutation based ant colony optimization load balancing algorithm is proposed. Here we focus on the mutation in Ant Colony Optimization (ACO) algorithm for efficiently balancing the load among various data centers to further develop the presentation boundaries like response time and while improving the overall fitness function.**

papers. ***Index Terms*—Mutative-Ant Colony Optimization, Ant Colony Optimization, Particle Swarm Optimization,makespan, energy consumption.**

I. Introduction

CLOUD computing is a computing paradigm that provides services to users having discrete requirements over internet. Cloud computing provides high performance computing by providing users with both software and hardware resources along with tools for software development and testing [1], [2]. It is progressively adaptable. Cloud computing is in its beginning phases, in order to obtain its full advantages, much exploration has been done and is yet to be done over an expansive space of subjects. Cloud Computing allocates the task on virtual machine(VM) which can be accessed by users [18].

To harness the advantage of cloud computing the resource must be used in the best possible manner. This can be accomplished by overseeing assets in cloud computing during booking and designation. For proficient asset the executives, load adjusting is basic. To harness the advantage of cloud computing the resource must be used in the best possible manner. This can be achieved by managing resources in cloud computing during the scheduling and the allocation. For productive asset the executives, load balancing is critical. In cloud, as the task that is coming for execution is at a varied pace, the resource consumption cannot be fixed. One of the benefits of cloud computing is to convert a physical machine operated by a single user to a virtual machine accessible to multiple users [3]. Cloud Service Providers (CSP) are responsible for allocation of task to the resources (VM) available with them. CSP has to ensure that the load on each virtual machine is either in the case of heavy traffic or in the case of light traffic is optimized [4].

Subsequently, the idea of loan balancing comes into picture. Load balancing is the way toward dividing load between virtual machines with the end goal that each machine has a similar load [5]. It ensures that no machine is ideal, overloaded or under loaded. Load balancing tries to maintain the performance of resources at CSP side while providing the promised QoS parameters as mentioned in Service Level Agreement (SLA) [19]. As the task arrives to the CSP, it utilizes some resources of VM. As the number of tasks increases, the resources at VM are exhausted. This situation is referred as over loading and it decreases the performance of the environment. In the event that more undertakings are submitted to a similar VM, make length season of errands will increment. In this situation, assignments those are moved to a VM that is either ideal or under-loaded. [6].

Load balancer disseminates the customer demands or organization load proficiently across the various workers .To distribute the load on a virtual machine manager (VMM), load balancer uses virtualization along with hypervisors. Load balancing are a NP hard problems. In cloud, the heap balancer can be static or dynamic [7]. The static calculations don't rely upon the framework state and have all the data, for example, task subtleties and framework assets in earlier. These calculations can't deal with an unexpected spike in burden or disappointment of any asset. The unique calculation takes the choice dependent on the present status of the framework and doesn't need any data ahead of time. The powerful calculations are intricate in nature however are deficiency open minded and have preferred execution over static calculation [8].

A. Metrics of load balancing

The different measurements that are utilized to gauge the presentation of load balancing calculations are:

Throughput The quantity of tasks that have finished their execution inside a given time limit. For better productivity, the throughput ought to be greatest [9], [10].

Overhead Overhead is the activity cost of the work caused during its execution. For better proficiency of the calculation, the overhead worth ought to be least.

Fault tolerance This boundary deals with the breakdown that has happened in a node verifiably and consistent way. On account of a defective node, it changes the hub. This boundary must be limited for better throughput..

Manuscript received June 30, 2021; revised July 01, 2021.

S.Singhal is Assistant Professor in the Department of Computer Engineering and Applications, GLA University, Mathura. e-mail: (saurabh.singhal@gla.ac.in).

A. Sharma is Professor in the Department of Computer Engineering and Applications, GLA University, Mathura. e-mail: (ashish.sharma@gla.ac.in)

Transfer time The time that is needed during the development and redistribution of the resources from a node to another node. The worth of this boundary ought to be least for accomplishing better productivity of the framework
Resource utilization This boundary guarantees the usage of the assets in load adjusting. This boundary ought to be pretty much as greatest as conceivable in a load adjusting climate to accomplish the expense minimization of assets.
Makespan The complete time needed to execute every one of the positions that are submitted to the framework for execution. Makespan is the greatest time taken for the execution of the cloudlets running on the data handling center. A small value of makespan shows the higher efficiency of the system.
The remainder of the segment of this examination is coordinated as follows: load balancing in cloud computing provided by the Section II, the proposed algorithm is discussed in section III, Simulation and result are discussed in section IV. Section V discusses the conclusion and future work of the proposed work.

II. Related Work

In [11] the creators have given a load adjusting strategy that depended on honey bee conduct. In this procedure, endeavors are moved from over-loaded machine to under-loaded machines for execution. It also considers the priority of a task for execution as to minimize the waiting time for tasks. The technique works on the principle of honey bees to find other honey bees i.e. to find under loaded machines.
In [12]the authors for load balancing considered various QoS parameters such as the response time and the number of migrations. They considered the task to be honey bee and food sources as under-utilized virtual machine. When a virtual machine is detected as high loaded, then some low priority are migrated from one machine to another one.
In [13]the authors for load balancing considered various QoS parameters such as the response time and the number of migrations. They considered the task to be honey bee and food sources as under-utilized virtual machine. When a virtual machine is detected as high loaded, then some low priority are migrated from one machine to another one. To improve response time of task, the authors gave an improved weighted round-robin algorithm. They considered size of the task, VM capacity, and inter-disciplinary nature of tasks. The proposed algorithm identified VM as least loaded by calculating the total completion time of jobs. VM having minimum completion time was considered as least loaded. The environment considered for running algorithm was homogeneous.
The authors in [14] have proposed a load balancing approach by combining two optimization algorithms i.e. Teaching-Learning-Based Optimization (TLBO) and Grey Wolves Optimization algorithms (GWO) to maximize the throughput by balancing load over VMs. The approach tries to overcome the problem of local optimum. The method is able to get a better performance when there is a high volume of data to cloud scheduler. In [15] to balance load in a dynamic environment, the authors proposed a technique based on external optimization (EO). The proposed method considered various factors to propose a solution for dynamic method. However, the algorithm does not work multi-objective optimization.
In [16] the creators talked about a methodology dependent on the conduct of natural ants that structures organizations while looking for food. In this methodology upon commencement, the fake subterranean insect begins its development toward a path by visiting the nodes individually and checking the situation with load in a node. If it finds an overloaded node then it starts backtracking to the previous node which is under loaded. The path is recorded and stored for further use.
In [17] the creators planned a decentralized burden adjusting calculation dependent on the conduct of honey bees. This algorithm aims to balance the load on cloud over heterogeneous nodes. In this algorithm, first the current load on each node is calculated. Based on this calculation the nodes are classified as over loaded, under loaded or idle. Then the load is shifted from over loaded node to the node that is lightly loaded.

III. Proposed Work

Figure 1 depicts the load balancing algorithm architecture. The client submits the jobs to the cloud via the web. The submitted jobs are collected by the Broker before disseminating them to different resources based on the load on each resource. Assuming that every resource is highly loaded, the balancer chooses the resource that has a fairly low load as compared to others and can execute the job.
The main thought behind an ACO model is that a good solution is obtained from the gradual increment of a partially good solution and not from the random approach. The Ant Colony enhancement is propelled by the scrounging conduct of insects. All the more extraordinarily, the correspondence that happens between the subterranean insects by the assistance of pheromone to discover an enhanced way between the food source and the insect province, as more routes between the colony and the food location takes place, the additional pheromone is laid upon. As pheromone decays naturally in environment, the chances of ant choosing a wrong path is minimum.
The artificial ants maintain the record of the last step and thus differ from their natural descended. To generate the candidate solution, the algorithm uses historic and heuristic information and keeps old solution in consideration while forming new solution. At a discrete time, using probabilistic step-wise manner the solutions are constructed. For a solution to be selected, its probability is calculated. The probability depends upon the heuristic component and historic data. The heuristic component is calculated as the cost of solution from that stage to final stage. Every time a better solution is obtained, the previous solution is discarded and history is updated.

A. Problem Representation

The issue can be addressed as a chart, G=(N, E) where N addresses the virtual machine in the environment and E signifies the planning of occupations to VMs. Initially, all the artificial ants are present in the virtual machine. At the iteration, ants move from one virtual machine to another building the load balancing solution, until they have visited the entire environment.

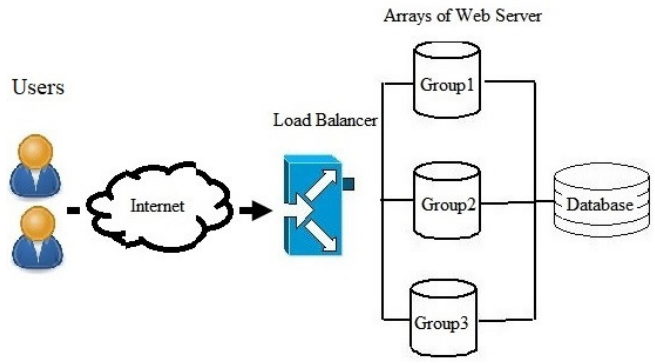

Fig. 1. Load Balancing Algorithm Architecture

B. Proposed Algorithm

The process flow of the proposed algorithm is depicted in figure 2. The pseudo code of the algorithm is described

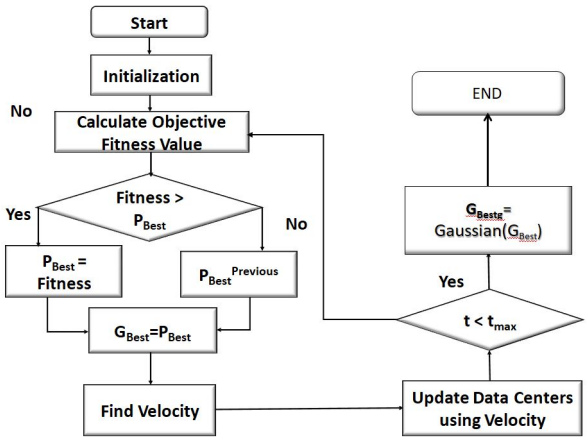

Fig. 2. Process Flow of the proposed algorithm

below.

Algorithm: Mutative ACO Algorithm

Input: List of Task represented as Cloudlets *n* and List of virtual machine *m*, initial state q_0, heuristic coefficient β, history coefficient α, decay factor γ

Output: P_{best}

Steps:

1. Initialize:

$i,j \leftarrow 0$

Set UpdateLocal=null.

$P_{best} \leftarrow$ Intial_Heuristic_Solution(VM)

2.$P_{bestcost} \leftarrow$ Intial_cost(C_h)

3. On *m* VM initialize *x* ants randomly

4. Initialize Pheromone as

5.$Pheromone_{init} \leftarrow$ 1.0/VM * $P_{bestcost}$

6.$Pheromone \leftarrow$ InitializePheromone($Pheromone_{init}$)

while ($P_{best} < P_{bestcost}$)

  for *j*=1 to *VM*, **do**

    $C_i \leftarrow$ Solution(Pheromone, VM, β, q_0)

    $C_{icost} \leftarrow$ Cost(C_i)

    if $C_{icost} < P_{bestcost}$ **then**

      $P_{bestcost} \leftarrow C_{icost}$

      $P_{best} \leftarrow C_i$

    end

    UpdateLocal(Pheromone, C_i, C_{icost},α)

  end

  UpdateGlobal(Pheromone, P_{best}, $P_{bestcost}$,γ)

end

return P_{best}

With the Constraints on jobs and resources, the fitness function is calculated based on route exploration. Since artificial ants are restricted to the digital world; they have a visibility factor of the solution for problem instance they are going to solve. The run-time for a job *j* from another task *i*, can be given by:

$$a_{ij}(t) = 1/(E_{timei} + I_{timej}) \tag{1}$$

where

a_{ij} is the artificial ant = 1....n

E_{timei} is the end time of task$_i$

I_{timej} is the idle time before starting task$_j$

For choosing a route by artificial ant, a probability is associated with the path. A probability P_{ij} is associated with the path i $\rightarrow$ j being selected by the ant.

$$p_{ij}^k = \begin{cases} \dfrac{{\tau_{ij}}^{\alpha}{\eta_{ij}}^{\beta}}{\sum\limits_{u \notin M_k} {\tau_{iu}}^{\alpha}{\eta_{iu}}^{\beta}} & , j \notin M_k \\ 0 & , j \in M_k \end{cases} \tag{2}$$

For the iterative construction of paths by artificial ants, the probability of decision is given by equation (2). This probability is used by an artificial ant at k, which is present at node i for choosing the next node j after it has traversed nodes in M_k.

In the case of known paths, artificial ant chooses the next node j by selecting the highest value of a path which is calculated using the quantity of pheromone and length of the path. In equations (2), τ_{ij} gives the value corresponding to the quantity of pheromone present on a path connecting i and j, and η_{ij} is the length of path ij. For calculating the pheromone and heuristic, predefined parameters α and β are used respectively. Two updates Global and Local are defined on the pheromone trail. If a better path is found, more pheromone is deposited on it as a reward, This is called Global update and can be defined as:

$$\tau_{ij} = (1-\lambda)\tau_{ij} + \lambda\Delta\tau^k, \quad \forall(ij) \in E^k \tag{3}$$

where E^k defines the set of overall edges in path ant k used, quality of solution is given by $\Delta\tau^k$, and λ is constant in interval $[0, 1]$. Not all the ants need to deposit pheromone on all the paths. To make the search more optimized and greedy, and the pheromone is deposited only on the best path that is available at that moment.

To avoid a situation where a large portion of pheromone is deposited on a single path, local updates are used. Local updates ensure that no path becomes too strong and thus uses pheromone evaporation. Therefore, every time a new path is discovered by ants, the previous path loses some pheromone. This can be given by 4:

$$\tau_{ij} = (1-\gamma)\tau_{ij} + \gamma\tau_0 \tag{4}$$

where γ is constant in interval $[0, 1]$, and τ_0 is the quality of the solution.

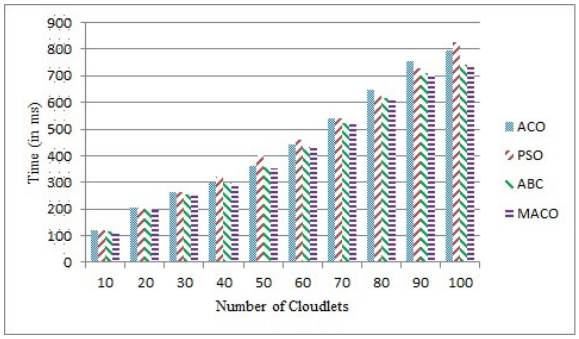

Fig. 3. Makespan Time when VM is 50

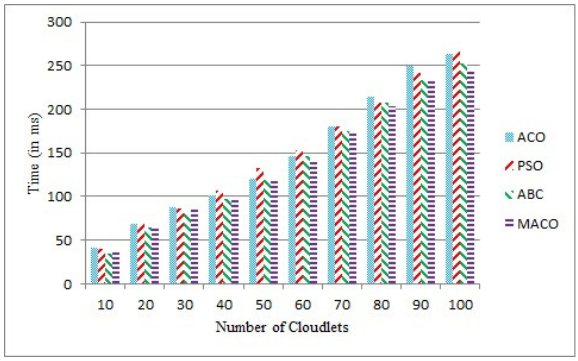

Fig. 4. Throughput when VM is 50

IV. Simulation And Result

This section discusses the simulation environment, parameter setting and results obtained after simulation.

TABLE I
Experimental Parameters for S0imulation

Entity	Parameters	Values
User Cloudlets	Cloudlets	50-500
	Length	500-10000
Host	Hosts	4
	RAM	8GB
	Storage	40GB
	Bandwidth	1000
Virtual Machine	Numbers of VMs	4
	RAM	2 GB
	Storage	10 GB
	Operating System	Windows
	Policy	Time sharing
	CPUs	2
Data centers	Numbers of Data Centers	10-50

1) Implementation Environment: The proposal is simulated and tested on CloudSim 3.0. CloudSim is built by CLOUDS laboratory, Australia and is completely written in Java. Along with Eclipse IDE, CloudSim allows users and researchers to model and simulate cloud infrastructure and services.

2) Parameters setting: For contrasting the exhibition of the proposed calculation, QoS boundaries like makespan time, throughput, and reaction time have been thought of. The QoS boundaries are contrasted and the proposed calculation with the current burden adjusting calculation for various upsides of server farms (DCs). The exploratory arrangement is displayed in table I. The scope of server farms is taken from 10-50. The QoS boundaries considered for looking at the exhibition of the calculation are makespan time, reaction time and throughput. The value of the virtual machine is kept fixed while varying the value of jobs. The different values of the virtual machine for which results are depicted are 50, 25 and 10. 3, 6 and 9 show the comparison of makespan time when the value of VM is 50, 25 and 0 receptively. 4, 7 and 10 shows the comparison for throughput while 5, 8 and 11 depicts the comparison for response time. The scope of

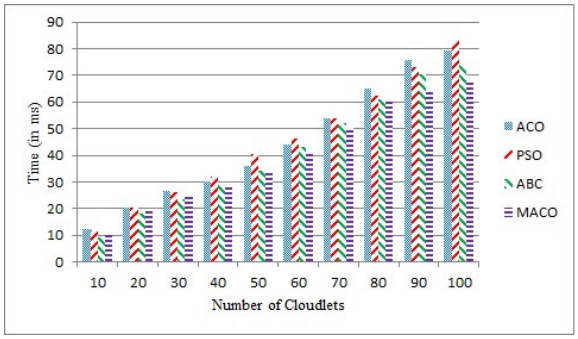

Fig. 5. Response Time when VM is 50

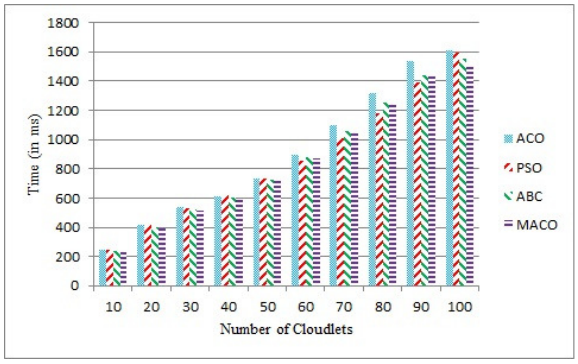

Fig. 6. Makespan Time when VM is 25

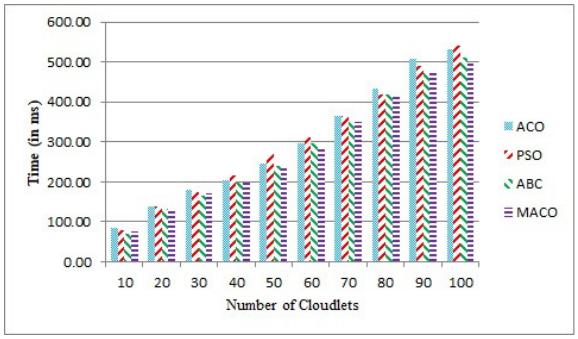

Fig. 7. Throughput when VM is 25

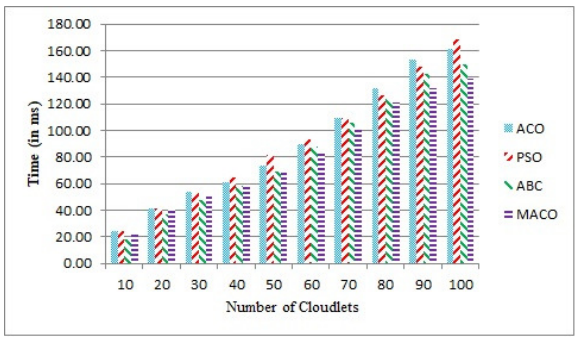

Fig. 8. Response Time when VM is 25

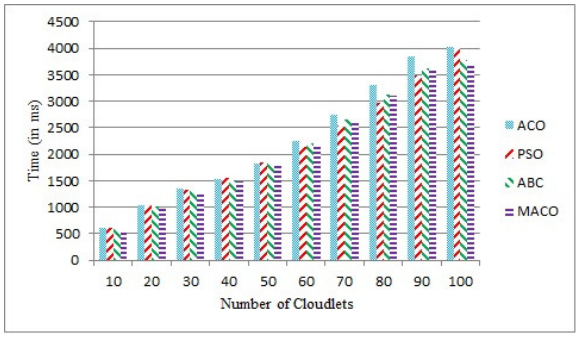

Fig. 9. Makespan Time when VM is 10

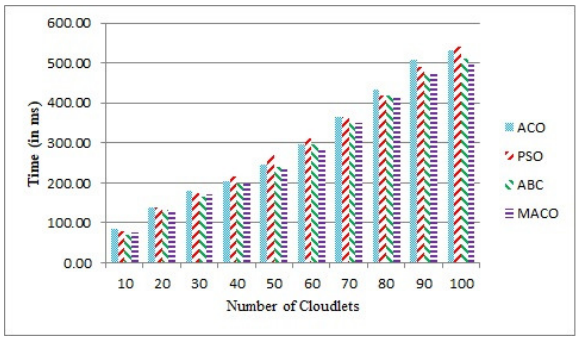

Fig. 10. Throughput when VM is 10

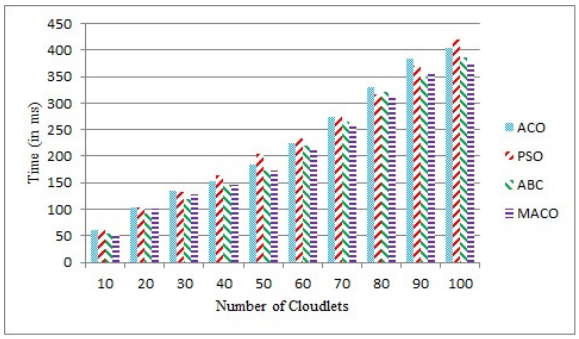

Fig. 11. Response Time when VM is 10

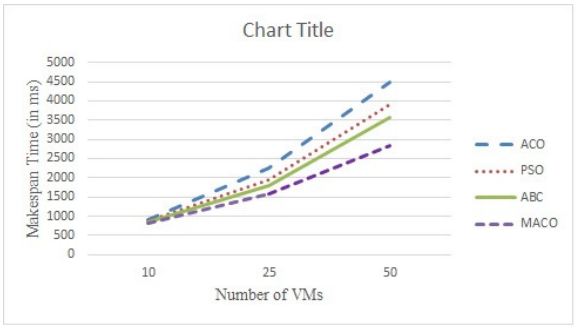

Fig. 12. Average Makespan Time

cloudlets is taken from 50-300 for each number of DCs

A. Experimental Result

This part diagrams the outcomes acquired from contrasting distinctive burden adjusting calculations and the proposed calculation. The proposed calculation is contrasted and ACO, PSO and ABC. The QoS boundaries considered for examination are makespan time, reaction time and throughput. The number of jobs is varied from 50-500 keeping the data centers constant. For different values of data centers, results are obtained.

Figure 3-11 shows the correlation of the proposed calculation with calculations present in the writing. In the event that the quantity of cloudlets is little, the presentation of the proposed calculation stays identical to the calculation utilized for examination. As the worth of the quantity of cloudlets increment the worth makespan time stay steady as the proposed calculation disposes of the nearby minima accordingly the exhibition of calculations such ACO, PSO which struck in neighborhood minima debases as the quantity of cloudlets increments though the proposed calculation picks the best wellness work that is determined while disposing of the other wellness work esteem limiting the worth of makespan. Figure 12 depicts the average makespan time of the prospered algorithm with the algorithm in literature. The proposed algorithm can improve existing algorithms by 8-20% for different QoS parameters.

V. Conclusion

In cloud computing, load balancing is an important issue as the service providers have to deal with numerous clients and their needs at the same time. As the request is irregular in nature, so the load balancing becomes more difficult. This paper presents an algorithm that tries to balance the load in the cloud environment. The proposal MACO reduces the makespan time and further develops the fitness function while keeping up with other QoS parameters. The algorithm deals with the resources successfully distributes over the data centers. The proposed algorithm shows further developed outcomes in the two instances of shifting server farms and undertakings. In the future, we can explore other parameters that help in predicting the future load for balancing the load at data centers.

References

[1] P. Pradhan, P.K.Behera, B.N. Ray, "Modified round robin algorithm for resource allocation in cloud computing," Procedia Computer Science, vol. 85, pp 878-90, Jan 2016.

[2] V.K. Reddy, B.T. Rao, L.S. Reddy, "Research issues in cloud computing," Global Journal of Computer Science and Technology, vol. 11, no. 11, pp 58-64, Jul 2011.

[3] M Alouane, H. El Bakkali, "Virtualization in Cloud Computing: NoHype vs HyperWall new approach," Proceeding of International Conference on Electrical and Information Technologies (ICEIT), pp 49-54, May 2016.

[4] S. Afzal, G. Kavitha, "Optimization of task migration cost in infrastructure cloud computing using IMDLB algorithm," Proceeding of International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET), pp 1-6, Dec 2018.

[5] S.L. Chen, Y.Y. Chen, S.H. Kuo, "CLB: A novel load balancing architecture and algorithm for cloud services," Computers & Electrical Engineering, vol. 58, pp 154-60, Feb 2017.

[6] D. Magalhães, R.N. Calheiros, R. Buyya, D.G. Gomes, "Workload modeling for resource usage analysis and simulation in cloud computing," Computers & Electrical Engineering, vol. 47, pp 69-81, Oct 2015.

[7] A. Aditya, U. Chatterjee, S. Gupta, "A comparative study of different static and dynamic load balancing algorithm in cloud computing with special emphasis on time factor," International Journal of Current Engineering and Technology, vol. 5, no. 3, pp 64-78, Jun 2015.

[8] T. Deepa, D. Cheelu, "A comparative study of static and dynamic load balancing algorithms in cloud computing," Proceedings of International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), pp 3375-3378, Aug 2017.

[9] R. Gupta, "Review on existing load balancing techniques of cloud computing," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 2, pp 168-71, Feb 2014.

[10] I. Nwobodo, "Cloud Computing: A Detailed Relationship to Grid and Cluster Computing," International Journal of Future Computer and Communication, vol. 4, no. 2, pp 82, Apr 2015.
[11] L.D. Dhinesh Babu, and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," Applied soft computing, vol. 13, no. 5, pp 2292-2303, 2013.
[12] K.R.R. Babu, P. Samuel, "Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud," In Proceedings of Innovations in bio-inspired computing and applications, pp 67-78, 2016.
[13] D.C. Devi, V.R. Uthariaraj, "Load balancing in cloud computing environment using improved weighted round robin algorithm for non-preemptive dependent tasks," The scientific world journal, vol. 2016, pp 1-14, 2016.
[14] S Mousavi, A Mosavi, A.R. Varkonyi-Koczy, "A load balancing algorithm for resource allocation in cloud computing," In Proceeding of International Conference on Global Research and Education, pp 289-296, Sep 2017.
[15] I. De Falco, E. Laskowski, R. Olejnik, U. Scafuri, E. Tarantino, M. Tudruj, "Extremal optimization applied to load balancing in execution of distributed programs," Applied Soft Computing, vol. 30, pp 501-513, May 2015.
[16] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K.P. Singh, R. Rastogi, "Load balancing of nodes in cloud using ant colony optimization," In Proceedings of UKSim 14th international conference on computer modelling and simulation, pp. 3-8, Mar 2012.
[17] J. Adhikari, S. Patil, "Load balancing the essential factor in cloud computing," International Journal of Engineering Research & Technology (IJERT), vol. 1, no. 10, pp 1-5, Dec 2012.
[18] M. Mohaupt, & A. Hilbert, "Integration of Information Systems in Cloud Computing for Establishing a Long-term Profitable Customer Portfolio," IAENG International Journal of Computer Science, vol. 40, no. 2, pp 124-133, 2013.
[19] G.B. Bindu, K. Ramani, and C. Shoba Bindu, "Optimized Resource Scheduling using the Meta Heuristic Algorithm in Cloud Computing," IAENG International Journal of Computer Science, vol. 47, no. 3, pp 360-366, 2020.