Proceedings of the Second International Conference on Computing Methodologies and Communication (ICCMC 2018)
IEEE Conference Record # 42656; IEEE Xplore ISBN:978-1-5386-3452-3

A MODERN APPROACH FOR LOAD BALANCING USING FOREST OPTIMIZATION ALGORITHM

Subasish Mohapatra[1] , Ishan Aryendu[2], Anshuman Panda[3], Aswini Kumar Padhi[4]

Department of Computer Science and Engineering
College of Engineering and Technology
Bhubaneswar, Odisha, India
[1]smohapatra@cet.edu.in, [2]ishanaryendu47@gmail.com, [3]ansh2frnds@gmail.com, [4]padhiaswinikumar@gmail.com

Abstract—**Load Balancing plays a dynamic role in keeping the tempo of the Cloud Computing framework. This research paper proposes a Forest Optimization Algorithm for load balancing in distributed computing structure. This depends on the conduct of the trees in the forest and uses the seed dispersal strategies for streamlining the makespan, which results in improved average response time and the total execution time. The simulation results prove that the proposed algorithm gives better outcomes than its counterpart load balancing algorithms.**

Keywords: **Load balancing, Cloud Computing, Forest Optimization Algorithm, makespan, average response time, total execution time.**

I. INTRODUCTION

Cloud computing is useful for managing and providing services over the web. It is defined as "a model for enabling ubiquitous, convenient, on-demandnetwork access to a shared pool of configurable computing resources like networks, servers, storage, applications, andservices, that can be rapidly processed and released with minimal management eff orts or service provider interactions" [1]. It is an integrated approach using parallel and distributed computing which shares resources like hardware, software, and information with computers or other devices on demand. With the aid of cloud computing and internet facility, the customer can access the resources by paying for the duration of use. Cloud models use virtualization technology. In this model, the virtual machine (VM) acts as an execution unit which inturn, serves as the foundation for cloud computing technologies. The Cloud computing must assign the computational tasks to the most suitable virtual machines from the dynamic pool of the VMs by considering the requirements of each task and the load on the VMs. The requests from the clients are directed to any of the data centers in the cloud. Then the same requests are directed by the data center to the most suitable VMs based on the cloud management policies depending on the loads on the individual VMs.

Load balancing is a strategy that distributes the workload among various nodes in each condition such that it guarantees that no node in the framework is over stacked or under stacked.An effective load balancing algorithm will ensure that each node in the framework carries out thepretty much same volume of work. [2] The duty of load balancing algorithm is to distribute work among the unutilized assets, so the average response time is enhanced and in addition, it gives productive resource usage. The eccentrics are expected to the consistently evolving conduct of the cloud. The primary concentration of load balancing in the cloud area is in apportioning the heap of requests evenly among the nodes keeping in mind the end goal to fulfill the client necessities and to maximize asset use by grouping the accessible load to different nodes. [3]

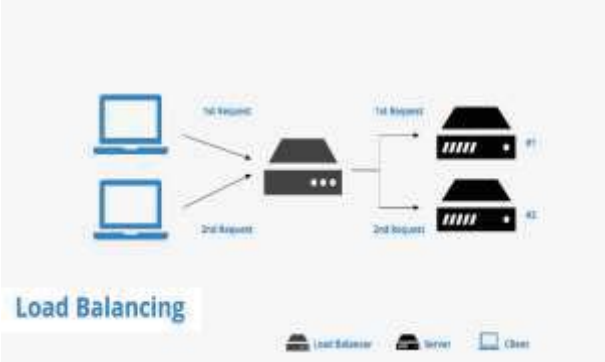

Fig. 1 Framework for load balancing

The seed dispersal is at the core of this optimization algorithm. A few seeds fall close to the parent tree and grow. This is named as local seed dispersal [4] or "Local seeding". Nevertheless, a clear majority of the seeds are diverted more distant from the parent tree by a few characteristic operators. In this way, the trees develop in various areas of the forest. This technique named as the "Global seeding". [4]

We perform a comparative analysis of evolutionary algorithms likeGA and PSO with the Forest optimization algorithm for load balancing. The techniques used by the plants and trees are employed to search for global and local optimums of the fitness function, i.e.

$$\boldsymbol{Fitness} = \frac{\mathbf{1}}{\boldsymbol{makespan}} \times \boldsymbol{AUM_i}.$$

II. RELATED WORK

Load Balancing is a process of distributing load with an aim of maximizing throughput and minimizing response time [6, 7]. It likewise avoids the situation in which a few of nodes are daintily

stacked and others are over-burden. In distributed computing condition, the jobs arrive randomly with random CPU utilization times. Because of this random arrival of jobs, specific resources are heavily loaded, and the other resources are less loaded. Different researchers have proposed several load balancing algorithms. Cloud framework has various data centers, spread over several topographical channels. Each of them comprises of several servers known as virtual machines (VM). At any point when a client presents an assignment, the cloudlets gather it. Data center regulator deals these endeavors. It utilizes a VM load balancer to figure out which VM should be responsible for processing the forthcoming requests. A VM load balancer utilizes different plans to adjust the load in complex cloud computing frameworks [8].

Task Scheduling principally comprises of a two-stage undertaking to meet the dynamic necessities of the clients. This calculation accomplishes load distribution by assigning tasks to virtual machines, thereafter the virtual machines to the host assets, respectively. It enhances the average response time of the framework. It likewise gives better use of resources. [9]

Opportunistic Load Balancing is an endeavor to keep every node occupied. It doesn't think about the present workload on every node. The calculation is very straightforward and aides in load balancing, however, its weakness is that for each task the normal execution time isn't considered. In this way, the entire finish time (i.e. makespan) is exceptionally poor. [10]

In Round Robin algorithm, every one of the tasks is partitioned amongst the processors. Here each task is allocated, just like in the round-robin tournament. The workload circulation is equivalent amongst the processing units. Distinctive tasks don't have a similar processing time. Sometimes a few nodes might be vigorously loaded, and others stay sit out of gear. Here the time quantum decides the task designation. [11]

Randomized algorithm kind of a static load distribution algorithms. Here, a procedure can be dealt by node 'n', having a probability 'p'. At the point when every one of the jobs is of equivalent load, at that point this algorithm functions admirably. The issue emerges when loads are over various computational intricacies. But it isn't implied for deterministic approaches. [12]

Min-Min Algorithm begins with an arrangement of every unassigned job and the minimum finish time for all assignments is found. From that point onward, the minimum value is chosen. Formerly, the execution time for every single other job is refreshed on that piece of equipment and a similar method is rehashed until the point that every one of the undertakings is relegated to the assets. With this algorithm, the primary issue is starvation. [13]

The max-min algorithm, practically the equivalent of the min-min algorithm. Primary contrast is that to start with, we should discover minimum execution time and afterward the maximum value is chosen. In the wake of finding the thoroughgoing time, the task runs on the chosen machine. At that point, the execution time for all the tasks is refreshed. This is finished by sampling the execution time of the tasks assigned, with the execution times of other tasks. At that point, all the assigned tasks are expelled from the itemize that has been executed by the framework. [14]

Honeybee algorithm is a nature-enlivened self-association algorithm. It uses local server operations to accomplish global load distribution. The execution of the framework canbe improved by an expansion in assorted variety. The significant downside of this algorithm is that throughput isn't expanded with an expansion in size of the framework. [15]

In Active Clustering, a similar set of nodes in the framework is assembled and they cooperate. It is a self-aggregation load distribution procedure where the grid is redone to adjust the burden on the framework. Framework enhancement utilizes comparative employment assignments by associating comparative administrations. Framework Performance enhances with enhanced assets. The throughput is enhanced by viable utilization of these resources. [16]

Compare and Balance algorithm is utilized for achieving a symmetry condition and oversee uneven framework load. This algorithm depends on the probability of various virtual machines running on the present host and the entire cloud framework. The present host is contrasted with the chosen one. On the off chance that a load on the current host is greater than the chose to have, it exchanges the additional load on that node. Each of the hosts plays out a similar methodology. [17]

Lock-free multiprocessing for load distribution proposes a lock-free multiprocessing model that maintains a strategic distance from the utilization of shared memory rather than other multiprocessing load reduction arrangements where the joint memory is utilized, and a lock is utilized to track a client session. It is accomplished by adjusting the kernel. Such arrangements help in enhancing the general execution of load equalizer in a multi-core condition by sampling numerous load distribution processes on a single load distributor. [17]

Ant colony optimization is a multi-specialist approach for troublesome combinatorial issues. Some cases utilizing this approach are TSP and QAP. The perception of the ant colonies enlivened them. Ant's conduct is guided towards the survival of their colonies. [18]

Load distribution issues are multifaceted and computationally immovable. It is very hard to discover the all-inclusive ideal arrangement by utilizing deterministic polynomial time algorithms. In this paper, we have put forward a Forest optimization algorithm to solve these problems in the cloud framework. The plans are assessed, and studies are made with other focused methodologies.

III. PROBLEM STATEMENT

Cloud system consists of distributed nodes interconnected by high-speed links. These nodes are responsible for executing different applications with diverse resources and computational requirements. Cloud systems are well suited to meet the computational demands of large, diverse groups of tasks. Consider a cloud system with m independent computing nodes spread across many DCs as

$$M = \{M_1, M_2, M_3 \ldots\ldots M_m\}$$

and let there be a set of n tasks represented as

$$T = \{t_1, t_2, t_3 \ldots\ldots t_n\}$$

Each task has an expected time to compute on node , denoted as tij .tij represents expected time to compute task on themachine . Let A(j) be the set of tasks assigned to node ;

and be the total time machine , needs to finish all the task in A(j). Hence, for all task in A(j), . This is otherwise denoted as Lj and defined as the load on node Mj . The basic objective of load balancing is to minimize the response time, which is defined as a maximum load on any node . Let corresponds to each pair (i,j) of node and task such that

(a) = 0; implies that the task i not assigned to node j
(b) = ; indicates the load of task i on node j

The load on node can be represented as . The load balancing problem aims to find an assignment that minimizes the maximum load. Let L be a load of m nodes. Hence, our objective is to minimize L. subject to

(a) $\sum_{j=1}^{m} x_i$
(b) $\sum_{i=1}^{n} x_{ij}$

The problem of finding minimum response time is intractable with a number of task and computing nodes. Hence, the sub-optimal solution is made by developing Forest optimization algorithm separately. Performance comparisons are made with the existing algorithms.

IV. PROPOSED ALGORITHM

FOA [19] is an evolutionary algorithm which starts with a preliminary set of trees, which are the potential solutions to the problem statement. A tree can be viewed as an array of x+1 variables, comprising of "Age" of the tree and several other attributes.

$$T_S = [age, v_1, v_2, ..., v_x]$$

Initially, the age of the tree is set to '0'. Thereafter, new trees are generated via local seeding and therefore the forest is updated with these trees. Their age is set as '0' and their parent's age is incremented by '1'. In every cycle of the algorithm, each parent tree generates "LSC" number of new trees. In this phase, the new solutions are variants of the parent were randomly selected attribute of the parent solution is incremented by a random value within a small interval, i.e. $[-d_x, d_x]$. This will result in exponential increase in the state space.

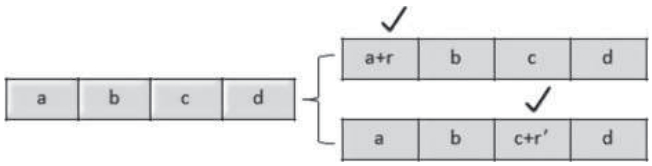

Fig. 2 Demonstration of local seeding with LSC=2

In order to restrict the state space to a finite range, limiting constraints namely "LT" and "AL" are used. When the age of a tree exceeds the value of "LT", it is eliminated to create the candidate population for the successive stage. These trees are sorted depending upon their fitness values. If the total number of solutions exceed the "AL" value, the surplus trees are removed from the initial population and are added to the candidate population.

In global seeding, new candidate population is taken into consideration and redundant local optimums are eliminated. The process is performed on a "TR" percentage of candidate population. "GSC" number of attributes are chosen randomly from the selected candidate solutions and replaced with a random value within its range. The newly formed solutions are added to the initial population. Later, the trees are sorted reliant on fitness values. The one with the most effective value is chosen and its age is set to '0'. It ensures that the tree doesn't age and perpetually stays within the forest.

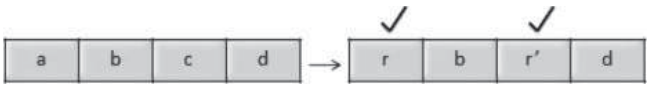

Fig. 3 Demonstration of global seeding with GSC=2

The process repeats till the termination criteria are met. These steps are depicted visually in the flowchart given in Fig. 4.

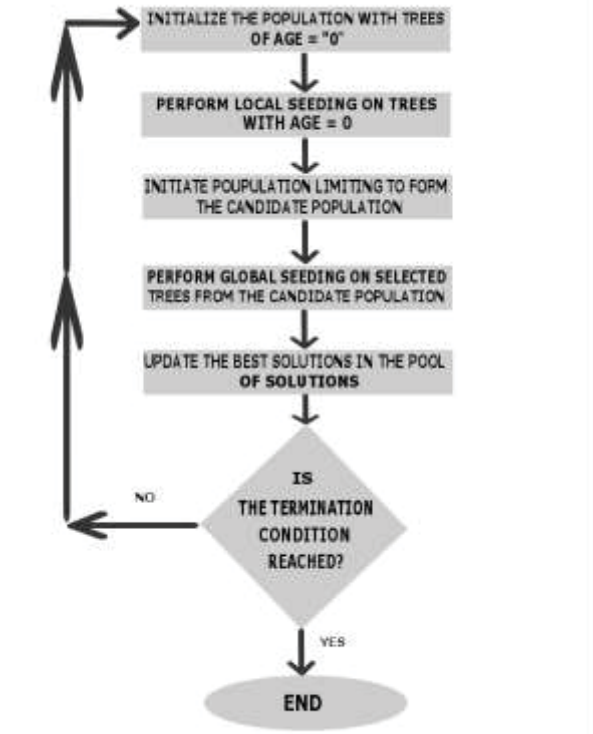

Fig. 4 Flowchart of FOA.

The lookup table for the shorthand notations used in the paper is given in Table 1.

Table 1: **LOOKUP TABLE FOR DIFFERENT PARAMETERS**

PARAMETERS	NOTATIONS
LSC	Local Seeding Changes
GSC	Global Seeding Changes
LT	Life Time

AL	Area Limit
TR	Transfer Rate

Algorithm FOA (life time, LSC, GSC, transfer rate, area limit)

Input: life time, LSC, GSC, transfer rate, area limit
Output: near optimal solution for objective function *f(x)*
1. Initialize forest with random trees
   1.1 Each tree is a (D+1)-dimensional vector x_i, x_i= (age, x_{i_1}, x_{i_2} . . ., x_{i_D}) for a D-dimensional problem
   1.2 The "age" of each tree is initially zero
2. While stop condition is not satisfied do
   2.1 Perform local seeding on trees with age 0
      - For i=1: "LSC"
        - Randomly choose a variable of the selected tree
        - add a small amount dx- dx ε [-Δx, Δx]- to the randomly selected variable
      - Increase the age of all trees by 1 except for new generated trees in this stage
   2.2 Population limiting
      - Remove the trees with age bigger than "life time" parameter and add them to the candidate population
      - Sort trees according to their fitness value
      - Remove the extra trees that exceed the "area limit" parameter from the end of forest and add them to the candidate population
   2.3 Global seeding
      - Choose "transfer rate" percent of the candidate population
      - For each selected tree
        - Choose "GSC" variables of the selected tree randomly
        - Change the value of each variable with other randomly generated value in the variable's range and add a new tree with age 0 to the forest
   2.4 Update the best so far tree
      - Sort trees according to their fitness value
      - Set the age of the best tree to 0
3. Return the best tree as the result

Fig. 5 Forest Optimization Algorithm

V. OBSERVATIONS AND GRAPHS

Cloudsim is an open source software that is used to simulate different aspects of a cloud framework. The proposed algorithm is implemented on an Alienware machine with core i7 processor, 8GB of RAM, 64-bit windows 10 operating system and 1TB of SSD. Its numerical simulation results were obtained on the machine mentioned above. Also, a comparative study has been done on a proposed algorithm with two evolutionary approach algorithms i.e. GA and PSO. The unit of measurement of time isin microseconds.

Table 2 represents the average response time of Forest optimization, GA and PSO algorithms using 1000 cloudlets respectively.

Table 2: COMPARATIVE ANALYSIS OF AVERAGE RESPONSE TIME FOR 1000 CLOUDLETS

VM	FOREST	GA	PSO
50	10130.6	10507.9	10366
100	6191.3	6456.7	6331.6
200	3871.3	3998.9	3889.8
500	2489.2	2697.5	2453.8

Table 3 represents the total execution time of Forest optimization, GA and PSO algorithms using 1000 cloudlets respectively.

Table 3: COMPARATIVE ANALYSIS OF TOTALEXECUTION TIME FOR1000 CLOUDLETS

VM	FOREST	GA	PSO
50	1121230	1119105	1114412
100	1169261.4	1169650	1153649.7
200	1164508	1161590	1165405
500	1176111	1175710	1176290

Table 4 represents the average response time of Forest optimization, GA and PSO algorithms using 1000 cloudlets respectively.

Table 4: COMPARATIVE ANALYSIS OF AVERAGE RESPONSE TIME FOR 100 CLOUDLETS

VM	FOREST	GA	PSO
50	2593.5	2329.7	2191.5
100	2334	1995.4	1914.67
200	1782.4	1599	1462.9
500	1606.09	1441	1329.1

Table 5 represents the total execution time of Forest optimization, GA and PSO algorithms using 100 cloudlets respectively.

Table 5: COMPARATIVE ANALYSIS OF TOTAL EXECUTION TIME FOR 100 CLOUDETS

VM	FOREST	GA	PSO
50	118626.5	118488	119566
100	109816	110028	111652.6
200	106201.9	106962	107831.7
500	105000	107943	109137

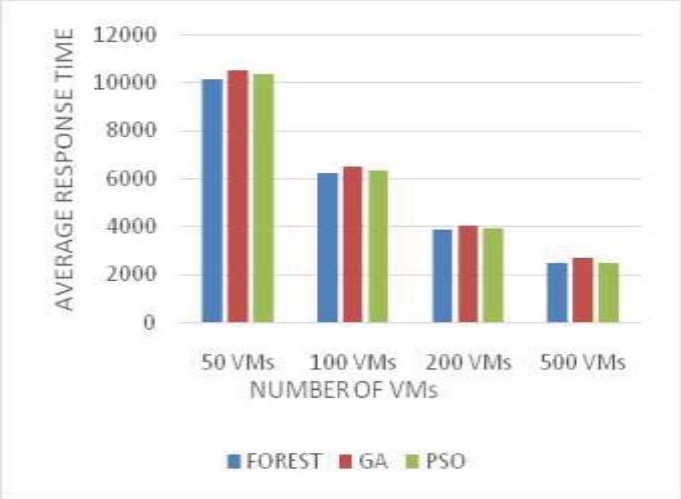

Fig. 6. Average response time for 1000 cloudlets

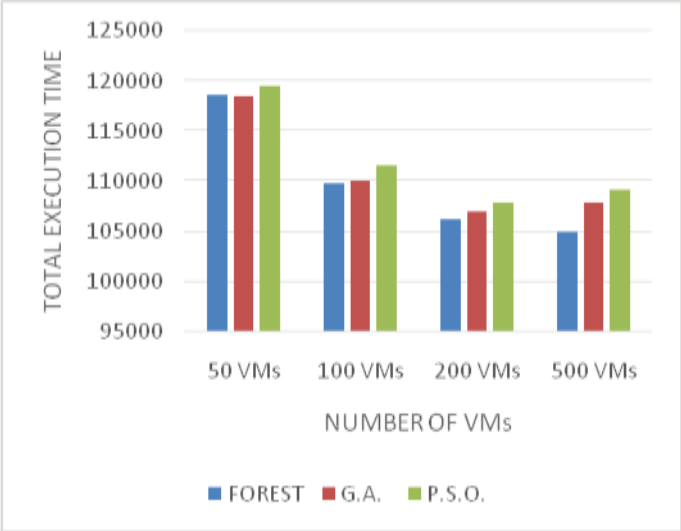

Fig. 7. Total execution time for 1000 cloudlets

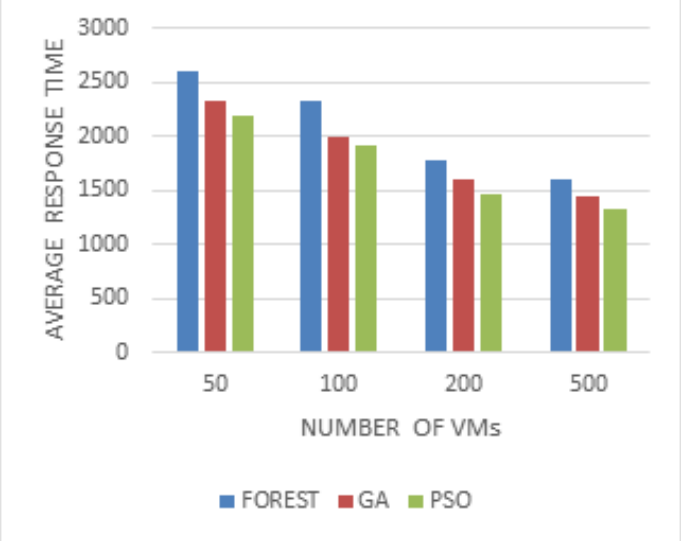

Fig. 8. Average response time for 100 cloudlets

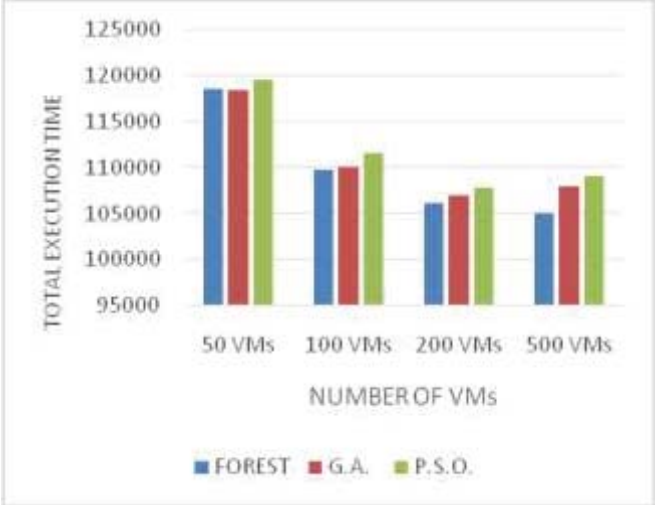

Fig. 9. Total execution time for 100 cloudlets

From the above graphs it is clear that Forest Optimization algorithm outperforms GA and PSO algorithms when the cloudlets to VM ratio significantly high. But as the ratio decreases, its performance is hindered due to the parameter presets.

VI. CONCLUSION AND FUTURE WORK

We have performed an exhaustive examination of FOA, GA and PSO calculations by leading investigations. The simulation result proves that the proposed FOA algorithmic program beats GA and PSO by minimizing the average response time and total execution time under significant load. On the off chance that the proportion of an aggregate number of tasks to that of the aggregate number of VMs is significantly high, we see extraordinary execution changes using the FOA. In any case, the parameters can be changed to perform better under lower loads. In the future, we may go for finding fitting estimations of these framework parameters for an adjusted execution under substantial, light and moderate loads.

REFERENCES

[1] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, 2013.

[2] Zenon Chaczko, Venkatesh Mahadevan, Sharzad Aslanzadeh, Christopher Mcdermid(2011) "Availability and Load Balancing in Cloud Computing" International Conference on Computer and Software Modeling IPCSIT VOL.14 IACSIT PRESS, SINGAPORE 2011

[3] Buyya R., R. Ranjan, and RN. Calheiros, "Intercloud: Utility oriented federation of cloud computing environments for scaling of application services," in Proc. 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), Busan, South Korea,2010.

[4] Cain, M. L., Milligan, B. G., & Strand, A. E. (2000). Long-distance seed dispersal inplant populations. American Journal of Botany, 87(9), 1217–1227.

[5] Ozsen, S., &Gunes, S. (2009). Attribute weighting via genetic algorithms forattribute weighted artificial immune system (AWAIS) and its application toheart disease and liver disorders problems. Expert systems with applications (Vol. 36, pp. 386–392). Elsevier

[6] Foster, I., Y. Zhao, I. Raicu and S. Lu, "Cloud Computing and Grid Computing 360-degree compared," in proc. GridComputing Environment Workshop.

[7] T. Desai and J. Prajapati, "A survey of various load balancing techniques and challenges in cloud computing," International Journal of Scientific & Technology Research, vol. 2, no. 11, pp. 158–161, 2013.

[8] S. B. Shaw and A. Singh, "A survey on scheduling and load balancing techniques in cloud computing environment," in Proceedings of the International Conference on Computer and Communication Technology (ICCCT). IEEE, 2014, pp. 87–95.

[9] Jinhua Hu, Jianhua Gu, Guofei Sun, Tianhai Zhao, "A Scheduling Strategy on Load Balancing of Virtual Machine source in Cloud Computing Environment", a 3rd International symposium on Parallel Architectures, Algorithms and Programming, IEEE 2010.

[10] S.-C. Wang, K.-Q. Yan, W.-P. Liao, and S.-S. Wang, "Towards a load balancing in a three-level cloud computing network," in Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), vol. 1, 2010, pp. 108–113.

[11] Nusrat Pasha, Dr. Amit Agarwal, Dr. Ravi Rastogi, ―Round Robin Approach for VM Load Balancing Algorithm in Cloud Computing Environment,‖ International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 5, May 2014.

[12] I. A. Mohialdeen, "Comparative study of scheduling algorithms in cloud computing environment," Journal of Computer Science, vol. 9, no. 2, pp. 252–263, 2013.

[13] H. Chen, F. Wang, N. Helian, and G. Akanmu, "User-priority guided min-min scheduling algorithm for load balancing in cloud computing," in Proceedings of the Parallel Computing Technologies (PARCOMPTECH). IEEE, 2013, pp. 1–8.

[14] S. B. Shaw and A. Singh, "A survey on scheduling and load balancing techniques in cloud computing environment," in Proceedings of the International Conference on Computer and Communication Technology (ICCCT). IEEE, 2014, pp. 87–95.

[15] S. M. Ghafari, M. Fazeli, A. Patooghy, and L. Rikhtechi, "Bee-mmt: A load balancing method for power consumption management in cloud computing," in Sixth International Conference on Contemporary Computing (IC3). IEEE, 2013, pp. 76–80.

[16] Randles, M., D. Lamb and A. Taleb-Bendiab, ―A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing,‖ in Proc. IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Perth, Australia, April 2010.

[17] Zhao Y. and Huang W. (2009) 5th International Joint Conference on INC, IMS and IDC, 170-175.

[18] Nishant, K. P. Sharma, V. Krishna, C. Gupta, KP. Singh, N. Nitin, and R. Rastogi, "Load Balancing of Nodes in Cloud Using Ant Colony Optimization." In Proc. 14th International Conference on Computer Modelling and Simulation (UKSim), IEEE, pp: 3-8, March 2012

[19] Ghaemi, Manizheh & Feizi Derakhshi, Mohammad Reza. (2014). Forest Optimization Algorithm. Expert Systems with Applications. 41. 6676-6687. 10.1016/j.eswa.2014.05.009.