Impact Factor Value: 4.029 ISSN: 2349-7084
International Journal of Computer Engineering In Research Trends
Volume 4, Issue 6, June-2017, pp. 277-284 www.ijcert.org

# Enhancing Load Balancing in Cloud Computing by Ant Colony Optimization Method

Prachi Verma[*1], Sonika Shrivastava[2], R.K. Pateriya[3]

[1] *MTech Scholar, Department of Computer Science & Engineering, MANIT, Bhopal, 462003, India*

[2]*Ph.D. Scholar, Department of Computer Science & Engineering, MANIT, Bhopal, 462003, India*

[3]*Associate Professor, Department of Computer Science & Engineering, MANIT, Bhopal, 462003, India*

{verma.prachi.92[1], ms271104[2], pateriyark[3]}@gmail.com

---

**Abstract:-** Cloud computing is an evolving technology which provides users "pay as you go" services on demand. Nowadays there is a tremendous increase in the use of the cloud by the clients due to its attractive features which results in a rapid growth of load on servers. Hence, load balancing has become a matter of concern in the domain of cloud computing. Load balancing is required to distribute the workload equally amongst all nodes in a network so that none of a node is overloaded or underloaded and each node does a similar amount of work in equal time. It minimizes the cost and time involved in the major computational models and helps to improve proper utilization of resources and system performance. Many approaches and algorithms are recommended by various researchers from all over the world to solve the problem of load balancing. In this paper, we present a technique built on Ant Colony optimization to address the issue of load balancing in a cloud environment.

**Keywords -** Cloud Computing; Ant colony optimization, Swarm intelligence; Load Balancing;

---

## 1. Introduction

Cloud computing is a newly progressing technique which offers online computing resources, storage and permits users to organize applications with enhanced scalability, availability and fault tolerance. Cloud computing is about storing the stuff on remote servers instead of on own computers or other devices. This information can be retrieved using the internet with any device, everywhere in the world as long as that device can support cloud computing systems. The cloud computing system is comprised of a front-end, which is the client side and a back-end which is a collection of the servers and computers owned by a third party which stores the data. A central server which is a fragment of the back-end follows protocols and uses middleware to communicate between networked computers. Cloud computing accumulates all the computing resources and manages them automatically [1]. Its characteristics describe a cloud computing system: on-need self-service, pooling of resources, access to the internet, the elasticity of service availability and measurement of services utilized by individual users. Cloud computing is everywhere with tools like Google Drives replacing Microsoft Office, Amazon Web Services replacing traditional enterprise data storage, banking websites replacing branch offices and Dropbox storing all our data and files. The cloud even provides different deployment models and service models.

The four deployment models present in cloud computing are: [2]

**1. Public cloud:** In the public cloud, the cloud provider provides resources for free to the public. Any user can make use of the resources; it is unrestricted. The public cloud is connected to the public internet for anyone to leverage.

 

**2. Private cloud:** In a private cloud, the planning and provisioning of the cloud are operated and owned by the organization or the third party. Here the hosted services are provided to a restricted number of people or group of individuals.

**3. Community cloud:** These type of cloud infrastructures exists for special use by a group of users. These are a group of users who share a common mission or have specific regulatory requirements, and it may be managed by the third party or organizations.

**4. Hybrid Cloud:** Hybrid Cloud provides the best of above worlds. It is created by combining the benefit of different types of cloud (private cloud & public cloud). In these clouds, some of the resources are provided and managed by public cloud and others as a private cloud.

The three different service models present in cloud computing are:

**1. Infrastructure as a Service (IaaS):** IaaS model provides just the hardware and the network. It allows users to develop and install their operating system, software and run any application as per their needs on cloud hardware of their own choice.

**2. Platform as a Service (PaaS):** In PaaS model, an operating system, hardware, and network are provided to the user. It enables users to build their applications on cloud making use of supplier specific tools and languages

**3. Software as a Service (SaaS): ):** In SaaS model, a pre-built application together with any needed software, hardware, operating system and the network is provided to the user.

# 2. Load Balancing

Load balancing is a serious concern in cloud computing. With the increase in attractiveness of cloud computing among users, the load on the servers and the quantity of processing done is surging drastically. There are multiple nodes in the cloud, and due to the random allocation of a request made by the client to any node, the nodes become unevenly loaded. So to avoid the condition where some nodes are either severely loaded or under loaded, the load balancer will evenly divide the workload among all the nodes [3]. Thus load balancing will equally distribute the workload among the nodes, and it can help in minimizing delays in communication, maximizing the throughput, minimizing execution time and maximizing resource utilization [3].

## 2.1 Goals of load balancing:

Some of the key purposes of a load balancing algorithm as pointed out by are:

1. It should possess fault tolerance.
2. It should be capable of modifying itself according to any change or expansion in the distributed system configuration [3].
3. Regarding system performance, it should give greater overall improvement at a minimal cost.
4. Regardless of the origin of job it must treat all jobs in the system equally.
5. It should also maintain system stability.

## 2.2 Issues of Load Balancing

The issues of load balancing are described below [4]:

1. Load balancing becomes critical because, in the middle of execution, the processes may shift amongst nodes to ensure equal workload on the system [5].
2. For a load balancing scheme to be good it should be scalable, general and stable and should add minimal overhead to the system. These requirements are interdependent [6].
3. One of the critical aspects of the scheduling problem is load balancing [7]. The challenge for a scheduling algorithm is to avoid the conflict between prerequisites: fairness and data locality.
4. Algorithms for load balancing have to be dependent on the hypothesis that the on hand information at each node is accurate to avoid processes from being continuously circulated the system without any progress[5].
5. How to accomplish a balance in load distribution amongst processors such that the computation can be done in the minimum possible time is one of the important problems to resolve.
6. Load balancing and task scheduling in distributed operating systems is a vital factor in gross system efficiency because the distributed system is not pre-emptive and non-uniform, that is, the processors may be different [7].

## 2.3 Components of Load Balancing Algorithms:

A load balancing algorithm has five major components [8]

1. Transfer Policy: The portion of the load balancing algorithm that picks a job for moving from a local node to a remote node is stated as Transfer policy or Transfer strategy.

 

2. Selection Policy: In this policy, it specifies the processors involved in the load exchange (processor matching) so that the overall response time and throughput may be improved.
3. Location Policy: The portion of the load balancing algorithm that is responsible for choosing a destination node for a task to transfer is stated as location policy or Location strategy.
4. Information Policy: The part of the dynamic load balancing algorithm that is in charge of gathering information about the nodes present in the system is started to as Information Policy or Information strategy.
5. Load Estimation Policy: In this policy, it determines the total workload of a node in a system.

## 3. Classification Of Load Balancing Algorithms:

Load balancing algorithms have been classified based on current state of system and who initiated the process
1. Depending on which user initiates the process:
A. Sender-Initiated: Sender or client initiates the execution of load balancing algorithm on identifying the need for load balancing.
B. Receiver-Initiated: Receiver or server initiates the execution of load balancing algorithm on identifying the need for load balancing.
C. Symmetric: This type of algorithm is a blend of sender-initiated type and receiver-initiated type algorithms.
2. Depending on current state of system
Static algorithm: In the static algorithm, there is a uniform distribution of traffic among the servers. This algorithm needs a prior understanding of system resources so that the judgment of shifting of the load does not depend on the current state of the scheme. The static algorithm is perfect in the system which has fewer inequalities in load [3].
Dynamic Algorithm: In the dynamic algorithm, for balancing the load the lightest server in the entire system or network is looked upon and preferred. For this real-time communication with the network is needed which can increase the traffic in the system. Here to make decisions for managing the load the current state of the system is used [3].

## 4. Related Work

Load Balancing is a vital part of Cloud Computing framework to accomplish maximum consumption of resources. Ant colony optimization (ACO) algorithm was projected by Marco Dorigo and his colleagues in 1992. It is motivated from real ants when finding for a food ant travels randomly, and in return trip, they deposited some chemical pheromone. By the quantity of this pheromone, other ants use the shortest path on which more pheromone value is deposited. The ants discover the shortest path from the nest to a food source by an indirect communication amongst the ants via pheromone trails.
Ekta Gupta et al. [9] Proposed technique based on the ACO where redistribution of overloaded nodes is done based on the threshold value. If the load on current node is less than the threshold ant will then search for overloaded node among the neighboring nodes of the current node and move to the underloaded node by checking its Foraging Pheromone value. Here ants move only in one direction at a time[17].
Mayanka Katyal et al. [10] discussed various load balancing pattern such as static load balancing, distributed and non-distributed dynamic load balancing, centralized and hierarchical load balancing. Although static load balancing patterns offer simplest and effort free simulation and monitoring of the environment, they are not capable of handling heterogeneous nature of the cloud. Considering dynamic load balancing algorithm, these are appropriate in a heterogeneous environment of cloud computing but are problematic to simulate.
Shagufta Khan et al. [11] implemented SALB algorithm. In this enhancement to ACO algorithm is proposed. Artificial ants move forward and backward direction to find the overloaded node and update value in pheromone table. Throughput, Response time, less energy consumption is achieved, but it gives less performance.
Kun Li et al. [12] proposes a cloud task scheduling strategy by Load Balancing Ant Colony Optimization algorithm. The core aid of the work was to balance the whole system load while trying to minimize the make span of a given task set.

## 5. Proposed Work

Macro Dorigo introduced this concept i.e. Ant Colony in his Ph.D. in 1992. Ant has proficiency to discover the path between nest and food. When ants find the path, they lay some chemical substances i.e. Pheromone and it is on the ground. All the ants can follow this pheromone while finding the path [13]. This chemical attracts the ants so that the ants can follow the same path with the highest probability of pheromone on the ground for searching the food and return to the nest. The ants subsequently reach the food sources by following the pheromone trails. The intensity of the pheromone can vary on different factors

 

like the quality of food sources, a distance of the food, etc. Ant's traverse from one node to the other node using the pheromone intensity and also simultaneously update the pheromone trail of that particular path, such that further, this path becomes more feasible if more ants follow this path [14]. Path with the maximum pheromone intensity is the shortest path between the best food source and the starting point. The movements of these ants independently update a solution set. For fast convergence, we employ two diverse ants in search of the slave nodes. At the same time, we leverage max-min rules to trigger ant generation, so as to improve the efficiency of the algorithm.

1. Forward movement- In the forward movement type of movement the ants travel forward to extract the food, or search for the food sources.
2. Backward movement- In the backward movement type of movements the ants go back to the nest for storing their food after collecting food from the food sources.

## 5.1 Working of ACO Algorithm

In our approach, the main procedure of load balancing with ACO consists of two steps before load balancing execution as below.

(1) Ant generation: After periodically checking the cloud platform, if there are overloaded or underloaded nodes, only then ants are generated.

(2) To find target node: According to searching rules, the ant is looking for the target nodes which meet the conditions of load balancing in its surrounding area. Candidate node is the other name for target node for load balancing.

## 5.2 Max-Min Rules

We define two distinct rules, named max-min rules, to trigger the forward ant generation, with the purpose of reducing the time for searching candidate nodes as below.

Rule 1: Maximum value trigger rule. A forward ant is generated from a slave node when the load in this node is greater than a certain threshold. It indicates that the node has been running close to or beyond its maximum load, which needs to dispense the tasks to idle nodes so as to achieve optimal resource utilization.

Rule 2: Minimum value trigger rule. A forward ant is generated from a slave node when the load in this node is smaller than a certain threshold. It denotes that the node is running in the light load state, which can accept a range of new tasks, to share its resource to the overload nodes.

## 5.3 Process of Load Balancing

Based on the above strategies, the initial steps of load balancing are described as follows.

(1) Compute moving probability for all of its neighbors and select the biggest one as its next destination;

(2) Now judgment is made that a new node is candidate node or not after moving to it. If yes, generate a backward ant and initialize this backward ant. For forward ant, go back to step 1;

(3) The backward ant goes back to the starting point of its forward ant, along with the path of its forward ant with the opposite direction. Update the information pheromone of each node the backward ant passes by, and remove the backward ant when reaching the starting point;

(4) Calculate the sum resources of the candidate nodes, and stop the process if they can meet the demand of load balancing and,

(5) Perform the load balancing operation. These steps are the same for max-min rules except the way to calculate the moving probability.

## 5.4 Dynamic Load Balancing Modelling with ACO

To explore both load allocation efficiency and network performance, two critical issues must be addressed. First, to meet the required QoS, there should be proper pheromone initialization in the cloud computing environment. Second, the pheromone update should fulfill the changing demand of the workload variation, with the intention of accelerating the convergence.

## 5.5 Pheromone Initialization

In cloud computing, the physical resources allocated to each virtual node are not the same and usually changing dynamically. Due to of this characteristic, we use the physical resources of virtual machines to measure a node's initial pheromone, as described in [15,16]. In pheromone initialization step, the physical resources involved are CPU, external storage, I/O interface, internal storage, and bandwidth. The CPU capability can be calculated by:

$$p_{CPU} = n \times p$$

The capability definitions of physical resources of virtual machines are defined as blow.

For CPU:

$$\tau_{CPU}(0) = \frac{p_{CPU}}{p_{\max}} \times 100\%$$

For internal storage:

 

$$\tau_{mi}(0) = \frac{m_i}{m_{i\max}} \times 100\%$$

For external storage

$$\tau_{me}(0) = \frac{m_e}{m_{e\max}} \times 100\%$$

For I/O interface:

$$\tau_{i/o}(0) = \frac{p_{i/o}}{p_{i/o\max}} \times 100\%$$

For bandwidth:

$$\tau_b(0) = \frac{p_b}{p_{b\max}} \times 100\%$$

Therefore, the pheromone initialization for one slave node is defined as:

$$\tau_i = \psi_1\tau_{CPU}(0) + \psi_2\tau_{mi}(0) + \psi_3\tau_{me}(0) + \psi_4\tau_{i/o}(0) + \psi_5\tau_b(0)$$

$$\sum_{n=1}^{5}\psi_n = 1$$

Where, $\Psi_n$ is a weight coefficient, which is used to adjust the influence of the physical resources in cloud computing.

## 5.6 Pheromone Update

There are two types of pheromone update that are available for ants movement. Ant can update this table after performing each task by ant. The type of movements of the ant is identified by the type of pheromone being updated by the ant, and it also tells about the kind of node the ant is searching for [15]. Two types of pheromone are given below:

1. **Foraging Pheromone**: The ants constantly move in the forward direction in the network coming across overloaded node or under the loaded node. Equation to update value in foraging pheromone is
   FP(t+1) = (1 - βeva) FP(t) + ΔFP
   Where,
   βeva = Pheromone evaporation rate
   FP = Foraging pheromone of the edge before the move
   FP( t+1 ) = Foraging pheromone of the edge after the move
   ΔFP = Change in FP

2. **Trailing Pheromone:** If an ant encounters an overloaded node in its movement, then it will move back to the underloaded node which was previously encountered. It will check if the node is still underloaded or not and if it finds it still under loaded, then it will redistribute the work to the under the loaded node. The vice-versa is also feasible and possible. Equation to update value in trailing pheromone is,
   TP(t+1) = (1 - βeva) TP(t) + ΔTP
   Where,
   βeva = Pheromone evaporation rate
   TP = Trailing pheromone of the edge before the move
   TP( t+1 ) = Trailing pheromone of the edge after the move
   ΔTP = Change in TP

Algorithm 1 The step by step procedure of the proposed load balancing

1. Beginning of proposed algorithm
2. Initialize pheromone for slave nodes;
3. Get-job-from-user ($job_n$) for master node;
4. Job-divides-into-tasks ($job_n$) by master node;
5. for(i = 0 to $n_t$){//$n_t$ is the distribution number of the tasks
6. Distribute-tasks-to-slaves($task_i$);
7. If-there-are-overload/underload-nodes(){
8. Generate-forward-ant();
9. Compute-moving-probability();
10. Move to next node;
11. if(node-is-candidate)
12. Generate-backward-ant();
13. Start-timer-for-backward-ant($timer_{na}$);
14. Update-pheromone-by-forward-ant();
15. if($timer_{na}$ > 0)
16. Update-pheromone-by-backward-ant();
17. if(task-in-slave-successful)
18. Increase- pheromone;
19. if(task-in-slave-failed)
20. Decrease- pheromone;
21. }
22. if(satisfy-load-balancing){
23. Do-load-balancing();
24. continue;
25. }
26. else if(need-new-tasks)
27. Go to 3;
28. }
29. End of algorithm

# 6. Results

CloudSim 3.0 [16] is chosen as the simulation toolkit for cloud computing, which is simply used to examine and

 

validate the achievability and stability of the proposed load balancing approach. Windows XP OS with 2.53 Ghz CPU, JDK 7.0 and Ant (1.8.4, Apache Software Foundation, Forest Hill, MD, US, 2011), 2 GB of memory are employed to run the simulations.

We assemble 10,000 slave nodes in the cloud platform, and each node is fixed with different processing capabilities randomly. After randomly setting slave nodes, we calculate pheromone initialization of every node. Each iteration represents a unit time in our experiment. In a unit time, a new job with 10,000 tasks is distributed to 10,000 slave nodes at random. Each task can be finished within [1,3] unit time. All ants can only move one edge from one slave node to the other, and the pheromone update is carried out once in a unit time.

During the process, 50 slave nodes that the task number is bigger than nine are selected as overload nodes and 50 slave nodes that the task number is smaller than two are selected as under load nodes at random. If the number of slave nodes is not enough to meet such conditions, the actual number of overload nodes and underload nodes are selected. The forward ants are generated from both the overload nodes and under load nodes to accelerate the searching process. The load balancing is performed when reaching the maximum searching step. The maximum search step stands for the maximum distance the ant can reach within one unit time. After load balancing operations, the cloud platform goes into the next unit time, and a new task with 10,000 tasks comes.

Table 1 gives the experimental results for the number of iterations (Num) and the convergence time (CT) with diverse parameters when reaching load balancing state. For each group of parameters, we carry out the simulation 20 times, and the results of Num and CT are the average values.

| NO | μ | N | Θ | Ω | σ | α | β | Γ | κ | Num | CT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.1 | 0.1 | ±0.1 | 0.3 | 0.1 | 1 | 1 | 1 | 1 | 334 | 15.432 |
| 2 | 0.1 | 0.1 | ±0.1 | 0.3 | 0.1 | 1 | 1 | 2 | 2 | 319 | 16.212 |
| 3 | 0.1 | 0.1 | ±0.1 | 0.3 | 0.1 | 2 | 2 | 1 | 1 | 394 | 20.134 |
| 4 | 0.1 | 0.1 | ±0.1 | 0.3 | 0.1 | 2 | 2 | 2 | 2 | 405 | 19.293 |
| 5 | 0.2 | 0.2 | ±0.2 | 0.4 | 0.2 | 1 | 1 | 1 | 1 | 315 | 14.289 |
| 6 | 0. | 0. | ±0. | 0. | 0. | 1 | 1 | 2 | 2 | 287 | 12.7 |
| | 2 | 2 | 2 | 4 | 2 | | | | | | 85 |
| 7 | 0.2 | 0.2 | ±0.2 | 0.4 | 0.2 | 2 | 2 | 1 | 1 | 376 | 16.112 |
| 8 | 0.2 | 0.2 | ±0.2 | 0.4 | 0.2 | 2 | 2 | 2 | 2 | 389 | 17.834 |
| 9 | 0.3 | 0.3 | ±0.3 | 0.2 | 0.3 | 1 | 1 | 1 | 1 | 329 | 14.115 |
| 10 | 0.3 | 0.3 | ±0.3 | 0.2 | 0.3 | 1 | 1 | 2 | 2 | 301 | 13.454 |
| 11 | 0.3 | 0.3 | ±0.3 | 0.2 | 0.3 | 2 | 2 | 1 | 1 | 392 | 16.298 |
| 12 | 0.3 | 0.3 | ±0.3 | 0.2 | 0.3 | 2 | 2 | 2 | 2 | 421 | 21.103 |

It can be seen from Table 1 that the result in row 6 has the minimum number of iterations and convergence time to achieve load balancing, which means the presented approach can reach an optimal state by these parameter settings. Thus, this group of parameter settings is used for the following simulations. As the number of ants has an influence on the results, we carry out the study with varied number of ants with 50 (Ant-50), 100 (Ant-100), and 200 (Ant-200) ants. The maximum searching step for each ant is set to 10. We adopt the degree of load balancing to describe the results, which stands for the balance level of the virtual machine in the cloud platform. The degree of load balancing is defined as:

$$LB = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Load_i - Load_{avg})^2}$$

Where i Load is the load of the virtual machine, that is, the load of the slave node and Avg Load indicates the average load of all virtual machines. The load gets more unbalanced with an increase in the degree of load balancing. We limit the number of iterations to below 600 and the simulation results of execution time and degree of load balancing by different numbers of ants are shown in Figures 1 and 2.

 

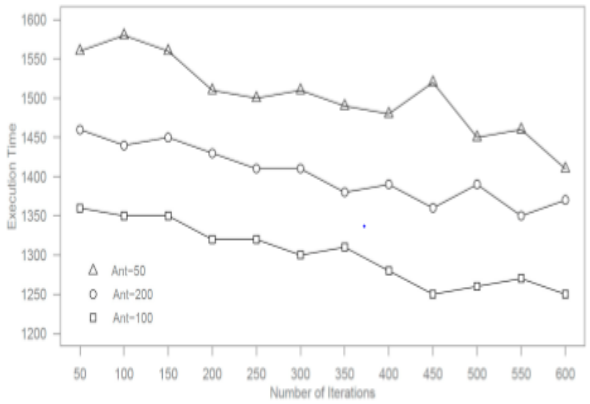


Figure 1: Execution time by various number of ants

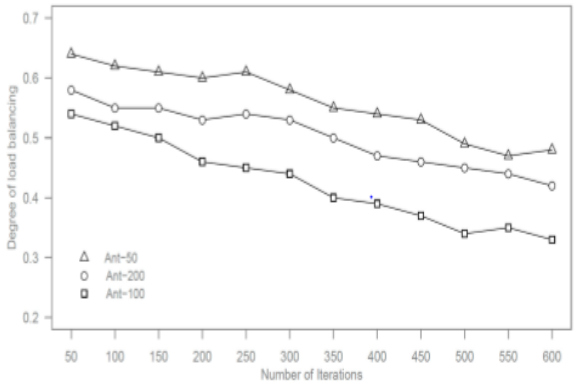


Figure 2: Degree of load balancing by different numbers of ants.

It can be observed from the results that the minimum values are attained for the number of iterations and the degree of load balancing when the number of ants is set to 100.

## 7. Conclusion And Future Work

In this paper, we discussed cloud computing and load balancing. Apart from this, we also discussed the various goals, issues, components, classification, different techniques and metrics for load balancing. Load balancing is one of the major concern in cloud computing, and the main purpose of it is to satisfy the requirements of users by distributing the load evenly among all servers in the cloud to maximize the utilization of resources, to increase throughput, to provide good response time and to reduce energy consumption. To optimize resource allocation and ensure the quality of service, this paper proposed a novel approach for dynamic load balancing based on the improved ant colony optimization.

Two dynamic load balancing strategies were applied with the max-min rules and forward-backward ant mechanism. According to the characteristics of cloud computing, we redefined and improved the ant colony optimization by pheromone update and pheromone initialization. Using such improvements, the speed for searching candidate nodes in load balancing operations can be greatly accelerated. Simulations were illustrated by the improved approach in a cloud computing platform. The results showed that the proposed approach is feasible and effective on load balancing in cloud computing and also has better performance than a random algorithm and LBVS algorithm. In future work, we will study the triggering procedure for ant generation and the approach for pheromone update so as to considerably reduce the searching time for candidate nodes. Furthermore, we will investigate how to introduce other intelligent algorithms into our approach, with the purpose of improving system performance and efficiency.

## 8. References

1 D. Saranya et.al, "Load Balancing Algorithms in Cloud Computing: A Review," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 5, Issue 7, July 2015.

2 S. Sethi et.al, "Efficient Load Balancing in Cloud Computing using Fuzzy Logic," IOSR Journal of Engineering (IOSRJEN) ISSN: 2250-3021 vol. 2, pp. 65-71, July 2012.

3 T. Desai et.al, "A Survey of Various Load Balancing Techniques and Challenges in Cloud Computing," International Journal of Scientific & Technology Research, vol. 2, Issue 11, November 2013.

4 S. Rajoriya et.al, "Load Balancing Techniques in Cloud Computing: An Overview," International Journal of Science and Research (IJSR), vol. 3, Issue 7, July 2014

5 Sharma S. et.al, "Performance Analysis of Load Balancing Algorithms," World Academy of Science, Engineering and Technology, 38, 2008.

6 Gross D. et.al, "Noncooperative load balancing in distributed systems", Elsevier, Journal of Parallel and Distributed Computing, No. 65, pp. 1022-1034, 2005.

7 Nikravan M. et.al, "A Genetic Algorithm for Process Scheduling in Distributed Operating Systems Considering Load Balancing", Proceedings 21st European Conference on Modelling and Simulation (ECMS), 2007.

8 M. Amar et.al, "SLA Driven Load Balancing for Web Applications in Cloud Computing Environment",

 

Information and Knowledge Management, 1(1), pp. 5-13, 2011.

9 Ekta Gupta et.al, "A Technique Based on Ant Colony Optimization for Load Balancing in Cloud Data Center", 13th International Conference on Information Technology, 2014 IEEE.

10 M. Katyal et.al, "A Comparative Study of Load Balancing Algorithms in Cloud Computing Environment", International Journal of Distributed and Cloud Computing Volume 1 Issue 2 December 2013

11 S. Khan et.al, "Effective Scheduling Algorithm for Load Balancing using Ant Colony Optimization in Cloud Computing", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 2, February 2014..

12 Kun Li et.al, "Cloud Task scheduling based on Load Balancing Ant Colony Optimization", 2011 Sixth Annual ChinaGrid Conference, 2011 IEEE.

13 D. Kashyap et.al, "A Survey Of Various Load BalancingAlgorithms In Cloud Computing", INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 3, ISSUE 11, NOVEMBER 2014.

14 Book: Ant colony optimization by Macro Dorigo and Thomas Stutzle.

15 R. Rastogi et.al, "Load Balancing of Nodes in Cloud Using Ant Colony Optimization." Proceedings of the 14th International Conference on Computer Modelling and Simulation (UKSim), March 2012, IEEE, pp: 3-8.

16 Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; de Rose, C.A.F.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software 2011, 41, 23–50.

17. T.Deepa, & S Sharon Amulya Joshi. (2016). A Survey on Load Balancing Algorithms in Cloud.
*International Journal of Computer Engineering In Research Trends, 3*(7), 371-374. Retrieved from http://ijcert.org/ems/ijcert_papers/3703.pdf

 