An effective load balancing procedure for cluster of virtual machines

1. ***Prof. Prabhakara B. K.,*** *Associate Professor, Department of Computer Science and Engineering, Vivekananda College of Engineering & Technology, Puttur, 574203, Karnataka, India, Email: prabhakarabk@yahoo.com Mobile: 9844526585*
2. ***Dr. Chandrakant Naikodi,*** *Professor, Department of Studies and Research in Computer Science (PG), Davangere University, Davangere, 577007, Karnataka, India. Email: chandrakant.naikodi@yahoo.in Mobile: 9901452550*
3. ***Dr. Suresh L.,*** *Principal and Professor, Department of Computer Science and Engineering, Cambridge Institute of Technology, Bengaluru, 560036, India. Email: suriakls@gmail.com Mobile: 9686001199*

Abstract: The cloud environment is ease for accessing remote servers; which can be used by the users for processing, storing and managing of data instead of the local host or personal computers. These servers are enabled by internet. Due to several advantages of cloud computation, many users prefer their computations in this environment. This huge demand of cloud environment computation is possible to achieve by adding more resources to the network. In such situation, some resources in this environment areutilized up to maximum and some other are very less. Hence a load balancing software is installed for effective utilization of resources in this environment. This increased demandof computation isfrequentlybalanced by providing the virtual nodes and dynamic load balancing algorithms. The proposed dynamic load balancing algorithm uses some features of two biologically inspired algorithms namely Honey Bee foraging algorithm which is used for allocating tasks to the servers. The Ant colony optimizations algorithm is used for routing the shortest path. The solution makes better utilization of VMs, by prioritizing the tasks they handle. It balances the overloaded and under loaded situations of VMs by grouping them into separate sets. The experimental results show the significant improvement in load balancing and effective utilization of VMs.

Keywords:Virtual Machines, foraging, Quality of Service, Ant Colony Optimization, Round Robin, under loaded, overloaded.

I. INTRODUCTION

The cloud computing is ease of network for remote servers, which can be used by the users for processing, storing and managing of data instead of the local host. It is an on-demand service for which the users can pay per use, which makes it very convenient. The “pay per use” is a concept where the user needs to pay only for his computation. This computation is possible by internet. When the number of user increases the requirements for the additional resources also increases. Then there is a need for load balancing. Load balancing is used for effective utilization of resources in cloud computation.When there are many users, resources which could be located far away from each other. Some of the servers may be overloaded where it could have load or tasks more than the threshold and some other servers are may be under loaded where the load or tasks is less than the threshold. In such situation load balancer helps in improving the response time.

Normally load balancers results are obtained by using some simulation softwares. These softwares provides virtual cloud environment for simulation. By using simulators the implementation burden of cloud environment is also reduces. The simulation helps to analyze the possible parameters for the implementation of different services of cloud environments. In general the load balancers are classified into two types. They are Hardware based load balancers and Software based load balancers.The load balancers make network more efficient and service delivery more reliable, enabling following to the users

- Ensure high availability and flexibility for applications.
- Better system capability.

- Allows more computation for many users and new applications even in peak of network traffic.
- It enhances performance of applications.

Because of above usefulness the load balancers are frequently deployed in standard computation practice for business critical web applications.

The load balancing algorithms: The load balancing methods deal with the control of traffic over the web or the server. Whereas task scheduling algorithms deal with assignment of task in the operating system (OS) so that the memory is used efficiently, and no deadlock is encountered. Both of them almost use same techniques like Round Robin (RR), First Come First Serve (FCFS) and Shortest Job First (SJF) etc. algorithms. But load balancing algorithms are related to networking are classified into two they areStatic load balancing algorithmsandDynamic load balancing algorithms

Static load balancing algorithms will not consider previous state of nodes in distributing the work load. These algorithms are suitable for small variation of work load. That's why these are not preferred in the cloud computation environment. These algorithm protocols are based on the mean behaviour of system. The work load transfer decisions are independent from real system state. These algorithms uses statistical information of computation before it utilize a system. This method of computation preferred for less execution time by reducing communication delays. The main advantage of these algorithms is selection of the host machine for computation is done only after creating a process. Once the process is allocated to a machine; during execution it is not possible to change system load.

Dynamic load balancing algorithmin thisalgorithm the work load distribution takes place during runtime. The allocation of work load is done exactly when a processor becomesunderloaded. The buffered work load will be in queue on the main host machine and dynamically allocated based on the requests from remote host machines. The dynamic load balancer monitors the load on all processors. When the load imbalance reaches to some threshold the work load is redistributed. This monitoring utilizes central processing cycles. So at the time of balancing monitoring task must be invoked. This redistribution of work load is additional burden that occurs during the execution time.

II. LITERATURE REVIEW

This survey includes honey bee foraging (Pham, Ghanbarzadeh et al. in 2005) and ant colony optimization (Marco Dorigo in 1992) algorithms and other major works related to these two algorithms. The Honey Bee foraging cloud load balancing algorithm [1] proposed by Harshit Gupta and KalicharanSahu. A collection of servers are arranged as Virtual Machines (VMs). Each server maintains the requests like service queue. By processing request each VM will calculates profit or reward. This measure resembles the quality of bee waggle dance. The quality of time spent by the VM for processing is considered as reward. The results show that process is better and powerful when compared with current algorithms. This algorithm has better execution time and less waiting time while scheduling tasks.

This algorithm [2] manages the workload in the server by verifying the current status of all available VMs for assigning the incoming requests. This algorithm considers the effective utilization of resources and VMs. The simulation result shows the distribution of load optimally and overcomes from underutilization and overutilization of VMs. By comparing this algorithm with existing algorithms result proves the VM load balancing problems are resolves imbalanced utilization of VMs.

This load balancing algorithm [3] is used for the computation balancing in date centre named Central Load Balancer (CLB). It avoids the overloading of VMs and under loading of VMs. The load distribution happens here by priority of request or state of request. Hence it is proved efficient algorithm in analysis.

The load balancing algorithm [4] introduced by Randles, Lamb and Taleb-Bendiab is decentralized honey bee algorithm. In this method the global load balancing is done by local server actions. The overall performance is increased through system diversity but throughput is hampered due to increase in system size. This algorithm is suitable when heterogeneous service types are required. In this process each severs depicts a bee role. The values are used to imitate the colony of honey bee where minimum number of bees are required for foragers (normally more number of bees are harvesters) to utilize existing sources. The server which is profitably fulfilling a request will place on the advert board. A server may arbitrarily

select a virtual servers queue or else it may check for the advert board (watching a waggle dance). In brief the idle server (waiting bees) follows either one among hence forth mentioned behaviour.

(1) A server which reads the advert board will follow the selected advert and then serves the request. Thus imitates the harvest bee behaviour.
(2) A server which is not reading the advert board will go back to forage bee behaviour. Thus serve it serves a random virtual server's queue request.

The server will execute and complete the request and calculates the profit of the severed virtual server.

The algorithm proposed by Babu, KR Ramsh, Amaya Anna Joy, and Philip Samuel [5] is based on foraging behaviour of honey bees. This technique resolves the overload VMs by assigning suitable under loaded VMs. The tasks are prioritized in the queues of VMs while waiting. Hence the time required for prioritizing eliminated. In the process of prioritizing the task with lowest priority is selected for the migration. This reduces the longer waiting time for processing. The result proves that the reduced time for initializing the environment, degree of imbalance and due to number of task migration the Quality of Service (QoS) enhanced. Thus this algorithm is considered priority as QoS parameter.

The Ant Colony Optimization algorithm (ACO) is invented by Macro Dorigo and his colleagues. This algorithm is based on the life of ants. It depicts the behaviour of ant in its colony. Ants live in colonies and they work for the survival of the colony. They optimize the path on which they travel. Ants keep track of every node in the path they visit and node data for future decision making. They have fixed amount of pheromone in them which is a chemical substance used by them to find out the shortest path. When the path is short or optimal, the pheromone droppings are more. This intense of pheromone droppings indicate that the particular path is optimal. But, when the path is long, the intensity is less indicating that the path is not short.The solution set is updated by movements of these ants. The ant's movements may be one of below two types.

- The ants will move away from their nest for searching the food resources. This movement is known as forward movement.
- The ants will move towards their nest after extracting food. This movement is known as backward movement.

Artificial ants are used to behave like the real ants. In the beginning, the lifetime of ants and the pheromone in ants are initialized. Then, a threshold load is defined. When the ants start moving, firstly the lifetime of an ant is checked. If the life of an ant is up, then the process is stopped. The visiting node load is compared with the threshold load; if more than threshold the algorithm transverses to the node having load less than the threshold and checks if the visiting node is overloaded. If yes, the pheromone table gets updated and the resources get reassigned. If not, the algorithm traverses to the node having load less than the threshold again and follows the same process.

Suppose the load of the visiting node is lower than the threshold load then, the algorithm traverses to the node having load above the threshold and checks if the visiting node is overloaded. If yes, the pheromone table gets updated and the resources get reassigned. If not, the algorithm traverses to the node having load less than the threshold again and follows the same process. This procedure is repeated until all the tasks in the meta-set are completed. The ants commit suicide after their journey. The result set is updated once.

Pheromones are updated constantly as the ants move throughout the network through the nodes. There are two types of pheromones namely Foraging pheromone and Trailing pheromone. The ants will move in the forward direction in the network to encounter under loaded node or overloaded nodes. In this move the foraging pheromones are used. In forward direction move if ant encounters the overloaded node, it moves back in the same direction to ensure the last visited under loaded node. Then ant will verify the current status of that under loaded node again. If it is still in same under loaded status, the work load is redistributed to it from the overloaded node. The pheromone used in the traversal is to classify the ants based upon their search. The default foraging pheromone is used by the ants to traverse from head node to the other nodes. Once they encounter overloaded node they follow the trailing pheromone to find the under loaded node. When all this happens, they update their data structure accordingly. If the ants find the under loaded node they start searching for the overloaded node and transfer the amount of load to the under loaded node. In the next stage of searching ants select a random neighbor node, if they encounter

under loaded node they follow the foraging pheromone for tracing an overloaded one. By repeating the same procedure the network performance is refined.

The paper by Hsiao, Ying-Tung, Cheng-Long Chuang and Cheng-ChihChien [6] gives the details of implementation of a routing algorithm based on ACO. This method distributes all computation on several nodes. By the help of delay time of each transmitted data packages, user can calculate cost of package transmission and is possible to refine the pheromone table of each node in network. This instant updating of pheromone table enhances the performance and also avoids the hotspots. The result shows that this algorithm is reliable and better in performance than RR algorithm.

The paper published by Nishant, Kumar, Pratik Sharma, Chhavi Gupta, KuwarPratap Singh, and Ravi Rastogi [7] is an enhanced approach of ACO. This approach initially detects all overloaded and under loaded nodes before assigning the computation. This enhanced approach of ant colony identifies the nodes by ants and the tracing its path consequently in search of different type of nodes.

III. IMPLEMENTATION

The forward and trialing pheromone concept of ACO is used according to the convenience. From the pseudocode one can observe the creation of ant with their functionalities according to the pheromone trials and node encountering visibility. Here each ant updates its own result set and later this result set is utilized to build entire solution. In the previous approach ants are capable of updating rather than their own result set. Here every node is updated instantaneously when it is necessary. The solution set gradually built in this approach. One more advantage here is the task of each ant is specialized rather than being general and task depends on the type of first node encountered based on overloaded or under loaded.

The paper published by S.Jyothsna [8] is inspired by the behaviour of honey bee for load balancing. In this balancing technique amount of waiting time of tasks in the queue is minimal. The average execution time and waiting time in the queue were improved. This approach is suitable for heterogeneous type of systems and for non-pre-emptive independent tasks.

3.1 Proposed Algorithm

1. Start of algorithm
2. Determine the work load of VMs
3. Create two groups of VMs; overloaded VMs and under loaded VMs based on threshold load set in the beginning of the process.
4. Estimate the possible load supply for the under loaded VMs.
5. Estimate the demand for the overloaded VMs.
6. Align overloaded VMs and under loaded VMs sets in required order.
7. Prioritize the task of overloaded VMs set based on priority.
8. For each task of overloaded VM find as best suited under loaded VM.
9. Update both VMs set and repeat form step 2.
10. End of algorithm

3.2 Pseudo code

```
functionnewAlgorithm()
i=first request on the request list
WHILE true
IF no server is free
Ti+=1
        ELSE IF any servers are free
                IF any servers are free at the current request's location
                        IF server Sk can handle the entire load
                                Assign i^th request to Sk
```

cost+=cost of the server Sk
Ti+=2
Else split the task based on low specification server
Then assign the chunks of requests one by one until the server Sk is reaches its threshold value or near to it.
Ti+=2
ELSE IF check other location AND server which are used
is not used up-to threshold
Ti+=2
IF server Sk can handle the entire load
Assign i^{th} request to Sk
ELSE split the task based on low specification server
Then assign the chunks of requests one by one until the server is reaches its threshold value or near to it
ELSE
Ti+=2
IF server Sk can handle the entire load
Assign i^{th} request to Sk
ELSE split the task based on low specification server
Then assign the chunks of requests one by one until the server is reaches its threshold value or near to it
IF no request on the request list
BREAK
ELSE
i=next request on the request list
END WHILE LOOP

The proposed algorithm uses the concepts of finding the shortest distance, then assigning as well as distributing the load on the servers that are used instead of using the servers that are not used at all. It also includes the concepts of dynamically distributing the task based on the low specification server.This implementation reads inputs from user. It contains server's characteristics such as primary memory capacity in megabytes and number of requests it can handle. It also receives requests from the various continents, which include number of requests and capacity of RAM required to handle those requests. Algorithm processes these inputs; implementation takes requests from one of the continents, then it attempt to find any of the servers is free at requested location or not. If server is free; the code verifies the capacity of server to handle the requested load entirely or not. If it can, then assign all the requests from the continent to that server. If it is not able to handle, then try to split the load based on the low specification server. Then try to feed the divided load to the server until it reaches the threshold completely or partially i.e. set by the administrator.

If the load reaches the threshold and if the current continent's requests are not completely assigned then, if servers are not free at request's source or if servers are completely utilized then, implementation find a server that is previously used in other location to reduce the overall cost by using the used servers. Then it assigns the load to the server until it reaches the threshold or load becomes empty. Finally, if any used servers are not found then it try to assign it to the other servers which are not used and located at different location. Then, assign the task if it is capable of handle that assignment else split the task.

3.3The Drawbacks which are overcome in proposed algorithm

- In this algorithm first priority goes to time and then to the cost. It will try to reduce the processing time as much as possible. In worst case if there is tie, it tries to reduce cost.
- Both honey bee and ACO algorithms are widely implemented because of their stability. Therefore this proposed algorithm also stable.
- This can perform load balancing even when the load is greater than the server capacity.

- This algorithm performs nearest node execution in the sense it tries to handle request by executing it in nearby server.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

The table 1 shows the inputs of default locations: North America, South America, Asia, Europe and Africa of the implementation. The default server capacities (based on RAM size in megabyte) are1024MB, 2048MB, 4096MB.

Location	Server CapacityRAM(MB)	Requests
North America	1024	2000
South America	1024	2000
Asia	1024	2000

Table 1. Adding & Deleting Servers

Location	Processor Size (MB)	Requested
North America	500	600
South America	700	800
Asia	900	900
Europe	800	890

Table 2. Adding & Deleting Processes

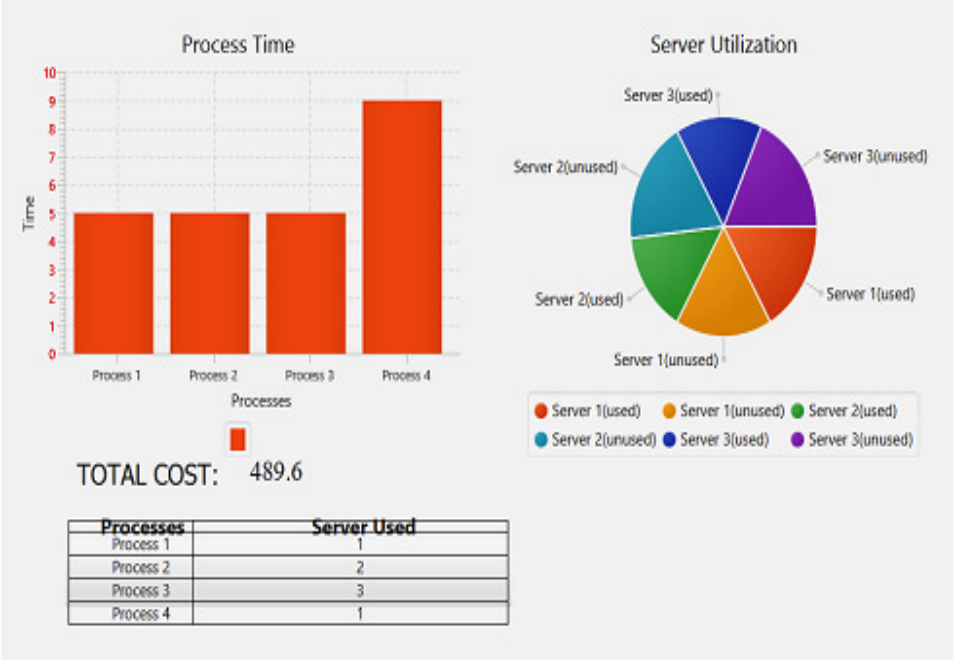

Figure1. Round Robin algorithm

The number of server request is specified by the user. Similarly processor size in megabytes and number of processor requests are entered by the user by selecting a server location as shown in table 2.

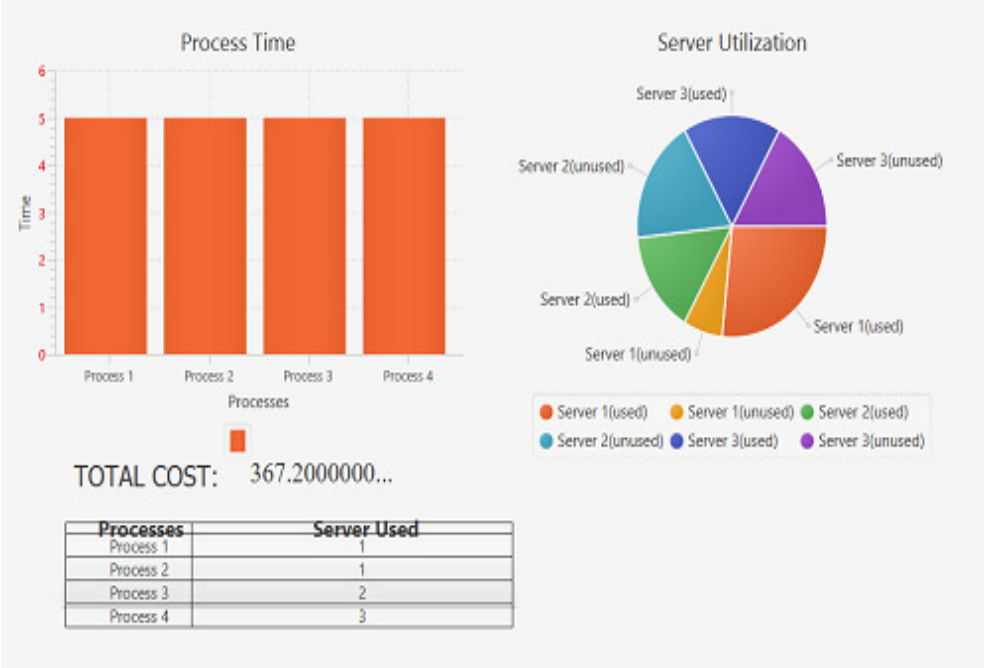

Figure2. Results for Honey bee foraging algorithm

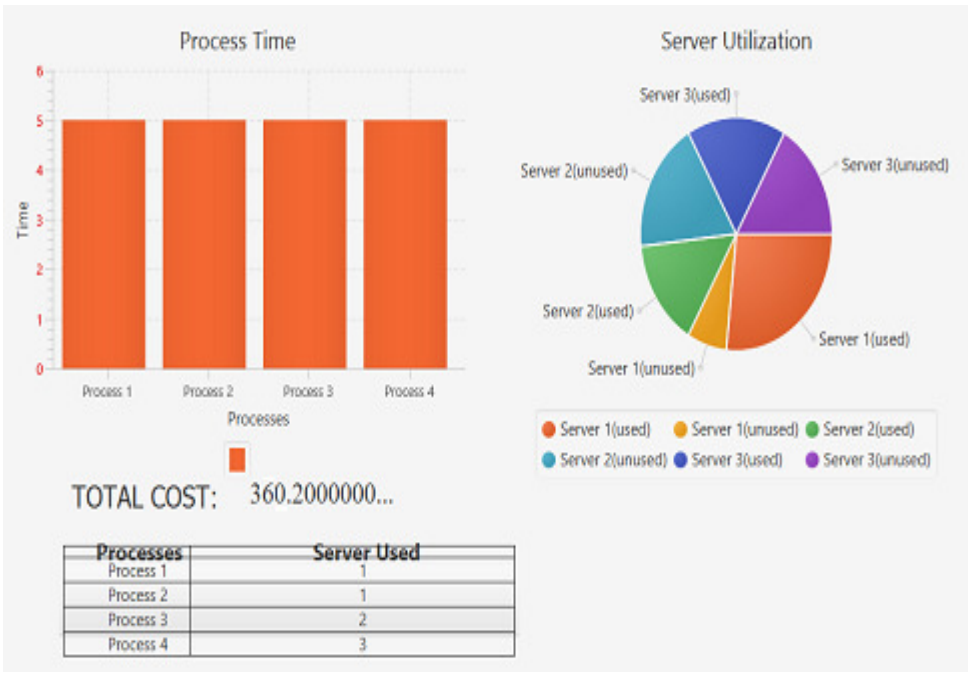

Figure3. Results for proposed algorithm

The figure 1, figure 2 and figure 3 shows process time in bar graph and server utilization in pie chart.The time taken by each process to completeis shownin the table. The figures give the total time taken by all processes andalso details of servers in which a process is executed.

Algorithm Used	Max server Utilization	Max Process Time	Total Cost
Round Robin	50.0	9.0	489.60
Honey Bee	80.0	5.0	367.20
Propose Algorithm	75.0	5.0	360.20

Table 3. Comparisons of different algorithm

The table 3 shows the comparison of RR, honey bee algorithms and proposed algorithm. From the table which it is clear that the total cost is improved compared with RR and honey bee algorithm. The server utilization is better in the case of Honey Bee foraging algorithm.

V. CONCLUSION

The proposed algorithm gives an effort of balancing the load among all the available VMs and the overloading of VMs is avoided. Allocation of process is done considering the location of each process and servers that already under loaded. Theidea of prioritizing the processeshelps in fair allocation of processes and efficient server utilization with less delay. The result of the implementation proves this algorithm is better than RR algorithm and Honey Bee algorithm.

It is still possible to improve this algorithm by analyzing some other factors like giving priority to the process after splitting the processes into more than one when it is bigger than the size of server. The proposed algorithm can be tweaked giving priority to time.

References

[1] Gupta, Harshit, and KalicharanSahu. "Honey Bee Behaviour Based Load Balancing of Tasks in Cloud Computing", International journal of Science and Research 3, no. 6, 2014.

[2] Domanal, Shridhar G., and G. Ram Mohana Reddy. "Optimal load balancing in cloud computing by efficient utilization of virtual machines", In Communication Systems and Networks (COMSNETS), 2014 6th International Conference on, pp. 1-4.IEEE, 2014.

[3] Soni, Gulshan, and Mala Kalra. "A novel approach for load balancing in cloud data center", In Advance Computing Conference (IACC), 2014 IEEE International, pp. 807-812.IEEE, 2014.

[4] Randles, Martin, David Lamb, and A. Taleb-Bendiab. "A comparative study into distributed load balancing algorithms for cloud computing", In Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on, pp. 551-556. IEEE, 2010.

[5] Babu, KR Remesh, Amaya Anna Joy, and Philip Samuel. "Load Balancing of Tasks in Cloud Computing Environment Based on Bee Colony Algorithm", In Advances in Computing and Communications (ICACC), 2015 Fifth International Conference on, pp. 89-93. IEEE, 2015.

[6] Hsiao, Ying-Tung, Cheng-Long Chuang, and Cheng-ChihChien. "Computer network load-balancing and routing by ant colony optimization", In Networks, 2004(ICON 2004).Proceedings. 12th IEEE International Conference on, vol. 1, pp. 313-318. IEEE, 2004.

[7] Nishant, Kumar, Pratik Sharma, Vishal Krishna, Chhavi Gupta, KuwarPratap Singh, and Ravi Rastogi. "Load balancing of nodes in cloud using ant colony optimization", In Computer Modelling and Simulation (UKSim), 2012 UKSim 14th International Conference on, pp. 3-8.IEEE, 2012.

[8] S. Jyothsna, “Distributed Load Balancing in Cloud using Honey Bee Optimization”, International Journal of Emerging Trends & Technology in Computer Science, 2016 Volume 5, Issue 6.