A NOVEL LOAD BALANCING MODEL FOR OVERLOADED CLOUD PARTITION

Mithra P B[1], P Mohamed Shameem[2]

[1]*Mtech Student, Dept of CSE, TKM Institute of Technology, Kerala, India*
[2]*Associate Professor, Dept of CSE, TKM Institute of Technology, Kerala, India*

Abstract

Load balancing is an efficient solution that distributes excess workload evenly to all nodes in cloud environment. The Load balancing model is used for the public cloud having numerous nodes in different geographic locations. The model divides public cloud into several cloud partitions. When partition status becomes overloaded, cloud partitioning is repeated. It reduces the working efficiency and expected response time of the system. To overcome this issue we propose a novel load balancing strategy for overloaded cloud partition. The overloaded load balancing strategy maintains two queues at overloaded condition. Priority queue is used when the cloud partition status is idle and normal and non priority queue is used when partition status is overloaded. At overloaded condition an overloaded partition scheduling algorithm is used for the allocation of jobs to Non priority queue. When a priory job ends, then one of the jobs in Non priority queue moves to priority queue based on arrival time and processing power required

Keywords: *Cloud Partition, Job Splitting Index, Non priority queue, Overloaded partition, Priority queue*

----------***----------

1. INTRODUCTION

Cloud computing is an emerging technology that brings many changes to the IT industry. Cloud computing allow users to take advantage from all these technologies, without deep knowledge about or expertise with them. Load balancing schemes depending on whether the system dynamics are important can be either static or dynamic [1]. It is an efficient solution that distributes excess workload evenly to all nodes in cloud environment [2]. The load balancing model is used for the public cloud having numerous nodes in different geographic locations [3]. The model for such a cloud computing environment leads to high cost when there is an increase in number of nodes. It is also difficult for the existing load balancing strategies to apply when the environment is large and complex. So cloud partitioning is chosen that divides the public cloud into several cloud partitions by the random selection of nodes. The model includes main controller and partition balancers to perform load balancing solution. When the cloud partition status is overloaded, cloud partitioning is repeated. It reduces the working efficiency and expected response time of the system.

In the proposed load balancing strategy each node maintains two queues Priority and Non-priority queue. It is a modified approach of existing load balancing model. Priority queue is used when the cloud partition status is idle or normal and Non priority queue is used when partition status is overloaded. At overloaded condition the jobs in idle and normal partition status are moved to Priority queue and the jobs after overloaded status are moved to non-priority queue. For the better allocation of jobs at overloaded situation we propose an overloaded partition scheduling algorithm. The main features of our algorithm can be listed as follows:

- Minimum response time at overloaded situation
- Provides better fault tolerance
- Simplifies load balancing

The rest of the paper is organized as follows: In section II, we survey related works of load balancing in cloud computing environment. In section III we do the proposed work. In section IV we do the performance analysis on our proposed algorithm. Finally, in section V summarizes our findings and concludes the paper.

2. RELATED WORK

Cloud computing has attracted considerable research attention, but only a small portion of the work has been done so far. There has also been much research in towards different styles of load balancing. Here, we survey those that proposed certain techniques and algorithms for load balancing in cloud environment.

Martin Randles, David Lamb (2010)[4] investigates three viable methods for load balancing. Firstly, nature-inspired algorithms for achieving global load balancing. Secondly, load balancing of all system nodes using random sampling of the system domain. Thirdly, optimizes job assignment by connecting similar services by local re-wiring.

Kumar Nishant (2012)[5] proposed an algorithm for effective distribution of workloads among the nodes of a cloud environment by the use of Ant Colony Optimization (ACO). This is a modified approach of ant colony optimization. The ACO is used for load balancing. The main advantage of this approach is the detection of overloaded and under loaded nodes. Nidhi Jain Kansal (2012)[6] study the existing load balancing techniques in cloud computing and further compares them based on various parameters like performance, scalability, associated overhead etc that are considered in different techniques.

Shantanu Dutt (1993)[7] presents a very efficient graph partitioning scheme that uses the basic strategy of the Kernighan-Lin algorithm to swap pairs of nodes to improve an existing partition of a graph G. The algorithm attempts to find a partition of a set of nodes (V) into disjoint subset A, B of equal sizes such that sum of the weights of the edges between nodes in A and B is minimized. For that take the initial partition and iteratively improve it. Vertex pairs with largest decrease or smallest increase in cut size are exchanged. These vertices are then locked. This process continues until all vertices are locked.

Tarun Kumar (2012)[8] proposed Load Balanced Max Min algorithm. The proposed algorithm outperforms Max-Min because it focuses on minimizing the completion time of tasks. The proposed algorithm is executed in two-phases. It uses the advantages of Max- Min and covers its disadvantages by reducing makespan and maximizing resource utilization.

Gaochao Xu (2013)[1] proposed a better load balancing model for public cloud based on the cloud partitioning. The model includes Main Controller and Balancers to perform load balancing solution. The Main Controller selects the best cloud partition and Balancers choose right load balancing strategy to distribute the jobs to cloud partition. Here, the idle partition status uses an improved Round Robin algorithm and the normal status uses a Game theory based load balancing strategy. When partition status becomes overloaded, cloud partitioning is repeated. It reduces the working efficiency and expected response time of the system

In reference to [1], we modified the load balancing model by maintaining two queues at overloaded condition and use a scheduling algorithm for the allocation of jobs in the way mentioned below.

3. PROPOSED WORK

The load balancing model is used for the public cloud which has numerous nodes in many different geographic locations. The model for such a cloud computing environment leads to high cost when there is an increase in number of nodes. It is also difficult to apply the load balancing strategy when the environment is very large and complex. So cloud partitioning is chosen. Cloud partitioning divides the public cloud into several cloud partitions by random selection of nodes. When the environment is very large and complex, these divisions simplify load balancing. The Load Balancing model includes Main Controller and Balancers, performs the load balancing solution. The Main Controller selects the best cloud partition and Balancers choose right load balancing strategy to distribute the jobs to cloud partition.

The load balancing model for public cloud using cloud partitioning concept uses a switch mechanism to choose different strategies during different situations. The idle status uses an improved Round Robin algorithm and normal status uses a game theory based load balancing strategy. When the cloud partition state is overloaded, a random node is selected as best node to perform the load balancing, cloud partitioning is repeated. It reduces the working efficiency and expected response time of the system. To overcome this issue we present an approach to develop a novel load balancing strategy for overloaded cloud partition.

3.1 Load Balancing Strategy for Overloaded Partition

It is evident that the working efficiency of cloud computing environment decreases when the cloud partition status is overloaded. So a novel load balancing model is proposed to avoid this problem by incorporating two queues. Figure 3.1 depicts the design of the cloud architecture for this approach.

According to this design each node maintains two queues, Priority queue and Non priority queue. Priority queue is used when the cloud partition status is idle or normal and Non priority queue is used when partition status is overloaded. When user submit job request job allocation is performed either by load balancing approach or by scheduling algorithm. When a new job arrives, if cloud partition status is idle or normal then load balancing approach is used to select a best node for executing the job. The load balancing approach uses priority queue for all arriving jobs. The jobs assigned to the nodes have the same priority and total CPU power is shared by all the jobs in the queue. When the cloud partition status is overloaded then queue is splitted into two. Then the jobs in idle and normal partition status are moved to Priority queue and the jobs after overloaded status are moved to non-priority queue. A separate scheduling algorithm is used for this allocation to Non priority queue.

ARCHITECTURE OF THE SYSTEM

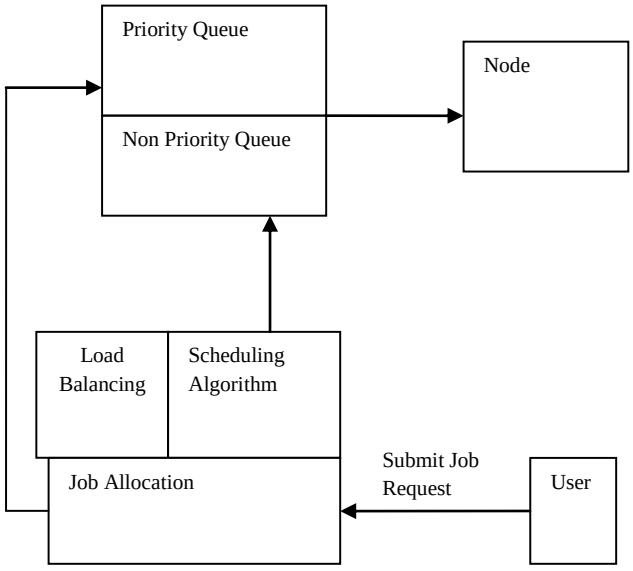

Fig 3.1 System Architecture

The overloaded load balancing model is a modified approach of existing load balancing model. Fig 3.2 shows the overall job assignment strategy for overloaded cloud partition. In this model when a new job arrives the main controller chooses best cloud partition for the arriving job. The cloud partition status is then evaluated. When the partition status is idle and normal the jobs are assigned to the nodes according to the existing load balancing strategy. The Idle cloud partition status uses improved round robin algorithm and Normal partition status uses game theory based load balancing strategy.

When an overloaded condition occurs the existing model selects a random node as best node to perform load balancing and cloud partitioning is repeated. It reduces the working efficiency and expected response time of the system. So overloaded load balancing model is used to avoid this problem by maintaining two queues, Priority and Non priority queue.

At overloaded condition the jobs in idle and normal cloud partition are moved to Priority queue and the jobs after overloaded status are moved to non-priority queue. A separate overloaded partition scheduling algorithm is used for this allocation to non priority queue. When a priory job ends, then one of the jobs in non priority queue moves to priority based on arrival time and processing power required. The modified job assignment strategy for overloaded cloud partition is shown below.

At overloaded condition node provides X% of its CPU power to priority queue and 1-X% to Non priority queue. For example if X=75% and if there are three jobs in priority queue then 75% of CPU power is shared by all three jobs in priority queue. Initially X=100% so there is no Non priority queue. When an overloaded situation occurs, queue is splitted into two. The node gradually increases CPU power in non priority queue and decreases CPU power in priority queue i.e. node provides a threshold of 75% CPU power to Priority queue and 25% to jobs in Non priority queue. This threshold value is calculated using overloaded partition scheduling algorithm. The main benefit of the overloaded load balancing model lies in response time and fault tolerance. It further improves the efficiency by considering all the jobs at overloaded status.

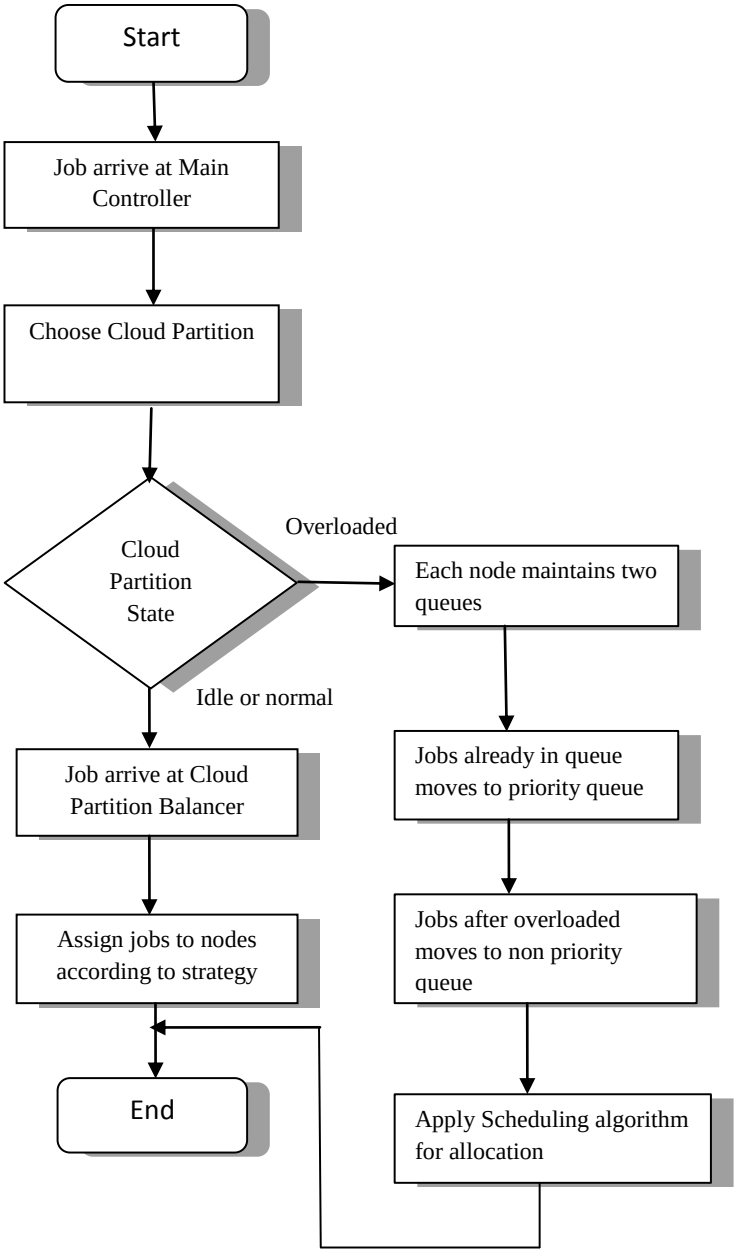

Fig 3.2 Job assignment strategy for overloaded status

3.2 Scheduling Algorithm

Scheduling algorithm is used for the allocation of jobs to non priority queue when overloaded situation occurs. When cloud partition status become overloaded, each node maintains two queues Priority queue and Non priority queue. The priority queue contains jobs when the cloud partition status is idle and normal. Here the total CPU power is shared by all jobs and equal priority is set for all jobs. When overloaded occurs queue is splitted into two. The jobs already in the queue moves to priority queue and jobs after overloaded condition moves to non priority queue. Scheduling algorithm is used for this allocation of jobs to non priority queue. In the scheduling algorithm the jobs are splitted according to a threshold. The threshold value is set as 75% and 25% for Priority queue and Non priority queue when cloud partition status is overloaded. At the initial stage 100% CPU power is shared by all the jobs in priority queue. When overloaded occurs 95% CPU power is given to jobs in priority queue and 5% CPU power is given to jobs in non priority queue. As jobs in non priority queue increases the CPU power given to them increases up to threshold value of 25% and CPU power given to jobs in priority queue decreases up to threshold value of 75%. The threshold value is calculated using Job Splitting Index.

$$\text{Job splitting index} = \text{Min}\left[25\ \left(5 * \frac{\text{Number of jobs in 2nd queue}}{\text{overload status}}\right)\right]$$

The Job splitting index is calculated for all the nodes in each partition. Then calculate the average Job splitting index from the node Job splitting index value.

$$\text{Job splitting index}_{avg} = \sum_{i=1}^{n} \text{Job splitting index(Ni)}/n$$

When overload occurs the threshold value is set as 75% and 25% for jobs in priority and non priority queue respectively. Allocation of job is done by selecting the best partition with minimum average job splitting index value. For each partition Pi select a partition if avg(Pi) < min. The selected partition is then assigned to partition controller. At partition controller the job is assigned to the node with minimum avg value.

Overloaded Partition Scheduling Algorithm

For each node i

For each partition Pi

Calculate Job splitting index of each node

$$\text{Job splitting index} = \text{Min}\left[25\ \left(5 * \frac{\text{Number of jobs in 2nd queue}}{\text{overload status}}\right)\right]$$

Calculate the average Job splitting index for each partition Pi

$$\text{Job splitting index}_{avg} = \sum_{i=1}^{n} \text{Job splitting index(Ni)}/n$$

1. Allocation of job at Main controller

Main controller chooses the best partition with min avg Job splitting index value.

Min=α

For each partition Pi

If avg(Pi) < Min

Selected= Pi

End if

End for

2. Allocation of job at Partition controller

On arrival of job the partition controller allocate job to the node with min Job splitting index value

4. PERFORMANCE ANALYSIS

The resulting load balancing model has been implemented and a graph has been plotted. The graph 4.1 shows the comparative performance of the response time of existing and overloaded load balancing model, with the Y axis showing the effect of improved response time on increased number of jobs in X axis. This graph demonstrates that overloaded load balancing model performs well as number of jobs increases. As the number of jobs increases the proposed overloaded load balancing model provides minimum response time.

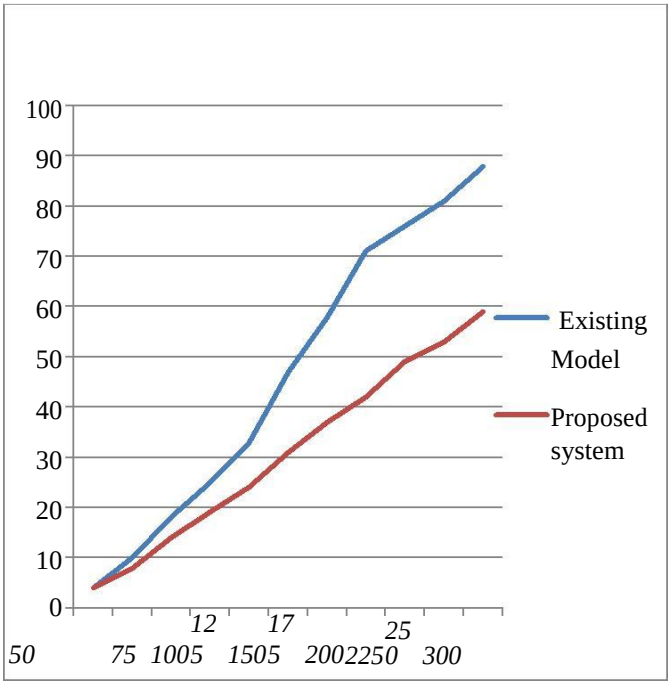

Fig 4.1 Number of Jobs against Response time

The graph 4.2 shows the comparative performance of the fault tolerance of existing and overloaded load balancing model,

with the Y axis showing the effect of improved fault tolerance on increased number of jobs in X axis. The overloaded load balancing model provides better fault toleranace when the number of jobs increases.

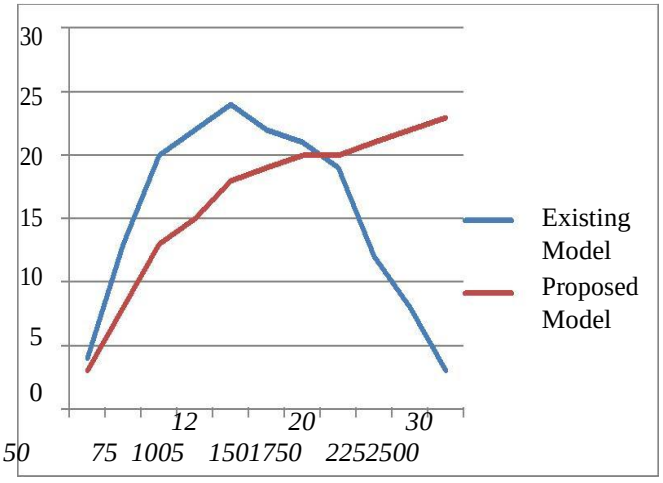

Fig 4.2 Number of Jobs against Fault tolerance

From the graph plotted it is proved that our overloaded load balancing model minimizes the load balancing in cloud environment and there by increases overall performance of the cloud system. The proposed model is fault tolerant when number of jobs increases

5. CONCLUSIONS

This is a modified approach of load balancing model aimed at the public cloud which has numerous nodes with distributed computing resources in many different geographic locations with the main aim of load balancing of nodes. The main benefit of this approach lies in the development of a load balancing strategy for overloaded cloud partition. When overloaded condition occurs the jobs in idle and normal partition status are moved to non-priority queue. An overloaded partition scheduling algorithm is used for this allocation to Non priority queue. When a priory job ends, then one of the jobs in Non priority queue moves to priority based on arrival time and processing power required.

ACKNOWLEDGEMENTS

I would like to express my gratitude to all those who gave me help to complete this project. A special thanks to my guide Prof. P Mohamed Shameem, H.O.D., CSE, TKM Institute of Technology. I am also thankful to staffs of the institution for guiding and providing me superior computing facilities. Last but not least I would like to thank almighty for making this project a reality.

REFERENCES

[1] N. G. Shivaratri, P. Krueger, and M. Singhal, “Load distributing for locally distributed systems”, Computer,vol. 25, no. 12, 1992.

[2] B.P Rima, E.Choi, and I.Lumb, “A Taxonomy and Survey of Cloud Computing Systems”, Proceedings of 5th IEEE International Joint Conference on INC,IMS and IDC,Seoul,Korea, 2009.

[3] B P Gaochao, “Load balancing model based on cloud partitioning for the public cloud”, IEEE transactions on cloud computing, 2013.

[4] M. Randles, D. Lamb, and A. Taleb-Bendiab, ” A comparative study into distributed load balancing algorithms for cloud computing”, IEEE 24th International Conference on Advanced Information Networking and Applications, Perth, Australia, 2010

[5] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, N. Nitin, and R. Rastogi, “ Load balancing of nodes in cloud using ant colony optimization” 14th International Conference on Computer Modelling and Simulation (UKSim), Cambridge shire, United Kingdom, 2012

[6] Nidhi Jain Kansal1, Inderveer Chana, “Cloud Load Balancing Techniques: A Step Towards Green Computing”, IJCSI International Journal of Computer Science, 2012.

[7] Shantanu Dutt, “New Faster Kernighan-Lin-Type Graph Partitioning Algorithms”, IEEE ,1993

[8] Tarun Kumar Ghosh, Rajmohan Goswami, “Load Balanced Static Grid Scheduling Using Max-Min Heuristic” , 2nd IEEE International Conference on Parallel, Distributed and Grid Computing, 2012.

[9] Zehua Zhang, Xuejie Zhang, “A Load balancing mechanism based on Ant colony and complex network theory in Open Cloud Computing Federation”, 2nd International Conference on Industrial Mechanism and Automation,2010.

[10] Z. Chaczko, V. Mahadevan, S. Aslanzadeh, and C. Mcdermid, “Availability and load balancing in cloud computing”, International Conference on Computer and Software Modeling, Singapore, 2011.

[11] Daniel Grosu, Anthony T.Chronopoulos,” A game theoretic model and algorithm for load balancing in distributed systems”, 16th International Parallel and Distributed Processing Symposium, 2002.

[12] Gowtham Gajala, “Cloud Computing: A State of Art of the Cloud”, International Journal of Computer Trends and Technology, 2013.

[13] Syed Tauhid Zuhori,Tamana Shamrin,Runia Tanbin,Firoz Mahmud, “ An Efficient Load Balancing approach in cloud environment by using Round Robin algorithm” , International Journal of Artificial Intelligence and Mechatronics, 2013.

[14] Rashmi K.S, Suma.V, Vaidehi.M, “Enhanced load balancing approach to avoid deadlocks in cloud” ,

International Journal of Computer Application on Advanced Computing and Communication Technologies for HPC Applications, June 2012.

[15] Tejinder Sharma, Vijay Kumar Banga, "Efficient and Enhanced Algorithm in Cloud Computing", International Journal of Soft Computing and Engineering, March 2013.

[16] Marios D. Dikaiakos and George Pallis, Dimitrios atsaros, Pankaj Mehra, Athena Vakali, "Cloud Computing: Distributed Internet Computing for IT and Scientifc Research", IEEE 2009.

[17] Yi Lu, Qiaomin Xie, Gabriel Kliot, Alan Geller, James R. Larus, Albert Greenberg, "Join-Idle-Queue: A Novel Load Balancing Algorithm for Dynamically Scalable Web Services".

[18] Rade Stanojevi´c, Robert Shorten, "Load balancing vs. distributed rate limiting: a unifying framework for cloud control".

[19] Hao Liu, Shijun Liu, Xiangxu Meng, Chengwei Yang, Yong Zhang, "LBVS: A Load Balancing Strategy for Virtual Storage", International Conference on Service Sciences, 2010.

[20] Che-Lun Hung, Hsiao-hsi Wang and Yu-Chen Hu, "Efficient Load Balancing Algorithm for Cloud Computing Network".

[21] Yi Zhao, Wenlong Huang, "Adaptive Distributed Load Balancing Algorithm based on Live Migration of Virtual Machines in Cloud", Fifth International Joint Conference on INC, IMS and IDC, 2009.

[22] Vlad Nae, Radu Prodan, Thomas Fahringer, "Cost-Efficient Hosting and Load Balancing of Massively Multiplayer Online Games", IEEE 2010.

[23] Aameek Singh, Madhukar Korupolu, Dushmanta Mohapatra, "Server-Storage Virtualization: Integration and Load Balancing in Data Centers".

[24] Shu-Ching Wang, Kuo-Qin, Wen-Pin Liao and Shun Sheng Wang, "Towards a Load Balancing in a Three-level Cloud Computing Network", IEEE, 2010.

[25] Amandeep Kaur Sidhu, Supriya Kinger, "Analysis of Load Balancing Techniques in Cloud Computing", International Journal of Computers & Technology, April 2013.

[26] Jeffrey M. Galloway, Karl L. Smith, Susan S. Vrbsky, "Power Aware Load Balancing for Cloud Computing", Proceedings of the World Congress on Engineering and Computer Science, 2011.

[27] Yang Xu, Lei Wu, Liying Guo, Zheng Chen, "An Intelligent Load BalancingAlgorithm towards Efficient Cloud Computing", AI for Data Center Management and Cloud Computing: Papers from the AAAI Workshop, 2011.

[28] Ratan Mishra and Anant Jaiswal, "Ant colony Optimization: A Solution of Load balancing in Cloud", International Journal of Web & Semantic Technology (IJWesT), April 2012.

[29] H K Sawant,Sachin Shelke, "A Non cooperative approach for non cooperative load balancing in distributed systems", Journal of Information, knowledge and Research in Information Technology.

[30] Md.Firoj Ali, Rafiqul Zaman Khan, "The Study on Load Balancing Strategies in Distributed Computing System", International Journal of Computer Science and Engineering Survey, April 2012.