Advances in Computing 2017, 7(2): 28-34
DOI: 10.5923/j.ac.20170702.04

Survey on Various Load Balancing Techniques in Cloud Computing

Athokpam Bikramjit Singh[1], Sathyendra Bhat J.[1,*], Ragesh Raju[1], Rio D'Souza[2]

[1]Department of Computer Applications, St Joseph Engineering College, Mangaluru, India
[2]Department of Computer Science and Engineering, St Joseph Engineering College, Mangaluru, India

Abstract Load Balancing is one of the most significant concepts in distributed environments. As Cloud Computing is considered to be one of the best platforms that gives storage of data at a minimal cost and is accessible all the time over the internet, load balancing for the cloud computing has turned into a very interesting and important study area. Load balancing aims at high user satisfaction and usage of resource ratio by guaranteeing a proficient and reasonable allocation of each computing resource. There are numerous difficulties in load balancing techniques such as security, fault tolerance etc which are prevalent in modern cloud computing environments. Many researchers have proposed several techniques to enhance load balancing and this paper too, portrays an overview on load balancing schemes in cloud environments. We explore the diverse types of algorithms that are proposed by a number of researchers to solve the problem of load balancing in cloud computing.

Keywords Cloud Computing, Load Balancing

1. Introduction

Cloud is the cluster of distributed computers that provides on-demand computational resources over a network. Cloud computing is becoming an advanced technology in recent years. It is conceptually distributed system where computing resources distributed through the network (Cloud) and services pooled together to provide the users on pay-as-needed basis.

Cloud Computing provides everything as a service and are deployed as public, private, community, and hybrid clouds. The three basic service layers of cloud computing are: Software as a Service (SaaS), where the user does not need to manage the installation and configuration of any hardware or software, such as Google Online office, Google Docs, Email cloud, etc; Platform as a Service (PaaS), where a service is a delivery of a computing platform over the web where users can create and install their own applications as they need. Configuration of computing platform and server is managed by the vendor or cloud provider. Example of PaaS is Google App Engine. Infrastructure as a Service (IaaS), where servers, software, and network equipment is provided as an on-demand service by the cloud provider [1].

The main function of sharing resources, software, information through the internet are the main interest in cloud computing with an aim to reduce capital and operational cost, better performance in terms of response time and data processing time, maintain the system constancy and to accommodate future adaptation of the system. So there are various technical challenges that need to be addressed like Virtual Machine (VM) relocation, server consolidation, fault tolerance, high availability and scalability. But the central issue is the load balancing [2, 3]. It is the mechanism of spreading the load among various nodes of a distributed system to improve both resource deployment and job response time while also avoiding a situation where some of the nodes are having a huge amount of load while other nodes are doing nothing or idle with very little work. It also ensures that all the processor in the system or each node in the network does approximately the equal amount of work at any instant of time [4, 5].

The goal of load balancing is to improve the performance by balancing the load among the various resources (network links, central processing units, disk drives, etc.) to achieve optimal resource utilization, maximum throughput, maximum response time, and to avoid overload. Below Figure 1 shows the block diagram of cloud architecture [10].

2. Classification of Load Balancing Algorithms

Load balancing algorithms can be broadly classified into two types: Static algorithms and Dynamic algorithms. In Static Scheduling the assignment of tasks to processors is done before program execution begins i.e. in compile time. Scheduling decision is based on information about task execution times, processing resources, etc., which are

* Corresponding author:
sathyendrab@sjec.ac.in (Sathyendra Bhat J.)
Published online at http://journal.sapub.org/ac

assumed to be known at compile time [6]. Static scheduling methods are non-preemptive. The goal of static scheduling methods is to minimize the overall execution time. These algorithms cannot adapt to load changes during run-time [7].

Dynamic scheduling (often referred to as dynamic load balancing) is based on the redistribution of processes among the processors during execution time. This redistribution is performed by transferring tasks from the heavily loaded processors to the lightly loaded processors with the aim to improve the performance of the application. It is particularly useful when the requirement of process is not known a priori and the primary goal of the system is to maximize the utilization of resources. The major drawback of the dynamic load balancing scheme is the run-time overhead due to the transfer of load information among processors and decision-making for the selection of processes and processors for job transfers and the communication delays associated with the task relocation itself.

The dynamic load balancing algorithms can be centralized or distributed depending on whether the responsibility for the task of global dynamic scheduling should physically reside in a single processor (centralized) or the work involved in making decisions should be physically distributed among processors [8].

The most important feature of making decisions centrally is simplicity [6]. However, centralized algorithms suffer from the problem of the bottleneck and single point failure. Distributed load balancing algorithms are free from these problems. Again distributed dynamic scheduling can be cooperative or non-cooperative. The last one is simple where individual processors act alone as autonomous entities and arrive at decisions regarding the use of their resources independent of the effect of their decision on the rest of the system. In the former one each processor has the responsibility to carry out its own portion of the scheduling task to achieve a common system wide goal [6, 8].

2.1. The Load Balancing Problem can be Divided into Two Sub Problems

1. Submission of new task for VM provisioning and placement of VMs on host.
2. Reallocation/migration of VMs. Different load balancing algorithms are there in the literature which are discussed below.

3. General Load Balancing Algorithms for Cloud Computing

3.1. Round Robin Algorithm

The algorithm works on random selection of the virtual machines. The data center controller assigns the requests to a list of VMs on a rotating basis. The first request is allocated to a VM picked randomly from the group and then the data center controller assigns the requests in a circular order. Once the VM assigns the request, the VM is moved to the end of the list [9]. The major issue in this allocation is this that it does not consider the advanced load balancing requirements such as processing times for each individual requests and if the VM is not free then incoming job should wait in the queue.

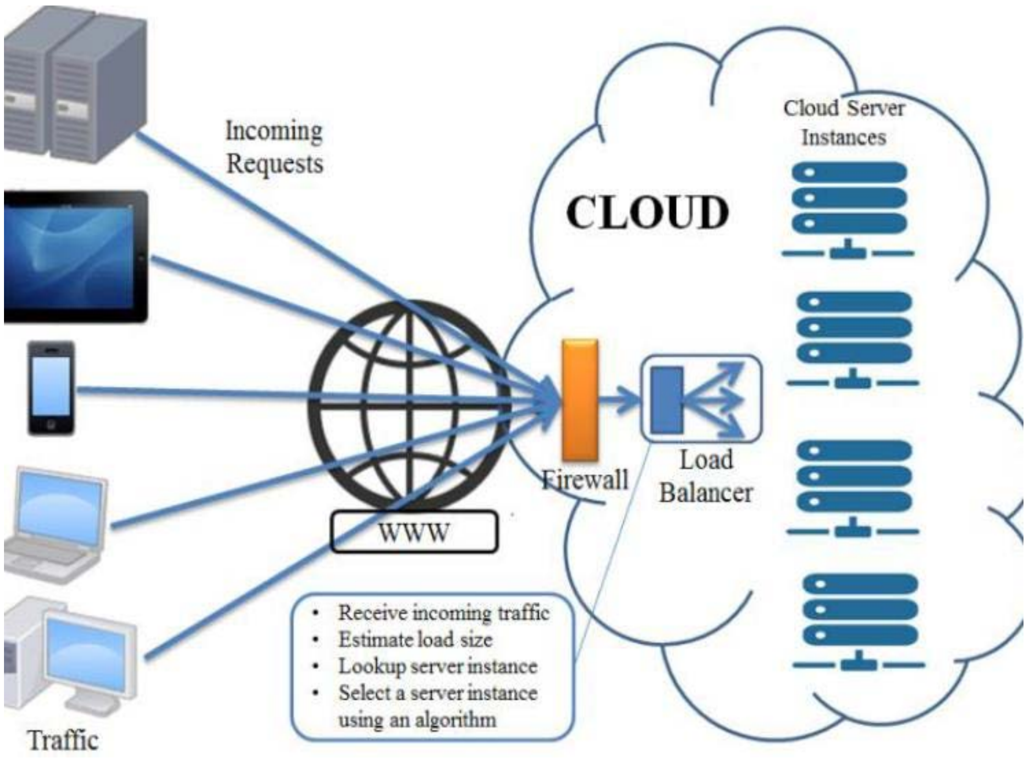

Figure 1. Block Diagram of Cloud Architecture

3.2. Throttled Load Balancing Algorithm (TLB)

In this algorithm the load balancer maintains an index table of virtual machines as well as their states (available or busy). The client/server first makes a request to data center to find a suitable virtual machine (VM) to perform the recommended job. The data center queries the load balancer for allocation of the VM. The load balancer scans the index table from the top until the first available VM is found or the index table is scanned fully. If the VM is found, the VM id is sent to the data center. The data center communicates the request to the VM identified by the id. Further, the data center acknowledges the load balancer of the new allocation and the data center revises the index table accordingly. During processing the request of the client, if appropriate VM is not found, the load balancer returns -1 to the data center [10, 11].

3.3. Modified Throttled

Like the Throttled algorithm, it also maintains an index table containing a list of virtual machines and their states. The first VM is selected in the same way as in Throttled. When the next request arrives, the VM at index next to already assigned VM is chosen depending on the state of the VM and the usual steps are followed, unlikely of the Throttled algorithm, where the index table is parsed from the first index every time the Data Center Queries Load Balancer for allocation of VM [12]. It gives better response time compare to the previous one. But in index table the state of some VM may change during the allocation of next request due to deallocation of some tasks. So it is not always beneficial to start searching from the next to already assigned VM.

3.4. Min-Min Scheduling Algorithm

It starts with a set of tasks. Then the resource which has the minimum completion time for all tasks is found. Next, the task with the minimum size is selected and assigned to the corresponding resource (hence the name Min-Min). Finally, the task is removed from set and the same procedure is repeated by Min-Min until all tasks are assigned. The method is simple, but it does not consider the existing load on a resource before assigning a task. So proper load balance is not achieved [13].

3.5. Load Balance Min-Min (LBMM)

This method uses Min-Min Scheduling algorithm as its base. It uses a three level hierarchical framework. Request manager which is in the first level of the architecture is responsible for receiving the task and assigning it to one service manager in the second level of LBMM. After receiving the request, service manager divides it into subtasks to speed up the processing. Then the service manager assigns the subtask to a service node for execution based on different attributes such as the remaining CPU space (node availability), remaining memory and the transmission rate. This algorithm improves the load unbalance of Min-Min and minimizes the execution time of each node, but does not specify how to select a node for a complicated task requiring large-scale computation [14].

3.6. Load Balance Improved Min-Min Scheduling Algorithm (LBIMM)

It starts by executing Min-Min algorithm at the first step. At the second step it chooses the smallest size task from the heaviest loaded resource and calculates the completion time for that task on all other resources. Then the minimum completion time of that task is compared with the makespan produced by Min-Min. If it is less than makespan then the task is reassigned to the resource that produces it, and the ready time of both resources are updated. The process repeats until no other resources can produce less completion time for the smallest task on the heavily loaded resource than the makespan. Thus the overloaded resources are freed and the under loaded or idle resources are more utilized. This makes LBIMM to produce a schedule which improves load balancing and also reduces the overall completion time. But still it does not consider priority of a job while scheduling [13].

3.7. User-Priority Aware Load Balance Improved Min-Min Scheduling Algorithm (PA-LBIMM)

User priority is incorporated with the LBIMM algorithm to develop PA-LBIMM. This algorithm will first divide all the tasks into two groups G1 and G2. G1 is for the VIP users' tasks having higher priority requirement. G2 is for the ordinary users' tasks. The higher priority tasks in G1 are scheduled first using the Min-Min algorithm to assign the tasks to the VIP qualified resources set. Then the tasks with lower priority are scheduled to assign them to all the resources by Min-Min algorithm. At the end, the load balancing function is processed to optimize the load of all resources to produce the final schedule. The algorithm is only concerned with the makespan, load balancing and user-priority. It does not consider the deadline of each task [13].

3.8. Cooperative Scheduling Anti-load Balancing Algorithm for Cloud (CSAAC)

Thiam et al [14] presented a decentralized dynamic scheduling approach entitled Cooperative Scheduling Anti-load balancing Algorithm for Cloud (CSAAC). CASA proposed algorithm adopts the promised job response time as the only criterion to evaluate the nodes capabilities. CSAAC adds load of node as another criterion to evaluate the nodes capabilities. Each responder node computes an estimated completion time, resource status, necessary energy, and also current load and delivers the information by means of an ACCEPT message. The node selected based on several parameters, such as the promised time to complete, energy consumed, the node load between under-load threshold and over-load threshold, node weight due to historical interaction records, etc. Selection policies take into account migration cost. The selected host was the one with the minimum energy consumed with best execution time, considered as migration cost to reduce the load of an overloaded host; it begins to migrate the slowest task. Selection policy will choose the task that will stay the longest on the host. Policy of localization will then identify the host that will receive the task without exceeding its capacities.

3.9. Load Balancing Strategy of Cloud Computing based on Artificial Bee Algorithm

J. Yao and J. H. [15] proposed an Artificial Bee Colony algorithm (ABC) based on the characteristics and requirements of cloud computing environments. Hundreds of thousands of simultaneous requests with the same type queued in the same server for the original ABC algorithm. Consequently, the local resource-intensive phenomenon raised and deteriorated load balancing. Due to the failure of this mechanism for the above case, an improved ABC is proposed. By replacing other types of requests with the next served request, the type of request is changed. It ended the accumulation of request and improved the system throughput. Experimental results show that ABC algorithm-based load balancing mechanism is applause for its stability and the improved ABC does well in the scalability.

3.10. Two-Phase Load Balancing Algorithm

This algorithm is the combination of the OLB and LBMM to have a better execution time and to balance the load more efficiently. A queue is used to store tasks that need to be carried out by manager. In the first phase OLB scheduling manager is used to assign job to the service manager. In the second phase LBMM algorithm is used to choose the suitable service node to execute the subtask by the service manager. The problem associated with this approach is that it is applicable only in static environment [16].

3.11. Adaptive Distributed Load Balancing Algorithm based on Live Migration of Virtual Machines in Cloud

R. Achar et al [17] proposed Distributed intra cloud load balancing algorithm to compare and balance based on sampling to reach equilibrium solution. The algorithm was executed concurrently on each host. Cost of running VM on each of the host was calculated and it assures that VM always migrates from pm with higher cost to those with lower one and larger was difference of costs, higher was migrated probability of VM's.

3.12. Load Balancing with Availability Checker and Load Reporters (LB-ACLRs)

P. B. Soundarabai et al. [18] proposed the concept of software based load balancer along with Availability-Checker and Load Reporters (LB-ACLRs) which reduces the overhead on server and the load balancer to improve performance in Distributed Systems. Motivated by the concept of CSMA Protocol Load Reporter stubs were deployed on each of the servers. This stub runs on the all server systems and provides current memory, CPU and network usage details to the Availability Checker at every time interval. Load Reporters (LRs) update the AC with all the load details from the various available servers which get stored in a hash table or database. Using this database, AC updates the main that was present in LB. LB was only responsible for selecting the available server based on the database information and redirecting the client requests to the next selected server. Only one TCP connection was required to update the servers' availability. So no multithreaded environment was used to collect the servers' load and availability details every time.

4. Cost Effective Load Balancing Algorithms for Cloud Computing

4.1. Load Balancing with Optimal Cost Scheduling Algorithm

Amanpreet Chawla and Navtej Singh Ghumman [19] used Round Robin algorithm to schedule incoming tasks and optimizes the cost and schedules the resources based on the cost. In the proposed algorithm resources were grouped as packages in each VM. When the user requests for the resource the VM consisting of that package was executed. This technique brings down the execution cost of the service provider.

4.2. Cost Effective Load Balanced Resource Allocation for Partitioned Cloud System

M. R. Sumalatha et al [20] proposed DBPS (Deadline Based Pre-emptive Scheduling) and a TLBC (Throttled Load Balancing for Cloud) load balancing model based on cloud partitioning using virtual machine. Once a task was submitted to the cloud server, it was divided into several sub tasks. The workload of the task was compared with the training set collected from various virtual machines and also the relative deadline of that task was predicted using samples.

These tasks were assigned to the Task Manager and Deadline Based Priority Scheduling was applied on the task set. The scheduled task set was given to the main controller which performs load balancing. The main controller maintains a status table where the status of all the nodes is stored. The task with higher priority in the scheduled list was submitted to the node with the exact amount of resources available for the task.

The node controller again checks the status of the Virtual Machine and submits the task to the virtual machine with appropriate CPU and I/O dimensions and then the tasks were executed. Reduction of Execution time and execution cost was demonstrated.

4.3. Power Aware Load Balancing for Cloud Computing

J. M. Galloway et al [21] first gathers the utilization percentage of each active compute node. In the case that all compute nodes were above 75% utilization, PALB instantiates a new virtual machine on the compute node with the lowest utilization number. Otherwise, the new Virtual Machine (VM) was booted on the compute node with the highest utilization (if it can accommodate the size of the VM).

4.4. The Load Balancing based on the Estimated Finish Time of Tasks in Cloud Computing

Y. Fahim et al [22] proposed a new improvement of the load balancing by the algorithm "estimated finish time load balancer", that takes into account, the current load of the virtual machine of a data center and the estimation of the processing finish time of a task before any allocation, in order to overcome the problems caused by the static algorithms. The algorithm allows cloud service providers, to improve the performance, availability and maximize the use of virtual machines in their data centers.

4.5. Load Balancing Strategy for Optimal Peak Hour Performance in Cloud Datacenters

A. K. Kulkarni and B. Annappa [23] proposed a VM load balancing algorithm that ensures uniform allocation of requests to Virtual machines even during peak hours when frequency of requests received in data center was very high to ensure faster response times to users was proposed.

4.6. Cloud Task Scheduling based on Load Balancing Ant Colony Optimization

K. Li et al [24] proposed a cloud task scheduling policy based on Load Balancing Ant Colony Optimization (LBACO) algorithm. The main contribution of their work was to balance the entire system load while trying to minimize the makespan of a given tasks set. The new scheduling strategy was simulated using the CloudSim toolkit package. Experiments results showed the proposed LBACO algorithm outperformed FCFS (First Come First Serve) and the basic ACO (Ant Colony Optimization).

4.7. Bee-MMT: A Load Balancing Method for Power Consumption Management in Cloud Computing

S. M. Ghafari et al [25] proposed a load balancing method called Bee-MMT (artificial bee colony algorithm-Minimal Migration Time), which using bee colony algorithm to detect over utilized hosts. Then with the MMT VM selection, selects one or more VMs to migrate from them to reduce their utilization. Meanwhile, it can find underutilized hosts and if it is possible, migrate all VMs which allocated to these hosts and then switch them to the sleep mode.

4.8. Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization

F. Ramezani et al [26] proposed a Task Based System Load Balancing method using Particle Swarm Optimization (TBSLBPSO) to achieve system load balancing by only transferring extra tasks from an overloaded VM instead of migrating the entire overloaded VM. Particle Swarm Optimization (PSO) optimization model was used to migrate the extra tasks to the new host VMs. It is shown that the TBSLB-PSO method significantly reduces the time taken for the load balancing process compared to traditional load balancing approaches and the overloaded VMs are not paused during the migration process, and there was no need to use the VM pre-copy process. It eliminated VM downtime and the risk of losing the last activity performed by a customer, and increased the Quality of Service experienced by cloud customers.

5. Cluster Based Load Balancing Algorithms for Cloud Computing

5.1. A Cluster-Based Load Balancing Algorithm in Cloud Computing

S. K. Dhurandher [27] proposed cluster based load balancing algorithm for cloud computing. The network was divided into clusters. Every cluster had at least one Inter Cluster Communication (ICC) node. A Slave was the computing element of the network. Every slave was connected to exactly one master directly. The slave periodically updates its master with the most recent values of the storage, bandwidth, processing. The master node maintains a table of load distribution among slaves. Each entry of the table describes the load on the respective slave.

5.2. Proposed Load Balancing Algorithm is Divided into Two Parts

1) Load distribution among masters and
2) Load distribution from master to slave

5.2.1. Load Distribution among Masters Based on

- **a.** On receiving the task, the master calculates a parameter called performance factor, which indicates the ability of the master to perform a specific task.
- **b.** Based on performance factor task is executed within the cluster or broadcasted to be executed by suitable cluster.

5.2.2. Load Distribution from Master to Slave is done with RR Scheduling

For Cluster Based Load Balancing in Cloud Computing, S. Kapoor, and C. Dabas [27] proposed a Cluster based load balancing algorithm which considered resource specific demands of the tasks and reduced scanning overhead by dividing the machines into clusters. Further K-means clustering approach was used to divide VMs into cluster. The algorithm is compared with the existing throttled and modified throttled algorithms and it is shown that the algorithm gives better results in terms of waiting time, execution time, turnaround time and throughput.

6. Conclusions

This has been an attempt to survey multiple algorithms and also to discuss about the different algorithms that exist for load balancing in cloud computing as well as metrics for the same. Load balancing is one of the most important aspects of cloud computing and is essential to distribute the extra dynamic local workload consistently to the entire node in the whole cloud to attain a high user satisfaction and resource utilization ratio. It also guarantees that every computing resource is distributed efficiently and fairly. A vast number of parameters and different types of soft computing techniques can be included in future for the better utilization and needs of the user. A comparison of several load balancing techniques has also been done here.

REFERENCES

[1] A. Beloglazov, and R. Buyya, Energy efficient resource management in virtualized cloud data centers, Proc. 10th IEEE/ACM international conference on cluster, cloud and grid computing, 2010, 826-831.

[2] Qi Zhang, Lu Cheng, and Raouf Boutaba, Cloud computing: state-of-the-art and research challenges, Journal of Internet Services and Applications, 1(1), 2010, 7-18.

[3] Amandeep Kaur Sidhu, and Supriya Kinger, Analysis of Load Balancing Techniques in Cloud Computing, International Journal of Computers & Technology, 4(2), 2013, 737-741.

[4] R. Mishra, and A. Jaiswal, Ant colony optimization: A solution of load balancing in cloud, International Journal of Web & Semantic Technology, 3(2), 2012, 33.

[5] E. Caron, L. Rodero-Merino, F. Desprez, and A. Muresan, Auto-scaling, load balancing and monitoring in commercial and open-source clouds, 2012.

[6] X. Evers, W. H. CSG, CR. B. SG, I. S. Herschberg, D. H. J. Epema, and J. F. C. M. de Jongh, A literature study on scheduling in distributed systems, Delft University of Technology, 1992.

[7] K. A. Nuaimi, N. Mohamed, M. A. Nuaimi, and J. Al-Jaroodi, A survey of load balancing in cloud computing: Challenges and algorithms, Proc. 2012 Second Symposium on Network Cloud Computing and Applications (NCCA), 2012, 137-142.

[8] Kuhl, A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems, IEEE Trans, on Software Eng., 14(2), 1988, 141-154.

[9] M. M. D. Shah, M. A. A. Kariyani, and M. D. L. Agrawal, Allocation of Virtual Machines In Cloud Computing Using Load Balancing Algorithm, IJCSITS, 2013, 2249-9555.

[10] B. Wickremasinghe, CloudAnalyst: A CloudSim-based tool for modelling and analysis of large scale cloud computing environments, MEDC project report, 22(6), 2009, 433-659.

[11] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications, Proc. 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), 2010, 446-452.

[12] S. G. Domanal, and G. R. M. Reddy, Load Balancing in Cloud Computing using Modified Throttled Algorithm, Proc. IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), 2013, 1-5.

[13] Elian, and G. Akanmu, User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing, Proc. National Conference on Parallel Computing Technologies (PARCOMPTECH), 2013, 1-8.

[14] C. Thiam, G. da Costa, and J. M. Pierson, Cooperative Scheduling Anti-load Balancing Algorithm for Cloud: CSAAC, Proc. 5th International Conference on Cloud Computing Technology and Science (CloudCom), 2013, 433-438.

[15] J. Yao, and J. H. He, Load balancing strategy of cloud computing based on artificial bee algorithm. Proc. 8th International Conference on Computing Technology and Information Management (ICCM), 2012, 185-189.

[16] Shu-Ching Wang, Kuo-Qin Yan, Wen-Pin Liao, and Shun-Sheng Wang, Towards a Load Balancing in a Three-level Cloud Computing Network, Proc. 3rd International Conference on Computer Science and Information Technology (ICCSIT), 2010, 108- 113.

[17] R. Achar, P. S. Thilagam, N. Soans, P. V. Vikyath, S. Rao, and A. M. Vijeth, Load balancing in cloud based on live migration of virtual machines, Proc. Annual IEEEI India Conference (INDICON), 2013, 1-5.

[18] P. B. Soundarabai, A. S. Rani, R. K. Sahai, J. Thriveni, K. R. Venugopal, and L. M. Patnalk, Load balancing with availability checker and load reporters (LB-ACLRs) for improved performance in distributed systems, Proc. 2nd International Conference on Devices, Circuits and Systems (ICDCS), 2014, 1-5.

[19] Amanpreet Chawla, and Navtej Singh Ghumman, Efficient Cost Scheduling algorithm with Load Balancing in a Cloud Computing Environment, International Journal of Innovative Research in Computer and Communication Engineering, 3(6), 2015.

[20] M. R. Sumalatha, C. Selvakumar, T. Priya, R. T. Azariah, and P. M. Manohar, CLBC-Cost effective load balanced resource allocation for partitioned cloud system, Proc. International Conference on Recent Trends in Information Technology (ICRTIT), 2014, 1-5.

[21] J. M. Galloway, K. L. Smith, and S. S. Vrbsky, Power aware load balancing for cloud computing, Proc. the World Congress on Engineering and Computer Science, 2011, 19-21.

[22] Y. Fahim, E. Ben Lahmar, E. H. Labriji, and A. Eddaoui, The load balancing based on the estimated finish time of tasks in cloud computing, Proc. Second World Conference on Complex Systems (WCCS), 2014, 594-598.

[23] A. K. Kulkarni, and B. Annappa, Load balancing strategy for optimal peak hour performance in cloud datacenters, Proc. International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2015, 1-5.

[24] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, August. Cloud task scheduling based on load balancing ant colony optimization, Proc. Sixth Annual Chinagrid Conference (ChinaGrid), 2011, 3-9.

[25] F. Ramezani, J. Lu, and F. K. Hussain, Task-based system load balancing in cloud computing using particle swarm optimization, International Journal of Parallel Programming, 42(5), 2014, 739-754.

[26] S. K. Dhurandher, M. S. Obaidat, I. Woungang, P. Agarwal, A. Gupta, A. and P. Gupta, A cluster-based load balancing algorithm in cloud computing, Proc. IEEE International Conference on Communications (ICC), 2014, 2921-2925.

[27] S. Kapoor, and C. Dabas, Cluster based load balancing in cloud computing, Proc. Eighth International Conference in Contemporary Computing (IC3), 2015, 76-81.