International Journal of Engineering & Technology, 7 (4.12) (2018) 13-19

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

A Software Agent Based Technique for Load Balancing in Partitioned Cloud

Mandeep Kaur[1], Dr. Rajni Mohana[2]*

[1] School of Computer Applications, Lovely Professional University, Phagwara, Punjab (India)
[2]Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat, Solan (India)
**Corresponding author E-mail: rajni.mohana@juit.ac.in*

Abstract

Large number of users are shifting to the cloud system for their different kind of needs. Hence the number of applications on public cloud is increasing day by day. Public clouds considered and is the most convenient platform for common cloud users with generic needs and lesser security concerns. Public cloud can cater to the needs of a large group of users and provide a variety of services. Lower cost and timely availability are the other advantages one expects from public clouds. These features make it very much convenient and attractive choice. But on the other hand, handling public cloud become unmanageable in comparison to other counterparts. Monitoring so many users, tasks and resources are difficult task. Sometimes public clouds are divided on geographically. Geographic partitioning of public cloud can resolve these issues by adding manageability and efficiency in this situation. But, partitioned clouds introduce different ends for submission and operations of cloudlets and virtual machines. This ends for task submission and resource allocation adds complexities also. A concrete mechanism is to be designed for handling the load allocation and processing of the nodes. The proposed work is addressing the same issue by advising a combination of centralized and decentralized load balancing. The main objective of this work is to fix a VM for a cloudlet, which can process it in minimum time and without overloading or underloading the datacenters. Another objective under consideration is to reduce the number of jobs left unhandled due to threshold constraints.

Keywords: *Cloud Computing, Public Clouds, Geographical Cloud Partitioning, Software Agents, Centralized Load Balancing, Decentralized Load Balancing, Static Load Balancing..*

1. Introduction

This Cloud Computing has the potential to affect a large part of IT industry. Nowadays developers need not to concern about the over-provisioning or under-provisioning of resources. Elasticity of resources without spending a large amount of funds is a unique concept of its kind [1]. Basic characteristics of Cloud Computing such as user friendliness, virtualization, automatic adaptation, scalability, resource optimization, pay-per-use, service SLAs, infrastructure SLAs etc. is attracting users in masses [2]. Cloud Computing is proving itself so beneficial, but the performance and efficiency of services is needed to be maintained. Handling concurrent jobs, users and processes through such a large set of machines is a very difficult task. Load balancing is one of the major concerns in success of cloud computing. Imbalanced load among servers is a major challenge in cloud computing. Many kinds of troubles occur due to overloaded as well as under load servers. Under loaded servers cause energy inefficiency, inefficient use of resources and add on to the management overheads. On the other hand overloaded servers can cause delay in response, low speed of processing, decreased throughput, increased makespan etc. To handle both these situations it is very important to adapt a load balancing mechanism which can distribute tasks evenly among all available servers. But load balancing is not a very simple task as it seems. Load management becomes even more challenging when we are considering a large public cloud. A public cloud has numerous nodes, scattered around in various geographic locations. Small partitions are more manageable as compared to a large group of nodes. So handling load will become simple and efficient.

Further, use of Software Agents can increase the efficiency and management of these partitions. By adding software agents to our load balancing mechanism we can expect all the advantages associated with a common set of characteristics of software agents. Software agents add intelligence and automation to the processes. Being autonomous in nature, agent based components do not demand much of the user intervention. Interoperability makes agents to communicate with users, applications and other agents. This feature also facilitates extensibility. An agent can learn from and respond to the environment, it is deployed in. Above all the agents can be trained as per individual requirements of its user entities. Involving all above features in load balancing mechanism can prove amazingly beneficial. This paper is an attempt to implement Software agents along with the concept of cloud partitioning. With the help of software agents we expect improved speed, better efficiency, reduced throughput and reduced makespan.

1.1 Contribution

Following a static threshold value to decide a service providing node in public cloud can cause less than optimal utilization of resources. Public cloud is an integration of a large number of resources and handling large number of users and their requests. At a given time there can be drastic changes in load status of the

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

nodes. The proposed algorithm considers a node for load allocation beyond the threshold values allocated to it. Another area aimed in current work is to reduce the execution and wait time of job by allocating cloudlets to the nodes with the best possible set of available resources and least execution time.

1.2 Paper Structure

In following sections of this paper, a decentralized load management technique is discussed. Section-II contains the background of partitioned public cloud and load balancing. Section-III contains the terminology, commonly used under current context. Section-IV discusses various metrics used for evaluation of load balancing algorithms. Section-V explains the literature review and existing work done in this area. In Section-VI contains the details about the proposed methodology of load balancing. Section-VII shows an experimental, simulation-based evaluation along with the retrieved. Section-VIII Finally, concludes this paper and identifies paths for future work. At the end all the references are mentioned in Section-IX.

2. Background

A public cloud is comprised of the infrastructure and computational resources, which are available for common public. Its ownership and operation rest with an operator. Consumer uses infrastructure external to his organization. Public cloud gives its consumers very limited control over the infrastructure and computing resources. Another attribute of public cloud is that it accommodates a large number of nodes in it. These nodes are geographically scattered around the world. Managing such a large number of geographically distant nodes is very complex. Implementing load balancing in such clouds is a very difficult task. Collecting load information of all these nodes and then using this information for balancing load is almost infeasible. It is a challenging job to decide which task to be allocated to which node, which node is overloaded or which node is underloaded. Hence, it is better to divide these large clouds in smaller partitions.

3. Terminology

3.1 Public Cloud

Provides scalable and elastic IT-enabled capabilities to customers who are not part of providers' organization. Public cloud computing promotes scaling and sharing of resources which helps users in saving cost and offers multiple options to choose an appropriate technology.

3.2 Cloud Partitioning

When a cloud environment is very large and complex it can be divided in small parts. It simplifies the cloud management, including load balancing. [3]

3.3 Cloud Partitions

A cloud partition is a Subarea in large cloud which can be managed separately. While load balancing every partition represents an individual, independent unit which can be assigned to tasks. [3]

3.4 Workload

The workload is the total time, which a processor takes to complete tasks allocated to it. [4]

3.5 Load Balancing

Process to systematically distribute the tasks to various nodes, ensuring that none of the nodes are overloaded or underloaded. [5]

4. Metrics

Following metrics can be considered to evaluate the performance of a load balancing algorithm.

4.1 Makespan

Makespan can be described as the longest time of processing by when all the jobs would have been finished on all the hosts in a cloud. This parameter is one of the most important criteria to evaluate performance of load balancing algorithms. Minimal makespan is desired from an algorithm. [6]

4.2 Number of Overloaded Hosts

A predefined threshold value defines if a host is overloaded or not. Count of such hosts, exceeding the threshold values is important to determine the overall status of the system.
Overloaded hosts are considered a risk to SLA fulfillment as these directly affect the performance of cloud. The aim of algorithm should be to minimize the number of overloaded hosts. [7]

4.3 Inter-host Communication Cost

For any migration, communication is required between host, VMs and other hosts. There is a communication cost involved in this in the form of the number of messages being passed, time taken to process those messages, resources being consumed in communication etc. Load balancing algorithm should be able to minimize this cost.

4.4 Percent of all VMs to be located in the Host

This parameter shows the minimum and maximum percentage of VMs to be located in a host. This parameter has a limitation that only a count of VMs is considered and not other parameters. Hence, in case VMs are heterogeneous, this parameter does not provide the exact load status of each cloud. [8]

4.5 Throughput

This parameter determines how fast a host can produce output to requests submitted to it. The maximum rate at which a host can finish jobs allocated to it is its throughput. Throughput depends upon various factors such as load at a time, data rate, resource availability etc. [9]

4.6 Average Imbalance Level

This is a parameter to determine deviation of multiple resources on all hosts and then combines them together with weights to denote the load balance effect. This parameter considers multiple resources like CPU, memory and bandwidth together. [10]

4.7 Capacity-Makespan

This parameter is derived from the makespan. It considers the required capacity and processing time. Capacity makespan is calculated by finding sum of product of required capacity and its processing time. [11]

4.8 Imbalance Score

This metric considers multiple resources. It helps in determining the extent to which a host is being over-utilized, above a pre-defined threshold. All hosts' individual imbalance score is to be summed up to calculate overall imbalance of a system. Any load balancing algorithm which cares about this metric has to minimize its value. [12]

4.9 Number of Migrations

VM migration is an effective way to implement load balancing. But contrary to this, it causes performance degradation. A trade-off is to be maintained between load balancing and performance level. This parameter cannot be used alone for affective load balancing. [13]

4.10 SLA Violation

In a cloud computing environment, resources are provided in the form of services, as per need specified in requests. While providing these services, clients as well as providers have to stick to an SLA. The main cause of SLA violation can be migrations. Hence efforts are required to be made to minimize the SLA violations. [14]

5. Potential of Use Of Software Agents

Software Agents are the kind of programs with ability to make decisions regarding what is required to be done in a particular situation. These possess the ability to work autonomously. When deployed in a system, multiple agents are put together to achieve system goals. The existence of multiple agents in the system creates need of abilities like Cooperation, Collaboration and Negotiation. There are various decision making points in the processes of cloud computing where user has to decide about the providers' selection, service and cost negotiations, finalizing the services according to specific needs and drafting SLAs. On the other hand, providers have their own part of decision making which involves selecting requests according to availability of resources, carrying out the execution of selected tasks, to predict future requirements of resources, etc. Software agents have ability to make all these decisions on behalf of both users, as well as providers. Multi Agents System (MAS) with pre-defines roles and responsibilities of each agent can contribute at large scale. [15]

Each Agent having pre-defined roles and responsibilities and capabilities to work together in a cooperative and collaborative manner. A large variety of software agents are available to choose from for a particular use. Figure 1 shows the classification of software agents.

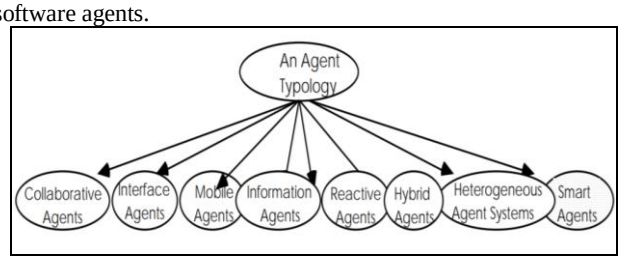

Fig. 1: Classification of Software Agent [16]

6. Related Work

Load balancing in cloud computing has been a very popularly discussed issue. Many researchers have focused their work towards resolving load balancing challenges in cloud computing. Hui Zhang et al. have discussed various issues relevant to this area such as service availability, reliability, SLA etc. [15] . Michael Pantazoglou et al. have suggested three algorithms for initial VM Placement, Partial VM Migration, and Full VM Migration. These algorithms are designed, focusing on the issues of elasticity, scalability, high cost etc. [16]. Matthias Sommer et al. have addressed the issue of energy consumption and its effects by proposing a novel proactive VM migration policy utilizing forecasts (PRUF) in Cloud data centers using a predictive overload detection. They have used short-term VM utilization [17]. Sadeghi Milani et al. have analyzed various existing load balancing techniques and presented a comparative approach to their applicability [18]. Stefano Sebastio et al. have suggested basic evaluation of a cloud partitioning approach to allocate requests in volunteer cloud. These requested tasks are validated using the Google workload data to trace [19]. Suguna R et al. have proposed a strategic model which dynamically partitions nodes on different cloud along with load balancing [20]. Abhay Kumar Agarwal et al. have presented a new algorithm based upon the existing load balancing algorithms and have concluded that Throttled Load Balancing Algorithm is best among all the existing algorithms in use [21]. As per the existing work discussed above, there is a need for simple as well as reliable method which can reduce the execution time, overheads, delays etc. and can efficiently utilize the available resources. In comparison to the above approaches, our proposed algorithm is combining centralized and decentralized methods of load balancing which provides control as well as flexibility and liberty to choose the best possible options available around the cloud. Finally the algorithm is evaluated with the help of common parameters.

7. Proposed Model

7.1 Assigning jobs to the cloud partition

We can use four partition status types:

1. Idle
2. Normal
3. Overload
4. Full

7.2 Significance of Full (fourth status type)

Partitioning concept is applied in case of large and public cloud. When idle, normal or overload is calculated it sees a particular percentage of nodes which are idle, and entire partition is declared idle. Similarly, if a particular percentage of nodes in a partition are overloaded the entire partition is declared overloaded. In case of public cloud, let us say even if 90% is the threshold for overloaded partition, 10% will hold a sufficient number of nodes to handle minor sized jobs. But in the current model, these 10% are being neglected. Hence, if a partition is calculated with an overload but still has some capacity, then this capacity can be used for job which have no time constraint, critical, small or are of high priority.

In short, in overload situation, chances of allocation are still there. Full shows that there is no way to allocate a partition to a job, i.e. even overloaded situation has been handled already.

For this functionality of Balancer Agent will include a tracking record of individual nodes also. Till now, what is considered is the sum of all the load of all nodes in a partition.

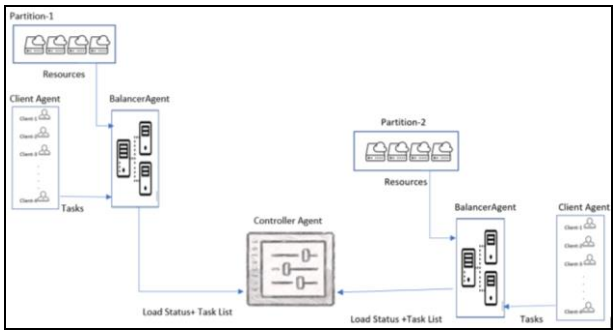

Fig. 2: Components of proposed model

7.3 Algorithm

1. While job do
2. Client Agent (CA) submits a request to the Balancer Agent (BA)
3. BA determines the resource requirements of the job
4. BA compares request of resources with availability
5. If current partition is capable to handle job itself by checking its load status (idle, normal, high).
6. If load status is idle or normal, then compare this partition's capacity with requirement of resources. And if the partition has sufficient resources, then allocate job to the same partition.
7. If current partition is overloaded then Control is sent over to the Central Controller Agent (CCA) traverse all other partitions and create 2 lists:
    a. List of all partitions with load status idle or normal.
    b. List of all overloaded partitions, i.e. partitions with load status high.
8. If the count of the first list is >=1 i.e. there is any node with load status idle or normal then

    a. Single partition: Compare resources with the requirement. If sufficient resources, then allocate.
    b. Multiple partitions:
        i. Compare the capacity of all the partition and find out one with maximum resources (for each resource individually)

        ii. Compare the requirements of resources with capacity of above partition. If sufficient "all" resources, then allocate tasks to this partition.

    c. No Partition: Step 6
9. If all other partitions are showing overloaded situation, then the node level search is required. So, collect individual node's load status from partition load balancers.
10. To traverse all nodes. Collect all nodes with idle or normal load status. Store their available resource capacity.
11. Fix the nodes/partitions (node or partition depends upon Point-1 in remaining) with maximum available capacities.
12. Compare these with the job's resource requirement.
13. If job's resource requirements can be fulfilled with available "all" resources of a node, allocate the job to that node and that partition as:
    a. Single partition: Compare resources with the requirement. If sufficient resources, then allocate.
    b. Multiple partitions:
        i. Compare the capacity of all the nodes and find out one with maximum resources (for each resource individually)
        ii. Compare the requirements of resources with capacity of above (max) partition. If sufficient resources, then allocate tasks to this partition.

    c. No nodes/Partition: Step 12
14. If no partition has sufficient capacity, then "No allocation is possible".
15. End

7.4 Agents Involved in the Process

For achieving above said goals 3 software agents can be created and deployed as follows:

Client Agent (CA)
- Submit the tasks to a resource provider
- Submit resource requirements of that task

Balancer Agent (BA)
- Located inside the partitions
- Invoked whenever a task is submitted to the relevant partition
- Makes load allocation decisions within a partition
- Does load related account keeping for a partition?
- Only access point for the controller agent to fetch, load status of a partition

Centralized Controller Agent (CCA)
- Centralized load distributor
- It interacts with balancers for load allocation decisions
- Balancer agents invoke a controller when a partition gets overloaded
- Invoke other partitions' balancers to assess and share load status
- The node is decided on the basis of the following criteria:
  - Maximum amount of available resources
  - Enough resources to cater the needs of the job

8. Experiments and Implementation

A multistep process is to be performed for allocating a specific task to a resource. The primary objective is the allocation of task in such a manner that processing time can be reduced and all the available resources are utilized equally. For attaining these objectives following set of processes are to be performed:

8.1 Overall Process

Fixing an Appropriate Partition

All the resource providers are grouped as per their geographical location. Whenever a task is submitted, first priority is always a provider available at the nearby location. Many economic, operational, and legal issues can be the reason behind it. But in case the nearest possible resource provider has been already busy then another resource provider from a neighboring partition is to be searched. This task is the responsibility of the Controller Agent (CCA) because that is the only global entity, having equal and open access to all the partitions in a public cloud. Here is how it is done:

- Acquire the list of resource providers, along with their respective partition ids

- Acquire the load status of all the partitions, which is the sum of load status of all the resource providers in a partition
- Choose a partition wherever the maximum number of resources are available.

Once a partition is chosen, updated load status of all the resource providers of that partition is collected. Finally, the list of resource providers in chosen partition are saved for further considerations.

Fixing an Appropriate ResourceProvider

Whether first step is performed or not, i.e. whether a local partition is chosen or a remote one, subsequent steps are always executed. Finding an appropriate resource provider is the responsibility of a Balancer Agent (BA). Balancer Agent keeps track of load status of all the resource providers under a partition and share this status with CCA whenever it is required. The system must have acquired the list of resource providers before initiating this step. This list either would have acquired from CCA or the current partition is the local partition itself. The next task is to fix one resource provider from this list. Following are the activities to be performed for under this process:
Acquire the list of resource instances with each resource provider in the list.

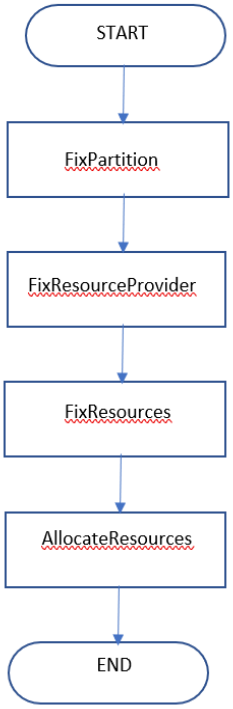

Fig. 3: Overall sequence of processes

Fixing an Appropriate ResourceProvider

Whether first step is performed or not, i.e. whether a local partition is chosen or a remote one, subsequent steps are always executed. Finding an appropriate resource provider is the responsibility of a Balancer Agent (BA). Balancer Agent keeps track of load status of all the resource providers under a partition and share this status with CCA whenever it is required. The system must have acquired the list of resource providers before initiating this step. This list either would have acquired from CCA or the current partition is the local partition itself. The next task is to fix one resource provider from this list. Following are the activities to be performed for under this process:

- Acquire the list of resource instances with each resource provider in the list.
- Acquire the load status of each instance of resources.
- Find out the resource provider with maximum number of resource instances having available resources with them
- Chosen resource provider and its resource instances will be considered in subsequent steps.

The output of this process will be a resource provider along with the list of resource instances belonging to it.

Fixing an Appropriate Instance of Resource

This step chooses the most appropriate instance of resources to which a task can be allocated. For this selection two aspects are to be considered: a) The task should be allocated to an instance where it will take minimum execution time. b) The chosen node should not be overloaded. The first aspect is straightly a relation between the available resource capacity vs resource requirement of the current task. Second aspect considered the already running load or number of tasks on the resource instance. An instance, fulfilling both the criteria, in best manner will be chosen as the final instance where tasks can be sent. In this process following activities are to be performed:

- Acquire the capacity of resources with each resource instance.
- Acquire the resource requirement of current task.
- Compare the capacity and requirement and find out which instance can finish the task in minimum time.
- Check if the chosen instance is already having some pending jobs. Also, the load status due to those jobs.
- Chose the instance with minimum execution time and appropriate load status.

The output of this process will be the resource instance to which the task can be allocated finally.

Allocation of Task

Once the resource instance is chosen now task will be associated with this instance. All the required parameters are set and load status of resource instance, corresponding resource provider and corresponding partition are updated.

8.2 Simulation setup

Table 1: Simulation Parameters

Parameter	Value
Partition Count	2
Datacenter Count	4
VM Count	24 (6 bound to each DC)
Cloudlet Count	40
Datacenter Broker Count	2
VM Resources	Heterogenous
Cloudlet Requirements	Heterogenous

8.3 Results

Simulations are performed using the cloudSim tool as per values specified in Table II. The proposed algorithm is evaluated based upon two parameters which are makespan and numbers of jobs detained (not served immediately after their arrival due to overloaded resources). These parameters are compared with SJF and FCFS algorithms, implemented in similar setups. As can be seen in Figure 2 proposed algorithm is capable to serve better count of jobs as compared to other 2 algorithms. Reason is introduction of 4th load state which has enhanced utilization of resources even better. 2 out of 5 times none of the jobs were left detained.

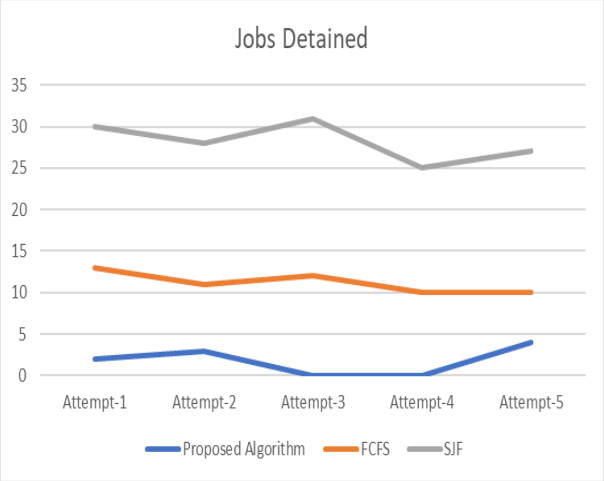

Fig. 4: No. of jobs detained due to overloaded resources

Second comparison is made between the average, maximum and minimum makespan of 3 of these algorithms. Results obtained from 5 times execution shows that SJF and the proposed algorithms produce almost similar results, i.e. takes a similar time to execute the assigned set of jobs. Also, as all the algorithms under consideration handle variable count of job every time, hence makespan is calculated considering the minimum number of jobs being handled by any of these three algorithms.

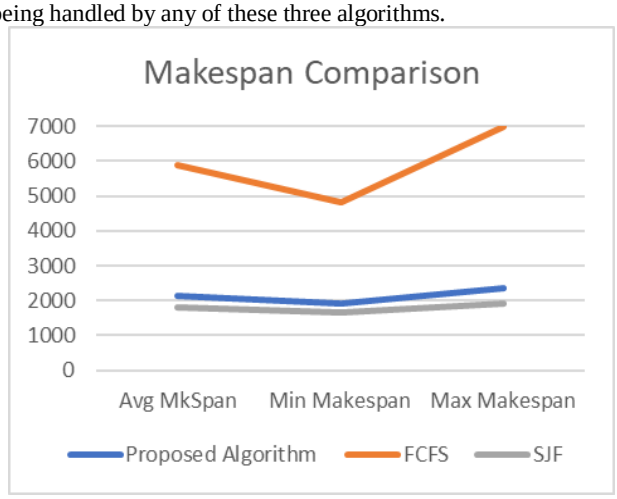

Fig. 5: Makespan Comparison

These results show that proposed algorithm is capable to produce better results when compared to FCFS algorithm, but more improvements are required to make its results better than SJF algorithm.

9. Conclusion

An attempt is made in this paper to improve the performance of job scheduling algorithms in cloudsim environment. Targeted cloud type is Public cloud and we are considering mainly geographical partitioned cloud where the number of nodes and resources are grouped together and considered for allocation to the tasks. An attempt is made to enhance the allocation process beyond threshold limits. During initial implementation attempts for achieving this target, the proposed algorithm is producing better results than basic algorithms such as FCFS. In future, further efforts can be put to implement advanced techniques of load balancing to make the performance of proposed work compatible to more modern and advanced algorithm under use. Another point of consideration in future work is to find a systematic method to determine the threshold values to measure load states and appropriate number of load states, to efficiently handle task execution as well as resource utilization.

References

[1] Michael Armbrust, "A View of Cloud Computing," Communications of ACM, vol. 53, no. 4, pp. 50-58, 2010.
[2] Luis M. Vaquero1, "A Break in the Clouds: Towards a Cloud Definition," ACsM SIGCOMM Computer Communication Review, vol. 39, no. 1, pp. 50-55, 2009.
[3] Xu, G., Pang, J. and Fu, X., "A load balancing model based on cloud partitioning for the public cloud," Tsinghua Science and Technology, vol. 18, no. 1, pp. 34-39, 2013.
[4] M. Xu, W. Tian and R. Buyya, "A survey on load balancing algorithms for virtual machines placement in cloud computing," Concurrency and Computation: Practice and Experience, pp. 1-22, 2017.
[5] K. Cho, P. Tsai, C. Tsai and C. Yang, "A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing," Neural Computing and Applications, vol. 26, no. 6, pp. 1297-1309, 2014.
[6] Y. M. a. D. T. X. Song, "A Load Balancing Scheme Using Federate Migration Based on Virtual Machines for Cloud Simulations," Mathematical Problems in Engineering, pp. 1-11, 2015.
[7] X. Song, Y. Ma and D. Teng, "Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers," Future Generation Computer Systems, vol. 28, no. 2, pp. 358-367, 2012.
[8] Chaudhary, A. Bhadani and S., "Performance Evaluation of Web Servers using Central Load Balancing Policy over Virtual Machines on Cloud," Proceedings of the Third Annual ACM Bangalore Conference ACM no. 16, pp. 1-5, 2010.
[9] Wenhong Tian, Yong Zhao, Yuanliang Zhong, "A dynamic and integrated load-balancing scheduling algorithm for Cloud datacenters," IEEE International Conference on Cloud Computing and Intelligence Systems, pp. 311-315, 2011.
[10] Wenhong Tian, Minxian Xu, Yu Chen, "A new paradigm for the load balance of virtual machine reservations in data centers," IEEE International Conference on Communications (ICC), pp. 4017-4022, 2014.
[11] A. Singh, M. Korupolu and D. Mohapatra, "Server-storage virtualization: integration and load balancing in data centers," Proceedings of the 2008 ACM/IEEE conference on Supercomputing, p. 53, 2008.
[12] Jinhua Hu, Jianhua Gu, Guofei Sun, Tianhai Zhao, "A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment," 3rd International Symposium on Parallel Architectures, Algorithms and Programming, pp. 89-96, 2010.
[13] Wei-Tao Wen, Chang-Dong Wang. De-Shen Wu, "An ACO-Based Scheduling Strategy on Load Balancing in Cloud Computing Environment," Ninth International Conference on Frontier of Computer Science and Technology, IEEE, pp. 364-369, 2015.
[14] Sanjay K. Dhurandher, Mohammad S. Obaidat, Isaac Woungang, Pragya Agarwal, Abhishek Gupta, Prateek Gupta, "A Cluster-Based Load Balancing Algorithm in Cloud Computing," IEEE ICC 2014 - Mobile and Wireless Networking Symposium, pp. 2921-2925, 2014.
[15] Hui Zhang, Guofei Jiang, Kenji Yoshihira, and Haifeng Chen, "Proactive Workload Management in Hybrid Cloud Computing," IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, vol. 11, no. 1, pp. 90-100, March 2014.

[16] Michael Pantazoglou, Gavriil Tzortzakis, Alex Delis, "Decentralized and Energy-Efficient Workload Management in Enterprise Clouds," IEEE Transactions on Cloud Computing, vol. 4, no. 02, pp. 196-209, April-June 2016.
[17] Matthias Sommer, Michael Klink, Sven Tomforde, J¨org H¨ahner, "Predictive Load Balancing in Cloud Computing Environments based on Ensemble Forecasting," IEEE International Conference on Autonomic Computing, pp. 300-307, 2016.
[18] Alireza Sadeghi Milani, Nima Jafari Navimipour, "Load balancing mechanisms and techniques in the cloud environments," Journal of Network and Computer Applications, vol. 71, pp. 86-98, 2016.
[19] Stefano Sebastio, Antonio Scala, "A Workload-Based Approach to Partition the Volunteer Cloud," IEEE Conference on Collaboration and Internet Computing, pp. 2010-2018, 2015.
[20] Xiaomin Zhu, Ji Wang, Hui Guo, Dakai Zhu, "Fault-Tolerant Scheduling for Real-Time Scientific Workflows with Elastic Resource Provisioning in Virtualized Clouds," IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, vol. 27, no. 12, pp. 3501-3517, December 2016.
[21] M. Xu, W. Tian and R. Buyya, "A Survey on Load Balancing Algorithms for Virtual Machines Placement," Wiley InterScience, pp. 1-22, Feb 2017.
[22] Suguna R, Divya Mohandass, Ranjani R, "A novel approach for Dynamic Cloud Partitioning and Load Balancing in Cloud Computing Environment," Journal of Theoretical and Applied Information Technology, vol. 62, no. 3, pp. 662-667, 2014.
[23] Abhay Kumar Agarwal, Atul Raj, "A New Static Load Balancing Algorithm in Cloud Computing," International Journal of Computer Applications, vol. 132, no. 2, pp. 13-18, 2015.