ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 9(48), DOI: 10.17485/ijst/2016/v9i48/96312, November 2016* ISSN (Online) : 0974-5645

Time stamp based Stateful Throttled VM Load Balancing Algorithm for the Cloud

Ansuyia Makroo* and Deepak Dahiya

School of Engineering and Technology, Ansal University, Gurgaon - 122003, Haryana, India;
ansuyia@gmail.com, deepakdahiya@ansaluniversity.edu.in

Abstract

Objectives: One of the essential requirements for improving Quality of Service of Cloud is to design load balancing algorithms that equally spread the load among the VMs such that neither of them is overloaded at any given point of time while ensuring that each of the VM is optimally utilised. The objective of this paper is to propose an efficient VM load balancing policy called Timestamp based Stateful Throttled VM load balancing algorithm. **Methods/Statistical Analysis:** The proposed algorithm deals with the space performance constraints of the existing Stateful Throttled VM Load balancing algorithm. It is based on the principle of locality of reference and deletes the old state information to reclaim space. The authors have carried out experimental analysis to compare the proposed algorithm with the existing algorithms. **Findings:** The comparative analysis shows that the algorithm behaves in the same manner as Stateful Throttled Algorithm while taking into consideration its space performance limitations. The authors have observed better performance in terms of response time and data center processing time for Timestamp based Stateful Throttled VM load balancing algorithm than Throttled VM load balancing algorithm in case of spatially distributed VMs. **Application/Improvements:** The salient features of the proposed algorithm are better response time and data center processing time in case of spatially distributed data centers in cloud.

Keywords: Cloud Computing, Data Center, Load Balancing, Virtualization, Virtual Machine

1. Introduction

Cloud computing has drawn extensive attention from both academia and industry for its flexible and on demand computing services[1]. Cloud is a computing paradigm that combines technologies such as Service-Oriented Architecture[2] (SOA), Abstraction and Virtualization to provide computing as a service to its growing customer base. Cloud computing enables users to access dynamically scalable and virtualized[3] computing resources over Internet. A Cloud consists of pool of data centers also known as nodes clubbed into clusters. Each node is partitioned virtually into Virtual Machines (VMs) to maximise the resource utilization of the underlying infrastructure. The virtualised cloud architecture is abstracted from its clients. Managing and providing computational resources to user applications is one of the main challenges for cloud providers. When users of cloud submit their execution jobs known as cloudlet to the cloud, it is the responsibility of Service broker to allocate an appropriate cluster to a cloudlet. The VM load balancer further allocates an appropriate VM[4] on a node in the allocated cluster for processing.

The increasing demand for infrastructure-less computing has given boost to the cloud computing industry. Considering the increasing number of cloud providers and growing competition among them, it is necessary to focus research on improving the QoS of Cloud. A client outsourcing computing requirements to Cloud expects the cloud to guarantee response time and processing time of the execution jobs on Cloud to be comparable to in-house data centers. Considering the increasing dependency of businesses on Cloud, it is necessary to design load balancing algorithms that

* *Author for correspondence*

equally spread the load among the VMs such that neither of them is overloaded at any given point of time while ensuring that each of the VMs is optimally utilised.

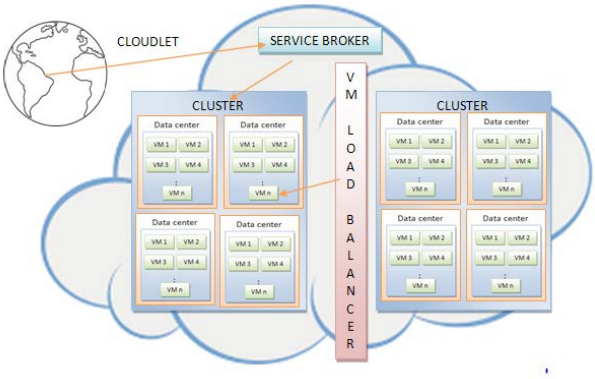

Figure 1. Generalized architecture of Cloud.

In order to save the state information of last VM allocated to a Userbase, State preserving[1] Virtual Machine Load balancing policies use a hashmap. This state information in the hashmap is used for lookup in the subsequent requests from the Userbase. If hashmap lookup returns the state information of the last VM allocation and if the particular VM is available, then there is no need to run the VM allocation algorithm, which significantly reduces the time complexity of the VM allocation. However the hashmap used in the state preserving algorithms has limitations in terms of space performance constraints. The authors have proposed the Timestamp based Stateful Throttled (TST) VM load balancing algorithm that prescribes a mechanism to deal with the space performance constraints of a hashmap used in the existing Stateful Throttled VM Load balancing algorithm.

The rest of the paper is organized as follows: Section 1 introduces the proposed work. Section 2 and 3 include motivation and review of the available Cloud VM load balancing algorithms. Section 4 describes the proposed algorithm i.e., Timestamp based Throttled VM Load Balancing algorithm in detail. Section 5 includes the experimental setup for testing and comparative analysis of the algorithm. This section further gives experimental and analysed results. Finally, Section 6 and 7 summarize the conclusion and the future scope of work.

2. Motivation and Problem Definition

The wide adoption of the cloud computing in IT industry has revolutionized the way industry uses computing resources. To meet the SLAs promised by cloud provider to users, the biggest hurdle for cloud developer is the efficient utilization of VMs in such a way that none of them are overloaded or under loaded. It is an important task to allocate the userbase requests to the VMs on the shared infrastructure while ensuring the optimal response time and processing time of the requests. The existing state preserving[6,7] Throttled VM load balancing algorithm uses a hashmap to store the state of previous allocation of a Userbase request to a VM in memory for faster access and retrieval. The hashmap is an in-memory data structure that stores the key, value pairs (Userbase, VM). The hashmap with open addressing[8] has been chosen for the given scenario of Throttled load balancing algorithm as it the most efficient data structure in terms time complexity. The time complexity of both insertion and retrieval operations for the hashmap is O(1). However, with the growing userbase, the limited in-memory userbase is an obstacle. Also, the increasing load factor degrades the performance of a hashmap.

The limitations of State preserving Throttled VM load balancing algorithm, leads us to the following problem definition:

"To develop a Timestamp based Stateful Throttled VM load balancing algorithm for Cloud that efficiently deals with the performance issues in the existing Stateful Throttled VM load balancing algorithm and carry out comparative analysis of the proposed algorithm with the existing algorithms."

3. Related Study

Cloud computing architecture is based on delivering computing resources like Infrastructure, Platform and Software[9] as a service to the clients on pay per use basis. The popular cloud providers including Google, Microsoft, Yahoo, and IBM are working on improving its QoS and resource utilisation[10]. The increasing research focus on these areas is to attract more customers in the environment of growing competition and maximise

their earnings. One area that impacts the performance of cloud and hence requires increased research focus is load balancing which deals with distributing load of the user request workloads among the underlying resources in the cloud computing environment. Cloud physical resources are virtually partitioned into Virtual machines. The user jobs are deployed on the virtual machines in such a way that the underlying shared infrastructure details as well the deployment details are abstracted from the user giving them an illusion that the application is running in isolation. However in the environment of resource contention, it is necessary that the jobs are deployed in a manner that maximises the throughput and minimises the response times. Currently, there are a number of cloud load balancing algorithms available. One of the most commonly used load balancing algorithm that allocates homogenous VMs to the user request in Round Robin fashion is the Round Robin load balancing algorithm. A variant of the Round Robin load balancing algorithm that is suitable for heterogeneous VMs is Weighted Round Robin Algorithm in which the weight of each VM is proportional to the size of the resources like CPU, RAM etc. The number of requests deployed on a VM are proportional to its weight. Another variant of Round Robin algorithm called Round Robin with server affinity uses the state information of the previous allocation of a Userbase request for subsequent requests. The Active Monitoring Load Balancer balances the load on the available VMs in a way that balances out the number of active tasks on each VM at any given point of time. Honeybee Foraging Algorithm draws its inspiration from the behaviour of honey bee foraging strategy[11–13]. This algorithm is useful in the case of heavily loaded VMs as it considers the prioritisation of tasks that have been removed from such VMs. Min-Min algorithm is based on relative prioritisation of unassigned jobs in the order of minimum completion times. It initially selects the job with least minimum completion time and assigns it to the node that produces the minimum completion time for the jobs and continues the process till all the unassigned jobs are allocated VMs. The Min-Min algorithm has a disadvantage that it may cause starvation of large jobs[14]. Another VM load balancing algorithm that is based on relative prioritisation of unassigned jobs and aims to reduce the waiting times of large jobs is Max-Min. However this algorithm, works in the reverse fashion i.e., the node with minimum completion time is assigned the job with the overall maximum completion time. This process is repeated until all the unassigned tasks are assigned. The Fuzzy-based load balancing algorithms migrates the jobs to another VM based on high load status of a VM in a heterogeneous cloud environment. Throttled load balancer ensures that at a given point time, maximum numbers of user requests allocated to a VM are less than or equal to a predefined threshold value. In case all the VMs are allocated maximum requests, then further requests are queued until one of them becomes available. A variant of Throttled Load Balancer that uses the state of previous allocation of a cloudlet to determine the next VM for the incoming cloudlets from the same userbase is Stateful Throttled Load Balancer. Load balancing in cloud has become a key area of research focus in academia and industry. However not all cloud researches get the opportunity to test their applications and algorithms on real cloud test bed. In order to deal with the limitation, a number of cloud simulation tools have been developed that can be used by researchers to test their applications and algorithms. CloudSim is one such simulator that supports simulation of virtualized cloud environment with data center partitioned into VMs and multiple data centers integrated to form a cluster. It allows the user to design and develop service broker algorithms and load balancing for mapping user requests to cluster and finally the VM respectively. Another simulator that extends the features of CloudSim is CloudAnalyser. It has an easy to use Graphical User Interface that makes simulation easy for the users. CloudAnalyser provides support for running multiple simulations and carrying out comparative analysis of the algorithms while abstracting the architecture details from the user. The simulator, GreenCloud[15] helps analyse the energy consumption of the distributed environments. NetworkCloudSim[16], another cloud simulator simulates data centers of the Cloud and generalized applications such as HPC, e-commerce and workflows. Out of all the above simulators, CloudAnalyser is the most suitable to analyze the proposed VM load balancing algorithm in different scenarios as it provides users with the capability to modify and test their algorithms with the help of user friendly GUI (Graphical User Interface). This motivated the authors to use CloudAnalyser for testing and comparison of the VM load balancing algorithm.

4. Proposed Algorithm: Timestamp based Throttled VM Load Balancing Algorithm

The scalable and elastic cloud model gives an illusion of infinite resources. However in reality the cloud has limited resources. The load on the cloud VMs varies from time to time based on active incoming cloudlets deployed on it from the userbase. At peak traffic times, there is a possibility that some of the VMs may be overloaded in the shared cloud environment. In such cases, the processing requirements of VM exceed capacity of resources that are available on it leading to poor performance and even failure of the cloudlets deployed on it. However, the cloud providers are obligated to meet the agreed SLAs and the failure of the cloudlets is unacceptable. Throttled VM load balancing algorithm is based on the strategy to define the threshold number of cloudlets that can be deployed on a VM at a given point of time. We have used two terms to explain the algorithm: Capacity and Load. The capacity of VM is the threshold value of Cloudlets that can be assigned to it at a given point of time. The Load of VM is the number of cloudlets currently assigned to it. A VM is available if the load is less than capacity else it is busy. When an incoming request arrives at Throttled Load balancer, it assigns the next sequentially available VM. The Stateful Throttled VM load balancing algorithm extends the Throttled VM load balancer by using the state of userbase request allocation to determine the VM to be allocated to the userbase request in future. The state information is stored in a Userbase Table based on hashmap with open addressing. A hashmap with open addressing is used to store the state information as it has a constant time complexity thereby decreasing the overall time complexity of the load balancing algorithm. However one of the major limitation of using a hashmap is the trade-off between the time and the space performance. The Stateful Throttled VM load balancing algorithm maintains a Userbase Table that has a linear space complexity. Considering the growing number of cloud clients, a constant sized hashmap is not feasible to store the Userbase state information. The size of the hashmap may pose a limitation to store the entries for the Userbases. Also, another important aspect to consider when using hashmap is the load factor of a hashmap. Load Factor is the ratio of the number of items in a hashmap to the table size. As the load factor increases, probe lengths grow longer. In open addressing the performance degrades badly as the load factor approaches 2/3.

The relationship between probe length (P) and load factor (L) for linear probing is given in the below equations.

For a successful search it is

$P = (1 + 1 / (1-L)^2)/2$

and for an unsuccessful search it's

$P = (1 + 1 / (1-L)) / 2$

Considering the case when the load factor is 1/2 i.e., half of the hashmap is full, a successful search requires 1.5 comparisons and an unsuccessful search requires 2.5 comparisons on an average. When the load factor increases to 2/3, the corresponding numbers rise to 2.0 and 5.0. Thus the increase in load factor leads to increase in search time. For a good performance, the load factor should be below 2/3. However, for a given amount of data, more memory needs to be allocated to the hashmap for a lower load factor. There is trade-off between the space and the time complexity of a hashmap and the optimum load factor depends on the requirement.

In order to deal with the trade-off between increasing space complexity and diminishing performance with increasing load factor, the authors have proposed the Timestamp based Throttled (TBT) VM load balancing algorithm. The Stateful Throttled VM load balancing algorithm is based on the principle of locality of reference. That is, at a given point of time, cloudlet generation is concentrated mainly to the same Userbase. The same principle is used in the case TBT VM load balancing algorithm, that uses a hashmap with threshold load factor of 2/3. It stores the timestamp for each entry in Userbase table and deletes the old entries from the hashmap when the load factor reaches the threshold value thus reclaiming space. The algorithm utilizes two the tables to store the state information:

- **Userbase Table**: Userbase table consists of two entries <*Userbase Id, VM id, Timestamp*>. The data structure used to implement a Userbase Table in this scenario is a hashmap which stores the key values <*Userbase id, (VM id, timestamp)*> such that the VM id corresponding to the key Userbase id can be retrieved and updated in the constant time.

An additional entry timestamp is used to store the instant of time of last access.

The timestamp is entered initially when the entry for Userbase in stored in the userbase table and updated on every access to the entry.

- **VM Status Table**: VM state table stores the current status of the VM i.e., Busy or available. A VM is available if the load is less than the capacity of the VM, else it is busy. To implement the VM state table, the ideal data structure that can be used is hashmap which stores the key value pairs <*VM id, Status*>. The VM status table will be able to retrieve and update the allocation status for a VM in constant time.

The algorithm for the TBT VM load balancer and the flowchart is given in Figure 2 and Figure 3 respectively. When a cloudlet from a userbase comes to a TBT VM load balancer, it searches the hashmap for its entry. If the entry is present, then it checks the status of the corresponding VM in the VM Status Table. If the VM Status Table shows that the VM is available, then the VM is assigned to the Userbase cloudlet and timestamp of the corresponding Userbase entry is updated. However if there is no entry for the Userbase in hashmap or the entry is present and the VM is busy, then the throttled algorithm needs to be run to find the next available VM. After the VM is allocated to it, the state information is stored in Userbase table. However in case the load factor of the hashmap is greater than 2/3 the space reclaiming algorithm needs to run that deletes the old entries with lesser timestamps.

Calculating the effective VM allocation time to a request from a given Userbase

The effective allocation time for Timestamp based Stateful Throttled algorithm is same as Stateful throttled algorithm.

Effective allocation time = (t1+t2)*(p*k) + (t1+t3)*(1-p) + (t1+t2+t3)*(p)*(1-k)

Where,

t1 = Time taken for searching an entry of a Userbase in Userbase hashmap

t2 = Time taken for checking the status of a VM in VM state list

t3 = Time taken for processing request using Throttled VM Load balancing algorithm

```
TBT_ Algorithm ( )
{
Initialize Userbase table with no entries;
Initialize VM Status Table with entries for all VM status as Available;
While (Cloudlets from a Userbase are received by Data Center Controller)
do
      {
      Search the Userbase table for an entry <Userbase id, VM id> of the given Userbase;
      if (Userbase table contains an entry for the Userbase)
            {
            Search for the corresponding VM entry <VM id, status> in VM Status Table;
            If (VM status = = Available)
                  {
                  Update the timestamp entry in the<Userbase, VM id, timestamp> in Userbase table;
                  Update the VM Status Table;
                  Continue;
                  }
            }
      Run the Throttled algorithm to assign an appropriate VM to it;
       if (Userbase table is full)
                {
                Delete the old entries from Userbase table;
                }
     Update the< Userbase, VM id, timestamp > entry in the Userbase table;
     Update the VM Status Table;
     }
}
```

Figure 2. The algorithm for TBT VM load balancer.

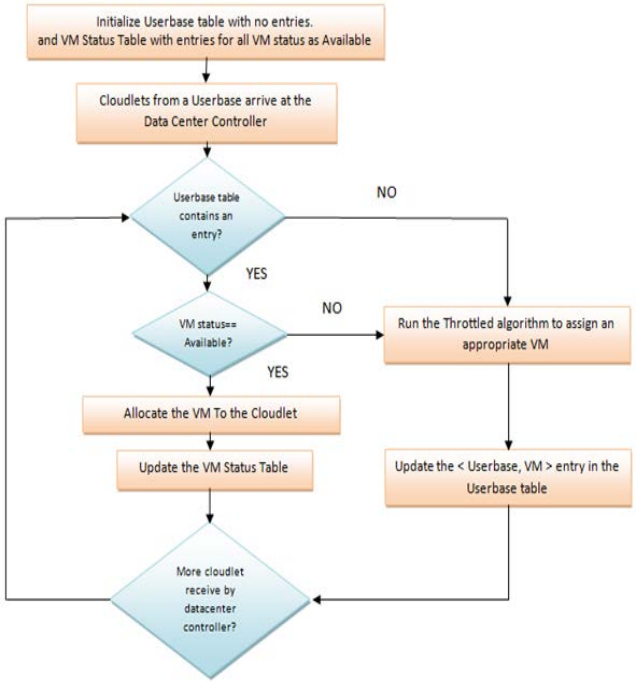

Figure 3. The flow chart for TBT VM load balancer.

t4 = Time taken for deleting old entries from the hashmap
p = Probability of finding an entry for the requesting Userbase ; p (0<=p<=1)
k = Probability of finding the corresponding VM status as available in VM state list

The time for deleting and storing the entries in hashmap is not included in the effective allocation time because this deletion and storage of entries happens in the background after the VM has been allocated to the Userbase.

Let t4 = Time for deleting old entries from the hashmap
t5 = Time for storing entries in a hashmap
m = Probability of finding the hashmap full which triggers deletion of old entries
Thus,

Storage time of the entries in the hashmap when it is full = t4 +t5

Storage time of the entries in the hashmap when it is not full = t5

$$\text{Hence, effective storage time} = m * (t4 + t5) + (1-m) * t5$$
$$= mt4 + mt5 + t5 - mt5$$
$$= mt4 + t5$$

Effective storage time, T= mt4 + t5

Consider the case when the size of hashmap is greater or equal to the number of Userbases. In that case, the hashmap would never be full i.e., m = 0.
In this case, effective storage time = t5
Also, $m \propto$ (Size of hashmap -Size of Userbase table)

$T \propto (1/($Size of hashmap -Size of Userbase table)

Hence, there is trade-off between the hashmap size and the effective storage time.

5. Experimental Setup

To develop and deploy the TST load balancing algorithm, the authors have used the simulator CloudAnalyser. The approach used for testing is in line with testing of Stateful throttled algorithm. One of the most popular social networking sites, Facebook can benefit by moving to cloud. So we have used the data of Facebook[17] for analysis of the performance and comparative analysis of the proposed algorithm.

Comparative Analysis: Throttled VM Load Balancer and Timestamp based Stateful Throttled VM Load Balancer.

The following parameters are used to analyze the performance and comparative analysis of the proposed algorithm:

- Overall Average Response Time (in ms).
- Overall Average Datacenter processing time (in ms).

We consider the scenario of homogeneous VM which are distributed across data centers. The numbers of data centers are increased to study the behaviour of the proposed algorithm with respect to the Social networking web application deployed on distributes data centers as given in Table 1.

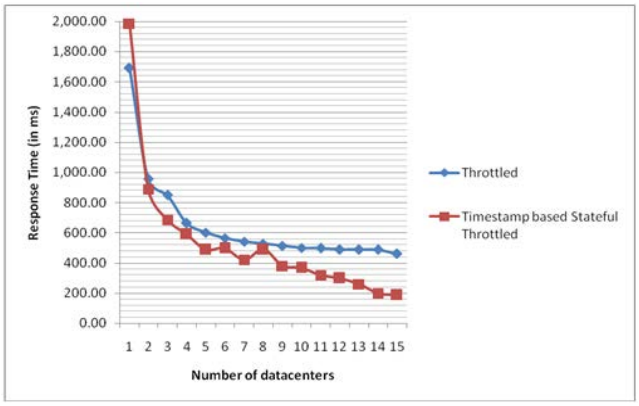

Figure 4. Graph showing comparative analysis of overall average response time.

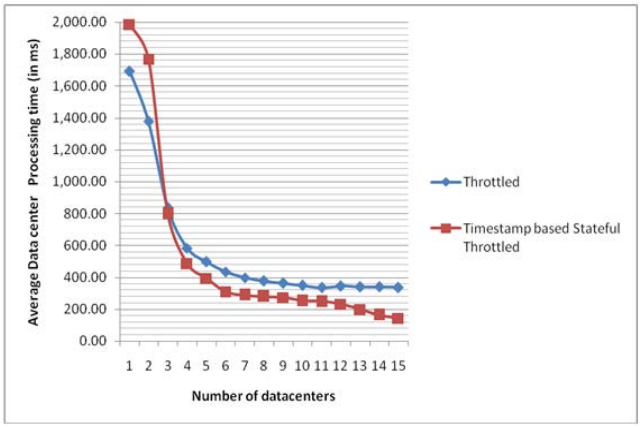

Figure 5. Graph depicting comparative analysis of the overall average data center processing time.

The comparative analysis is summarized in Table 1 and Figure 4 , 5 is given:

The comparative analysis shows that the algorithm behaves in the same manner as Stateful Throttled Algorithm while taking into consideration limitation of same. The overall average response time and overall average datacenter processing time decreases considerably on increasing the number of spatially distributed data centers. The authors have observed better overall average response time and data center processing time for TST algorithm Throttled than Throttled VM load balancing algorithm in case of spatially distributed VM.

To conclude, the experimental analysis shows that the proposed VM load balancing algorithm i.e., TST VM load balancing algorithm performs better than Throttled VM load balancing algorithm in terms of in terms of Overall average response time and Overall average datacenter processing time for spatially distributed VMs.

6. Conclusion

The emergence of cloud computing as an economic and viable solution to the computing needs of enterprises is the reason for its immense popularity. So it is very crucial for the industry and academia to focus on building more reliable and robust cloud infrastructures. One such area that requires focus is the performance unpredictability in the cloud. A cloud consists of pool of data centers also known as nodes clubbed into clusters and each node is virtually partitioned into Virtual Machines (VMs). The user requests are deployed on the VMs. The user request should be deployed in such a way that maximise the resource utilization of the underlying infrastructure

Table 1. Overall comparative results

Scenario (DC with 100 VMs each)	Average Response Time (in ms)		Average Data center Processing time (in ms)	
	Throttled VM load balancing algorithm	Timestamp based Stateful Throttled VM load balancing algorithm	Throttled VM load balancing algorithm	Timestamp based Stateful Throttled VM load balancing algorithm
1 Data center (DC)	1,693.08	1,986.13	1378.13	1,768.80
2 Data centers	955.86	888.90	835.36	799.06
3 Data centers	849.17	686.23	583.44	487.32
4 Data centers	663.82	590.67	497.23	394.11
5 Data centers	602.10	489.62	436.45	310.54
6 Data centers	564.65	498.78	399.14	290.65
7 Data centers	542.23	420.79	377.22	280.20
8 Data centers	527.98	490.12	363.11	273.33
9 Data centers	514.70	380.55	350.00	253.78
10 Data centers	500.25	368.05	335.70	249.19
11 Data centers	499.02	320.12	345.89	231.89
12 Data centers	489.78	298.58	340.45	199.54
13 Data centers	489.56	260.48	339.89	164.23
14 Data centers	487.54	198.89	337.25	143.76
15 Data centers	460.89	188.98	331.58	129.56

while ensuring that none of the VMs are over utilised. This is the task of Load balancing algorithms. This paper discusses the Timestamp based Throttled VM load balancing algorithm that improves the existing Throttled VM Load balancing algorithm by dealing with its space performance limitations. The TBT VM load balancing algorithm uses a hashmap with threshold load factor of 2/3. It stores the timestamp for each entry in Userbase table and deletes the old entries from the hashmap when the load factor reaches the threshold value thus reclaiming space. The major contribution of the paper is that authors have proposed an efficient VM load balancing algorithm for the cloud. The authors have carried out a comparative analysis of the proposed algorithm with the existing algorithms. The salient features of the proposed algorithm are better Overall Average Response Time and Overall Average Data center processing time in case of spatially distributed data centers in cloud. The testing of the proposed algorithm has been done by deploying the algorithm on a cloud simulator. So the results may vary in case of real cloud environment. The results obtained using simulator serve as a basis for the comparison and analysis of the proposed work. The authors plan to test the proposed algorithm on a private cloud environment set up using Eucalyptus. Secondly, homogeneous VMs have been assumed to test the proposed algorithm. The working of the proposed algorithm has not been studied in case of heterogeneous VMs. The authors plan to test the proposed work on heterogeneous VMs also. The authors have not taken into account the fault tolerance in the data center. However, in future they plan to incorporate fault tolerance mechanisms in the proposed VM load balancing algorithm.

7. References

1. Mahajan K, Makroo A, Dahiya D. Round robin with server affinity: A VM load balancing algorithm for cloud based infrastructure. Journal of Information Processing Systems. 2013; 9(3):379–94.
2. Foster I, Zhao Y, Raicu I, Lu S. Cloud computing and grid computing 360-degree compared. In Grid Computing Environments Workshop; Austin, TX. 2008. p. 1–10.
3. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M. A view of cloud computing. Communications of the ACM. 2010 Apr 1; 53(4):50–8.
4. Makroo A, Dahiya D. A systematic approach to deal with noisy neighbour in cloud infrastructure. Indian Journal of Science and Technology. 2016 May 31; 9(19):1–9.
5. Shen Z, Subbiah S, Gu X, Wilkes J. Cloudscale: Elastic resource scaling for multi-tenant cloud systems. Proceedings of the 2nd ACM Symposium on Cloud Computing ACM; USA. 2011 Oct 26. p. 5.
6. Zhang Q, Cheng L, Boutaba R. Cloud computing: State-of-the-art and research challenges. Journal of Internet Services and Applications. 2010 May 1; 1(1):7–18.
7. Mahajan K, Dahiya D. A cloud based deployment framework for load balancing policies. IEEE 7th International Conference on Contemporary Computing (IC3); Noida. 2014 Aug 7. p. 565–70.
8. Girbea A, Suciu C, Nechifor S, Sisak F. Design and implementation of a service-oriented architecture for the optimization of industrial applications. IEEE Transactions on Industrial Informatics. 2014 Feb; 10(1):185–96.
9. Buyya R, Yeo CS, Venugopal S. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. 10th IEEE International Conference on High Performance Computing and Communications, HPCC'08; Dalian. 2008 Sep 25. p. 5–13.
10. Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson DA, Rabkin A, Stoica I, Zaharia M. Above the clouds: A berkeley view of cloud computing [Technical report]. University of California; 2009. p. 1–12.
11. Nuaimi AK, Mohamed N, Nuaimi AM, Al-Jaroodi J. A survey of load balancing in cloud computing: Challenges and algorithms. IEEE 2nd Symposium on Network Cloud Computing and Applications (NCCA); London. 2012 Dec 3. p. 137–42.
12. Lee R, Jeng B. Load-balancing tactics in cloud. IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC); Beijing. 2011 Oct 10. p. 447–54.
13. Wang L, Von Laszewski G, Younge A, He X, Kunze M, Tao J, Fu C. Cloud computing: a perspective study. New Generation Computing. 2010 Apr 1, 28(2), pp. 137-46.
14. Wu MY, Shu W, Zhang H. Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems. Heterogeneous Computing Workshop; Cancun. 2000 May 1. p. 375–85.
15. Kliazovich D, Bouvry P, Khan SU. Green Cloud: A packet-level simulator of energy-aware cloud computing data centers. The Journal of Supercomputing. 2012 Dec; 162(3):1263–83.
16. Garg SK, Buyya R. Networkcloudsim: Modelling parallel applications in cloud simulations. IEEE 4th Utility and Cloud Computing (UCC); Victoria, NSW. 2011. p. 105–13.
17. Garg SK. NetworkCloudSim: Modelling parallel applications in cloud simulations International Conference on IEEE; Victoria, NSW. 2011 Dec 5. p. 105–13.