Research Article

Load Allocation of Virtual Machines based on Enhanced Throttled Load Balancing Algorithm

[1]Vandana Sivaraj, [2]A. Kangaiammal

[1]Part-Time Ph.D. Research Scholar,
Govt. Arts College(Autonomous), Salem-7
Periyar University
Salem ,India
vandansivaraj369@gmail.com
[2]Assistant Professor of Computer Applications,
Govt. Arts College (Autonomous), Salem-7
Periyar University
Salem ,India
indurath2007@gmail.com

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 16 April 2021

Abstract: Today's world is fully dependent on the cloud services and it has become more than mandatory to prevent the faults that occur during the process. Cloud computing has grown tremendously in the past few years and the demand for its services are also drastically increasing all over the world. Virtualization technology is the base for cloud computing. Virtual machines are provisioned during virtualization and care should be taken to handle the faults that arise during this process. To facilitate the proper functioning of the Virtual Machines (VMs), the proposed work uses the technique of an Enhanced Throttled Load Balancing Algorithm to efficiently allocate load to the VMs to the particular task and proactively handling faults. Cloud analyst, a GUI simulator is used to compare and analyse the performance and load balancing factors during virtualization and allocate load to the VM based on the outcome.

Keywords: Load Allocation, Load Balancing, Fault Tolerance,Virtualization, Cloud Computing, Cloud Analyst.

I. INTRODUCTION

According to NIST, Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.[1] Since cloud computing has become the need for all operations of todays' world, care should be taken towards maintaining its security and authenticity by reducing its faults. It is a known fact that Virtualization is the backbone of cloud computing and is one of the foundational elements of cloud computing technology that helps utilize the capabilities of cloud computing to the full.

The fragmentation of our hard drive into different drives is the best example of how virtualization works in our daily life. While we may have only one hard drive, our system sees it as two or more different and separate segments. Virtualization is a similar technology which started as the ability to run multiple operating systems on one hardware set and now it has become an important part of testing and cloud-based computing.

Virtualization Architecture is presented in Figure 1. In general, it has been seen that operating system acts as an interface between the physical machine or the hardware and the applications. Virtualization is a process where one can create multiple virtual machines with different operating systems to be run on each of the machine. But there is only one physical machine. Therefore, during virtualization, Hypervisor acts as a layer of interface between the operating systems (i.e.Virtual Machines) and the physical machine.

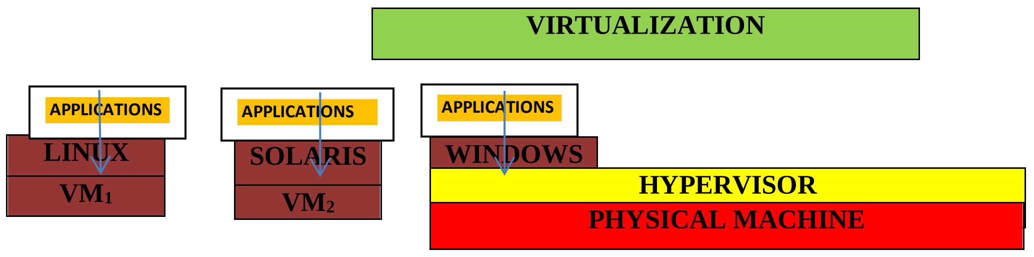


Figure 1: Architecture of Virtualization

II. LITERATURE REVIEW

In order to protect the services offered by the cloud from fault, provisions have to be done to save the virtual machines from faults. This can be achieved by many of the existing algorithms which prevent faults from occuring either in a proactive way or a reactive way. Many proactive techniques such as Software Rejuvenation, Self-healing and Pre-emptive Migration have been proposed to effectively handle faults in a proactive manner. Techniques like Replication, Retry, Sgaurd and Resubmission are some of the proposed techniques that try to avoid the faults in a reactive manner. [2]

As the basic idea of cloud computing is to provide resources such as VMs, as services on demand, during this process it becomes mandatory that the Virtual machines are also free from faults. Although the same techniques of proactive and reactive fault tolerances of cloud computing seems good for the VMs also. The most commonly used technique is the Load Balancing (LB) algorithm which takes care of balancing of loads of different VMs when task is allocated to them. Allocating efficient VM on demand is being carried out with the help of the various Load Balancing algorithms.[12]

There are a number of techniques proposed for Load balancing during virtualization. The basic categories underlying Load Balancing technique based on the state of the machines are Static, Dynamic and Hybrid. Static Load balancing techniques has the benefit of division of load amongst machine in a uniform way whereas this technique has its own disadvantages. Though the complexity of implementation is simple, it is always hard to move the scheduled tasks to different VMs when the process in being executed. [13]

On the other hand, Dynamic load balancing takes into consideration the current state of system and has capacity to deal with unpredictable processing load. The advantage of dynamic load balancing is that tasks can move dynamically from an overloaded machine to under-loaded one but are much complex in nature and much complicated to design compared to static LB algorithms. [4]

III. METHODOLOGY

This paper proposes to work on the concept of load allocation and load balancing during virtualization by bringing in various cases which require provisioning of VMs with different requirement. This paper is based on the previous study relating to an open source initiative Liquid Galaxy which requires provisioning of multiple VMs to simulate panoramic view of a landscape or any other display.[9]

In [9], a proposal was made to identify the size of memory needed for the proper simulation of the display in multiple VMs. A master machine with the necessary size was first created using Virtual Box and then one Slave machine was created by studying the required size necessary for simulation of that particular software such as the Google earth. It was then concluded that if the size is same in all VMs, then provisioning of all the VMs will depend on the size of one VM, ie., the first slave VM created. All the remaining VMs can directly use that size which will tolerate any fault that would happen if this allocation was not made. This was a proactive way of handling fault in virtual machines. But, the paper concludes with a single scenario of having multiple VMs with the same requirement. The paper states that the future work will handle provisioning of different VMs with different requirement. To handle the imbalance, the VMs have to be applied with some load balancing algorithms in a dynamic manner. For this, there have been algorithms proposed such as the Round Robin Algorithm and Throttled Algorithm and Active Monitoring Load Balancing algorithm. [10]

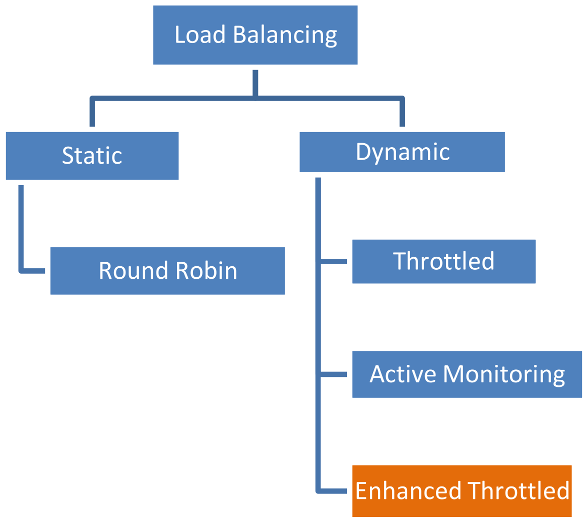


Figure 2: Types of Load Balancing Algorithms

Figure 2 depicts the different types of Load Balancing Algorithms. Out of the three dynamic algorithms, it has been identified that Throttled Algorithm is the best in terms of performance.[6] But, in throttled algorithm also only one job can be assigned to a VM. This means that if the task allocated is not utilizing the VM to its maximum capacity then it cannot be a perfect solution. If the task allocated demands heavy load and if the VM is not going to manage, again it will lead to a fault due to overload. Therefore, in this paper, an algorithm is proposed which is an extension of the Throttled algorithm called as the Enhanced Throttled Algorithm.

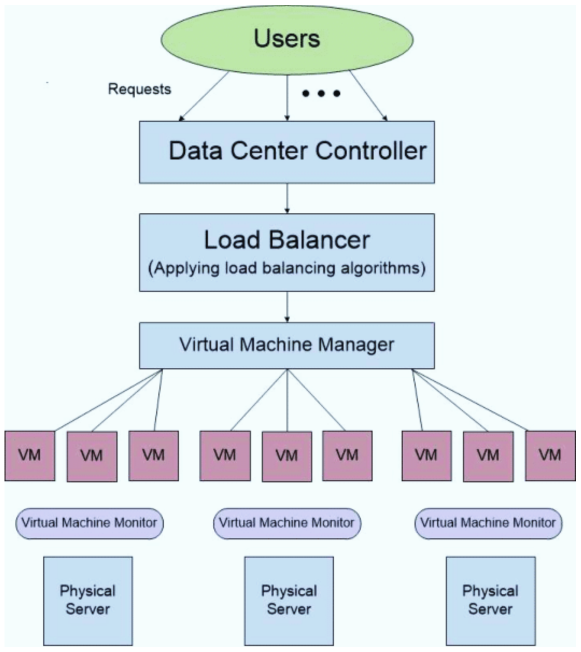


Figure 3: Load Balancer Architecture

Figure 3 presents the Load Balancer Architecture. It is seen from Figure 3 that the Load balancer architecture is well organised and multi-levelled. The user's request is sent to the data center controller and here is where the entire load is initially known. It then seeks the help of a load balancer that has a lot of load balancing algorithms in hand to efficiently distribute load to the VM Managers according to the needs of the applications. Once the calculations are done, the VM managers that manage multiple VMs send over data to be distributed to the different VMs. The VM Monitor takes care of aggregating the tasks from the VMs and sends them to the physical servers.

IV. PROPOSED TECHNIQUE

This paper focuses in proposing a technique on the basis of the various requirements of different VMs. Imbalance resource usage can be observed in cases, such as a VM is running a computation-intensive application while with low memory requirement [5]. An analysis is made and calculation is carried out so that the existing algorithm is modified in such a way that it can handle multiple VMs with varied requirements. The proposed work includes identifying the system which has maximum configuration and which is idle. This will be allocated to the Master VM. For this, the existing 'Throttled Load Balancing' algorithm is used.

In this algorithm, the load balancer maintains an index table of virtual machines as well as their states whether available or busy. First a request is made to the data centre by the client/server to find out a suitable virtual machine (VM) so that the necessary task can be accomplished. The load balancer takes care of the allocation of the VMs. So, the data centre requests the load balancer for VM allocation. Scanning of the index table is done by the load balancer from the top till the first available VM is found or the index table is fully scanned. If the VM is found, the load balancer intimates it to the data centre. The data centre then communicates to the VM with the request identified by the id. Further, with the information from the load balancer, the data centre acknowledges and revises the index table accordingly.

In case if virtual machine is not found, then the balancer returns '-1' for the data center to queue the request. When the virtual machine finishes with the processing of the allocated request then the data center controller receives the response cloudlet and then it sends the notification to balancer to virtual machine for de-allocation [7, 8].

In the proposed model, since the basic requirement of the slave is calculated before hand, instead of allocating the VMs based on the efficient throttled algorithm [3] directly, a check is made to find out if the VM can handle the required load. If yes, it is allotted otherwise it checks with the next VM in the index table. This will ensure one additional layer of fault tolerance, when the load of the VM is not able to support the slave's requirement.

```
Algorithm for Enhanced Throttled Model:
1. Start
2. Find the expected response time.
3. Find the efficient VM with the least load
4. An id with the least load is returned by this algorithm.
5. This id is compared with the required size of the slave VM
6. If the size of the VM returned from the index table is more than the required size, we continue with the
   allocation.
7. Else, the steps 3 to 6 are repeated.
8. Stop
```

Figure 4: Enhanced Throttled Model Algorithm

Figure 4 describes the proposed algorithm. It is ensured that after identifying of the efficient VM, an additional check to identify VM which satisfies the bare minimum request is made and proceeded accordingly.

V. EXPERIMENTAL SETUP OF SIMULATION USING CLOUD ANALYST

To verify the working and the authenticity of the algorithm proposed, a simulator is used to study the behaviour.

Cloud Analyst [10,7] is a Graphical User Interface based tool that is developed on CloudSim architecture. CloudSim is a very popular simulator which helps in modeling, simulation and other experimentation. The main disadvantage of CloudSim is that all the tasks have to be done programmatically. On the other hand, the cloud analyst allows setting location of users that are generating the application and also the location of the data centers. The various parameters that can be set in this configuration include number of users, number of request generated per user in an hour, number of VMs, number of processors, amount of storage, network bandwidth and other parameters that are required. Based on the parameters the tool computes the simulation result and depicts them in graphical form as shown in Figure 5.

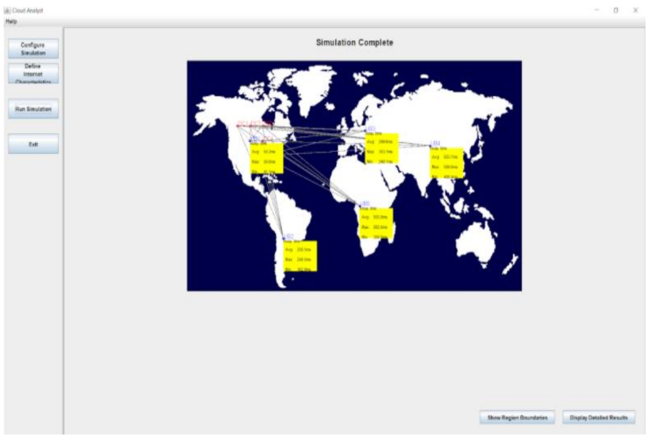
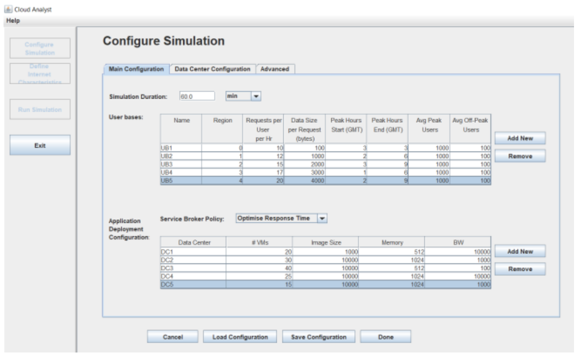


Figure 5: Common Visual Representation of the Configuration

Figure 6 describes the Cloud Analyst common values for Throttled, Round Robin and Enhanced Throttled Model with simulation duration set as 60 minutes with five User Bases and five Data Centers.
The role of cloud analyst is to set the location of users which generate applications and also sets the location of the data centers.
During this process different configuration parameters are set which includes number of users, number of request generated per user per hour , number of VMs, number of processors, amount of storage, network bandwidth and other necessary parameters. The tool calculates and returns the response time, processing time, cost etc as results based on the parameters and displays them graphically.



Configure Simulation

Simulation Duration: 60.0 min

User bases:

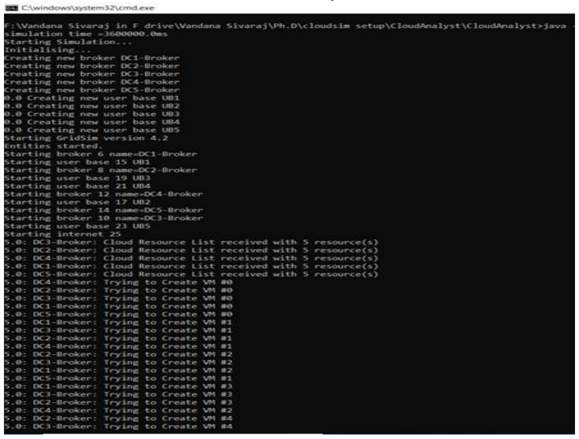
Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
UB1	0	10	100	3	3	1000	100
UB2	1	12	1000	2	6	1000	100
UB3	2	15	2000	3	9	1000	100
UB4	3	17	3000	1	6	1000	100
UB5	4	20	4000	2	9	1000	100

Application Deployment Configuration:

Service Broker Policy: Optimise Response Time

Data Center	# VMs	Image Size	Memory	BW
DC1	20	1000	512	10000
DC2	30	10000	1024	1000
DC3	40	10000	512	100
DC4	25	10000	1024	10000
DC5	15	10000	1024	1000

Figure 6: Cloud Analyst common values for Throttled, Round Robin and Enhanced Throttled Model



```
C:\windows\system32\cmd.exe
F:\Vandana Sivaraj in F drive\Vandana Sivaraj\Ph.D\cloudsim setup\CloudAnalyst\CloudAnalyst>java
simulation time =3600000.0ms
Starting Simulation...
Initialising...
Creating new broker DC1-Broker
Creating new broker DC2-Broker
Creating new broker DC3-Broker
Creating new broker DC4-Broker
Creating new broker DC5-Broker
0.0 Creating new user base UB1
0.0 Creating new user base UB2
0.0 Creating new user base UB3
0.0 Creating new user base UB4
0.0 Creating new user base UB5
Starting GridSim version 4.2
Entities started.
Starting broker 6 name=DC1-Broker
Starting user base 15 UB1
Starting broker 8 name=DC2-Broker
Starting user base 19 UB3
Starting user base 21 UB4
Starting broker 12 name=DC4-Broker
Starting user base 17 UB2
Starting broker 14 name=DC5-Broker
Starting broker 10 name=DC3-Broker
Starting user base 23 UB5
Starting internet 25
5.0: DC3-Broker: Cloud Resource List received with 5 resource(s)
5.0: DC2-Broker: Cloud Resource List received with 5 resource(s)
5.0: DC4-Broker: Cloud Resource List received with 5 resource(s)
5.0: DC1-Broker: Cloud Resource List received with 5 resource(s)
5.0: DC5-Broker: Cloud Resource List received with 5 resource(s)
5.0: DC4-Broker: Trying to Create VM #0
5.0: DC2-Broker: Trying to Create VM #0
5.0: DC3-Broker: Trying to Create VM #0
5.0: DC1-Broker: Trying to Create VM #0
5.0: DC5-Broker: Trying to Create VM #0
5.0: DC1-Broker: Trying to Create VM #1
5.0: DC3-Broker: Trying to Create VM #1
5.0: DC2-Broker: Trying to Create VM #1
5.0: DC4-Broker: Trying to Create VM #1
5.0: DC2-Broker: Trying to Create VM #2
5.0: DC3-Broker: Trying to Create VM #2
5.0: DC1-Broker: Trying to Create VM #2
5.0: DC5-Broker: Trying to Create VM #1
5.0: DC1-Broker: Trying to Create VM #3
5.0: DC3-Broker: Trying to Create VM #3
5.0: DC2-Broker: Trying to Create VM #3
5.0: DC4-Broker: Trying to Create VM #2
5.0: DC2-Broker: Trying to Create VM #4
5.0: DC3-Broker: Trying to Create VM #4
```

Figure 7: Simulation background process started

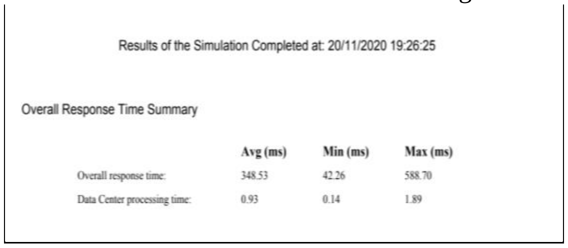
The simulator is fed with common values that are tested for the three models, Round Robin, Throttled and the Enhanced Throttled. Figure 7 shows that the simulation running in the background and also shows that the Cloud Analyst has started the simulation process.

Figure 3 presents the Load Balancer Architecture. It is seen from Figure 3 that the Load balancer architecture is well organised and multi-levelled. The user's request is sent to the data center controller and here is where the entire load is initially known. It then seeks the help of a load balancer that has a lot of load balancing algorithms in hand to efficiently distribute load to the VM Managers according to the needs of the applications. Once the calculations are done, the VM managers that manage multiple VMs send over data to be distributed to the different VMs. The VM Monitor takes care of aggregating the tasks from the VMs and sends them to the physical servers.

VI RESULTS AND DISCUSSION

The simulation results for the three different algorithms Round Robin, Throttled Algorithm and Enhanced Throttled Algorithm are presented in Table 1 through 3 respectively as generated in the Cloud Analyst simulator. The result includes response time, processing time, cost, etc. As GUI interface, after performing various simulation operations, the cloud provider can determine the best way to allocate resources, based on request which data center to be selected and can optimize cost for providing services.

Table 1: Simulation for Round Robin Algorithm

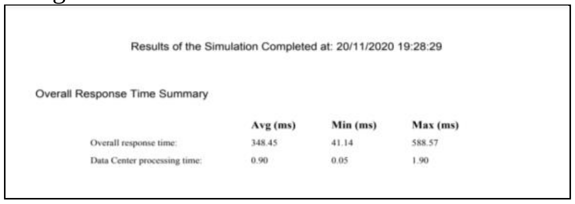


Results of the Simulation Completed at: 20/11/2020 19:26:25

Overall Response Time Summary

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	348.53	42.26	588.70
Data Center processing time:	0.93	0.14	1.89

Table 2: Simulation for Throttled Algorithm

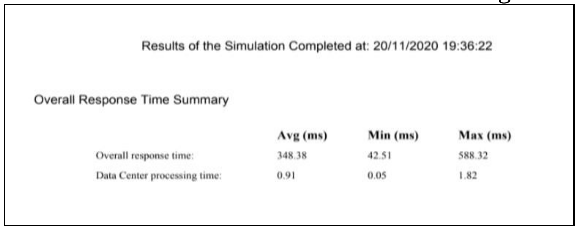


Results of the Simulation Completed at: 20/11/2020 19:28:29

Overall Response Time Summary

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	348.45	41.14	588.57
Data Center processing time:	0.90	0.05	1.90

Table 3: Simulation for Enhanced Throttled Algorithm



Results of the Simulation Completed at: 20/11/2020 19:36:22

Overall Response Time Summary

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	348.38	42.51	588.32
Data Center processing time:	0.91	0.05	1.82

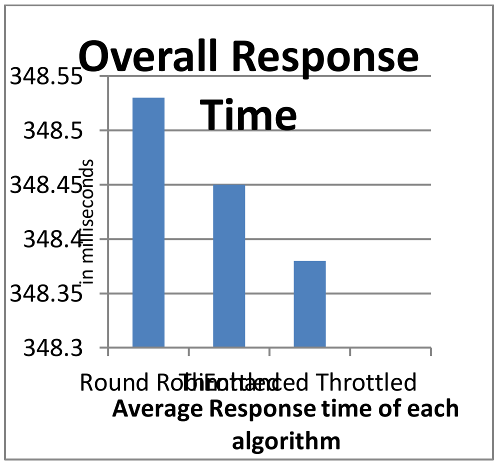
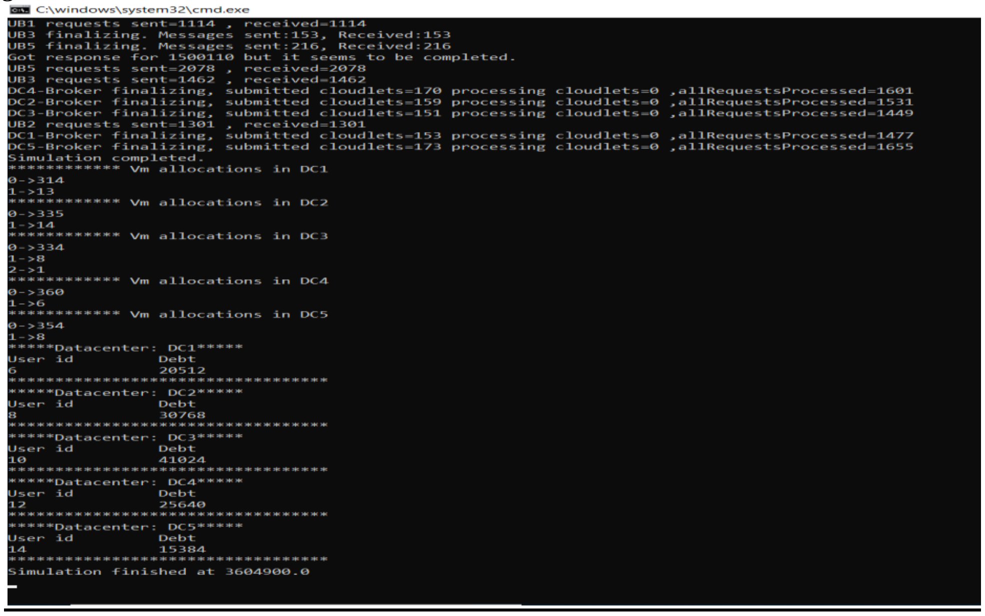


Figure 8: Chart showing the difference in Overall Response Time

Based on the parameters set in the cloud analyst tool, it has computed the simulation result and the comparison of overall response time is derived and is depicted in Figure 7. It is observed that the overall response time is slightly reduced in Enhanced Throttled Algorithm over the other two approaches namely Round Robin and Throttled Algorithm.



```
C:\windows\system32\cmd.exe
UB1 requests sent=1114 , received=1114
UB3 finalizing. Messages sent:153, Received:153
UB5 finalizing. Messages sent:216, Received:216
Got response for 1500110 but it seems to be completed.
UB5 requests sent=2078 , received=2078
UB3 requests sent=1462 , received=1462
DC4-Broker finalizing, submitted cloudlets=170 processing cloudlets=0 ,allRequestsProcessed=1601
DC2-Broker finalizing, submitted cloudlets=159 processing cloudlets=0 ,allRequestsProcessed=1531
DC3-Broker finalizing, submitted cloudlets=151 processing cloudlets=0 ,allRequestsProcessed=1449
UB2 requests sent=1301 , received=1301
DC1-Broker finalizing, submitted cloudlets=153 processing cloudlets=0 ,allRequestsProcessed=1477
DC5-Broker finalizing, submitted cloudlets=173 processing cloudlets=0 ,allRequestsProcessed=1655
Simulation completed.
************ Vm allocations in DC1
0->314
1->13
************ Vm allocations in DC2
0->335
1->14
************ Vm allocations in DC3
0->334
1->8
2->1
************ Vm allocations in DC4
0->360
1->6
************ Vm allocations in DC5
0->354
1->8
*****Datacenter: DC1*****
User id          Debt
6                20512
**********************************
*****Datacenter: DC2*****
User id          Debt
8                30768
**********************************
*****Datacenter: DC3*****
User id          Debt
10               41024
**********************************
*****Datacenter: DC4*****
User id          Debt
12               25640
**********************************
*****Datacenter: DC5*****
User id          Debt
14               15384
**********************************
Simulation finished at 3604900.0
_
```

Figure 9: Simulation background process completed

The simulation running in the background shows that for each of the data centre creation and the algorithm used there is a difference in the output by the Cloud Analyst as seen from the Figure 9 representing simulation background process completed screen.

Based on the results and observations from simulation, it is concluded that the average response time is lesser in case of Enhanced Throttled Algorithm compared to Round Robin and Throttled algorithms. The outcome of the simulator shows that with the help of the Enhanced Throttled algorithm, load allocation to virtual machines is possible with reduction in faults. However data centre processing output does not show such improvement as compared to the response time.

VII CONCLUSION AND FUTURE WORK

Efficiently allocation of load to the VMs with respect to a particular task needs to be done in a proactive load balancing approaches during virtualization to reduce faults with improved response time. This research work has proposed an Enhanced Throttled Algorithm for load balancing and thereby improves the response time. It is simulated and observed that the proposed algorithm outperforms the existing two algorithms - Round Robin and Throttled Algorithm. Since the data centre processing time has not improved, the future work can consider some modification in the algorithm which will take care of that also.

REFERENCES

1. Peter Mell, Timothy Grance, "The NIST Definition of Cloud Computing", Computer Security Division, National Institute of Standards and Technology, NIST Special Publication 800-145, September 2011.
2. Atul Kumar, Deepti Malhotra ,"Study of Various Reactive Fault Tolerance Techniques in Cloud Computing", International Journal of Computer Sciences and Engineering Open Access, Vol-6, Special Issue-5, Jun 2018.
3. MR.Manan D. Shah, Mr.Amit A. Kariyani, Mr.Dipak L. Agrawal, "Allocation of Virtual Machines in Cloud Computing using Load Balancing Algorithm", IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN: 2249-9555 Vol. 3, No.1, February 2013.
4. Afzal, S., Kavitha, G. Load balancing in cloud computing – A hierarchical taxonomical classification. Journal of Cloud Computing, Vol.8, No.22, 2019.
5. Minxian Xu ,Wenhong Tian, Rajkumar Buyya, "Survey on Load Balancing Algorithms for Virtual Machines Placement in Cloud Computing", Published online in Wiley InterScience (www.interscience.wiley.com).

6. Durgesh Patel, Anand S Rajawat, Efficient Throttled Load Balancing Algorithm in Cloud Environment, International Journal of Modern Trends in Engineering and Research (IJMTER).e-ISSN: 2349-9745, p-ISSN: 2393-8161,Vol. 2, No.3, March 2015.
7. Bhathiya, Wickremasinghe,"Cloud Analyst: A Cloud Sim-based Visual Modeller for Analysing Cloud Computing Environments and Applications", IEEE, 2010.
8. Isam Azawi Mohialdeen , "Comparative Study of Scheduling Algorithms on Cloud Computing Environment", Journal of Computer Science, 2013.
9. Vandana Sivaraj, A Kangaiammal, Avinash S Kashyap, Enhancing Fault Tolerance using Load Allocation Technique during Virtualization in Cloud Computing, International e-Conference on recent trends and Technologies in Soft Computing(ICRTSC 20), 2020.
10. Tanveer Ahmed, Yogendra Singh, "Analytic Study Of Load Balancing Techniques Using Tool Cloud Analyst", International Journal of Engineering Research and Applications, Vol. 2, No. 2, pp. 1027-1030, 2012.
11. Veerawali Behal, Anil Kumar, "Comparative Study of Load Balancing Algorithms in Cloud Environment using Cloud Analyst", International Journal of Computer Applications ISSN:0975 – 8887, Vol. 97, No.1, 2014.
12. Kumar, Anit and Chawla, Priyanka, "A Systematic Literature Review on Load Balancing Algorithms of Virtual Machines in a Cloud Computing Environment", Proceedings of the International Conference on Innovative Computing & Communications (ICICC), March 30, 2020.
13. Ghasemi, A., Toroghi Haghighat, A. "A Multi-objective Load Balancing Algorithm for Virtual Machine Placement in Cloud Data Centers based on Machine Learning". Computing, Vol. 102, pp. 2049–2072 (2020).