Utilizing Round Robin Concept for Load Balancing Algorithm at Virtual Machine Level in Cloud Environment

Stuti Dave
B H Gardi College of Engineering & Technology
Rajkot
Gujarat - India

Prashant Maheta
B H Gardi College of Engineering & Technology
Rajkot
Gujarat - India

ABSTRACT

Day by Day increasing traffic on internet introduces the requirement of load balancing concept to get the most utilization of the resources available on Cloud. There are so many complex calculations and concepts implemented to achieve better and better resource utilization and performance. Out of all this complexity the Rounrobin algorithm provide the simplest solution for load balancing in cloud environment. So we are going to use roun robin as a base algorithm and propose an improved version of load balancing algorithm in the paper.

General Terms

Cloud load balancing, Round robin algorithm.

Keywords

Round robin, Load balancing, Virtual machine balancing, Cloud Analyst.

1. INTRODUCTION

Current cloud computing environment serves in almost every field of our life. But while fulfilling lots and lots of user requests it faces few limitations to be overcome. Along with providing us facilities like virtualization, resource sharing, ubiquity, utility computing it ask us to focus on issues like security, authentication, fault tolerance, load balancing, and availability.

Lot of work is being carried out to achieve load balancing in cloud infrastructure now days. There are so many load balancing algorithms performing their task on different layers of cloud with different level of complexities. To achieve great level of load balancing people use many complex algorithms. But as a result it adds more processing load on the system executing such logic.

2. LOAD BALANCING

Implementing load balancing in your application simply means to spread the incoming request load among the available request executing nodes. Load balancing takes care of the executing nodes that they do not get overloaded with the user request. This simple care taken by the cloud manager improves the performance drastically.

Basically there are 2 types of load balancing algorithm depending on their implementation method according to [1].

A. Static Load Balancing Algorithm:

It does not depend on the current state of the system. Earlier before setting up the request it decides that at which host the request will be executed.

B. Dynamic Load Balancing Algorithm:

Decisions on load balancing depend on current state of the system. The load balancer analyses the current load statistics at each available host and executes request at appropriate host.

In the current cloud infrastructure we can introduce load balancing concept at few levels. We can implement load balancing at Service Broker, Server cluster and the Virtual Machine Monitor (VMM). In Fig-1 we have shown a basic cloud infrastructure as shown in [10]. Region N indicates geographic location of the users. User bases are group of users sending request traffic. With the help of Service Broker user request will identify the suitable server which can fulfill their requests. Here we have shown VM load balancer at executing node cluster. In worse situations VM load balancer takes decision to migrate VM rather than VM load with help of VMM [1].

In basic round robin algorithm the Load balancer allocates a VM to requesting node in cyclic manner equally among all available nodes [3]. Main advantage of this algorithm is that it utilizes all the resources in a balanced order and equal numbers of VMs are allocated to all the nodes which ensure fairness.

Load balancing algorithm, implemented at any level of cloud, must try to fulfill any or all of the following characteristics according to [2]:

a)Maximum context switches, CPU utilization, throughput and

b) Minimum Turnaround time, Waiting time, Response time.

With context switches we mean switching among the CPU states stored earlier so that execution can be resumed from it. Throughput indicates the number of processes completed per unit time. Utilization quotient of CPU is specified by CPU utilization so we wish to keep it as high as possible. Turnaround time is sum of the time a process spend in ready queue waiting for memory along with doing input output processes and executing on CPU. Waiting time is the amount of time a process waits in ready queue. Response time is the time the system starts responding the request, not the time it completes responding.

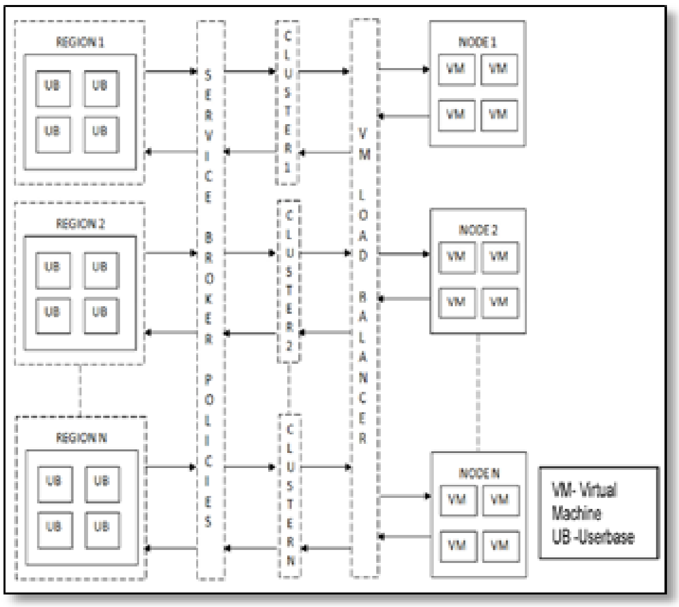


Fig 1: Basic Cloud Environment

3. SIMULATORS AND ALGORITHMS

To perform check on load balancing algorithms there are plenty of simulators available. Cloudsim [6, 7] is the first tool to simulate a complex cloud infrastructure for load balancing purpose. CloudAnalyst, CloudReport are simulators which are developed using Cloudsim packages.

In the CloudAnalyst terminology, [5] the user base is a cluster of users sending requests and Data center is the cluster of capable nodes which can perform according to the requests arrived. Each Data Center instance hold Datacenter controller. Datacenter controller utilizes VM Load Balancer to balance the load by assigning specific VM to the incoming cloudlet. Here cloudlet means request to the cloud for any service. And the VM Load balancer follows basically 3 types of popular load balancing techniques [4, 5]:

3.1Round Robin Algorithm

This is the simplest algorithm out of all available algorithms for load balancing and hence do not require complex algorithm implementations. It simply maintains a queue of incoming requests and allocates them VM in Time scheduling manner. Thus each request is allowed to be executed for specific time quantum only then after if it is still incomplete, it has to wait for its next round and if the request is complete it allows other process to take charge of that VM based on the algorithm.

3.2 Throttled Algorithm

This algorithm checks for suitable and available VM in the list of VMs along with state (BUSY/ AVAILBLE). When a request arrives at load balancer, it scans VM list available with it. If suitable VM is found free it is allocated for request execution otherwise the request is queued. Here the VM searching process is speed up with help of introducing indexing at each row of the VM list. Thus Throttled algorithm is good as it does not introduce any implementation complexity and though performs significantly.

3.3 Active monitoring Algorithm

This algorithm load balances the requests between available VMs in a way that manages almost same number of active requests on each VM at any given time. This algorithm is similar to throttled [5]. But as it allocates multiple tasks/ requests on VM the complexity increases.

has to wait for its next round and if the request is complete it allows other process to take charge of that VM based on the algorithm.

3.4 Throttled Algorithm

This algorithm checks for suitable and available VM in the list of VMs along with state (BUSY/ AVAILBLE). When a request arrives at load balancer, it scans VM list available with it. If suitable VM is found free it is allocated for request execution otherwise the request is queued. Here the VM searching process is speed up with help of introducing indexing at each row of the VM list. Thus Throttled algorithm is good as it does not introduce any implementation complexity and though performs significantly.

3.5 Active monitoring Algorithm

This algorithm load balances the requests between available VMs in a way that manages almost same number of active requests on each VM at any given time. This algorithm is similar to throttled [5]. But as it allocates multiple tasks/ requests on VM the complexity increases.

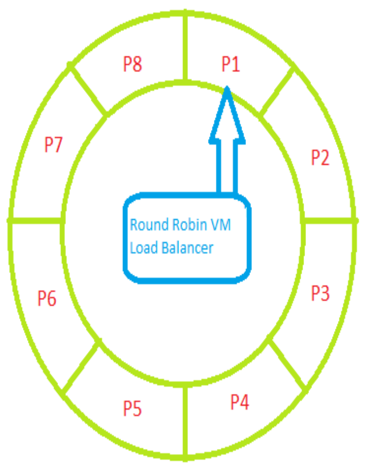


Fig 2 Round Robin Algorithm

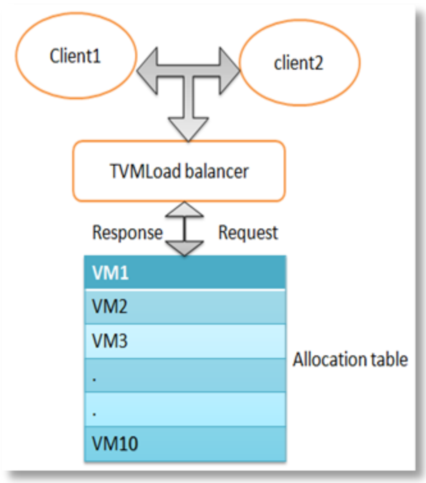


Fig 3 Throttled Algorithm

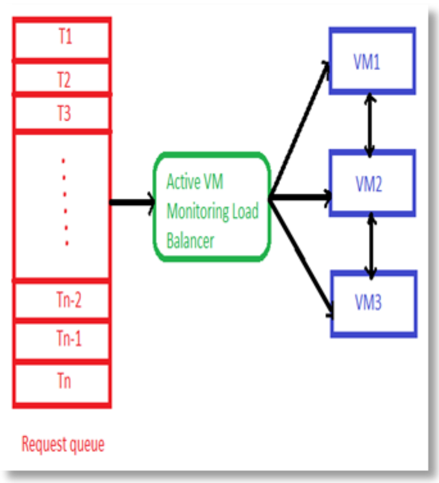


Fig 4 Active Monitoring Algorithm

4. ROUND ROBIN ALGORITHM AND ITS VARIANTS

The round robin algorithm is one of the most popular and simplest algorithm in any field we consider. The user is always sure that the request processing will be simpler and faster when round robin is working for the Load Balancing goal. Because of these feature there has been lot of research carried out to improve performance of this algorithm.

There are so many variants of round robin algorithm available introducing different techniques focusing on different measuring parameters of it. When one wishes to work on or use Round robin algorithm, he must be familiar with its terminology:

a) Burst Time: BT is the time duration which a request requires to complete.

b) Time Quantum: TQ is the time duration for which a request is allowed to access a VM.

4.1 PBDRR

In Priority Based Round Robin (PBDRR) algorithm [8], they have calculated intelligent time slice which allocates a different TQ to each process according to the priority given. In their future work the deadline is to consider one of the input parameter in addition to the priority in the algorithm.

4.2 MRR

In Modified Round Robin (MRR) algorithm [9], here Time Slice (TS) is calculated based on (range× total no of process (N)) divided by (priority (pr) × no of process in queue(p)). Range is calculated by (maximum burst time +minimum burst time) divided by 2. This algorithm improved a scheduling up to some extent but not much.

4.3 TSPBRR

In Load balancing based on Time Slice Priority Based RR (TSPBRR) [2], they have initially taken Time Quantum (TQ) as half of the first request BT and then after time quantum is calculated using equation TQi = TQ i-1 + ½ TQ i-1,i indicating execution round number. This algorithm improved over results of MRR but still they expect more improvement in response time.

4.4 WRR

In Weighted round robin algorithm [3], it allocates all incoming requests to the available virtual machines in round robin fashion based on the weights without considering the current load on each virtual machine. This may lead to wait few request wait for a long time if it is allocated to a busy VM.

4.5 RR with Server Affinity

In round robin algorithm with server affinity [10], The Round Robin with server affinity VM load balancer maintains two data structures, which are as listed below.

- Hash map: These stores the entry for the last VM allocated to a request from a given Userbase.
- VM state list: this stores the allocation status (i.e., Busy/Available) of each VM.

When a request is received from the Userbase, if an entry for the given Userbase exists in the hash map and if that particular VM is available, there is no need to run the Round Robin VM load balancing algorithm, which will save a significant amount of time.

5. PROPOSED ALGORITHM – FairRR

All algorithms included in above section follows different strategies to improve round robin algorithm's performance trying to make its simplicity intact. Because a complex algorithm also increases executing node's load and degrades its performance.

Our algorithm follows quite simple strategy of implementing dynamic TQ based on algorithm execution round. Our algorithm steps can be enlisted as shown in fig -5.

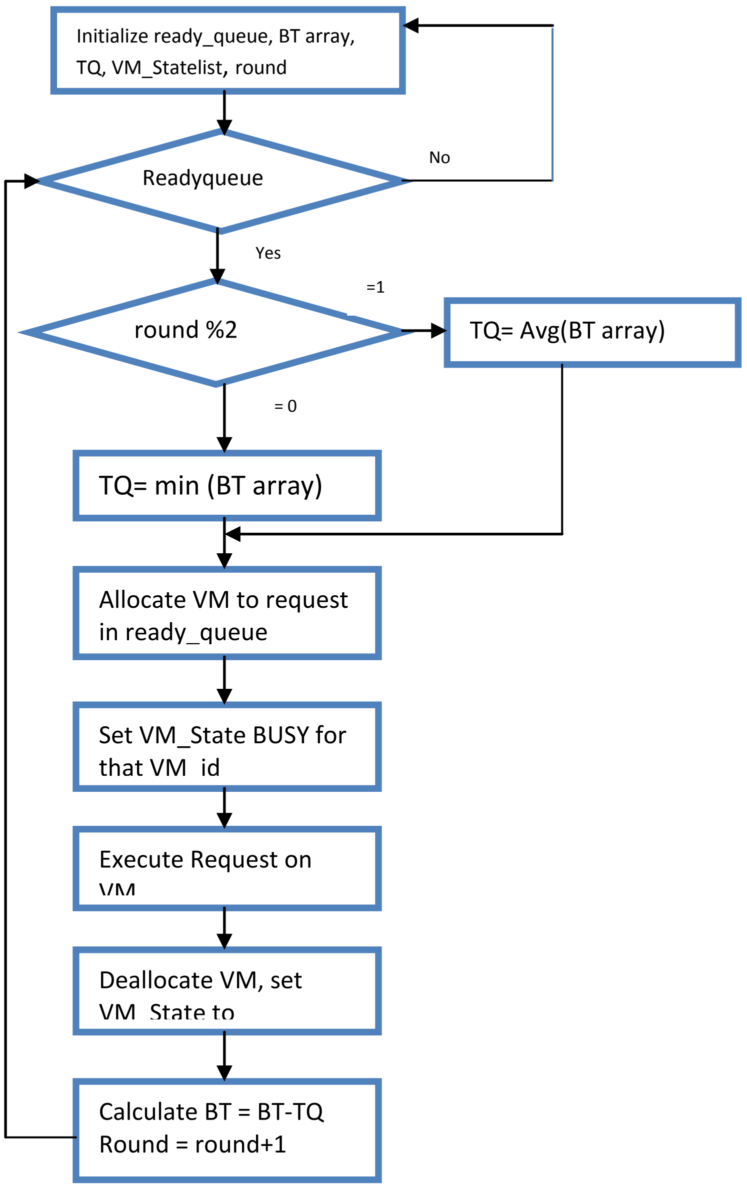

Initialize ready_queue, BT array, TQ, VM_Statelist, round
Readyqueue
No
Yes
round %2
=1
TQ= Avg(BT array)
= 0
TQ= min (BT array)
Allocate VM to request in ready_queue
Set VM_State BUSY for that VM id
Execute Request on VM
Deallocate VM, set VM_State to
Calculate BT = BT-TQ
Round = round+1

Fig 5 Flowchart for Proposed Algorithm

```
FairRR()
{
        Initialize ready_queue, BT list, TQ, VM_State
        list, round;
        While(ready_queue != NULL)
        {
                if ( round%2 == 0)
                        TQ= min(BT list);
                else
                      TQ= avg (BT list);
                Allocate  VM to request in
                ready_queue, set VM_State to BUSY;
                Execute request;
                Deallocate VM and set VM_State to
                AVAILABLE;
                round++;
                New BT=BT-TQ;
        }
}
```

Fig 6 Pseudo code for Proposed algorithm

Table 1 Example Execution of FairRR algorithm

Task Or Request	BT	At R0 TQ= min(BT)= 4	At R1 TQ= avg(BT)=8.5	At R2 TQ= min(BT)=8
R1	20	16	8	Complete
R2	4	Complete	Complete	Complete
R3	12	8	Complete	Complete
R4	8	4	Complete	Complete
R5	10	6	Complete	Complete

Pseudo code for our proposed algorithm is shown in Fig-6.

Proposed algorithm is fair round robin algorithm because it provides enough fair scheduling when the burst time of incoming request load is having great variance. To better explain this, consider the example shown here:

Example in Table-1 shows how the request gets quick response as compared to normal round robin algorithm independent of its request size or Burst time.

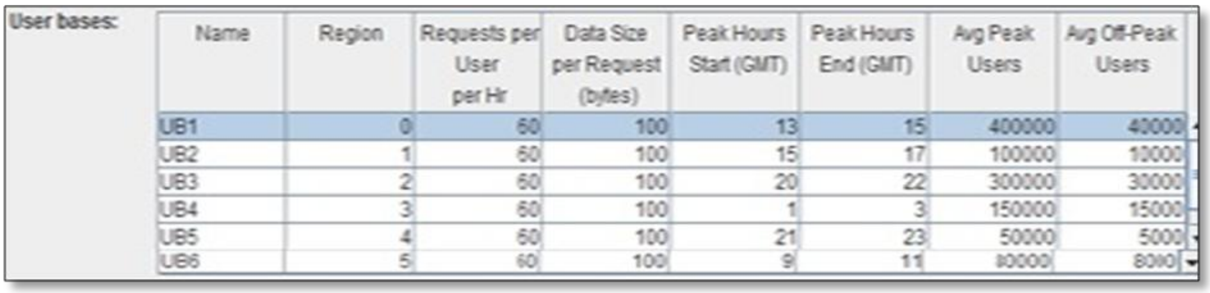
The example here also shows that the overall algorithm execution requires less number of rounds to follow thus the algorithm when applied to large number of request performs faster providing better response time to any request.

FairRR(Fair Round Robin) algorithm thus provides fairness to larger requests and smaller one also to the load coming at executing node in the cloud.

6. SIMULATION ENVIRONMENT

To analyze result and compare them with existing algorithm, we used CloudAnalyst tool for simulating FairRR algorithm. The cloud environment set up generated was having following configuration.

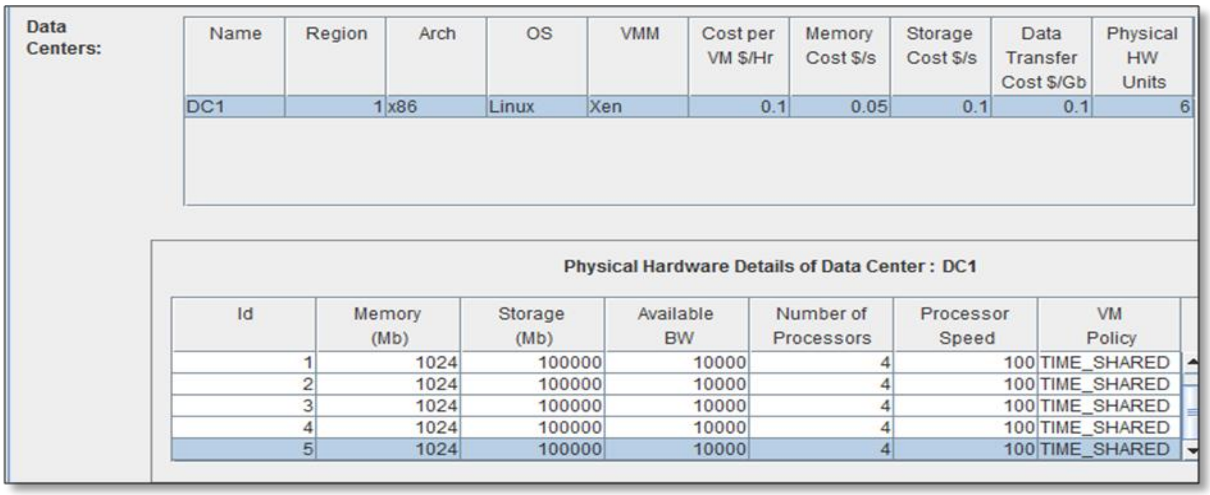
UserBase Settings, Datacenter Settings are shown in fig –7 and fig-8 respectively.



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	60	100	13	15	400000	40000
UB2	1	60	100	15	17	100000	10000
UB3	2	60	100	20	22	300000	30000
UB4	3	60	100	1	3	150000	15000
UB5	4	60	100	21	23	50000	5000
UB6	5	60	100	9	11	80000	8000

Fig 7 UserBase Settings



Data Centers:

Name	Region	Arch	OS	VMM	Cost per VM $/Hr	Memory Cost $/s	Storage Cost $/s	Data Transfer Cost $/Gb	Physical HW Units
DC1	1	x86	Linux	Xen	0.1	0.05	0.1	0.1	6

Physical Hardware Details of Data Center : DC1

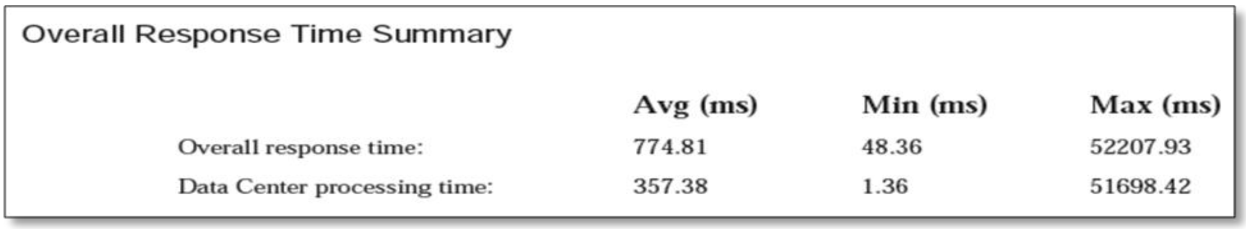
Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
1	1024	100000	10000	4	100	TIME_SHARED
2	1024	100000	10000	4	100	TIME_SHARED
3	1024	100000	10000	4	100	TIME_SHARED
4	1024	100000	10000	4	100	TIME_SHARED
5	1024	100000	10000	4	100	TIME_SHARED

Fig 8 Datacenter Settings

User grouping factor which indicates number of simultaneous users from a userbase is taken 1000. And Request grouping factor indicating simultaneous requests a single application server can support is taken 100. Executable instruction length per request is taken 250 Bytes. There are 5 VMs working for a Datacenter.

7. SIMULATION RESULT AND COMPARISON

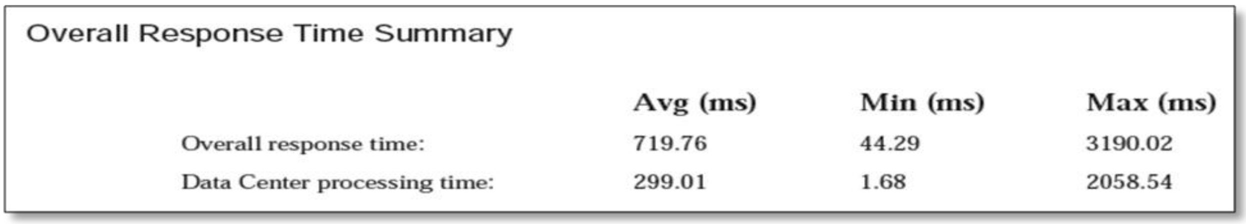
Fig-9, fig-10, fig-11, fig-12 represents results of round robin algorithm, Throtled algorithm, active monitoring algorithm and proposed algorithm. Comparison of overall response time and Data Centre processing time is shown in the graph shown in fig 13.



Overall Response Time Summary

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	774.81	48.36	52207.93
Data Center processing time:	357.38	1.36	51698.42

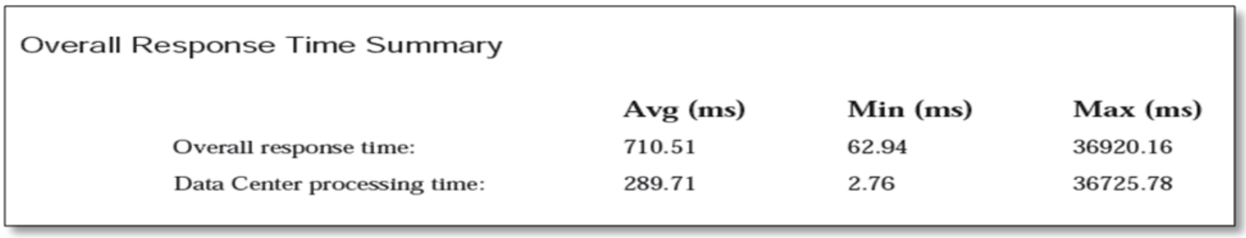
Fig 9 Round robin algorithm result



Overall Response Time Summary

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	719.76	44.29	3190.02
Data Center processing time:	299.01	1.68	2058.54

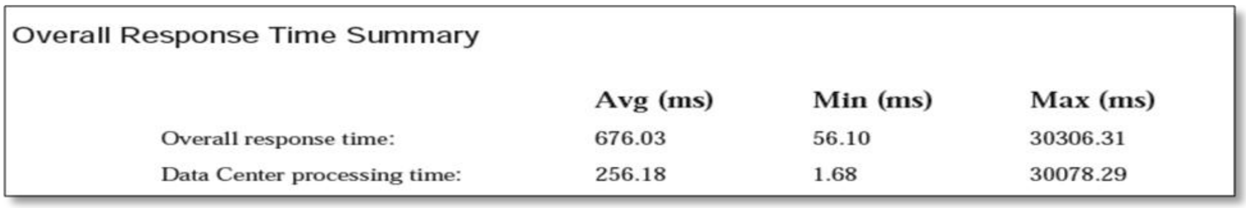
Fig 10 Throttled Algorithm result



Overall Response Time Summary

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	710.51	62.94	36920.16
Data Center processing time:	289.71	2.76	36725.78

Fig 11 Active monitoring Algorithm result



Overall Response Time Summary

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	676.03	56.10	30306.31
Data Center processing time:	256.18	1.68	30078.29

Fig 12 Fair RR Algorithm result

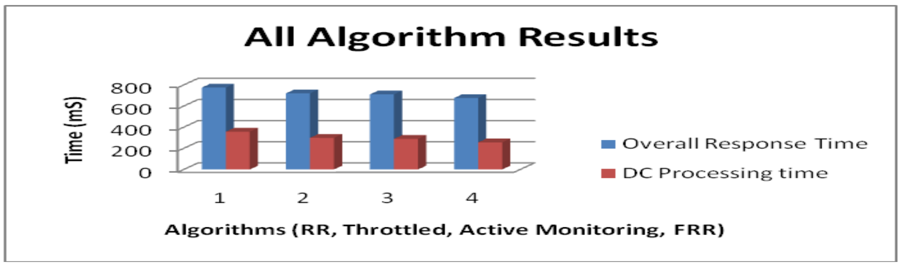


Fig 13 All algorithm result Comparison

8. CONCLUSION AND FUTURE WORK

Existing algorithm in cloud load balancing field are either complex or not providing much promising results. By introducing this algorithm we would like to present simple yet efficient algorithm for load balancing in cloud environment.

Basically this algorithm keeps in mind the variations in request size and the effect of number of rounds of algorithm implementation so as to achieve fairness and faster processing on incoming cloud requests resulting in faster load balancing.

9. ACKNOWLEDGMENTS

We would like to express our deep sense of gratitude towards Prof. Rushit D. Trivedi of VVP Engineering College of Rajkot (Gujarat) for providing us constant motivation and support as and when required.

10. REFERENCES

[1] Stuti Dave, Prashant Mehta, “Role of Virtual Machine Live Migration in Cloud Load Balancing”, IOSR Journal of Computer Engineering (IOSR-JCE),e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume X, Issue X (Nov. - Dec. 2013).

[2] Subasish Mohapatra, Subhadarshini Mohanty, K.Smruti Rekha, “Analysis of Different Variants in Round Robin Algorithms for Load Balancing in Cloud Computing”, International Journal of Computer Applications (0975 – 8887),Volume 69– No.22, May 2013.

[3] Supreeth S, Shobha Biradar, “Scheduling Virtual Machines for Load balancing in Cloud Computing Platform”, International Journal of Science and Research (IJSR), India Online ISSN: 2319-7064, Volume 2 Issue 6, June 2013.

[4] Subasish Mohapatra, K.Smruti Rekha, Subhadarshini Mohanty, “A Comparison of Four Popular Heuristics for Load Balancing of Virtual Machines in Cloud Computing”, International Journal of Computer Applications (0975 – 8887), Volume 68– No.6, April 2013.

[5] Bhathiya Wickremasinghe, Rajkumar Buyya , “CloudAnalyst: A CloudSim-based Tool for Modelling and Analysis of Large Scale Cloud Computing Environments”, 433-659 distributed computing project, csse dept., university of melbourne, 22/6/2009.

[6] Rodrigo N. Calheiros, Rajiv Ranjan, César A. F. De Rose, Rajkumar Buyya, “CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services”, 2009.

[7] R. Buyya, R. Ranjan, and R. N. Calheiros, “Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities,” Proc. of the 7th High Performance Computing and Simulation Conference (HPCS’ 09), IEEE Computer Society, June 2009.

[8] Prof. Rakesh Mohanty, Prof. H. S. Behera, Khusbu Patwari, Monisha Dash, M. Lakshmi Prasanna, “Priority Based Dynamic Round Robin (PBDRR) Algorithm with Intelligent Time Slice for Soft Real Time Systems”, International Journal of Advanced Computer Science and Applications, Vol. 2, No.2, February 2011.

[9] C. Yaashuwanth, R. Ramesh, “Design of Real Time Scheduler Simulator and Development of Modified Round Robin Architecture for Real Time System”, International Journal of Computer and Electrical Engineering (IJCEE), Vol. 10, No. 3, pp 43-47, March, 2010.

[10] Komal Mahajan, Ansuyia Makroo, Deepak Dahiya, “Round Robin with Server Affinity: A VM Load Balancing Algorithm for Cloud Based Infrastructure”, http://dx.doi.org/10.3745/JIPS.2013.9.3.379, J Inf Process Syst, Vol.9, No.3, September 2013.