www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issues 8 Aug 2016, Page No. 17375-17380

A Round-Robin based Load balancing approach for Scalable demands and maximized Resource availability

Poonam Kumari[1], Mohit Saxena[2]
M. Tech Scholar[1], Assistant Professor[2]

[1]Apex Institute of Engineering & Technology,
Sitapura, Jaipur, Rajasthan, India
Hunny.punam@gmail.com

[2]Apex Institute of Engineering & Technology,
Sitapura, Jaipur, Rajasthan, India
Mohit.saxena234@gmail.com

Abstract: **In a communication scenario where the capacity of one *running service is not enough to serve high demands and it is desired to increase the performance by distributing the workload across multiple resources i.e. services. The solution for such scenarios could be using of load balancer. Load balancer receives hundreds of requests at same instant from client and it distributes the load to the different instances/machines. This Load distribution is independent of the number of instances/machines. To simulate the scenario, this research implements a client application using a service to Sort the n numbers. The calculation is split into many small intervals. Therefore, the service gets overloaded and the performance drops down. In this research, a load balancer is proposed based on the Round-Robin approach. The service processes the message and sends back the response that is routed via the load balancer back to the client. Due to any reason if one of the instance/machine fails during process execution, this does not lead the complete process to failure. The Round-Robin approach is suitable for load balancing because the processing of requests takes approximately same time.***

Keywords: Load Balancing, Round-Robin approach, Distributed systems.

1. Introduction

In today's communication scenario, where traffic is continuously growing up and up so single service is not enough to handle it. So, to cost effectively scale to meet these high volume traffic, modern computing requires several concurrent servers which serves these concurrent requests from clients and return responses.

A load balancer works between client and server and it routes client requests across all available servers that helps to maximize speed and capacity utilization of each server. Load balancer acts as a reverse proxy. Load balancer also ensures that no any server is overloaded, which could degrade the performance of system. If any server goes down, then load balancer redirects the incoming traffic to the remaining available online servers [1]. When a new server is added to the system then the load balancer automatically begin to send traffic to it.

Load balancing is an approach which distributes the workloads across the available multiple computing resources so it increases capacity and reliability of system. Load balancer also improves the overall performance of system by minimizing the burden on servers associated with system by managing and maintaining network sessions. These resources may be computers, a computer cluster, network links, central processing units or disk drives. The aim of Load balancing is to optimize the resource use, maximize the throughput, minimize their response time, and avoid the overloading situation of any single resource. A load balancing with multiple components may increase the reliability and availability of the system through redundancy than a system with single component. Usually load balancing consists a dedicated software or a hardware. It is also known as Server Pool or Server Farm [2].

Load balancer ensures that it sends traffic to those servers which are online so it provides high availability and reliability to the system. Load balancer also provides flexibility to the system by adding or subtracting servers to system as demands needs.

Load balancer are divided into categories: Layer 4 and Layer 7. Layer 4 load balancer works upon the data found at network and transport layer while layer 7 load balancer acts on data associated with application layer protocols like HTTP. Incoming requests are received by both type of load balancers and these requests are distributed to a particular server decided by a configured algorithm. For example: a multilayer switch or a Domain Name System server process [2].

Load balancing sounds same as channel bonding but Load balancing is different from the channel bonding. In a load balancing, balancer divides the traffic or workload between the network interfaces. These network interfaces rely on network socket which works in layer 4 of OSI model. Whereas, in channel bonding, traffic is divided between the physical interfaces. These division may be as per packet (which comes under layer 3 of OSI model). It also may be on data link basis (which is layer 2 of OSI model) with a protocol such as shortest path bridging [3].

1.1 Overview of load balancing algorithms

There are five common load balancing algorithms, discussed below. In this section, some important characteristics and suitable environment for using these algorithms are given below.

1.1.1 Round Robin

Round Robin is the most widely used algorithm in computer science. Main thing about it is that it is easy for the implementation and easy to understand. It is also worked in a load balancing. Suppose, two servers are waiting for the requests behind load balancer in system. Assume a scenario in which the first request arrives, then balancer will forward this

request to the first server. And when the second request arrives to the balancer then (assuming it is coming from a different client), this request will be forwarded to the second server [4].
As mentioned above there are two servers in this cluster. So, the next request (i.e. third one) will be moved back to the first server. And then next upcoming fourth request will forward back to the second server, and so on, in a circular way.
The method used in round robin is very simple. But when this algorithm gives best results or for which environment it does not work well, for the answers of these questions can be understand by following examples.
Suppose, Server 1 is having more CPU, RAM, and other specifications compared to the Server 2 means now Server 1 should be able to handle a higher workload than Server 2, right? But, a load balancer with round robin algorithm would not work accordingly to the capacity of these two servers. So it may give inefficient results because the load balancer still distribute workload (means requests) equally to the both servers by using round robin algorithm [4].
According to this scheduling server 2 gets overloaded faster and probably it may go down. So, it is not a suitable approach for this situation. So, to handle these servers according to their capacity wise, a weighted round robin algorithm is introduced.
It is clear that round robin algorithm is suitable for a cluster consisting of servers with identical specifications. But if specifications of servers are not same then weighted round robin approach is used.

1.1.2 Weighted Round Robin

As it is mentioned above in second scenario in which server 1 is having higher capacity than the server 2 means Server 1 is having higher specifications than Server 2. By using weighted round robin algorithm, load balancer assigns more requests to the server 1 which is having higher capability to handling greater load than server 2. This algorithm is known as Weighted Round Robin algorithm.
This Weighted Round Robin is same as Round Robin algorithm in a manner by which incoming requests are assigned to the server is still cyclical. The server with the higher specifications will be apportioned to a greater count of requests.
But how would a load balancer know about the capacities of the nodes available in the network? It is set beforehand. Basically, when a load balancer is set up in network, then it is assigned by the "weights" to each node. These weights are given according to the capacity of nodes which is available in system. A node which is having higher specifications should be given a higher weight and a node which is having lesser specifications should be given lesser weight accordingly.
Usually weights are specified in proportion to the actual capacities. For an example, if the capacity of server 1 is 5x more than the Server 2's, so, weight of the server is usually given as 5 and for the Server 2 a weight is given to 1.
So, when clients are started to come in. The first five requests will be assigned to the server 1 and then after the upcoming sixth request will be assigned to the server 2.
If more clients are come in system, then this same sequence will be followed by balancer to distribute incoming requests. That means, the upcoming 7th, 8th, 9th, 10th, and 11th requests will all assign to the Server1, and the next 12th request will be assigned to the Server 2, and so on [5].
Weighted round robin is suitable for such system in which above mentioned capacity of server is an issue. It is also suitable for such situations, in which capacities of the servers are equal but we want any server from the network to get a substantially lesser number of connections than an equally capable server. This is useful for such scenario in which the first server is running a business-critical applications and we don't want this to be get overloaded easily.

1.1.3 Least Connections

This can be understand by assuming a scenario in which two servers are in a cluster and these two servers are having exactly the same specifications. Suppose one server may still get overloaded faster than the other server. The one possible reason for such scenario would be that clients which are connecting to the server 2 stays connected much longer than those clients which are connecting to the server 1. So, the total current connections in server 2 is increased. While the clients of server 1 are connecting and disconnecting over the shorter times. So that, the total current connections in server 1 would be remain same and having lesser current connections compare to the server 2.
So, for a result, resources of server 2 can run out speedily. As in an example given below, where we can see that clients 1 and 3 are already disconnected, while client 2, 4, 5 and 6 are still connected.
For such situations, the Least Connections algorithm is better. This algorithm consider the total number of current connections of each node has. So when a client try to connect with a node so, the load balancer will determine a server which is having a least number of current connections in system firstly and then it will assign the incoming request to that server. And make a new connection to that server [5].
For example (continuing to last example given above), client 6 now attempts to connect after the both 1 and 3 have already terminated but both 2 and 4 are still in connection, So, load balancer will now assign next client 6 to the Server 1 instead of the Server 2.

1.1.4 Weighted Least Connections

The improvement of Round Robin algorithm is a Weighted Round Robin. The Weighted Least Connections algorithm do similar to the Least Connections. It uses a "weight" component in algorithm. This weight can be find according to the respective capacities of an each server. Same as in the Weighted Round Robin algorithm. So, before started we have to specify the "weight" of each server.
If a load balancer works according to the Weighted Least Connections algorithm then there are two things which should be consider by it are following: (1) the weights/capacities of each server (2) the current number of clients that are currently connected to each server.

1.1.5 Random

There is no such particular method is used in assigning a task to the node. As it name tells, this algorithm randomly matches clients and nodes available. This can be done by using random number generator. Suppose there is a high traffic in system, then the load balancer will receive these requests and distribute these requests evenly to the nodes available by using random algorithm. So, it works here like Round Robin algorithm. The Random algorithm is sufficient for those clusters which consist nodes with the similar configurations such as (CPU, RAM, etc).

2. Literature Survey

2.1 Brief Overview of Taxonomy of Load Balancing Policies

The different categories of the load-balancing policies are displayed in this part. A detailed overview of these different taxonomies are given in later sub parts.

2.1.1 Static versus Dynamic

Static load distribution also called as a deterministic scheduling. In this scheme load balancer assigns a given job to a fixed processor or node. In static scheme, the system is restarted every time and the same binding of task-processor (means the allocation of a task to the same processor which is predefined) is followed without considering the changes that may be occurred during lifetime of a system. Moreover, static load distribution may also characterized in the strategy which is applied at runtime. In this strategy, it does not follow the same task-processor assignment as previous, but it assigns newly arrived jobs in a sequential or in a fixed fashion. For example, by using a simple static strategy, arrived jobs can be assigned to the nodes in a round-robin manner. So that, each processor executes approximately the same number of tasks.

Fixed or static scheme produces poor results because it does not consider any changes occurred in a system. So, Dynamic load-balancing scheme is introduced. In this scheme, system parameters may not be fixed before. A dynamic policy is usually executed several times in system and can change the binding of task-processor. It may change/reassign a previously scheduled job to a new processor/node according to the current dynamics of the system environment.

2.1.2 Distributed Versus Centralized

Load distribution normally falls under the category of dynamic load-balancing scheme, where a basic question arises that where the actually decision is made. In centralized policies, global information is store at a central node/location and by using some computations on this information load balancer takes the scheduling decisions and store resources of one or more processors. This scheme is most suitable for such systems where a central station/node can easily collect an individual processor's state information at little cost, and the new jobs first arrived at this centralized location and then redirected to the subsequent nodes. But it has a single point of failure. This is the main drawback of this scheme [6].

In distributed scheduling, the state information is distributed among the nodes that are responsible in managing their own resources or allocating tasks residing in their queues to other processors. In some cases, the scheme allows idle processors to assign tasks to themselves at runtime by accessing a shared global queue. Note that failures occurring at a particular node will remain localized and may not affect the global operation of the system.

Another available scheme which lies between the above mentioned two types is hierarchical scheme. In this scheme, some nodes are selected as a decision makers. These nodes are responsible for scheduling of task by providing task to the set of processors. These selected nodes are arranged in the form of tree. In which, the selected nodes are the roots of a sub tree domains.

2.1.3 Local Versus Global

The both Local and the global load-balancing falls under the category of distributed scheme. Since a centralized scheme should always work as a global. Each processor polls with other processors in its neighborhood in this local load balancing scheduling. And by using this local status information load balancer makes decision upon a load migration. This local neighborhood area is generally called as migration space in the local load balancing scheduling. The primary objective of this scheme is to minimize the remote communication between nodes and to maintain an efficiently balance of load on the processors. But in the case of global balancing scheme, status information of the entire or a part of the system is shared which is used to make decision about load-balancing. A considerable information set is needed to be exchanged within a system which may be affect its scalability [6].

2.1.4 Cooperative versus Non-Cooperative

There are two mechanisms which involves the cooperation level between the parts of the system within the distributed dynamic global scheduling. These are cooperative and non-cooperative schema. Non-cooperative is also called autonomous scheme. Each node is autonomous and maintain its own resource scheduling in the non-cooperative scheme. That means, decisions are made by each node independently without involving the rest of the system. So that nodes may transfer or allocate the tasks according to their local performance. But in cooperative scheduling, processes work together towards a common system-wide global balance. So that, decisions are depend upon some global measures. So scheduling decisions are take place after considering their effectiveness on global measures (such as global completion time).

2.1.5 Adaptive versus Non-Adaptive

Dynamic load-balancing policies has two parts: Adaptive and non-adaptive schemes. Scheduled decisions are made by considering the past and the present system performance in an adaptive scheme. These decisions are also affected by the previous decisions or the changes occurred in environment. If any system parameter does not correlated to the performance of program, it is weighted low for next time. But in non-adaptive scheme, system parameters which are used in the scheduling, are not affected by the system's past behavior and remains same. An example may be of such policy which always weighs it's given inputs to the same without considering the past behavior of the system.

Main thing which helps to distinguish between both dynamic scheduling and adaptive scheduling is that a dynamic scheduling takes an environmental inputs into consideration while making the decisions, whereas an adaptive scheduling takes an environmental stimuli into the account to alter scheduling policy itself . An adaptive policy is also a dynamic [7].

2.1.6 One-Time Assignment versus Dynamic Reassignment

In this part, scheduling of an entities are considered. If task is assigned to node under one time assignment scheme, it may be dynamically done but it is scheduled once to the available node. It cannot be rescheduled to another processor/node. Whereas in the dynamic reassignment process, jobs can be transfer after an initial binding is done from one processor to another processor. But it may have a negative aspect such as tasks may endlessly moving into the system without having much progress.

2.1.7 Sender/Receiver/Symmetrical Initiated

In distributed systems, techniques for task scheduling are divided into sender-initiated, receiver-initiated, and symmetrically-initiated. In sender-initiated algorithm, a node which is overloaded can transfer their one or more tasks to the under- loaded nodes. In receiver-initiated scheme, a node which is now under-loaded can request for tasks to be sent from nodes which is having higher loads to them. In a symmetric approach, both under-loaded and over loaded nodes can initiate about load transfers.

2.2 Related Work

In 2014, Sunil K S, Ravichandra A J and Dr. H S Guruprasad, researchers studying Load Balancing in Three Tier Cloud Computing concluded that Load Balancing Algorithm reduces the average execution time of user tasks by increasing machine availability time which leads uniform distribution of workload in a cloud infrastructure [8].
In their approach they did not use the fault tolerant concept if a server crashes then it may lead to system fault.
In 2015, Geethu Gopinath P P and Shriram K Vasudevan , in their research paper with title “An in depth analysis and study of load balancing techniques in the cloud computing environment" have given the performance metrics of load balancing algorithms in cloud are response time and waiting time [9].
In their research, they compare two algorithms but not able to find the approach for dynamic load balancing.

2.3 Motivation

Load balancing covers different types of architectures that are not possible in traditional approach to handle large amount of requests. The motivation behind this approach is that it is having simplified design, scalable and enough control over the availability of large data. Load balancing provides results more efficient for different types of tasks.
Load balancing can be applied in many ways as according to user requirements like here Round Robin method is used to implement the load balancing. When there are thousands of requests at a time over the single server then it is not able to handle all the requests then it fails. It needs more servers to handle them at single instance so it needs something that distribute the load of all requests to among the servers. To solve this problem load balancing is used here to distribute the load and provide more efficient and accurate result.

2.4 Contribution

The contribution of this research work can be summarized as the improved performance of Load balancing using Round Robin method with different number of servers over the single server.
The load balancing makes the performance better when the number of servers added to the load balancer network. It improves the efficiency of the result generated by the server, also reduce the load traffic from the single server and distribute these traffic loads to all the servers which are connected to the load balancer. It also care about the system crashes, it never cause the system failure if any server crashes and all other servers perform their execution without interrupt and provide the efficient result.

3. Proposed Approach & Experimental Setup

3.1 Proposed Approach

In this research, a load balancer is developed based on Round-Robin Approach. Load balancer receives hundreds of requests at same instant from client and it distributes the load to the different instances/machines. Load distribution is independent of the number of instances/machines.
In this research, a load balancer is developed which is distributing total workload among available machines or instances of an application. This distribution is done according to the Round Robin approach. Round Robin is one of the most popular approach which is best suitable for those applications which uses context switching concept somewhere in their program. By using its approach, context switching is implemented based on some criteria. This helps to load balancer to divide a total number of tasks equally to instances of an application. So that, a particular part (i.e. range of task) of a program is executed by one instance and next part is executed by next instance and so on. And round robin also responsible for parallel execution of a process. This can minimize the response time of a program.
Round Robin is suitable for such systems where all available machines are having similar configurations and able to process equal number of tasks by taking approximately equal time. Here, the instances of an application is used to process tasks. Which shows a similar capacity to process. If any system is having dissimilar type of specifications of machines means machines may be of different capacities then also we can used extent approach of Round Robin. This is known as weighted round robin. Here in this research, instances of same application is used. So we can assume that it all are takes approximately same time to process same number of tasks. These all the things are implemented based on the concept of Round Robin algorithm.
The service processes the message and sends back the response that is routed via the load balancer back to the client. Due to any reason if one of the instance/machine fails while process execution, this does not lead the complete process to failure.
To simulate the scenario, this research implements a client application using a service to Sort the n numbers. The calculation is split into many small intervals. Therefore, the service gets overloaded and the performance drops down. The solution for such scenarios could be using of load balancer.

3.2 Experimental Setup

In this research, all the tests are performed under following specifications:
Host System: Intel i5 processor with 6 GB RAM and 1000 GB Hard disk.
Operating Environment: Windows 10
Microsoft Tool: Visual Studio 2013 is used for Task Parallelism.

4. Results And Analysis

4.1 Performing sorting function with load balancing on 5k Numbers

Here in the first experiment of this research, sorting function is applied on 5k random numbers in three different ways. Implementation is done on the single server, two servers and three servers which are having similar type of configurations. Here number of executions are five on all type of servers. After performing the operation time is calculated of all servers.

Table 4.1
Execution Time for Single Server, Two Servers & Three Servers

Experiment No.	**Single Server**	**Two Servers**	**Three Servers**
1	14.256	9.54	7.652
2	13.35	8.96	6.544
3	12.66	8.486	6.452
4	12.8	8.796	6.398
5	12.68	8.656	6.55

Table 4.1 presents the execution time of load balancing executed on all these servers. When load balancing is done on the single server the execution is much higher than as compare to that when load balancing is done on two servers and three servers. Maximum speed is achieved when there are three servers. Along with the speed accuracy is also maintained perfectly through the load balancer. These results show the efficiency and speed of load balancing over the different servers.

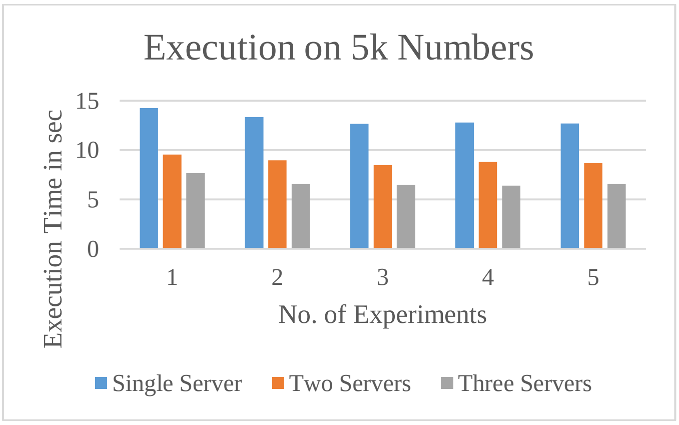

Figure 4.1: Execution Time for Single Server, Two Servers & Three Servers

From the figure 4.1 it is showing the execution time of load balancing executed on all these servers. When load balancing is done on the single server the execution is much higher than as compare to that when load balancing is done on two servers and three servers. Maximum speed is achieved when there are three servers. Along with the speed accuracy is also maintained perfectly through the load balancer. These results show the efficiency and speed of load balancing over the different servers. The resulting graph displaying the gap between these operations and prove that load balancer is producing efficient and high speed performance and also it is seen that as number of servers increase then the speed and performance of the operations also increase.

4.2 Performing sorting function with load balancing on 10k Numbers

Here in the first experiment of this research, sorting function is applied on 10k random numbers in three different ways. Implementation is done on the single server, two servers and three servers which are having similar type of configurations. Here number of executions are five on all type of servers. After performing the operation time is calculated of all servers.

Table 4.2

Execution Time for Single Server, Two Servers & Three Servers

Experiment No.	Single Server	Two Servers	Three Servers
1	22.56	16.75	13.75
2	20.98	15.896	13.445
3	20.458	15.458	12.369
4	19.784	14.785	12.856
5	19.633	14.536	12.476

Table 4.2 presents the execution time of load balancing executed on all these servers. When load balancing is done on the single server the execution is much higher than as compare to that when load balancing is done on two servers and three servers. Maximum speed is achieved when there are three servers. Along with the speed accuracy is also maintained perfectly through the load balancer. These results show the efficiency and speed of load balancing over the different servers.

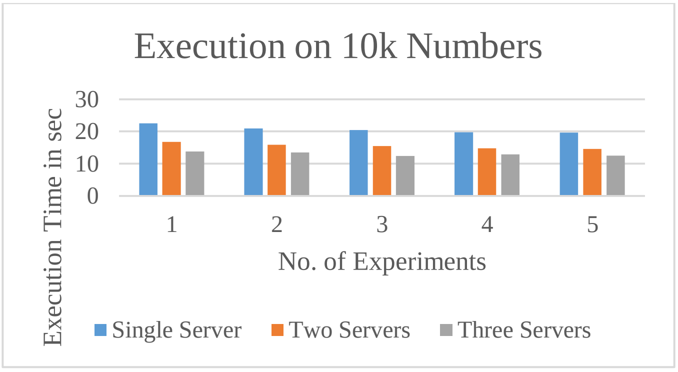

Figure 4.2: Execution Time for Single Server, Two Servers & Three Servers

From the figure 4.2 it is showing the execution time of load balancing executed on all these servers. When load balancing is done on the single server the execution is much higher than as compare to that when load balancing is done on two servers and three servers. Maximum speed is achieved when there are three servers. Along with the speed accuracy is also maintained perfectly through the load balancer. These results show the efficiency and speed of load balancing over the different servers. The resulting graph displaying the gap between these operations and prove that load balancer is producing efficient and high speed performance and also it is seen that as number of servers increase then the speed and performance of the operations also increase.

4.3 Performing sorting function with load balancing on 15k Numbers

Here in the first experiment of this research, sorting function is applied on 15k random numbers in three different ways. Implementation is done on the single server, two servers and three servers which are having similar type of configurations. Here number of executions are five on all type of servers. After performing the operation time is calculated of all servers.

Table 4.3

Execution Time for Single Server, Two Servers & Three Servers

Experiment No.	Single Server	Two Servers	Three Servers
1	29.784	22.698	17.488
2	28.636	22.865	16.589
3	28.411	21.569	16.48
4	27.96	21.856	16.963
5	27.458	22.58	16.865

Table 4.3 presents the execution time of load balancing executed on all these servers. When load balancing is done on the single server the execution is much higher than as compare to that when load balancing is done on two servers and three servers. Maximum speed is achieved when there are three servers. Along with the speed accuracy is also maintained perfectly through the load balancer. These results show the efficiency and speed of load balancing over the different servers.

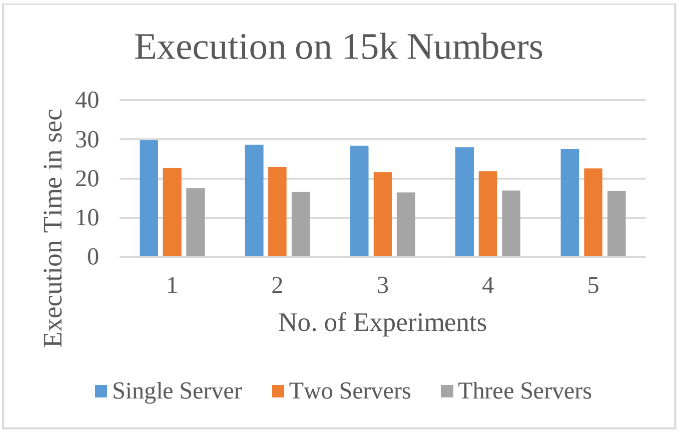

Figure 4.3: Execution Time for Single Server, Two Servers & Three Servers

From the figure 4.3 it is showing the execution time of load balancing executed on all these servers. When load balancing is done on the single server the execution is much higher than as compare to that when load balancing is done on two servers and three servers. Maximum speed is achieved when there are three servers. Along with the speed accuracy is also maintained perfectly through the load balancer. These results show the efficiency and speed of load balancing over the different servers. The resulting graph displaying the gap between these operations and prove that load balancer is producing efficient and high speed performance and also it is seen that as number of servers increase then the speed and performance of the operations also increase.

5. Conclusion

5.1 Conclusion

This research shows the load balancing working, how the load balancer distribute the load with using the round robin approach. It is very clear to see the results that show the load balancer is capable to distribute the load on the different servers.

Results prove that load balancer give the more efficient results in terms of speed and accuracy. The results prove that when the number of servers increase then the speed increases and accuracy also maintained.

It does not effect if some server crashes while execution or in middle of operation and then other server's execution is not affected by this and their operation will be executed continuously.

5.2 Future Scope

The present and future of this area is bright, and full of opportunities and great challenges as it processes high demands.

In future it can be used for the auto sharding for the high scalable demands.

References

[1] N. Ajith Singh, M. Hemalatha, "An approach on semi distributed load balancing algorithm for cloud computing systems" International Journal of Computer Applications Vol-56 No.12 2012

[2] Shanti Swaroop moharana, Rajadeepan d. Ramesh & Digamber Powar ," Analysis of load balancers in cloud computing" International Journal of Computer Science and Engineering (IJCSE)ISSN 2278-9960 Vol. 2, Issue 2, May 2013, 101-108.

[3] Gaurav R. et al. "Comparative Analysis of Load Balancing Algorithms in Cloud Computing." International Journal of Advanced Research in Computer Engineering & Technology, Vol. 1, No. 3, pp.120-124, May 2012.

[4] A.Khiyaita, M.Zbakh, H. El Bakkali & Dafir El Kettani ,"Load Balancing Cloud Computing : State of Art" IEEE Network Security and Systems (JNS2), 2012 National Days of , 978-1-4673-1050-5.

[5] Ektemal Al-Rayis, Heba Kurdi," Performance Analysis of Load Balancing Architectures in Cloud Computing" IEEE Modelling Symposium (EMS), 2013 European, 20-22 Nov. 2013, 978-1-4799-2577-3.

[6] Shridhar G.Damanal and G. Ram Mahana Reddy," Optimal Load Balancing in Cloud Computing By Efficient Utilization of Virtual Machines", IEEE 978-1-4799-3635-9/14

[7] D. Fernández-Baca: Allocating modules to processors in a distributed system, IEEE Transactions on Software Engineering, Vol. 15, No. 11, pp. 1427-1436 (1989).

[8] Suneel K S, Ravichandra A J and Dr. H S Guruprasad, "Enhanced load balancing algorithm in three tier cloud computing", International Journal of Engineering Sciences & Research Technology, December 2014

[9] Geethu Gopinath and Shriram K Vasudevan, "An in depth analysis and study of load balancing techniques in the cloud computing environment",2nd International Symposium on Big data and Cloud computing, 2015

[10] Rodrigo N. Calheiros, Rajiv Ranjan,Anton Beloglazov and Rajkumar Buyya,Cesar A. F. De Rose," CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms" Published online 24 August 2010 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/spe.995

[11] Suneel K S - Department of Computer Engineering, BMSCE Ravichandra A J - Department of Computer Engineering, ,BMSCE and Dr. H S Guruprasad - Department of Computer Engineering, ,BMSCE "Enhanced load balancing algorithm in three tier cloud computing",International Journal of Engineering Sciences & Research Technology,December- 2014