**Improvements in Load Balancing Algorithm**

**Sakshi Mishra(015721810)**

**Computer Software Engineering, San Jose State University, San Jose, California, sakshi.mishra@sjsu.edu**

**ABSTRACT**

Load Balancing is a method of distributing network traffic between different servers. It collects requests from clients and distributed those requests to a suitable server. In this paper, we will be figuring out the different algorithms that could be used by load balance to distribute the load between different servers and how one algorithm outperforms another algorithm in different scenarios. For detailed analyses, we will be comparing round robin and weighted round-robin algorithms and will try to improve the round-robin algorithm further.

**Keywords:**Load balancing, load, nodes, server, client

**1. Introduction**

There are different types of load balancing algorithms like the round-robin algorithm, Honeybee-based load balancing technique, Throttle load balancing algorithm, Random Sampling but they all are majorly divided into two subcategories i.e. Static load balancing algorithm and Dynamic load balancing algorithm.

[5]Type of Load-balancing algorithms :

**1.1. Static load balancing algorithm**

In Static load balancing algorithm load distribution happens based on predefined rules and methods. The algorithm is not concerned with the current state of the system and keeps on inundating services with requests. The advantage of this method is that the load balancer does not need to do any computation to distribute requests but this may also lead to any specific server being overloaded with requests which may slow down the efficiency of the whole application

**1.2. Dynamic load balancing algorithm**

In Dynamic load balancing algorithm load distribution happens based on the current state of the system. The algorithm usually computes and decided which server to direct request based on network traffic and the amount of load on each server. This method has the advantage when there is a heavy load and we want to evenly distribute requests on each server but decision making before each redirection can take time to decide perfect server.

There are a few very popular dynamic load balancing algorithms that we will discuss, compare and improve in this paper:

**1.2.1 Round Robin Algorithm**

In the round-robin algorithm, load distribution happens Request are redirected to servers in cyclic order. A queue of all available servers is registered at the load balancer. Once a request is received by the load balancer it redirects the request to the node present in front of the queue and the node that has freed up from computing its task is later added back to the queue for the next request.

**2.2.2 Weighted Round Robin Algorithm**

Round Robin algorithm works well in cases where all servers have the same capacity and can take a similar load but In a real-life scenario, all registered servers have a different load, and hence to distributing request according to the capacity of the server become necessary. For such a case, the Weighted round-robin algorithm can be used where the load is distributed unevenly based on the capacity of the server. The request is given out in round-robin but the capacity of each server is decreased in each iteration and servers with capacity zero are removed from the queue so that redirection does not happen to the server till it is again available for request handling.

**2.2.3 Throttled Load Balancing Algorithm**

According to [2] in this load balancing algorithm, the load balancer maintains an index table that keeps track of active servers and their current state as available or busy. Whenever a request comes to the load balancer it will iterate over the table to check the next available server, mark that server as busy, and redirect the request to that server. Once the request is served the state of that server is changed back to active.

**3 . Attempt to Improve Round Robin and Weighted Round Robin algorithm(Random choice of N random Nodes LB)**

In the round-robin algorithm, we are selecting nodes in a cyclic manner but this could lead to an issue where if the load on server A is more than server B but when the next request came we are still redirecting the request to server A which could lead to more computation time for system or even crash of server A. One way to improve this is to select nodes before redirecting requests to any server but this could lead to more computation time on the server. Another way is to predetermine load and select k nodes with the least current load from the complete set of n nodes and redirect requests in a round-robin to these nodes for a specific amount of time and after this time is over recompute to select k random nodes to perform these operations. The advantage of this method is that the probability of only a single node getting inundated with requests reduces as the set of nodes selected to do computation is always changing and computation time will not affect much as computation is happening on completion of specific intervals. This concept has been mathematically proven in [3] which says the probability of choosing 1 out of n uniformly and iteratively is higher than the probability of selecting k out of n and then selecting 1 out of this k.

**4 Experiment performed**

To compare the performance of round-robin, weighted round-robin, and random k round-robin algorithm fixed 6 nodes are given to each algorithm test and load is kept on increasing.

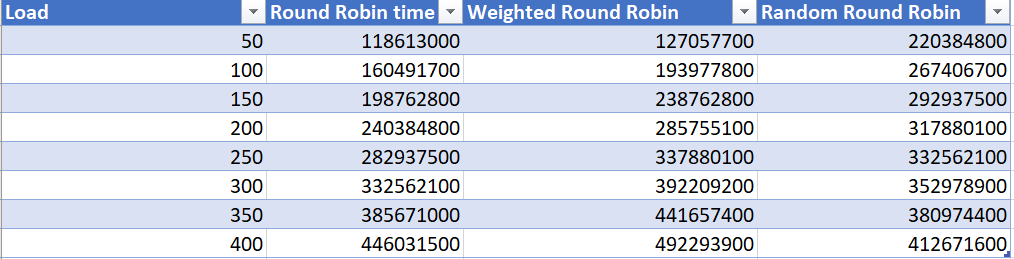
For the round-robin algorithm, each node is selected in cyclic order and each node is provided with a fixed load

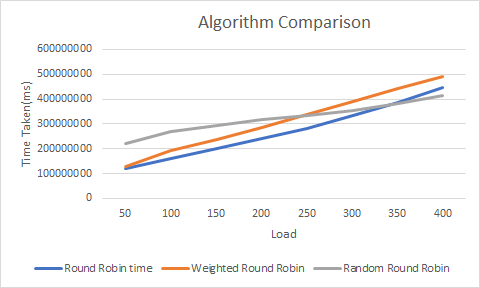
In the weighted round-robin algorithm, each node has a different capacity. So, node selection is a bit different. Node is selected in cyclic order and the capacity of the selected node is reduced. Once the capacity of the node is zero, the node is removed from round-robin distribution and the load is distributed on other nodes.

In a random k round-robin, nodes are selected randomly then randomly selected nodes are sorted based on their load and one with the least load is selected. This process is respected 6 times to select 6 nodes.

**4.1 Performance of Load Balancing Algorithms**

In this graph, we can observe that with an increase in load with the number of nodes being a constant increase in time taken by Random k is less than a round-robin and weighted round-robin algorithm. Since the extra time is taken to dynamically select n nodes is overcome by the amount of bottleneck happening on servers in the round-robin algorithm. Hence, We can say that if the load is less round-robin or weighted round-robin are a better choice but if request load is heavy then we should opt for a random k selection algorithm for load balancing.

****

****

**5. Conclusion**

We can observe that as the load on the system increases random selection is performing better but in less load round-robin is taking less time. The reason for such result is that in less load single node is not getting inundated with task and round-robin over-perform as it does not spend any time in selecting a set of k nodes but as load increases, time taken by round-robin also increases since node A with large load is getting inundated more with work which leads to increase in time, more than the amount required by the load balancer to compute k most efficient on specific intervals.

**6. References**

**[1]**[**https://www.researchgate.net/publication/277941229\_A\_Comparative\_Study\_of\_Load\_Balancing\_Algorithms\_in\_Cloud\_Computing**](https://www.researchgate.net/publication/277941229_A_Comparative_Study_of_Load_Balancing_Algorithms_in_Cloud_Computing)

**[2]**[**https://www.ijmter.com/papers/volume-2/issue-3/efficient-throttled-load-balancing-algorithm-in-cloud-environment.pdf**](https://www.ijmter.com/papers/volume-2/issue-3/efficient-throttled-load-balancing-algorithm-in-cloud-environment.pdf)

**[3]**[**https://www.eecs.harvard.edu/~michaelm/postscripts/handbook2001.pdf**](https://www.eecs.harvard.edu/~michaelm/postscripts/handbook2001.pdf)

**[4]**[**https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.1918&rep=rep1&type=pdf**](https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.1918&rep=rep1&type=pdf)

**[5]**[**https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.3487&rep=rep1&type=pdf**](https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.3487&rep=rep1&type=pdf)