**IS 651 Distributed System**

**Project Final Report**

**Load Balancing Techniques and Algorithms in a Distributed Systems**

**By**

**Sonal Ingle (YV16051)**

**Shubhi Shrivastava (NN46376)**

**Tanvi Kulkarni (UV 93494)**

**Karan Dhamecha (PS91095)**

**Instructor**

**Dr. Sisi Duan**

**The Graduate School**

**University of Maryland, Baltimore County**

**Table of Content**

1. **Introduction…………………………………………………………………………3**
2. **Scope…………………………………………………………………………………3**
3. **Load Balancing Algorithms……………………………………………………....4**
   1. **Round Robin………………………………………………………………...4**
   2. **Weighted Round Robin……………………………………………………5**
   3. **Least Connections………………………………………………………....6**
   4. **Weighted Least Connections …………………………..………………..6**
   5. **Random…………………………………………….………………………...7**
4. **Approaches to Improve Load Balancing...………………………….…………7**
5. **Proposed Design…………………………………………………………………...9**
   1. **Flow Chart…………………………………………………………………...11**
   2. **Application of Proposed Design…………………………………………11**
   3. **Advantages and Limitations……………………………………………...12**
6. **Conclusion…………………………………………………………………………..12**
7. **References…………………………………………………………………………..13**
8. **Introduction:**

Load balancing can be described as intelligent data distribution (workloads) between multiple computing services (server pool) to provide reliability, redundancy, and performance enhancement. A load balancer acts as your server's "traffic manager," routing client requests through all available servers fairly ensuring that no server is overwhelmed which could reduce performance. It also ensures that no server is idle and gives server add-on or subtraction flexibility as per demand. Different types of load balancing algorithms are meant for different benefits and the choice depends on your needs (Khan, Haroon, & Husain, 2015). A load balancer coupled with at least one more additional server is required to implement load balancing. Algorithms differ widely depending on the distribution of the load, whether it's on the network or application layer.

In our project we are discussing how load balancer plays a crucial role in distributing incoming network traffic across a group of backend servers for network and application layers and what are the different algorithms it uses for doing so. Load balancing algorithms falls in two main categories – weighted and non-weighted algorithms. Weighted algorithms use a calculation based on weight or preference to make the decision for example servers with more weight receives more traffic (Khan et al., 2015). Whereas non-weighted algorithms make no such distinctions, instead it assumes that all the servers have the same capacity. We will be discussing in detail about various algorithms that are implemented at Network layer which includes Round Robin, Weighted Round Robin, Least Connection, Weighted Least Connection and Random Algorithms.

Also, we have proposed a designed algorithm that improves the efficiency of Weighted Least Connections Algorithm by adding a scalability parameter. Using this parameter, you can scale up the number of servers when you experience huge amount of traffic and you can scale it down when the traffic is low. It uses Binding Method for scalability purpose. Hence, this model not only provides scalability but also improves other factors such as performance and latency and hence making it cost efficient. Thus, we can say that depending upon the scenario or the type of business different load balancing algorithms can be implemented and each of these algorithms has its own advantages and disadvantages.

**2. Scope:**

One of the key features of using a load balancer is that it ensures availability of services. Making sure that your device and services are up and running. Its advanced traffic management functionality can help a business steer requests more efficiently to the correct resources for each end user. There are many different use cases for a Load Balancer (LB). It is important to know how to effectively manage your LB configuration so that it performs optimally in your environment (Shukla & Suryavanshi, 2019). There are various factors that we must take into consideration before choosing the load balancing algorithms. These factors include business environment, business budget, volume of traffic etc. However, when faced with several good options, you still have to decide which one is best suited for your organization. Obviously, there is no one-size-fits-all solution. Still, diligent evaluation can lead you to the best solution for your environment. For instance, simple load balancing algorithms such as round robin, random or least connections can be implemented when all the backend servers have same specifications such as CPU and Memory. More sophisticated load balancers may take additional factors into account, such as a server's reported load, least response times, up/down status, number of active connections, geographic location, capabilities, or how much traffic it has recently been assigned. In such scenarios we can use weighted least connections algorithm which take these parameters into consideration while distributing the traffic (Shukla & Suryavanshi, 2019).

Choosing a load balancing platform can be a complex undertaking requiring a comprehensive understanding of the business content and application delivery infrastructure. The method is further complicated by the overwhelming array of options available. The aim is to choose a solution only after taking full account of the impacts of future growth and scalability in terms of efficiency, availability, performance and security.

**3. Technical Details:**

In this section, we will talk about the various load balancing algorithms we came across. Given below are the characteristics and drawbacks of these techniques.

**3.1** **Round Robin:**

It is the simplest to implement technique for load balancing. Therefore, it is widely used. The requests are assigned in a rotating sequential manner to the servers.

As seen in the diagram, client requests from 1 is assigned to server 1 then the net request goes to server 2. Similarly, the next request goes to server 1 and so on. This approach works well if the servers are of similar configurations. The round-robin approach is not suitable in case of disproportionate capacity servers. This is where weighted round-robin comes into the picture.

A close up of a map

Description automatically generated

Figure 3.1 Round Robin

**3.2** **Weighted** **Round Robin:**

In weighted round-robin, nodes are assigned a certain weight as per their CPU and RAM capacity. So, in this way, nodes with higher capacity are assigned a higher weight and thus receive more requests from clients.

Also, capacity is not the only criteria. Sometimes we need a lower number of connections for one server than an equally capable server so that we can run some other applications on that server and this way we can avoid it from getting overloaded.

As we can see from the diagram, server 1 and server 2 have weights assigned as 5 and 1 respectively, so the first 5 client requests will be assigned to server 1 and then when there is a new request it will go to server 2.

A close up of a map

Description automatically generated

Figure 3.2 Weighted Round Robin

**3.3** **Least Connections:**

Now in some cases servers may have similar specs but can still get overloaded faster than others because clients stay connected to a particular server for too long. Other servers, as shown in the fig. maybe having clients connecting and disconnecting. This way the total connections virtually remains the same. The total connections to a server 2 may get stacked up.

So, in this situation we have the Least Connection algorithm in which we consider the current number of connections each server has.

From the diagram, we can see client 1 and client 3 were initially assigned to server 1 but disconnected. Now server 1 has lesser current connections than server 2. If a new client tris to connect after 1 and 3 have already disconnected, then it will be assigned server 1 instead of server 2.

A close up of a map

Description automatically generated

Figure 3.3 Least Connections

**3.4** **Weighted Least Connections:**

An improvement to the least connection is the weighted least connection. It takes into consideration two things: weights or capacity of each server as well as the number of clients currently connected to each server. So, if two servers have the same number of active connections but one has more capacity than the other, then the next request is assigned to this one. As shown in the diagram, clients 1 and 3 have disconnected and so the current connections are the same for both. Here, the one with a higher weight will get the request.

A close up of a map

Description automatically generated

Figure 3.4 Weighted Least Connections

**3.5** **Random:**

In this algorithm, the clients and servers are matched based on a random number generated. This is good where we have a large number of client requests coming up. In this case the requests will be assigned evenly to the servers. So like round-robin this is better when we have servers with identical specs and configurations and the drawback is it may crash when there are servers of different capacities.

A close up of a map

Description automatically generated

Figure 3.5 Random

**4. Approaches to Improve Load Balancing:**

Web clusters contain load balancers for managing web requests and distributing them to various servers. We analyzed several load balancing algorithms as described above; then, we realized none of the conventional algorithms such as least connections, weighted least connections, round-robin, weighted round-robin, and random does not provide scalability. We propose that the Least weight connections algorithm for load balancing offers scalability to the load balancing system. It is an improvement to the weighted least connection scheduling algorithm for load balancing. All the previous load balancing algorithms do not allow us to add a new server or new IP addresses or release existing servers or old IP address dynamically; we need to restart the load balancer while uploading the changes to it. LWC balancing algorithm provides an extra element of scalability (Zhu, Cui, & Xiong, 2018).

We also inherit all the advantages of weighted least connections; our LWC load balancing algorithm provides an excellent performance better than most of the algorithms present, for instance, Random and round-robin load balancing algorithms that provide average or low performance. Our throughput is better than most of the algorithms with less overhead time.

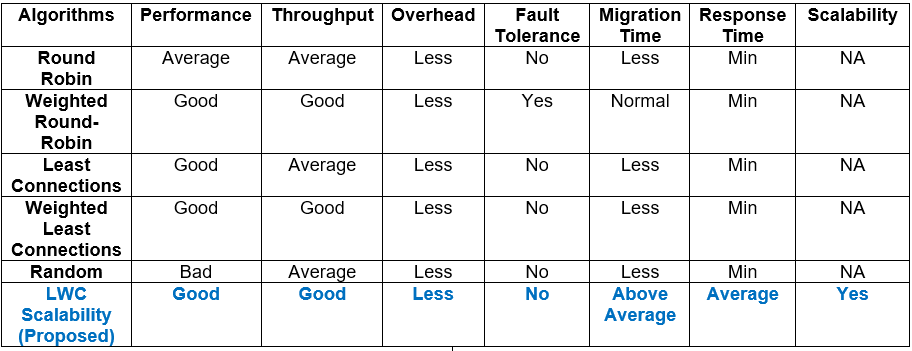


Figure 4.A Comparison Chart

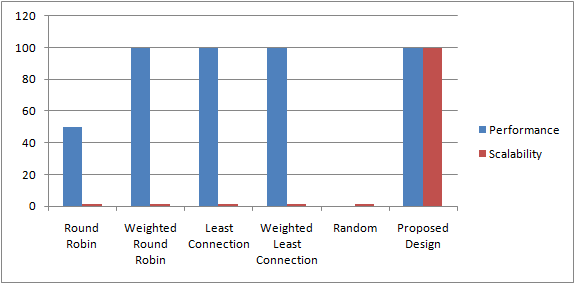


Figure 4.B Graphical Comparison of Algorithms (Performance and Scalability)

**5. Proposed Design:**

The weighted least connection is an advancement over the least connection load balancing technique. In weighted least connection, the server with more weight value will receive a larger percentage of active connections at a time. The server with fewer connections will be granted the new request. If more than one server has equal connections which are least then the algorithm will check the weight of the server. The node with the highest weight will be given the request. The metrics upon which the weight of the server is decided are connecting time and first response time, transfer time, throughput, errors at the network level, etc. Weight distribution is a simple method to deal with heterogeneous frameworks which may emerge after adding new back end servers and has different hardware configuration as compared to existing servers. The weight data ordinarily reflect the handling limits of a back end server and the corresponding application running on. Our proposed system adds a new server whenever there is a resource crunch and assigns weight to it. None of the nodes are overloaded in the system because of this scalable system. Hence the probability of a system crash becomes negligible.

Our proposed work is designed for n nodes(N1,N2…Nn). Considering Load = L, CPU = C, Memory = M, bandwidth = B, Disk usage = D. Prerequisites for ith node are:

● The weight for node Ni is W=ai, which is pre-assigned to each machine. It is calculated on the basis of machine parameters C,M,D,B.

● The threshold T will be set for each machine depending on the machine capacity.

● Server capacity is pre-calculated on all the nodes based on the parameters C,M,B,D.

Supposing there is a server set S = {N0, N1, ..., Nn-1}, W(ai) is the weight of server Ni; C(Ni) is the current connection number of server Ni; CSUM = ΣC(Ni) (i=0, 1, .. , n-1) is the sum of current connection numbers. The new connection is assigned to the server j, in which

(C(Nm) / CSUM)/ W(Nm) = min { (C(Ni) / CSUM) / W(Ni)}  (i=0, 1, . , n-1), where W(Ni) isn't zero.

Since the CSUM is a constant in this lookup, there is no need to divide by CSUM, the condition can be optimized as

C(Nm) / W(Nm) = min { C(Ni) / W(Ni)}  (i=0, 1, . , n-1), where W(Ni) isn't zero

Since division operation eats much more CPU cycles than multiply operation, and Linux does not allow float mode inside the kernel, the condition C(Nm)/W(Nm) > C(Ni)/W(Ni) can be optimized as C(Nm)\*W(Ni) > C(Ni)\*W(Nm). The below pseudo code is only to check the weight of the server and return the server id which is picked for the particular request. The algorithm will not assign any request to the node if its weight is zero. The condition C(Nm)\*W(Ni) > C(Ni)\*W(Nm) is similar to the condition L(Ni)<Ti in our flowchart. The load L(Ni) is calculated on the number of connections i.e. C(Ni) and Ti is the threshold which depends on the weight W(Ni). The below algorithm is only a part of our proposed algorithm which returns server id for the node with least connection. The advancements like binding method and adding new servers have been represented in the flowchart.

for (m = 0; m < n; m++) {

if (W(Nm) > 0) {

     for (i = m+1; i < n; i++) {

         if (C(Nm)\*W(Ni) > C(Ni)\*W(Nm))

             m = i;

     }

     return Nm;

}

}

return NULL;

**5.1** **Flow Chart:**

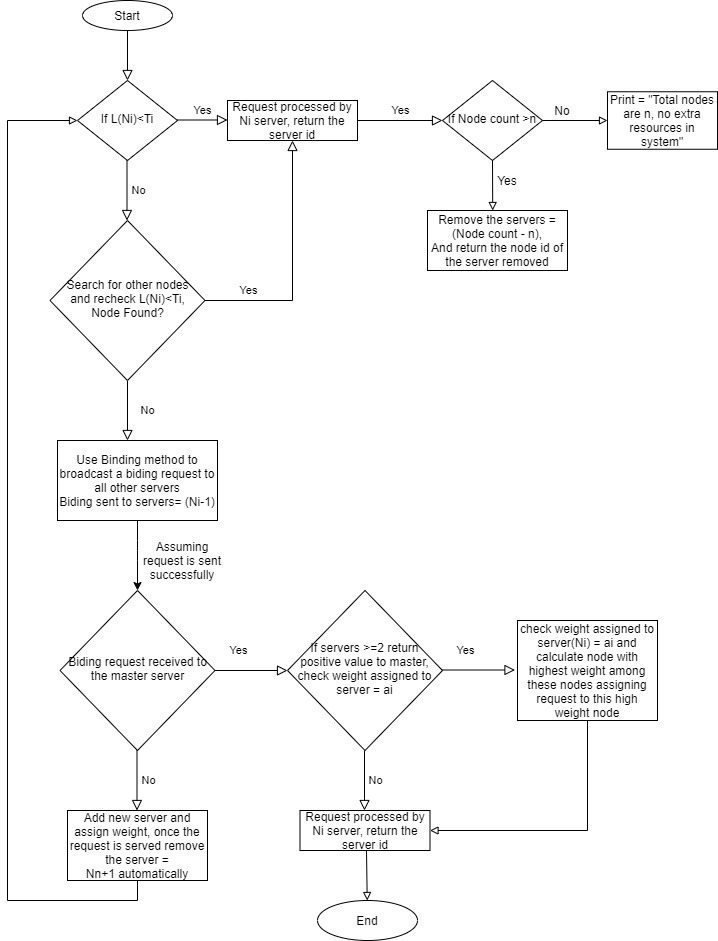


Figure 5.1 Flow Chart (Proposed Design)

**5.2** **Application of Proposed Design:**

The major application of our proposed design will be used is e-commerce. Our proposed design is majorly used when the company has a sudden exponential increase in the customer’s requests. During the “Black Friday Sale”, the retailers like Walmart, Kohl's, Macy's, Amazon, Target, and Best Buy have to handle huge requests. These increased requests demand more resources to function without any issues. This design allows the scalability of the system increasing the throughput of the system. Whenever the number of requests increases more than the threshold of each node, then there is a requirement of more resources. Hence a new server is added from the pool into the system which will be ready to handle further requests. Once the customer is done with online shopping, the extra resources are not required. Once the request is complete, the extra server is removed. This minimizes the cost of the system while providing extra resources.

The response time is a little high as compared to others because after the addition of new servers the local communication time will increase slightly. The other factors like performance and latency are improved in this model with scalability as an additional factor. Hence while serving such huge requests, the probability of the system going offline is negligible.

**5.3** **Advantages and Limitations:**

Due to the scalability factor added in the proposed design, the probability of system crash   is negligible. The overall performance of the system will be good. Although the system compromises over the response time and migration time compared to other techniques, this system is great when the application demands the its working 24\*7 and cannot afford the system crash at any point of time.

**6. Conclusion:**

There are various algorithms used for Load Balancing, Round robin, weighted round-robin, Least Connection, Weighted least connection, Random. This Paper explains how Least weight connection is a useful Load balancing technique to improve Scalability. We have presented our proposed Solution and its Design and evaluation of Least weight connection for those organizations which have exponentially increased in sales. We also compare these algorithms based on different aspects such as performance, throughput, response time, etc. And we can say each proposed algorithm has its own advantages and disadvantages. For example, for the web-based applications, a simple round-robin technique would be enough to handle the download requests for the static web pages. Whereas in applications for example real-time interactive applications, where the requests take more time than the others to process, advanced load balancing algorithms such as Least Connections or Weighted Least Connections can be used.

Whatever load balancing technique we choose for an application, the chosen Algorithm should be an efficient technique to handle the workload of the application in an efficient way. Several opportunities exist for potential future work. These include: evaluating our algorithms on large Scale to further test their scalability; adding a fail-over mechanism to ensure that the load balancer is not a single point of failure; and looking at other algorithms workloads.

**7. References:**

Khan, R., Haroon, M., & Husain, M. S. (2015, 23-24 April 2015). *Different technique of load balancing in distributed system: A review paper.* Paper presented at the 2015 Global Conference on Communication Technologies (GCCT).

This paper presents a characteristic analysis and pros and cons of different types of dynamic load balancing techniques on the basis of the location of decision making, information used for decision making, scalability factor and overhead of profile switching.

The Centralized technique presents that the responsibility of load balancing stays with the master node while other information is gathered from other nodes known as slave nodes. In Distributed Non-Cooperative technique, accountability is distributed over all the working nodes or workstations instead of a single master. The info load is on-demand to provide more scalability but may increase interconnection traffic and info exchange overhead. In Distributed Cooperative Optimal technique, the decision of load balancing is disseminated over all workstations instead of master nodes. This provides more efficient scalability as load balancing info strategy is dependent on demand. Distributed Cooperative Semi Optimal Heuristic technique presents that the accountability of load balancing decision is assigned over all the workstations together with demand-driven information strategy and average profile information exchange transparency and moderate scalability. Distributed Cooperative Semi Optimal Approximation Dynamic technique, the load balancing accountability, information strategy and scalability remains same but there is an uncontrollable profile information exchange overhead which increases the traffic over the interconnection networks.

The issues in performance evaluation of these techniques include measuring the resource workload, criteria to define this workload, how to avoid it’s harming effects of resources dynamicity towards workload and determining the heterogeneity of resources to get the average workload of the system. The paper concludes by discussing wide scope and research potential, and the need for automation in routing and other areas.

Shukla, S., & Suryavanshi, R. (2019). *Survey on Load Balancing Techniques*.

We are reviewing this paper to learn about the different algorithms used in load balancing. As load balancing is very important in distributed systems, it is very important to choose the correct technique for load balancing to reduce the system load with minimum use of resources. In this paper, we will learn about various load balancing techniques with the help of node movement and replication. To manage the proper distribution of load we have various techniques that will help to improve the response time and resource utilization. Load balancing is generally placed in the network layer for proper load distribution. We will discuss static and dynamic load balancing algorithms and will compare these algorithms. From this paper, the static algorithms - Round Robin Algorithm, Randomized Algorithm, Central manager Algorithm and dynamic algorithms - Nearest Neighbor Algorithm, Dynamic ratio Algorithm, Least connection Algorithm are being extracted for the studying and comparing these algorithm techniques. (Shukla & Suryavanshi, 2019). This study will help us to decide the algorithm to be used in our project implementation.

Zhu, L., Cui, J., & Xiong, G. (2018, 14-16 Dec. 2018). *Improved dynamic load balancing algorithm based on Least-Connection Scheduling.* Paper presented at the 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC).

The paper proposes a dynamic load balancing algorithm with a collection server static load factor and dynamic load factor on the basis of the least connection algorithm. It will be used for calculating the weight of the closest server real-time performance parameters, introduced the response time and the number of connections in statistics class load factor and calculate the current server load. Classical load balancing algorithms include round-robin algorithm, weighted round-robin algorithm, random allocation algorithm, and dynamic feedback schedule. In this paper, the load efficiency and the load of the node are determined on the basis of the load information gathered and the weight of the node is measured. Ultimately, the cumulative load will be determined based on the reaction time and the number of node connections. The larger the weight of the node, the smaller the composite load, the more likely it is to be allocated to the order. The study of the weight allocation to each node will be extracted from this paper to include in proposed solution (Liangshuai Zhu, 2018).