IS 651: Distributed Systems
Project Progress Report

1. **Team Members:**
   - Sonal Ingle (YV16051)
   - Shubhi Shrivastava (NN46376)
   - Tanvi Kulkarni (UV93494)
   - Karan Dhamecha (PS91095)

2. **Topic:** Load Balancing Techniques and Algorithms

3. **Type:** 2 (Review and Design)

4. **We have achieved so far:**

   We have reviewed **3** papers associated involving basic concepts of distributed file systems, security issues and methodologies to improve the security. We have summarized each paper in detail as below.

5. **Project Description:**

   In our project we are discussing how load balancer plays a crucial role in distributing incoming network traffic across a group of backend servers and what are the different algorithms it uses for doing so.
   Different load balancing algorithms provide different benefits and depending upon the situation you can choose the one that is best suited in that situation. 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. 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 Source IP Hash. We will be implementing these algorithms in Python and then in the end compare the results.

6. **Review of Papers:**

   a. **Paper Name:** Improved dynamic load balancing algorithm based on Least-connection Scheduling.

      Summary: 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,
      - calculated the current server load.

      It is found that the improved load balancing strategy makes more efficient use of resources in the server cluster and which improves the throughput and processing capacity of the system.

Classical load balancing algorithms include round-robin algorithm, weighted round-robin algorithm, random allocation algorithm, and dynamic feedback schedule.

Nonetheless, with the advancement of load balancing technology, several new load balancing algorithms have been suggested by scholars at home and abroad.

The load weight of the node is determined by the ratio of the load of the node to the output of the node, such that the function is assigned by the weight of the load, proposed a consistent hash load balancing algorithm based on critical acceleration and decrement.

This algorithm uses virtual nodes for actual server node allocation and accelerates the decrement of the critical factor mech.

In addition, some researchers apply genetic algorithm, particle swarm algorithm, and ant colony algorithm to swarm intelligence technologies to load balancing problems. For example, the advantages of the genetic algorithm and the simulated annealing algorithm for the load balancing of server clusters are considered.

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.

Therefore, based on the minimum value of the composite load and the weight ratio of the nodes in the back-end cluster, a node is chosen to forward the order. When there are several nodes with a minimal weight, the principle of round robin is followed. Choose a node from it.

For Nginx's built-in load balancing algorithm, there is a concern of unbalanced load delivery. Centered on the least conn, an advanced load balancing algorithm is proposed to enhance conn. The weight of the node WE is determined on the basis of the static load factor and the dynamic load factor which is measured on the basis of the mathematical load factor (Liangshuai Zhu, 2018).

b. Paper Name: Survey on Load balancing techniques

Summary: 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 analyze the differences between them. This study will help us to decide the algorithm to be used in our project implementation.

Static Load balancing: This load balancing technique deals with providing prior information about the system. The node calculates their assigned work and informs the remote node, post which the workload is distributed. There are 6 algorithms but we will discuss the 3 preferable ones that are categorized under static:

1. Round Robin Algorithm: This algorithm deals with ring-type allocation which has multiple servers and clients in a system. This has 2 types namely Classic round-robin and weighted round-robin algorithm. In the classic method, a specific time slice is allocated to each process while weighted, weight is given to every node depending on its

configurations. This algorithm is very easy for implementation and has minimum internal communication between the processes. Hence this is good for special-purpose algorithms. But its disadvantage is that if the request has an unequal processing time, it does not offer good performance.

2. Randomized Algorithm: In this method, the node is selected randomly and the load is allocated to it. This algorithm works best with the system which has nodes with equal nodes. As the node does not have information about the current or previous load, the load may be distributed unevenly causing performance issues on the system.
3. Central manager Algorithm: This has a master-slave configuration where the master node allocates the work to the node with the least load. When the load changes, the slave nodes communicate with the master. Hence this requires more internal communication between the processes. This algorithm works best where we have a system when the loads on the nodes are changing frequently. This has a disadvantage that it cannot work for large systems as it is dependent only on a central node for load distribution.

Dynamic Load balancing: The dynamic load balancing works with multiple nodes and monitors the changes in the nodes and allocates the job accordingly. All nodes know its neighboring nodes and failure of a node is taken care of in this algorithm. Hence this offers the fault-tolerant distribution of jobs. The dynamic load balancing offers 18 types of the load distribution from which we will discuss 3 types which are more preferable than others based on factors – implementation, proper distribution with minimum time, inter-process communication, etc.

1. Nearest Neighbor Algorithm: In the nearest neighbor algorithm, the nodes will look into their neighboring node, if succeeded to find the overloaded node then the load is distributed to the node having less node. This algorithm is very easy to implement and the execution time is very less. But it has a disadvantage that it can miss the shortest path to transfer load. The communication is locally between the nodes, hence faster response is guaranteed with this algorithm.
2. Dynamic ratio Algorithm: In this algorithm, we will analyze the node capacity and how much load it can handle. The central unit stores the node RAM and its capacity in a table. The load distribution is done about this table according to the dynamic ratio. Each time a request is processed the ratio changes. The response time for this algorithm is not good but it offers fault tolerance.

3. Least connection Algorithm: In this algorithm, the load balancer keeps the information about the number of connections on each node. The distribution is done here according to the number of connections. The node having fewer connections will be allocated the request first. This algorithm offers fault tolerance, but it does not take into consideration the node capacity which might lead to overloading a node having low capacity (Shukla & Suryavanshi, 2019).

c. Paper Name: Different Technique of Load Balancing in distributed system: A Review Paper

Summary: 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.

1. 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.
2. 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 basis i.e. whenever any node tries to change its current balanced working state to an overloaded

state, the specific node distributes the load information to rearrange. This provides more scalability but may increase interconnection traffic and info exchange overhead.

3. 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.
4. 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.
5. 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 (Riyazuddin Khan, 2015).

7. Design goals and requirements:

Load balancing plays an important role in the security domain. It prevents the organization from distributed denial-of-service (DDoS) attacks. We will focus on the load balancing technique considering the security factor. SSL is the standard security layer which helps to encrypt the data between the server and the web browser. The data is decrypted either at load balancer or web server. We will consider the security factor as well while designing the algorithm at load balancer end.

References

Liangshuai Zhu, J. C. (2018, 12 14). *Improved dynamic load balancing algorithm based on Least-Connection Scheduling.* Retrieved from IEEE: https://ieeexplore.ieee.org/document/8740642.

Riyazuddin Khan, M. H. (2015). *Different Technique of Load Balancing In Distributed System: A Review Paper.* Retrieved from IEEE: https://ieeexplore-ieee-org.proxy-bc.researchport.umd.edu/stamp/stamp.jsp?tp=&arnumber=7342686&tag=1

Shantanu Shukla, R. S. (2019, 02). *Survey on Load Balancing Techniques.* Retrieved from Researchgate: https://www.researchgate.net/publication/331034889_Survey_on_Load_Balancing_Techniques

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 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 lest 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;

Applications:

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.

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.