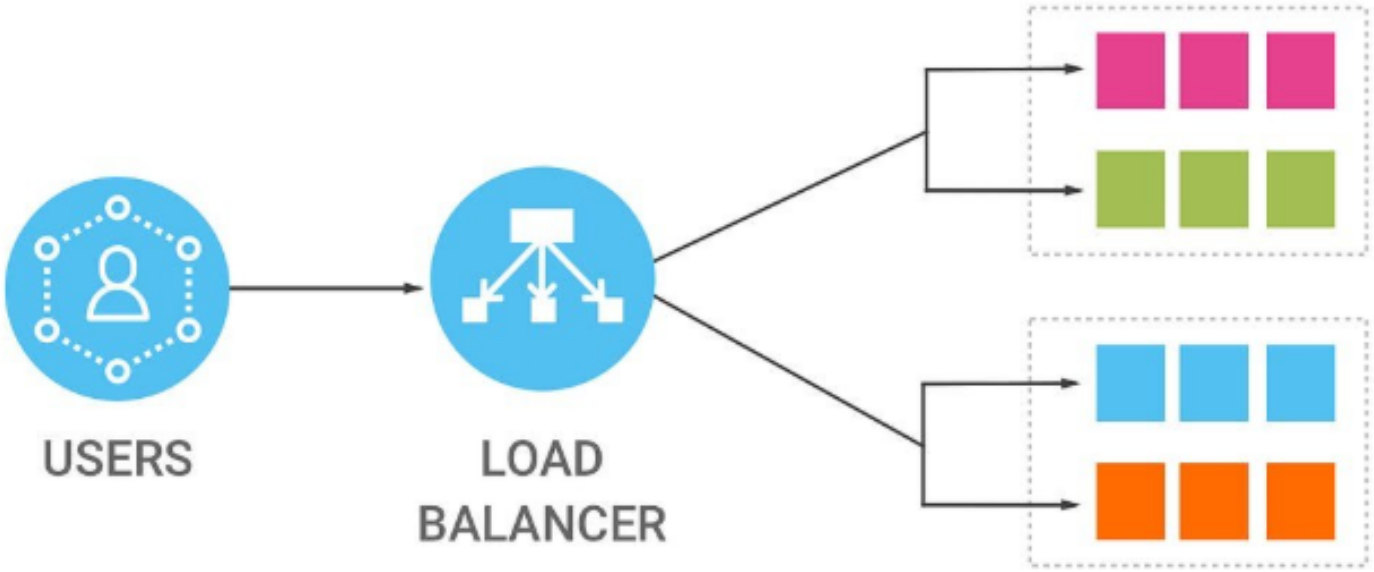
Mapping Techniques for Load Balancing in Parallel and Distributed Systems

Load balancing is a computer networking method designed to distribute incoming network traffic or computing workloads across multiple servers or resources that supports an application.

What is Mapping in Load Balancing?

Mapping refers to the process of assigning tasks or data to processing units (e.g., CPUs, GPUs, or nodes) to distribute the workload. Effective mapping helps in:

- Maximizing performance by utilizing resources efficiently.
- Minimizing idle time by distributing tasks evenly.
- Reducing communication overhead in distributed systems.



Load Balancing Technologies Types

There are two kinds of load balancers: hardware load balancers and software load balancers.

Hardware load balancers

A hardware-based load balancer is a hardware appliance that can securely process and redirect gigabytes of traffic to hundreds of different servers. You can store it in your data centers and use virtualization to create multiple digital or virtual load balancers that you can centrally manage.

Software load balancers

Software-based load balancers are applications that perform all load balancing functions. You can install them on any server or access them as a fully managed third-party service.

Load Balancing Benefits

Load balancing plays a pivotal role in creating a robust and high-performing IT infrastructure, offering benefits such as enhanced performance, scalability, fault tolerance, resource utilization, and an overall improved user experience.

Load balancing ensures that incoming tasks or network traffic are distributed evenly among multiple servers. This prevents any single server from becoming overloaded, optimizing overall system performance and response times.

As user demands increase, load balancing facilitates the seamless addition of new servers or resources to the network. This scalability ensures that the system can handle growing workloads without compromising performance.

Load balancers enhance system reliability by redirecting traffic away from failed or underperforming servers. In the event of a server malfunction, the load balancer automatically routes traffic to healthy servers, minimizing downtime and maintaining service availability.

Load balancing provides an additional layer of security for your internet applications, offering built-in security features to safeguard against various threats. Particularly adept at handling distributed denial of service (DDoS) attacks. Load balancers also offer traffic monitoring and malicious content blocking and can be integrated with network firewalls.

Load Balancing Algorithms

Load balancing algorithms come in two main flavors: static and dynamic. In the static approach, servers are assigned fixed weights, and traffic distribution remains constant over time. On the other hand, dynamic algorithms adapt to changing conditions,

continuously monitoring server performance, and adjusting traffic allocation in real-time.

Static load balancing

Round robin: Round robin distributes incoming traffic or requests evenly across a set of servers. Each server in the pool takes turns handling a request, and the algorithm cycles through the servers in a circular order.

Round Robin is a simple and widely used load balancing algorithm that **distributes incoming traffic evenly** across multiple servers.

Each server handles requests **in turn** following a **circular order**.

This ensures that no server is **overloaded** while others **remain idle**, providing **fair resource utilization** and **improving system performance**.

Scenario: A system has **three servers (S1, S2, S3)** and **nine incoming requests (R1 to R9)**.

Request	Assigned Server
R1	S1
R2	S2
R3	S3
R4	S1
R5	S2
R6	S3
R7	S1
R8	S2

Example of Weighted Round-Robin Distribution

Scenario:

Three servers (**S1, S2, S3**) have **weights** assigned based on their **processing capacity**:

- **S1:** Weight **4** (highest capacity)
- **S2:** Weight **3**
- **S3:** Weight **2**

Requests: 9 requests (**R1** to **R9**) need to be distributed.

Weighted Distribution Calculation

- S1: 4/(4+3+2) = 4/9 → ~44% of requests → 4 requests
- S2: 3/9 = 33% → 3 requests
- S3: 2/9 = 22% → 2 requests

Request	Assigned Server
R1	S1
R2	S2
R3	S3
R4	S1
R5	S2
R6	S1
R7	S3
R8	S1
R9	S2

Final Allocation:

- Server S1: 4 requests (R1, R4, R6, R8)
- Server S2: 3 requests (R2, R5, R9)
- Server S3: 2 requests (R3, R7)

Dynamic load balancing

Least connections: The Least Connections load balancing algorithm directs incoming traffic to the server with the fewest active connections. The rationale behind this approach is to distribute new connections to servers that are currently less busy, promoting a more balanced distribution of the workload.

Least response time: The Least Response Time load balancing algorithm selects the server with the fastest response time to handle incoming requests. The goal is to route traffic to the server that can process requests most quickly, thereby minimizing the time it takes to complete a transaction or deliver content.

Example for Least Connections Load Balancing Algorithm

The **Least Connections** algorithm directs incoming traffic to the server with the fewest active connections. Let's work through a mathematical example to understand how it distributes traffic.

Example Scenario:

Suppose we have **3 servers** handling incoming requests. The current active connections on each server are:

- **Server A**: 10 active connections
- **Server B**: 5 active connections
- **Server C**: 7 active connections

Now, let's assume **5 new incoming requests** need to be distributed.

Least Connections Rule

new request is assigned to the server with the fewest active connections at each step.

Step	Server A (10)	Server B (5)	Server C (7)	Assigned to
1	10	6	7	Server B
2	10	7	7	Server B
3	10	7	8	Server C
4	10	8	8	Server B
5	10	8	9	Server C

Final Distribution After 5 New Requests

After distributing 5 new requests using the **Least Connections** method, the updated active connections on each server are:

- **Server A**: **10** connections
- **Server B**: **8** connections
- **Server C**: **9** connections

Observation:

- **Server B** received 3 new requests since it had the lowest number initially.
- **Server C** received 2 requests because, after two assignments to Server B, Server C became the next lowest.
- **Server A** was not assigned any requests because it started with the highest number of active connections.

Selecting the Server

To determine the server for a new request, we use:

where:

$$S = \arg\min(C_i)$$

- S is the selected server
- C_i is the number of active connections on server i

Each new request is assigned to the server with the smallest C_i.

Example for Least Response Time Load Balancing Algorithm

The **Least Response Time** algorithm directs incoming traffic to the server with the **fastest response time** (i.e., the lowest latency). This ensures that requests are processed quickly and efficiently.

Example

Suppose we have **3 servers** with different response times and active connections:

Server	Current Active Connections	Response Time (RT) in ms
Server A	12	250 ms
Server B	8	150 ms
Server C	10	180 ms

Now, assume **4 new incoming requests** need to be assigned using the **Least Response Time** strategy.

Assign Requests Using the Least Response Time Rule

- The request is **always assigned to the server with the lowest response time.**

Step	Server A (250 ms)	Server B (150 ms)	Server C (180 ms)	Assigned To