Load Balancing

Load Balancing is a technique used in computer networks and distributed systems to distribute incoming network traffic or requests across multiple servers or resources. The goal is to ensure no single server becomes overwhelmed, improve performance, reduce latency, ensure high availability, and achieve fault tolerance.

How it works

1. **Client Request**: A user device sends a request to an application or website.
2. **Load Balancer Intercepts**: Instead of going directly to a server, the request is received by a load balancer.
3. **Server Selection**: The load balancer analyzes incoming requests and the current status of its connected servers.
4. **Traffic Distribution**: It then forwards the request to an available and appropriate server within the server group.
5. **Response**: The selected server processes the request and sends the response back to the load balancer, which then forwards it to the client.

Benefits of Load Balancing

- **Improved Performance**: By dividing the workload, load balancers reduce the burden on individual servers, resulting in faster response times and lower latency for users.
- **Increased Availability**: If one server fails, the load balancer automatically reroutes traffic to the remaining operational servers
- **Enhanced Scalability**: Load balancing allows businesses to add or remove servers from the pool as demand changes
- **Optimal Resource Utilization**: It prevents individual servers from becoming overloaded while others remain idle

Load Balancing Strategies

1) Round Robin

Definition:
Round Robin is a simple and widely used algorithm. It distributes client requests sequentially across the server pool in a circular manner.

How it Works:

- Each new request is sent to the next server in line.
- After reaching the last server, it loops back to the first.
- Does not consider the current server load, just distributes evenly.

Example:
Assume 3 servers: S1, S2, S3
Requests 1 → S1
Request 2 → S2
Request 3 → S3
Request 4 → S1
...and so on.

Pros:

- Simple to implement.
- Works well when all servers have equal capacity and requests take roughly equal time.

Cons:

- Doesn’t account for load or response time.
- Not ideal if server performance varies or requests are uneven.

2) Least Connections

Definition:
This method forwards incoming requests to the server with the fewest active connections at the time.

How it Works:

- The load balancer maintains a count of active connections per server.
- New requests are sent to the server with the least number of current connections.

Example:
If:

- S1 has 5 active connections
- S2 has 2 active connections
- S3 has 3 active connections

The next request goes to S2.

Pros:

- Balances traffic based on current server load.
- Ideal for long-lived or uneven request workloads.

Cons:

- Slightly more complex to implement.
- Can cause uneven distribution if not combined with capacity awareness.

Variation: *Weighted Least Connections* — where servers are assigned weights based on capacity.

3) Random

Definition:
Random load balancing sends each request to a randomly selected server from the pool.

How it Works:

- Each new request is assigned to any server using a random number generator.
- No pattern, just probabilistic distribution.

Example:
With servers S1, S2, S3:

- Request 1 → randomly S2
- Request 2 → randomly S1
- Request 3 → randomly S3

Pros:

- Very easy to implement.
- Sometimes useful in large distributed systems with many nodes.

Cons:

- No guarantee of even distribution.
- Doesn't consider load or performance.
- Can lead to overloading some servers by chance.