Load Balancing

What is Load Balancing?

Load balancing is a computing process that **distributes incoming network traffic** efficiently across a group of backend servers, often referred to as a server farm or server pool. The goal is to optimize resource utilization, maximize throughput, minimize response time, and prevent any single server from becoming overloaded.

Think of a load balancer as a **"traffic cop"** sitting in front of your servers. It directs client requests to the most appropriate server based on specific algorithms and server health checks.

How Load Balancing Works?

A device or service called a **load balancer** manages the distribution process:

1. **Client Request:** A user (client) sends a request to access an application or website.
2. **Load Balancer Intercepts:** The load balancer, which has a single virtual IP address (VIP) advertised to the client, intercepts this request.
3. **Algorithm-Based Routing:** The load balancer uses a pre-configured load balancing **algorithm** to decide which healthy backend server should receive the request.
4. **Server Response:** The selected server processes the request and sends the response back to the client, usually through the load balancer.

What are the different load balancing strategies?

Round Robin: Round Robin is the most simplest and straightforward strategy for load balancing. It distributes client requests to the servers in the pool **sequentially**, one after the other. It treats the list of servers as a circular queue, cycling through them repeatedly. Best for environments where all backend servers have **identical (homogeneous) processing capacity** and all incoming requests are relatively **uniform** in terms of resource consumption.

Though it has a drawback as it is a **static** algorithm, meaning it doesn't consider the current **load** or connection count of a server. If one server is much slower or has a very long-running request, Round Robin will still send it the next request, potentially causing it to become overloaded while other servers are idle

Least Connection: The **Least Connection** method is a dynamic algorithm that bases its decision on the **current, real-time workload** of each server. The load balancer sends a new request to the server that has the **fewest active connections** at that moment. Best for applications where client connections or sessions can have **varying durations** (e.g., video streaming, long API calls, file uploads). By checking active connections, it ensures that long-lived sessions don't disproportionately tie up a single server. It's **dynamic** and smarter than Round Robin because it adapts to the actual server load. This leads to a more balanced distribution of work over time, minimizing the chance of any single server becoming a bottleneck. The drawback is that it requires the load balancer to **actively monitor** and track the connection count for every single server, which adds a small amount of overhead compared to the static methods.

Random: The **Random** load balancing strategy is another static and simple method. The load balancer selects a server from the pool **randomly** to handle the incoming request. It is best for situations where the server pool is very large and the connections are short, the randomness, over a large number of requests, tends to **average out** to a roughly equal distribution, similar to Round Robin. The drawback is that like Round Robin, it is a static method that is **unaware of server load or capacity**. A random selection might land on a server that is already struggling with a heavy workload, leading to performance issues for that request.