Load balancing

Load Balancing: What & Why

Load balancing is a crucial technique in network management and web hosting. Its goal is to spread incoming traffic across multiple servers so that no single server becomes overloaded. This ensures high availability, efficient resource utilization, and a smoother user experience.

What Is Load Balancing?

Load balancing is the practice of distributing incoming requests or workloads to multiple servers or resources to:

- Prevent any single server from being overwhelmed
- Minimize latency and response time
- Improve reliability and uptime

Key Benefits:

- **Scalability:** You can scale horizontally (add more servers) instead of continually upgrading one server.
- **High Availability:** If one server goes down, traffic is rerouted to healthy servers, reducing downtime.
- **Better Performance:** Work is spread more evenly, so each server handles a manageable load.
- **Fault Tolerance:** Load balancers can detect server failures and stop sending traffic to those servers until they recover.

Strategies for Load Balancing

Here are common algorithms used to decide *which server* should handle a given request:

a. Round Robin

Requests are distributed in a cyclical fashion: Server 1 → Server 2 → Server 3 → back to Server 1 → ...

- **Pros:** Simple, even distribution under equal server capacity

- **Cons:** Doesn't consider current load or server health
- **Best For:** Homogeneous servers with similar capacity

Example: With 3 servers, incoming requests might go 1, 2, 3, 1, 2, 3, ...

---

b. Least Connections

The load balancer sends the request to the server that currently has the fewest active connections.

- **Pros:** More dynamic than Round Robin; adapts to load differences
- **Cons:** Doesn't account for the nature of connections (some may be heavier than others)
- **Best For:** Environments with varying durations of connections or mixed loads

Example: If Server A has 5 active connections, B has 10, and C has 3, a new request goes to C.

---

c. Random

Each incoming request is assigned to a server chosen at random.

- **Pros:** Very simple to implement
- **Cons:** Can lead to uneven distribution, especially under high or burst traffic
- **Best For:** Light loads or when load distribution is not critical

Example: Every request is sent to a random server, without considering current load.

More Advanced Strategies

Beyond the basic three, here are a few more sophisticated techniques:

1. IP Hashing

Uses the client's IP address (or a hash thereof) to consistently route that client to the same server. Useful for session persistence.

2. Weighted Algorithms

- **Weighted Round Robin / Weighted Least Connections**: Assigns weights to servers (based on capacity), so more powerful servers get more traffic.

- **Least Response Time / Least Load**: Use metrics like server response time or resource usage to guide routing.

Important Considerations

- **Health Checks:** The load balancer should periodically check whether servers are healthy. If one fails, it shouldn't receive traffic until it recovers.
- **Session Persistence (Sticky Sessions):** Some applications need a client's requests always to go to the same server (e.g. shopping cart). Load balancers can use cookies, IP affinity, or other means to support this.
- **Global Load Balancing:** For applications spread across regions or data centers, load balancing can operate across geographic locations, sending users to the "closest" or best-performing cluster.
- **SSL Offloading, Content Awareness, and Protocol Handling:** Many advanced load balancers can decrypt SSL traffic, inspect content, and route based on path, headers, etc