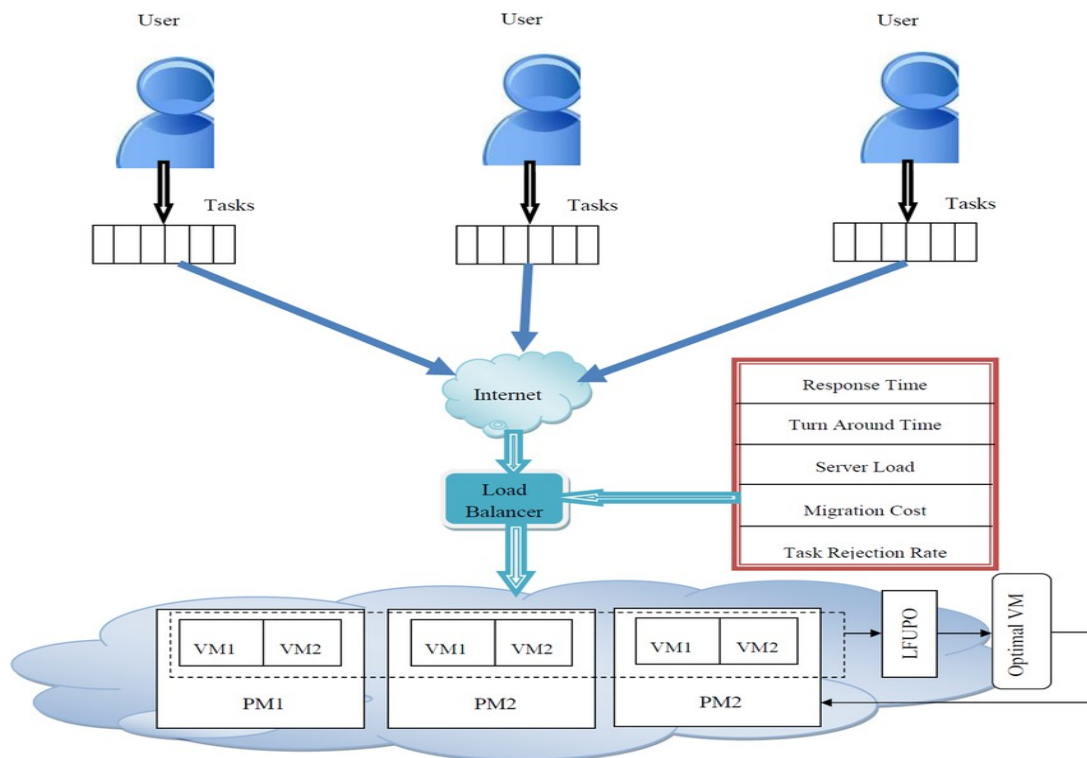


What Is Load Balancing?

Load balancing is the process of efficiently distributing incoming network traffic or computational tasks across multiple servers or resources. Its primary goal is to prevent any single server from becoming a bottleneck, thereby improving system responsiveness, fault tolerance, and scalability.

In modern architectures—especially cloud-native and microservices-based systems—load balancing is essential for handling millions of concurrent requests. A load balancer acts as a gateway between users and backend servers, deciding which server should handle each request based on predefined algorithms.



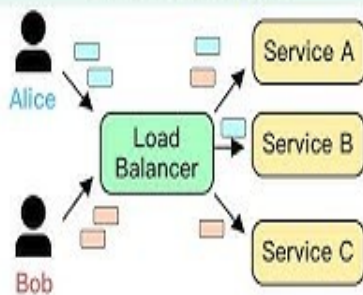
Why Is Load Balancing Important?

- **High Availability:** If one server fails, traffic is rerouted to others.
- **Scalability:** Easily add or remove servers without disrupting service.
- **Performance Optimization:** Distributes workload evenly to avoid slowdowns.
- **Fault Tolerance:** Reduces risk of system crashes due to overload.
- **Cost Efficiency:** Maximizes resource utilization, reducing infrastructure costs.

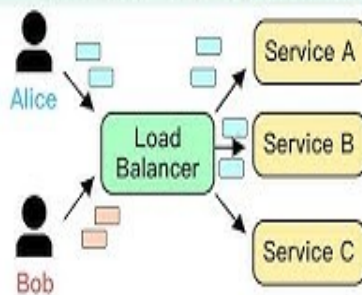
Example: Imagine an online shopping site during a festive sale. Thousands of users access the site simultaneously. Without load balancing, one server might crash under pressure. With load balancing, requests are spread across multiple servers, keeping the site responsive and stable.

Top 6 Load Balancing Algorithms

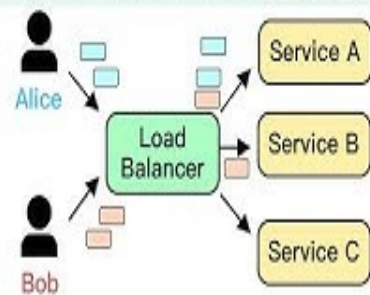
1.Round Robin



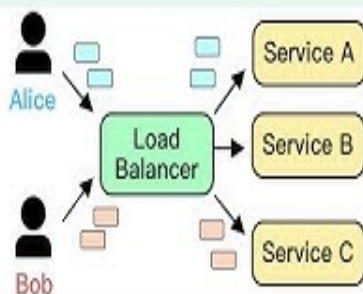
2.Sticky Round Robin



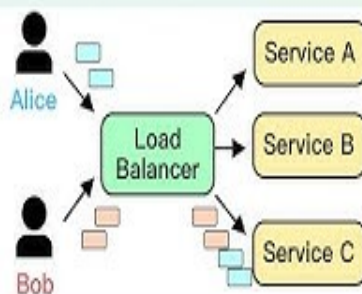
3.Weighted Round Robin



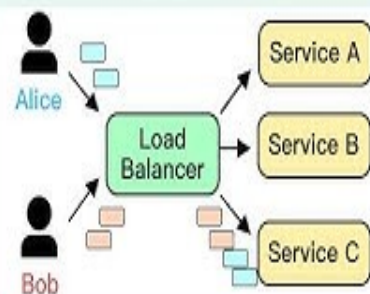
4.IP/URL Hash



5.Least Connections



6.Least Time



Load Balancing Strategies

Different algorithms determine how traffic is distributed. Here are three widely used strategies:

1. Round Robin

How it works: Requests are distributed sequentially across all available servers in a circular order.

Example: If there are three servers—A, B, and C—the first request goes to A, the second to B, the third to C, and the fourth back to A.

Advantages:

- Simple to implement
- Effective when servers have similar capacity

Limitations:

- Doesn't account for server load or performance

Use Case: A blog site with static content hosted on identical servers can use Round Robin to evenly distribute traffic.

2. Least Connections

How it works: The load balancer sends the request to the server with the fewest active connections.

Example: If Server A has 2 active connections, B has 5, and C has 1, the next request goes to C.

Advantages:

- Dynamic and responsive to server load
- Ideal for long-lived or variable-duration connections

Limitations:

- Requires real-time tracking of active connections

Use Case: A video conferencing platform where some users stay connected longer than others benefits from Least Connections to avoid overloading any single server.

3. Random

How it works: Requests are assigned to servers randomly, without considering order or load.

Example: Each incoming request is routed to a randomly selected server, regardless of its current usage.

Advantages:

- Easy to implement
- Works well when server loads are balanced

Limitations:

- Can lead to uneven distribution if not monitored

Use Case: A gaming app with lightweight requests and uniform server capabilities can use Random strategy for simplicity.

Conclusion

Load balancing is a foundational element of modern IT infrastructure. Whether you're running a high-traffic website, a cloud-native application, or a distributed microservices system, choosing the right load balancing strategy is crucial.

- **Round Robin** is best for uniform workloads.
- **Least Connections** suits dynamic, long-lived sessions.
- **Random** offers simplicity for balanced environments.

By implementing these strategies, organizations can ensure their systems remain fast, reliable, and scalable—even under heavy load.