Load Balancers in System Design

We often use web services that quickly respond to our requests, but we may not be aware of the complexity of the systems that are responsible for providing this fast response. To understand one of the critical reasons, we need to understand the concept of load balancing.

What is a load balancer?

Load balancing is a technique to distribute incoming traffic or workloads across multiple servers. For this, we use load balancers. A load balancer is a device or software component that acts as an intermediary between clients and a group of servers. Its purpose is to evenly distribute workloads to prevent any server from becoming overloaded.

In simple words, the load balancer is one of the key components in modern distributed systems to optimize resource usage, maximize performance, and ensure high availability, scalability and reliability.

Why do we need a load balancer?

Suppose we have a single server setup, where several clients are sending requests to a single server. When number of requests increases, there will be two critical issues:

- **Server overloading:** There is a limit to the number of requests a single server can handle. If the number of requests exceeds this limit, the server may become overloaded and unable to function properly.
- **Single point of failure:** If the single server goes down for any reason, the entire application will become unavailable to users. This can result in a poor user experience and impact the reliability of the system.

We can solve the above problems in two ways:

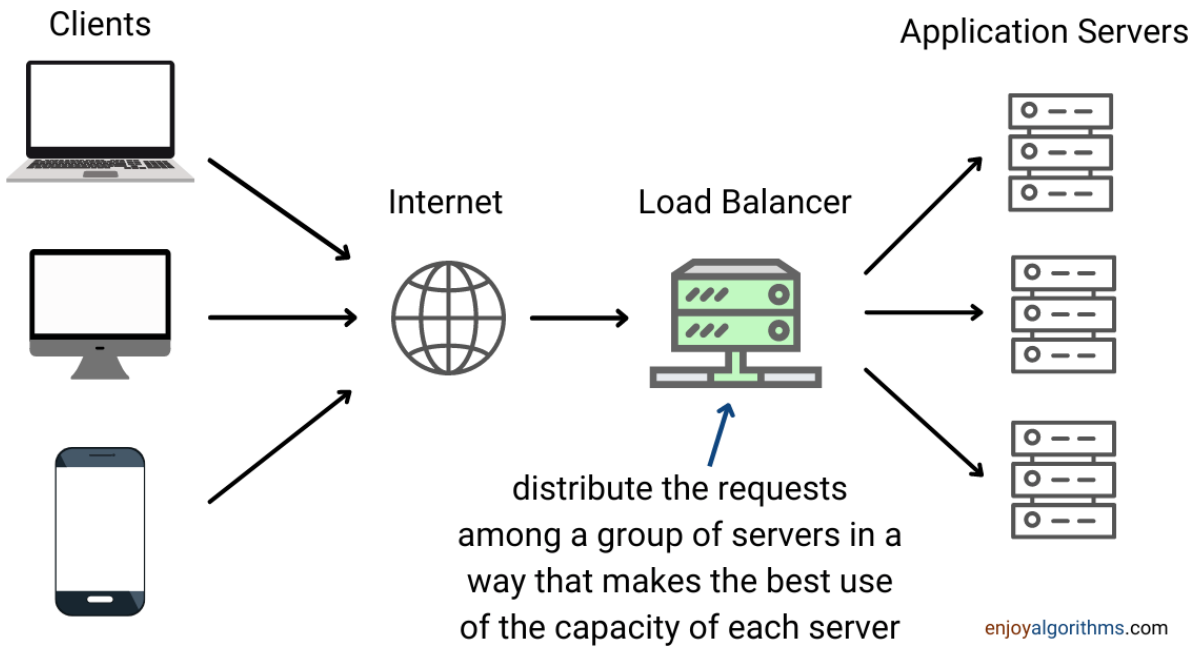
- **Vertical scaling**: We can increase the power of our current server.

But there are limits to how much we can increase the capabilities of a single machine.

- **Horizontal scaling**: We can add more servers to our system. Now this will bring a new challenge: How to distribute requests evenly across these servers? The answer is: We should use load balancers!

The load balancer will not only help us to distribute requests across multiple servers but also increase system capacity to add more servers if the number of requests increases in the future.

On the other side, load balancer also continuously checks the health of each server. If one of the servers goes offline due to some reason, it will redirect the traffic going to that server to some other available server. So load balancers help us to ensure that the service remains available, even in the case of the failure of one server or multiple servers.



How a system with a load balancer works?

Now there are some critical questions: How user requests are served when there is a load balancer in the system? What are the key steps involved in this process? Let's understand the whole process starting from the DNS request.

- **Step 1:** User device sends a DNS request to a DNS server. Now DNS server maps the domain name in the request to an IP address.
- **Step 2:** DNS server responds to the device with the IP address of the requested domain. Here is one important thing: This IP address is the IP address of the load balancer! Due to this, web servers are not reachable directly by the end user and only the load-balancer layer is visible.
- **Step 3:** Now user device establishes a connection with the load balancer using the provided IP address. This connection is initiated over the HTTP(S) protocol.
- **Step 4:** Now load balancer receives the incoming request from the user device. At this point, the load balancer acts as the entry point for the traffic.
- **Step 5:** Now load balancer determines which web server should handle the request based on a load-balancing algorithm. This decision depends on factors like server availability, server load, and the logic of load-balancing algorithm.
- **Step 6:** Load balancer forwards the incoming request to the selected web server through private IPs. Here private IP address is unreachable for the end device and reachable only between servers in the same network. They are used for communication between servers.
- **Step 7:** The selected web server processes the request and generates a response. Now web server sent back the response to the load balancer.
- **Step 8:** Load balancer receives the response from the web server and forwards it back to the user device.
- **Step 9:** Finally, user device receives the response from the load balancer. They treat it as if it originated directly from the requested server.

These steps may be different based on the load balancer configuration and the protocols. Most of the time, load balancers also perform additional functions like SSL termination, session persistence, traffic monitoring, caching, etc.

Where do we add a load balancer?

Load balancers can be placed at different points in a system to distribute the workload. Some common places to use load balancers are:

- **Between clients and frontend web servers:** This is often the first point of contact between the client and the system. So load balancer receives incoming requests from clients and distributes them across frontend web servers.
- **Between frontend web servers and backend application servers:** In a system with multiple frontend web servers, a load balancer can be used to distribute incoming requests from the web servers to the backend application servers.
- **Between backend application servers and cache servers:** Load balancers can be used to distribute requests from the application servers to cache servers, which store frequently accessed data in memory to reduce response times.
- **Between cache servers and database servers:** In systems with multiple cache servers, a load balancer can be used to distribute requests from the cache servers to the database servers, which store the actual data. This helps to ensure that the database servers are not overwhelmed with requests.

Types of load balancers

There are two main types of load balancers: software load balancers and hardware load balancers. The main difference between them is the level of customization and scalability they offer.

Software load balancers are implemented as software applications that are installed on physical servers or virtual machines. They are configured to meet the specific needs of a system and provide greater flexibility in terms of customization. Scaling with software load balancers is also easier because we can add more capacity by adding more servers.

Hardware load balancers are physical devices that are installed in a

network or in data centres. They are generally less flexible and offer fewer options for customization. But they are often faster and more reliable than software load balancers because they are dedicated hardware devices that are designed specifically for load balancing.

Overall, the choice between a software or hardware load balancer depends on the specific requirements of a system. Software load balancers are more suitable for systems that require a high level of customization and scalability, while hardware load balancers are better for systems that require high performance and reliability.

Pros and cons of software load balancers

Pros

- Provide more options for customization and configuration.
- Scale horizontally to handle more traffic by adding more instances.
- Cheaper than hardware load balancers because they can be installed on commodity hardware. The best thing is: They do not require the purchase and maintenance of physical hardware.
- They can be deployed in the cloud for easy scaling and cost savings.
- Provide a simple interface or API to configure, monitor, and manage multiple load balancer instances from a single control point. This simplifies administration and reduces operational overhead.

Cons

- There can be some delay when scaling beyond the initial capacity of a software load balancer because the software needs to be configured and set up.
- They introduce an additional layer of processing in the networking stack. Due to this, there can be some performance overhead in terms of throughput, latency and concurrent connections. This performance can also be constrained by the processing capabilities of the host machine.
- Require regular maintenance and updates to ensure good

performance and security.

Examples of software load balancers

- HAProxy: A TCP load balancer.
- NGINX: An HTTP load balancer with SSL termination support.
- mod_athena: Apache-based HTTP load balancer.
- Varnish: A reverse proxy-based load balancer.
- Balance: Open-source TCP load balancer.
- LVS: Linux virtual server offering layer 4 load balancing.

Pros and cons of hardware load balancers

Pros

- Hardware load balancers offer consistent performance because load-balancing logic runs on specialized hardware. They can handle large concurrent connections, and provide fast throughput and low latency.
- Built on well-optimized and tested hardware, with an underlying operating system that is optimized for performance and stability. This makes them less prone to failure compared to software load balancers.
- Increase security, as only authorized personnel can physically access the servers.

Cons

- Require a higher upfront cost for purchase and maintenance.
- Struggle to scale beyond a certain number of requests because they are limited by the hardware.
- Require more human resources to configure and manage, compared to software load balancers.

Examples of hardware load balancers

- F5 BIG-IP load balancer
- CISCO system catalyst

- Barracuda load balancer
- Coytepoint load balancer
- Citrix NetScaler

Load balancing algorithms

Load balancers use various load-balancing algorithms to distribute incoming network traffic across multiple servers in a balanced manner. So, it is the responsibility of these algorithms to select a server from a pool of available servers to direct each incoming request.

Here are some popular load-balancing algorithms. If you want to learn these algorithms in detail, explore this blog: load balancing algorithms.

- **Round Robin Method:** Each request is sequentially distributed to the next server in a circular manner.
- **Least Connections Method**: Directs new requests to the server with the fewest active connections.
- **Weighted Round Robin Method:** Similar to the Round Robin but there is a weight associated with each server (weight represents the server capacity). So the servers with higher weights receive a larger proportion of the load.
- **Least Response Time Method:** Considers the response times of the servers and directs the request to the server with the lowest response time.
- **IP Hash Method:** Calculates a hash value based on the client's IP address to select the server from the pool. This will ensure that requests from the same client will be directed to the same server.

Different algorithms have different strengths and weaknesses. So the correct choice of load-balancing algorithm depends on the characteristics of the workload and the goals of the load-balancing strategy. On the other side, load balancers often provide various configuration options to choose the load-balancing algorithm.

Advantages of load balancing

- Ensure that the application is always available and can scale as needed.
- Prevent a single server from being overloaded with requests.
- Provide encryption, authentication, and other functionalities to secure, manage, and monitor the application. They can also provide efficient protection against DoS attacks.
- End users only need to know the address of the load balancer, rather than the addresses of every server in the cluster, which provides a layer of abstraction.
- Minimize server response time and maximize throughput by distributing requests evenly across servers.
- Load balancers can do health checks and monitor the request handling capability of servers to ensure that they are functioning properly.
- We can add or remove servers based on the number of requests.
- Load balancers can be used to roll out software updates without taking the whole service down by taking out one server at a time.

Critical concepts to explore further

- What is the difference between Load Balancer and Reverse Proxy?
- Different Categories of Load Balancing: 1) Layer 4 (L4) load balancer 2) Layer 7 (L7) load balancer 3) Global server load balancing (GSLB)
- Health check feature of the load balancer.
- DNS load balancing vs Hardware load balancing
- The application load balancer in designing several systems
- Cloud load balancing

In this blog, we have covered the basic concepts of load balancers. In the coming future, we will add advanced insights. Thanks to Navtosh for his contribution in creating the first version of this content. If you have any queries or feedback, please write us at contact@enjoyalgorithms.com. Enjoy learning, Enjoy system design!