What is a load balancing algorithm?

A load balancer is a software or hardware device that keeps any one server from becoming overloaded. A load balancing algorithm is the logic that a load balancer uses to distribute network traffic between servers (an algorithm is a set of predefined rules).

There are two primary approaches to load balancing. *Dynamic load balancing* uses algorithms that take into account the current state of each server and distribute traffic accordingly. *Static load balancing* distributes traffic without making these adjustments. Some static algorithms send an equal amount of traffic to each server in a group, either in a specified order or at random.

What are the different types of load balancing algorithms?

Dynamic load balancing algorithms

Least connection: Checks which servers have the fewest connections open at the time and sends traffic to those servers. This assumes all connections require roughly equal processing power.

Weighted least connection: Gives administrators the ability to assign different weights to each server, assuming that some servers can handle more connections than others.

Weighted response time: Averages the response time of each server, and combines that with the number of connections each server has open to determine where to send traffic. By sending traffic to the servers with the quickest response time, the algorithm ensures faster service for users.

Resource-based: Distributes load based on what resources each server has available at the time. Specialized software (called an "agent") running on each server measures that server's available CPU and memory, and the load balancer queries the agent before distributing traffic to that server.

Static load balancing algorithms

Round robin: Round robin load balancing distributes traffic to a list of servers in rotation using the Domain Name System (DNS). An authoritative nameserver will have a list of different A records for a domain and provides a different one in response to each DNS query.

Weighted round robin: Allows an administrator to assign different weights to each server. Servers deemed able to handle more traffic will receive slightly more. Weighting can be configured within DNS records.

IP hash: Combines incoming traffic's source and destination IP addresses and uses a mathematical function to convert it into a hash. Based on the hash, the connection is assigned to a specific server.

Static Load Balancing Algorithms

Static algorithms assign workloads ahead of time, without adapting to changes in server load or resource availability. Examples include:

- Round Robin: Cycles through servers, assigning each request to the next one in the list. Best when all servers are similar, sessions are stateless, and workload is predictable.
- Weighted Round Robin: Like round robin, but some servers get more requests based on pre-assigned weights (good for a mixed-capacity setup).
- IP Hash: Uses a hash function on the client's IP address to consistently route requests from the same client to the same server. Useful for sticky sessions.

When to use static algorithms:

- Your servers have similar hardware and capacity.
- The workload is predictable (little fluctuation during the day).
- Session persistence is important, but you don’t need dynamic adaptation.
- You want simplicity and low overhead.
- Example: A simple image server cluster behind a round-robin DNS setup, serving stateless requests where every server is equally powerful.[cloudflare+2]

Dynamic Load Balancing Algorithms

Dynamic algorithms monitor the real-time state of the servers (like open connections, response times, or even CPU/RAM load) and adjust traffic distribution accordingly. Examples:

- Least Connection: Sends new requests to the server with the fewest open connections. Great when time per connection varies.
- Weighted Least Connection: Adjusts least connections for the capacity of each server (more powerful servers naturally take more load).
- Weighted Response Time: Considers both connection count and actual measured response times, routing requests to the fastest/least loaded server presently.
- Resource-based: Checks each server's live resource stats (CPU, memory) via an agent before directing new requests.

When to use dynamic algorithms:

- Your traffic pattern is unpredictable or bursty.
- Server capacities vary or fluctuate (some servers may be upgraded/downgraded on demand).
- User sessions and file/image requests are not uniform—some require more backend work than others.
- Downtime or overload must be minimized, even as traffic changes quickly.
- Example: A global image/file server cluster where peak traffic varies by hour and location; some servers are regularly upgraded; file sizes and processing time vary.

Choosing the Right Algorithm: Practical Guidelines

- Use static algorithms when: You want minimal configuration, consistent workload, stateless traffic, and servers are similarly sized.
    - Example: Simple photo website, rarely changing traffic, all servers in the same datacenter.
- Use dynamic algorithms when: Traffic is spiky or unpredictable, servers are upgraded independently, or sessions/connections are far from uniform in duration or resource needs.
    - Example: File upload/download service needing fair distribution during viral spikes or upgrades.
- For high-performance, cloud-native systems: Resource-based and response-time-aware dynamic algorithms maximize utilization, but at the cost of more complexity and possible monitoring overhead.
- For session persistence (files in progress, user logins): IP hash or consistent hashing (static) ensures users stay pinned to the same backend, which may be easier than configuring a dynamic approach.

In summary, pick static when simplicity outweighs adaptability, and dynamic when adaptability and performance outweigh configuration simplicity.[geeksforgeeks+3]