Load Balancers

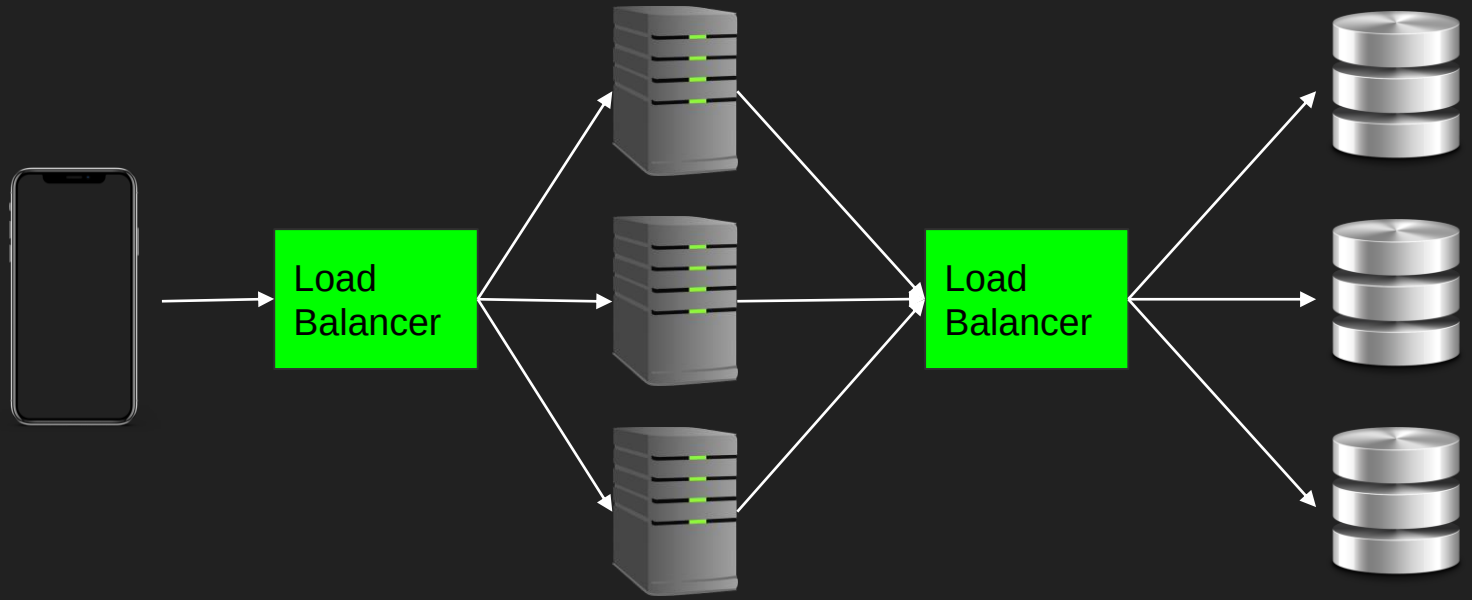
Background

As systems scale, there reaches a point where you can no longer use more powerful hardware (vertical scaling) and have to use more nodes (horizontal scaling). We have first seen this with databases via partitioning for increasing storage, as well as replication for increasing read (and sometimes even write) throughput. However, databases are not the only thing being horizontally scaled, application servers can be too!

Any time that there are a variety of nodes in a cluster that can serve the same requests, using something like a load balancer can help to ensure that none of the servers become overwhelmed.

Load Balancers

Load balancers are a type of reverse proxy, that can sit between layers of an application.



Load Balancing Overview

- Locally keep state of the health of each server via heartbeats
  - Number of open connections
  - Load
  - Time of last heartbeat
  - Done locally as opposed to in a coordination service to speed up routing
- Route traffic to each server based on some sort of routing policies

Routing Policies

- Least connections
  - Traffic routed to server with fewest active open TCP connections
- Least response time
  - Traffic routed to server with lowest average response time
- (Weighted) round robin
  - Create ordering of the servers, send each subsequent request in order to next server
  - Can be weighted by compute capacity of the servers
- Hashing
  - Based on the contents of the request

Layer 4 vs. Layer 7 Load Balancing

Layer 4:

- Requests are routed based on the details exposed by the network transport layer
  - Source IP address, destination IP address, ports in the header
  - Computationally faster

Layer 7:

- Requests are routed based on header contents
  - More flexibility in routing, can hash on more relevant information

Consistent Hashing

Recall: Consistent hashing is an algorithm that distributes items into buckets in a way such that adding/removing buckets only shifts around the hash of a small portion of the items.

This is extremely useful for load balancing, as routing the same/similar requests to the same nodes allows nodes to locally cache certain results, thus reducing the amount of network calls/computation needed.

Availability

Up until this point, we have acted like there is only one load balancer, which would act as a single point of failure - if the load balancer went down we would not be able to serve any requests.

Can either run two machines with some shared network attached storage for configuration, both acting as load balancers (active-active), or have one machine running as a load balancer with a second that occasionally exchanges heartbeats with it and takes over if it is down (active-passive).

Conclusion

Load balancers are a relatively essential part of applications at scale, and in reality they often perform even more functionality than just load balancing - doing things like encryption/decryption, and acting as a reverse proxy.

Pros:

- Make sure that your various servers do not become overwhelmed by incoming requests
- Maximize local cache performance via consistent hashing

Cons:

- Can become a performance bottleneck
- Adds more complexity to the system