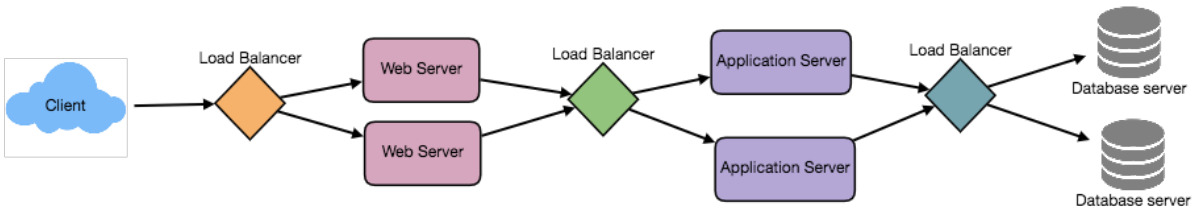
Load Balancing

Load balancer (LB) is another critical piece of any distributed system. It helps to distribute load across multiple resources according to some metric (random, round-robin, random with weighting for memory or CPU utilization, etc.). LB also keeps track of the status of all the resources while distributing requests. If a server is not available to take new requests or is not responding or has elevated error rate, LB will stop sending traffic to such a server.

To utilize full scalability and redundancy, we can try to balance the load at each layer of the system. We can add LBs at three places:

- Between the user and the web server
- Between web servers and an internal platform layer, like application servers or cache servers
- Between internal platform layer and database.



There are many ways to implement load balancing.

1. Smart Clients

A smart client will take a pool of service hosts and balances load across them. It also detects hosts that are not responding to avoid sending requests their way. Smart clients also have to detect recovered hosts, deal with adding new hosts, etc.

Adding load-balancing functionality into the database (cache, service, etc.) client is usually an attractive solution for the developer. It looks easy to implement and manage especially when the system is not large. But as the system grows, LBs need to be evolved into standalone servers.

2. Hardware Load Balancers

The most expensive–but very high performance–solution to load balancing is to buy a dedicated hardware load balancer (something like a Citrix NetScaler). While they can solve a remarkable range of problems, hardware solutions are very expensive, and they are not trivial to configure.

As such, even large companies with large budgets will often avoid using dedicated hardware for all their load-balancing needs. Instead, they use them only as the first point of contact for user requests to their infrastructure and use other mechanisms (smart clients or the hybrid approach discussed in the next section) for load-balancing for traffic within their network.

3. Software Load Balancers

If we want to avoid the pain of creating a smart client, and since purchasing dedicated hardware is excessive, we can adopt a hybrid approach, called software load-balancers.

HAProxy is a one of the popular open source software LB. Load balancer can be placed between client and server or between two server side layers. If we can control the machine where the client is running, HAProxy could be running on the same machine. Each service we want to load balance can have a locally bound port (e.g., localhost:9000) on that machine, and the client will use this port to connect to the server. This port is, actually, managed by HAProxy; every client request on this port will be received by the proxy and then passed to the backend service in an efficient way (distributing load). If we can't manage client's machine, HAProxy can run on an intermediate server. Similarly, we can have proxies running between different server side components. AProxy manages health checks and will remove or add machines to those pools. It also balances requests across all the machines in those pools.

For most systems, we should start with a software load balancer and move to smart clients or hardware load balancing as need arises.