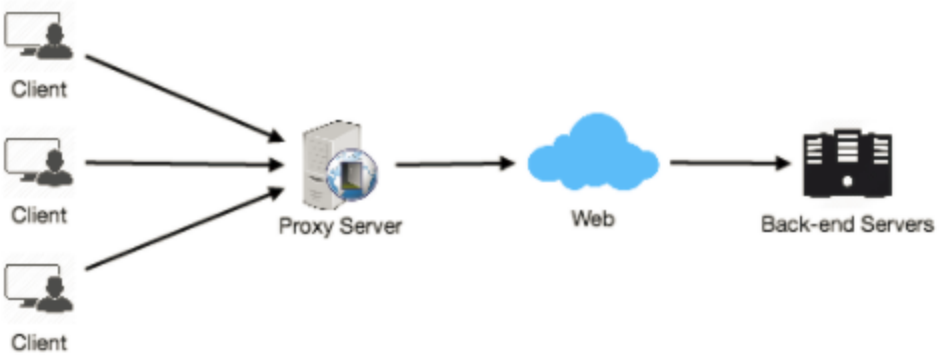
Proxies

A proxy server is an intermediary piece of hardware/software that sits between the client and the back-end server. It receives requests from clients and relays them to the origin servers. Typically, proxies are used to filter requests or log requests, or sometimes transform requests (by adding/removing headers, encrypting/decrypting, or compression). Another advantage of a proxy server is that its cache can serve a lot of requests. If multiple clients access a particular resource, the proxy server can cache it and serve all clients without going to the remote server.



Proxies are also extremely helpful when coordinating requests from multiple servers and can be used to optimize request traffic from a system-wide perspective. For example, we can collapse the same (or similar) data access requests into one request and then return the single result to the user; this scheme is called collapsed forwarding.

Imagine there is a request for the same data across several nodes, and that piece of data is not in the cache. If these requests are routed through the proxy, then all them can be collapsed into one, which means we will be reading the required data from the disk only once.

Another great way to use the proxy is to collapse requests for data that is spatially close together in the storage (consecutively on disk). This strategy will result in decreasing request latency. For example, let's say a bunch of servers request parts of file: part1, part2, part3, etc. We can set up our proxy in such a way that it can recognize the spatial locality of the individual requests, thus collapsing them into a single request and reading complete file, which will greatly minimize the reads from the data origin. Such scheme makes a big difference in request time when we are doing random accesses across TBs of data. Proxies are particularly useful

under high load situations, or when we have limited caching since proxies can mostly batch several requests into one