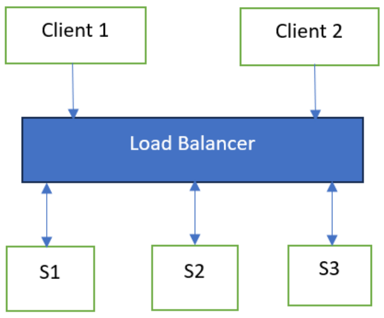
DAVID MALAN LECTURE SHORT NOTES

Vertical scaling: If low on RAM or Disk space, you can get more processor with more RAM and more Disk space.

Horizontal scaling: If low on RAM or Disk space, you can add more machines and distribute traffic across them.

Load Balancing: IP address of the load balancer is returned to the client. So, load balancer will have public IP address and servers can either have public/private IP address.



DNS and Round Robin:
DNS servers can use Round Robin to cycle through multiple IPs (server1, server2, etc.).
This approach bypasses a centralized load balancer, so the client directly talks to one of the backend servers.
But without sticky sessions or session sharing, this can lead to broken session state issues.

Shared session state: Sessions are temporarily saved as text files and are implemented per server. You can use Round Robin DNS, but session stickiness (also called session affinity) must be maintained, or the sessions must be shared. You can use factorization. Factor out a service like session states. So, if we have a file server connected to all the servers, anytime they store session data, they store it there instead of their own hard drive. Or they can put sessions on load balancer.

RAID: Redundant array of independent discs. They are used in desktop computers. All the variants of RAIDs assume you have multiple hard drives on your computer.
RAID 0: It's good in performance. No fault tolerance.
RAID 1: Data is written in 2 hard drives. If 1 hard drive dies, you can go and buy a new hard drive and the data gets restored.
RAID 10: Combination of RAID 0 and RAID 1.
RAID 5: Any number of hard drives can be used but only 1 hard drive is used for redundancy.
RAID 6: Any number of hard drives can be used but only 2 hard drives are used for redundancy.

Problem for the need of sticky sessions:

Sticky sessions: When you visit a website multiple times, your session is somehow preserved.

The problem can be overcome by using cookies. These cookies have a finite size. You can store the ID of the server in a cookie so that the user can visit the website a number of times as by following links and goes back to the same back-end server.

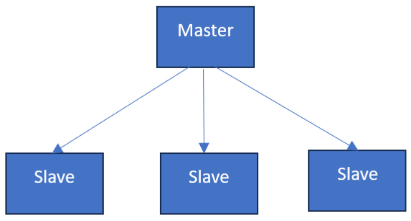
The load balancer similarly could be inserting cookies with the set cookie header that the end user then subsequently sends back so we can remember what back-end server to send the user to.

PHP Accelerator: Compiles php every time but throws a result. They precompile into bytecode.

Caching: It can be implemented in context of dynamic websites generating the HTML and saving it as xyz.html and storing it on the disk.
Downside of file-based caching approach: space, redundancy.
Much better performance from just saving up the static content but the price you pay is more disk space.
MySQL caching: MySQL results of identical queries.
Memcached: Used by Facebook over the years. Stores whatever you want in RAM.

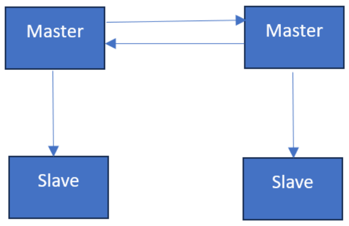
Replication: Making automatic copies of something.

Master-Slave:



The purpose of the slave is to get a copy of every row that's in a master database. Here, the master and slaves are identical. If the master fails, you can make any of the slave as the new master.

Master-Master:



If either the master fails, you will still have a master.

Load Balancing + Replication:
Active-Active: Pair of load balancers that are constantly listening to connections either one of which can receive pockets from the outside world and relay them to backend servers.
Active Passive: If active is not present, passive becomes active and handles all the requests from the outside world.

How to get sticky sessions using only a load balancer and no shared state yet?
Have the load balancer listen at the HTTP level and when the response comes back from the first web server, the load balancer can insert some cookie that allows it to remember.