APPROACH:

1. Outline use cases, constraints and assumptions
2. Create a high level design.
3. Design core components.
4. Scale the design

SCALABILITY:

WEB HOST:

eg: GoDaddy.com

Check if

* They use SFTP, not FTP to communicate between host and server)
  (Username, password needs to be encrypted).
* Virtual Resources → Are the server resources "shared" with others.
* If we use VPS (Virtual Private Server), the Server has multiple VMs. We get one. So our data cannot be accessed by other users. But the hosting company still has physical access.

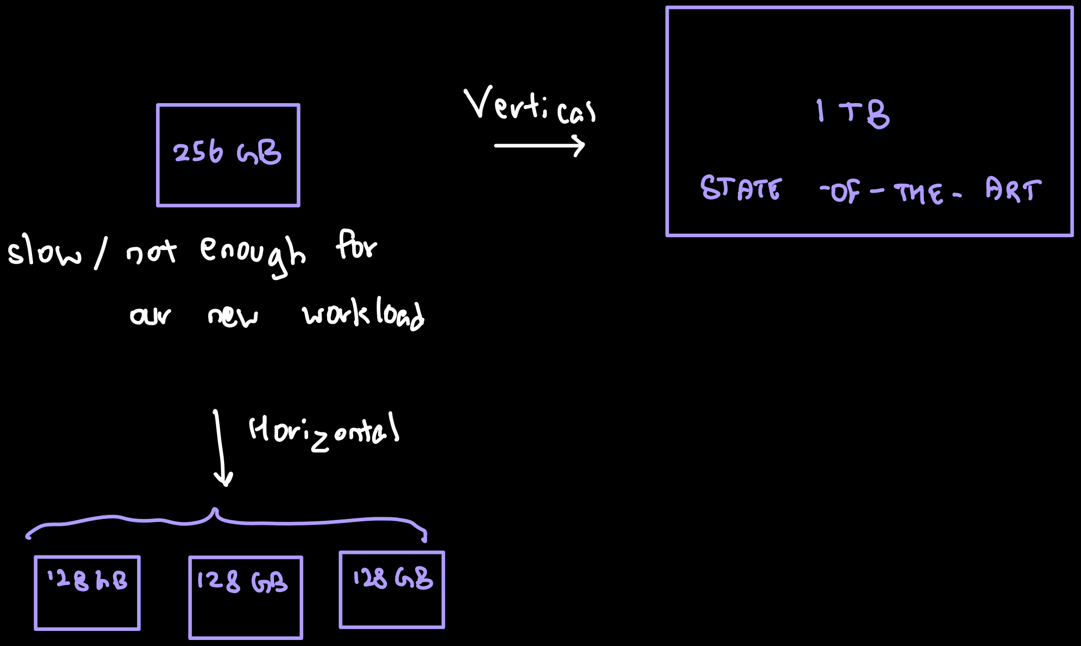
USE A SERVER THAT ONLY YOUR COMPANY HAS PHYSICAL ACCESS TO!

VERTICAL SCALING:

Increase the RAM or memory etc.

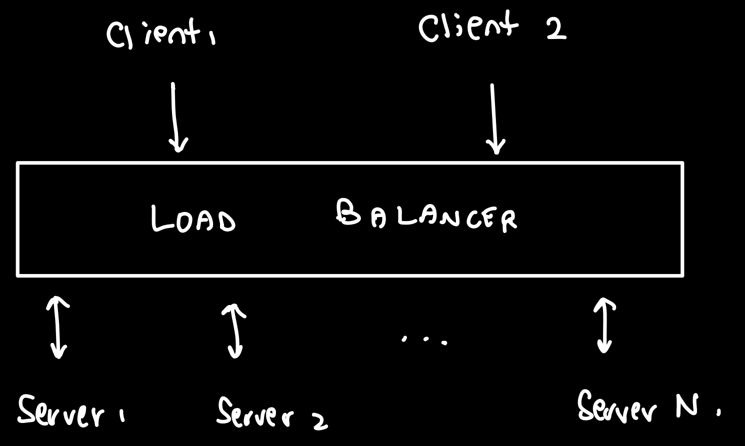
HORIZONTAL SCALING:

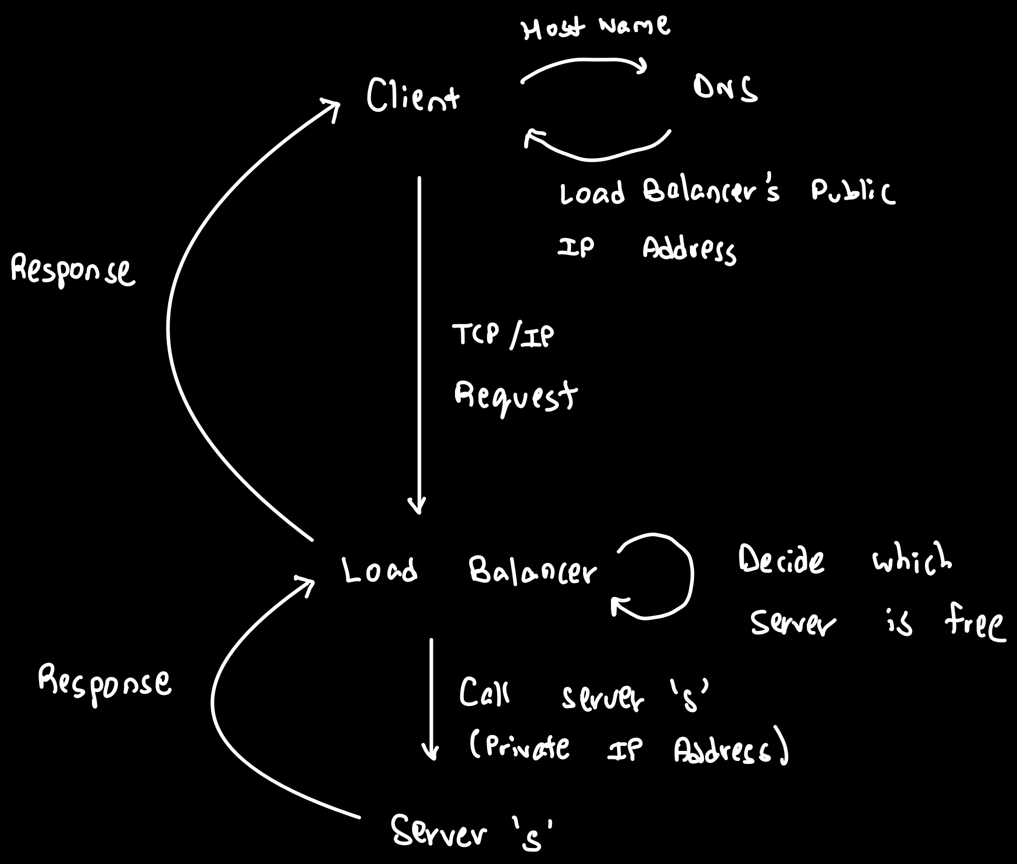
Use many small not-so-advanced servers to get the work done.



LOAD BALANCER:

Now there are multiple servers. Which to call?




Host Name
Client
DNS
Load Balancer's Public IP Address
Response
TCP/IP Request
Load Balancer
Decide which server is free
Response
Call server 's' (Private IP Address)
Server 's'

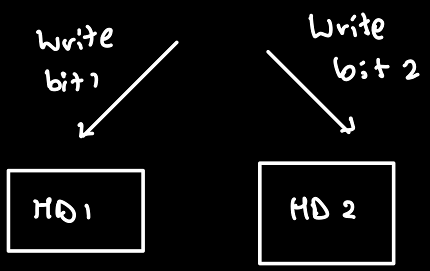
Note: Servers → Private IP Addresses → "Safer"

So data like session details Disk are shared by the servers. What if it fails?

RAID [REDUNDANT ARRAY OF INDEPENDENT DISKS]:

REDUNDANCY WITHIN THE SERVER

→ RAID 0:

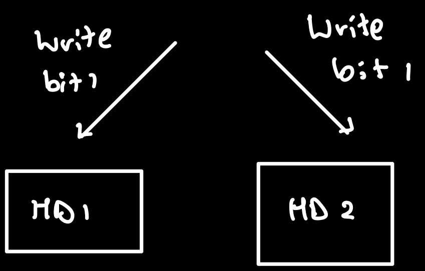


HD: Hard Drive

Write alternate bits in other HD. [STRIPING]

↑ Performance.

→ RAID 1:



Redundancy!

If HD1 dies, bring new HD → connect →

Automatically copies RAID Arrays

from HD 2 to HD 3 (now HD1).

→ RAID 10:

STRIPING + REDUNDANCY

4 HDs.

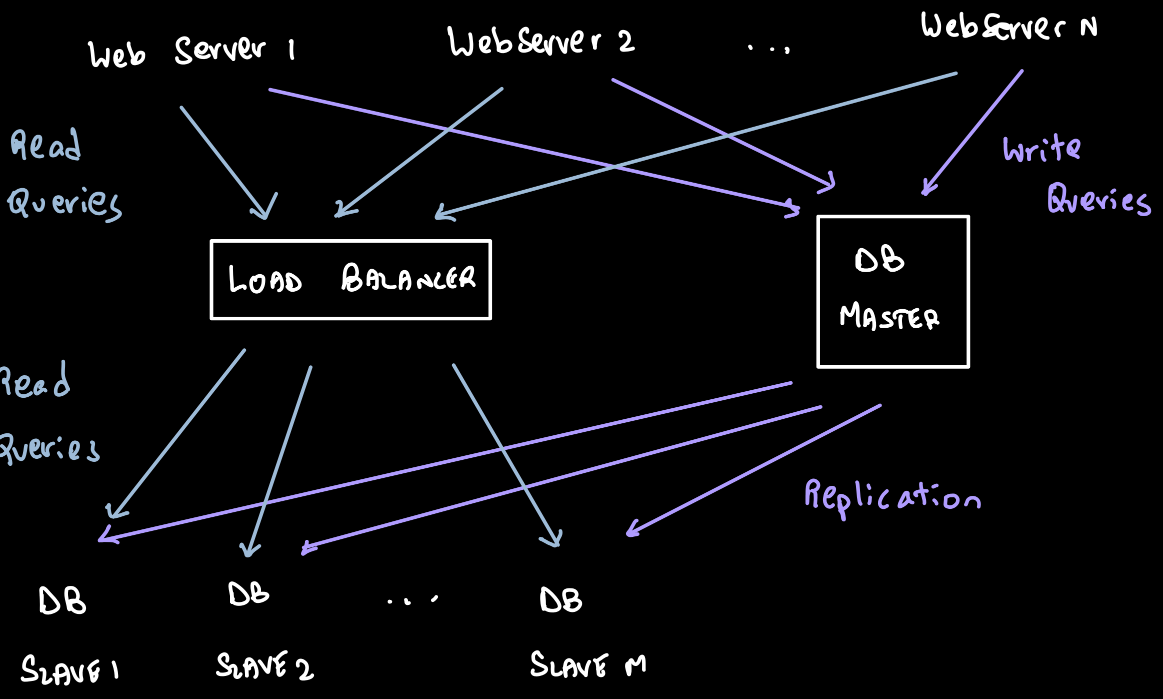
→ RAID 5: PARITY

One Redundant HD that provides redundancy to (equivalent) 5-6 HDs.

→ RAID 6: DOUBLE PARITY

2 Redundant HD. Now even if 2 HDs fail, we can replace with no data loss!

REDUNDANCY THROUGH MULTIPLE STORAGE SERVERS:


Web Server 1
Web Server 2
...
Web Server N
Read Queries
Write Queries
LOAD BALANCER
DB MASTER
Read Queries
Replication
DB SLAVE 1
DB SLAVE 2
...
DB SLAVE M