Scaling Mindset

* scale ⇒ high traffic
  ↓
  Scalable ⇒ capable of handling huge traffic

[
* dummy project
* code was in your laptop
]

reallife → • work with other engineers
• TBs of data
• test
• deploy on multiple machines
• changing requirements

* google question

SDE 1 → 0-3 yrs
SDE 2 → 3-6 yrs
SDE 3 → 6-8-10 yrs ⟶ 7.5 yrs
↓
BLR (1.6 cr)
Staff → 8 - 10+
Principal Arch → 12-14+
⋮
HOE / VP. -

Q ⇒ given a list of strings, you need to sort it alphabetically

ant

top

ball

blue

cat

⇒ ant ball blue cat top $\longrightarrow$ sorted

strs = [" ", " ", " ", " —]

Python ⇒ strs.sort();

Java ⇒ Arrays.sort[strs];

C++ ⇒ sort[strs.begin(), strs.end()];

: Catch ⇒ Data is 50PB long.

↓

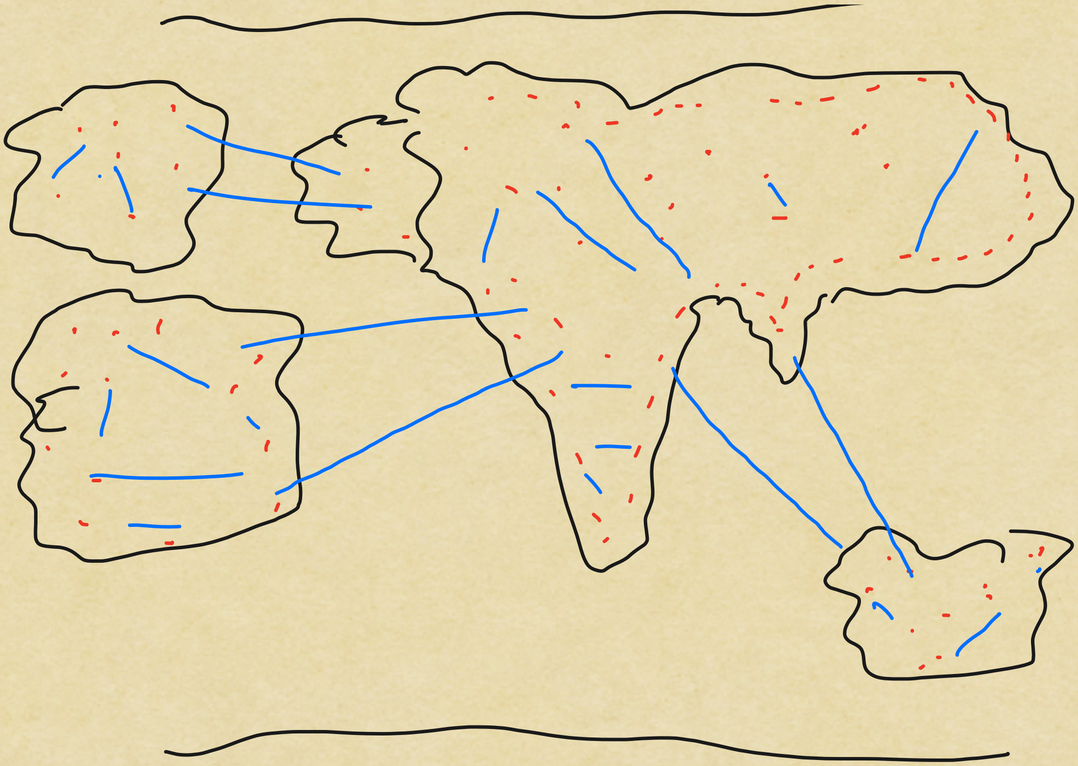
Can we store this data in RAM / HDD / SSD in a single machine $\longrightarrow$ X

↓

to solve, we will need to distribute this across multiple machines

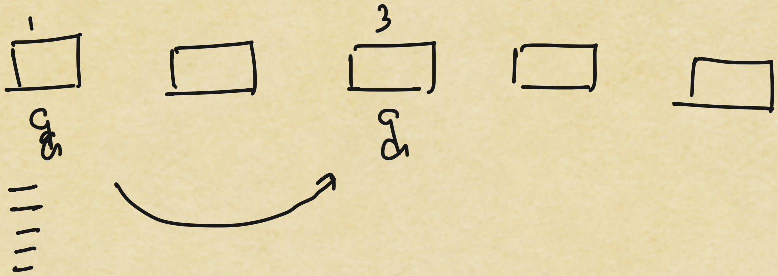
↓

Distributed Systems



- Internet can go down
- Natural disasters
- Machines might crash
- Cyber attack
- HDD might fail

⇒ <u>2006</u>



(bookmarks)

Joshua ⇒ Del·icio·us ⇐ url

signup ⇒ email, password

⇒ store ⇒ bookmarks

CRUD ⇒ list of URLs

↓

Create

Read

Update

Delete

⇒ IP address ⇒ $\underbrace{10}_{0-255}$·11·12·13

Scaler ⇒ $\underbrace{111·112·110·53}$ ⇒ IP addr server → running Scaler

↓

website

↓

domain ⇒ $\underbrace{\text{scaler.com}}_{\text{domain name}}$ ⇒ 111·112·110·53.

Hashmap <K,V>

key ⟶ value

↓ ↓

[name] [IP]

ICANN → maintains the entire worlds DNS servers.

↓

non-profit orgs

→ maintains all domain names for the world

↓

godaddy
namecheap
hostinger

} brokers

DNS

=> domain

www.scaler.com → IP → server

↓

browser (cache)

↓

laptop (cache)

↓

ISP (cache)

↓

DNS → 13 (cache)

↓

name servers (cache)

↓

detail server

domain → register → IP

ICANN

domain	static IP

laptop

Joshua laptop

- code
- domain
- static IP
- mapping → domain + IP

2006

laptop ⇒
- RAM ⇒ 128 MB
- HDD ⇒ 20 GB
- CPU ⇒ 1 GHz 2 core

1 KB data per row

8 bytes

UserId	Url
1	www.scaler.com
2	www.google.com
1	www.youtube.com
3	www.facebook.com
1	www.faceboo.com/friend/xyz

Size estimations

1 row → 1kB

1 user ⇒ 10 bookmarks

per user ⇒ 10 kB

↓

1000 users ⇒ 10 MB

↓

10^6 users ⇒ 10GB

↓

$\underbrace{2 \times 10^6 \text{ users}}_{2M \text{ users}}$ ⇒ 20GB

↓

HDD full

1 lakh ⇒ 10^5 users

↓

5 bookmarks → $5kB \times 10^5$

→ $5MB \times 10^2$

→ 0.5GB / day

10 days ⇒ 5GB

20 days ⇒ 10GB

40 days ⇒ 20GB (X)

~~2 months~~

2) Scaling a system :-

Vertical

20GB | 128MB | 2 core

↓

200GB | 512MB | 4 core

↓

1TB | 2GB | 8 core

↓

server

↓

10TB | 128GB | 64 cores

⋮

single machine

SPOF → single point of failure

* easy to manage

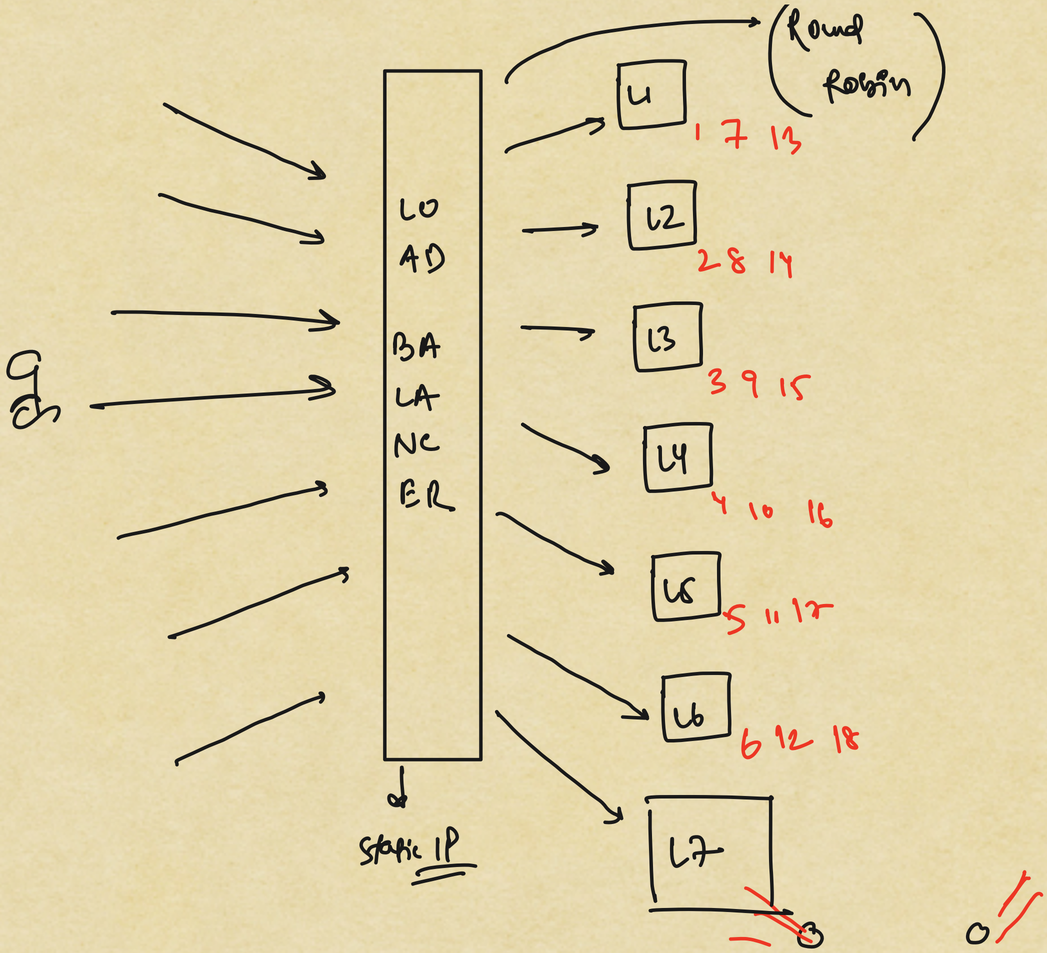
Horizontal

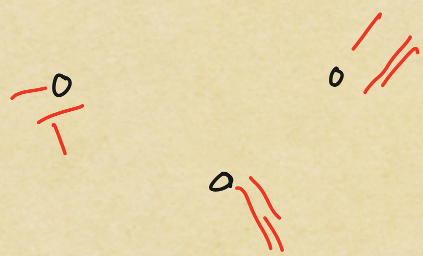
→ multiply

20GB | 128MB | 2 core

. . . .

* cheaper
* easily available
* can add as many as we want
* No SPOF
* management becomes very difficult





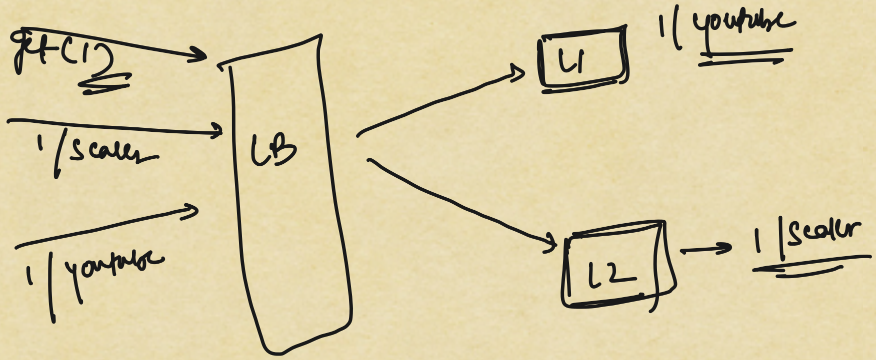
→ Round Robin

→ weighted Round Robin

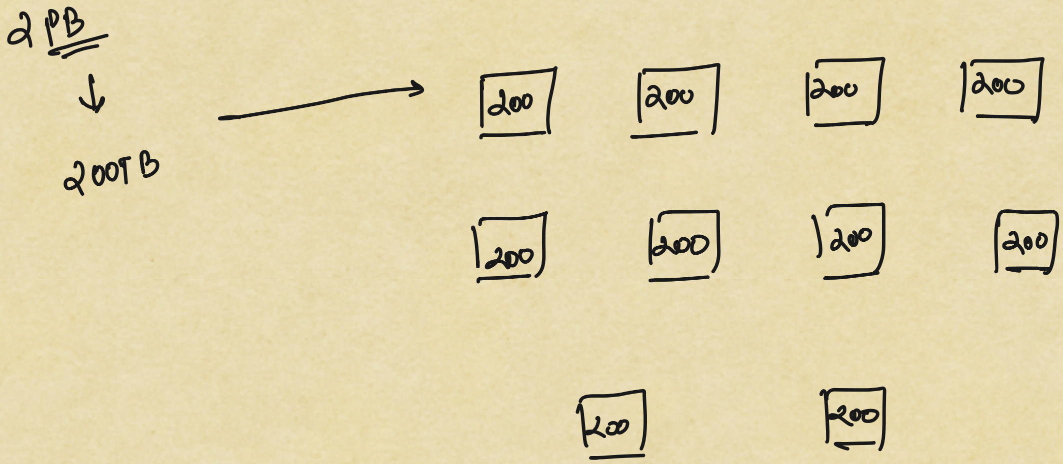
→ Latency → [fastest]

→ smart LBs → [ML algo running to predict and decide servers]

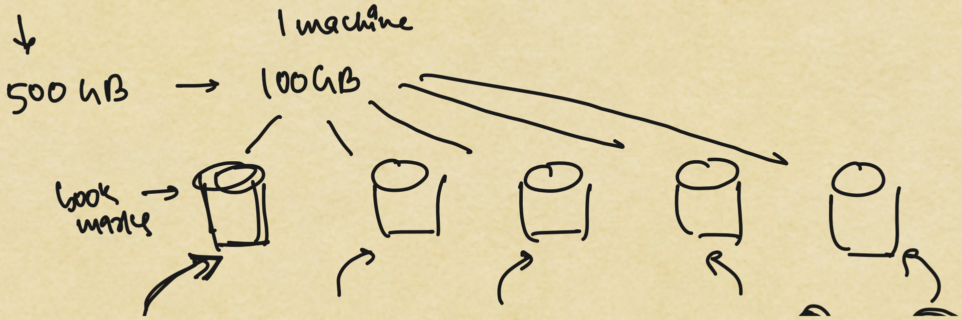
⇒ Distributing data:



⇒ Sharding → distributing data across multiple machines



Usernames



[ABCDE] [FGHIJ] [KLMNO] [PQRST] [UVW XYZ]

high — high — low

↓

even distribution and scalability

=> **Consistent Hashing**

=> HLD Curriculum [tentative]

i) Load Balancing + Consistent Hashing

ii) Caching
- local vs global | distributed
- eviction policies
- algorithms
- invalidation policies
- case studies

iii) CAP | PACELC Theorem

iv) SQL vs NoSQL

SQL:
- work
- internal
- case studies

NoSQL:
- key value
- Document
- columnar
- internal

Sharding

V) Case Studies

→ Build a typeahead

→ Messaging (WA | slack)

→ Elastic Search

→ Ride booking → UBER

↓

[Quad tree]

→ File Storage ⇒ (S3)

→ Video Streaming (Hotstar)

↓

Live + recorded

VI) Messaging Queue ⇒ Kafka

VII) Popular Interview Question