Agenda

1. What & Why of HLD ✓
2. Case Study - del.icio.us ✓
3. Scaling challenges ✓
4. Stateful vs Stateless servers
5. Consistent Hashing
6. HLD curriculum overview

overflow → next class

support@scaler.com

+91 7351769231 (whatsapp)

Traditionally write & execute code on 1 machine (your machine)

↳ handle 100s of requests/sec

How do you scale to million / billion users

↳ how to architect the solution

For senior roles

HLD is a must!!

Google's Staff Soft Eng position.

↳ file → a bunch of strings (search queries)

↳ sort this file in asc.

There is 50 Peta Bytes of data.

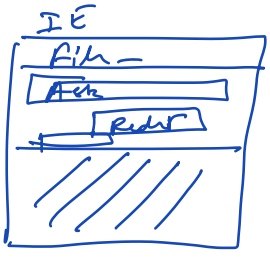
10^3 – KB
10^6 – MB
10^9 – GB
10^{12} – TB
10^{15} – PB

50 Million GB of data.

Del.icio.us Case Study

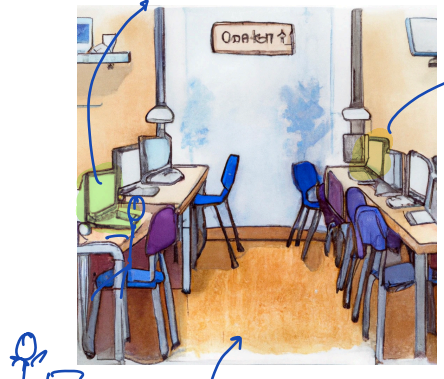
Bookmarking Service.

? Launched in Joshua, 2003



2003 2005 2006 2008

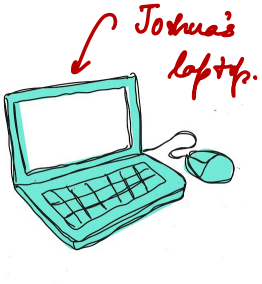
Motivation



access bookmarks from any device!

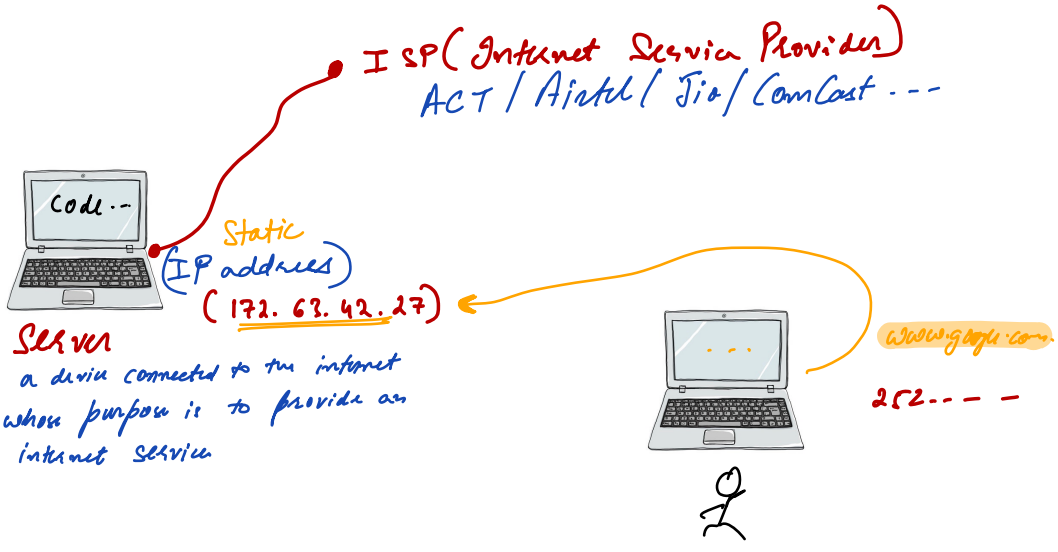
Features → Minimal Viable Product (MVP)

1. User registration & Login (Auth)
2. add_bookmark (url, user_id)
3. get_all_my_bookmarks (user_id)



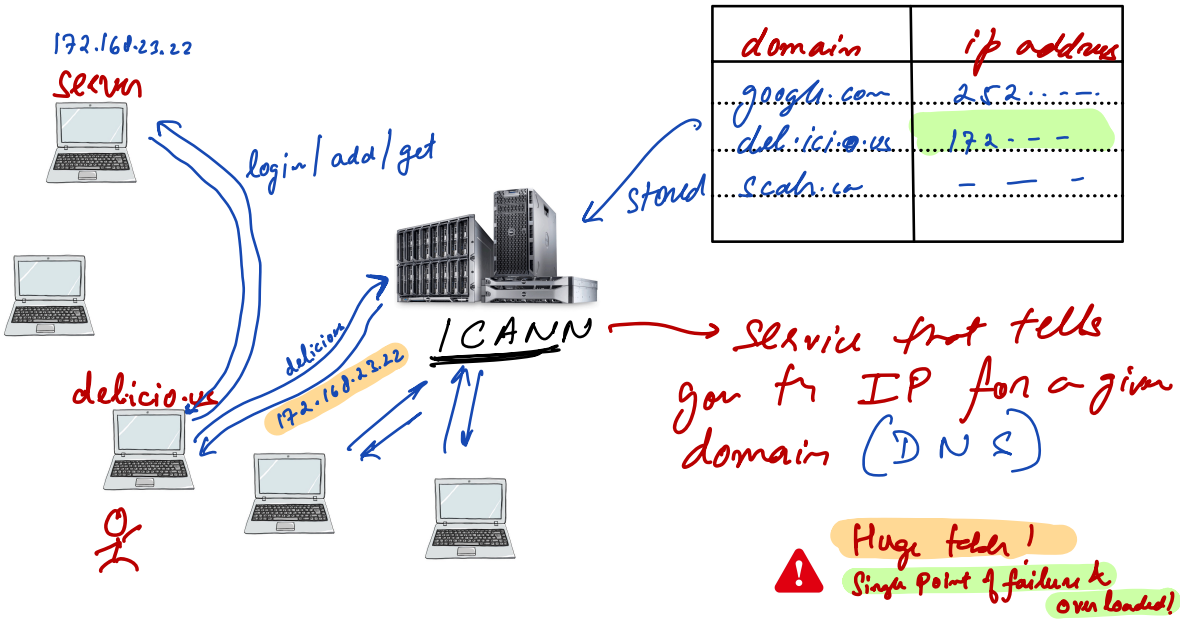
to be continued..

Fantastic websites & where to find them



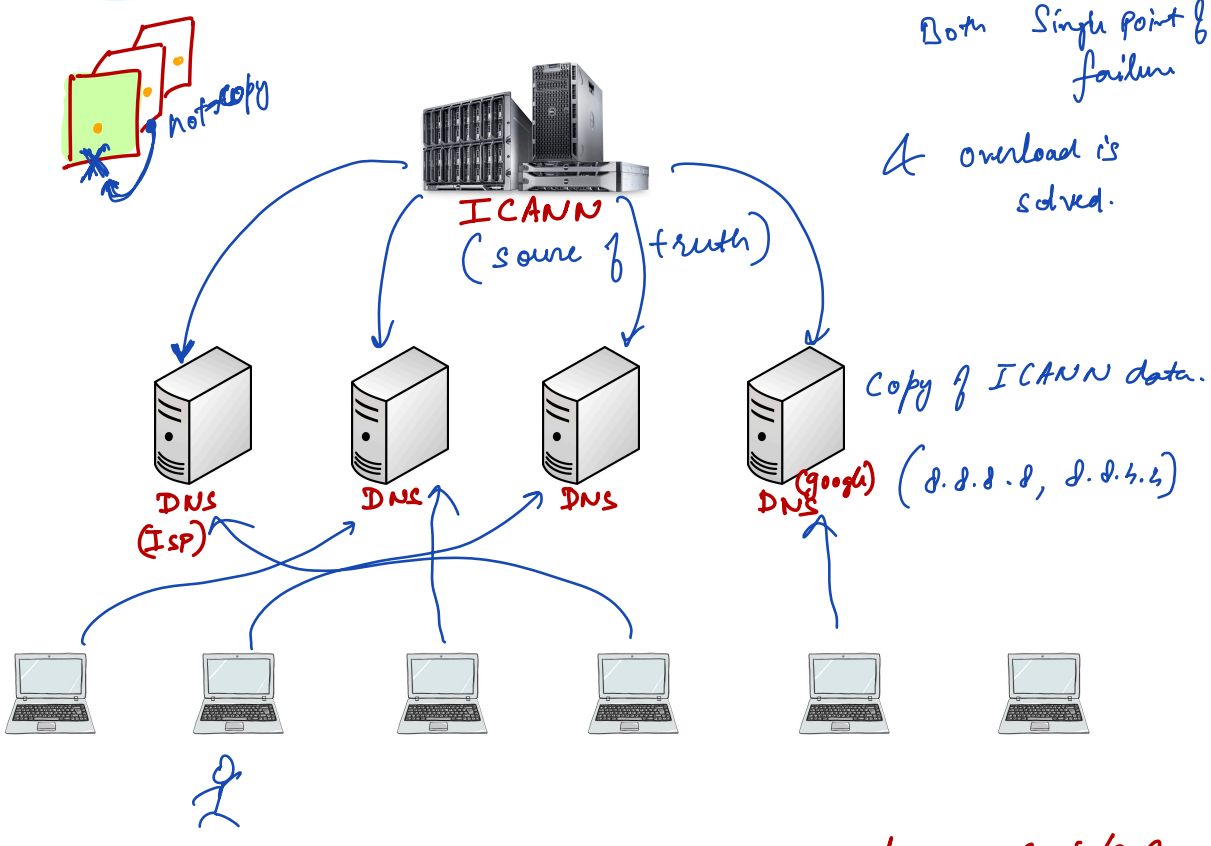
? Quiz - how does the browser know?

↳ somewhere there is a mapping



Domain Name Service

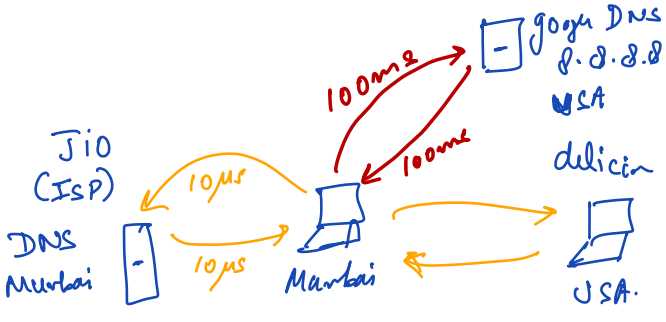
https://www.cloudflare.com/en-gb/learning/dns/what-is-dns/



Who maintains the DNS?

domain registrar
godaddy
namecheap
google domains

Demos



Del.icio.us continued ..

2008

System Configuration

RAM 128 MB RAM

Disk 40 GB

CPU Intel Pentium Dual Core 3.2 GHz.

Postgres

delicious DB Apps/OS/Tools

Issues

① goes to sleep? system is shutdown website goes down!!

② game / downloading website runs slow

③ Power outage website goes down

? Quiz - space (disk)

2 Bytes → # 2^{16} = 65,536 values
4 bytes → # 2^{32} ≈ 4 Billion
8 bytes → # 2^{64} ≈ 16 BB

user_bookmarks

4 bytes user_id	1000 chars url
2	google.com
31	dbdb.io
2	Scaler.com
101	netflix.com

index

1 row ≈ 1 KB (1000 bytes)

1 KB per bookmark
1 Million bookmarks / day
amount of data / day
= 1 KB * 1 M
= 1 GB of data

Quiz - time

1 GB data / day

40 GB HDD available

time to run out = $\frac{40\ GB}{1\ GB / day}$ = 40 days.

Solutions

1. Upgrade Machine

128 MB
40 GB
duo-core

} 30,000 Rs

Personal Computing ←

512 MB
100 GB
duo-core

} 2 lakh Rs

≈ 100 days (3 months)

IBM dedicated Server hardware

2 GB
500 GB
4 duo-core

} 40 Lakh Rs

≈ 500 days (< 2 years)

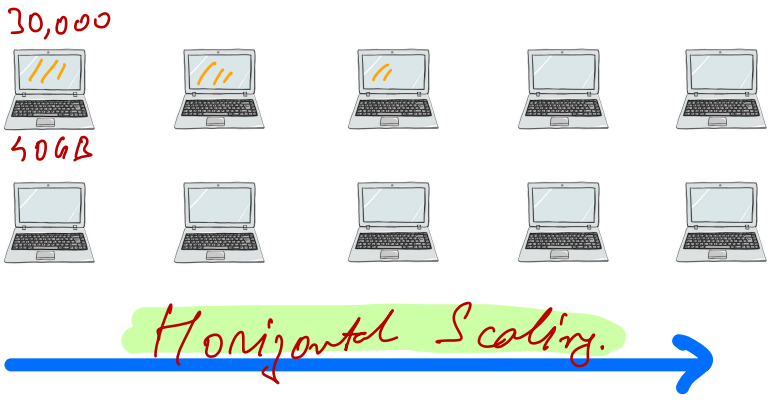
5 GB
1 TB
20 duo-core

} 2 crore Rs

a few years.

VERTICAL SCALING

2. Buy a lot of cheap laptops.



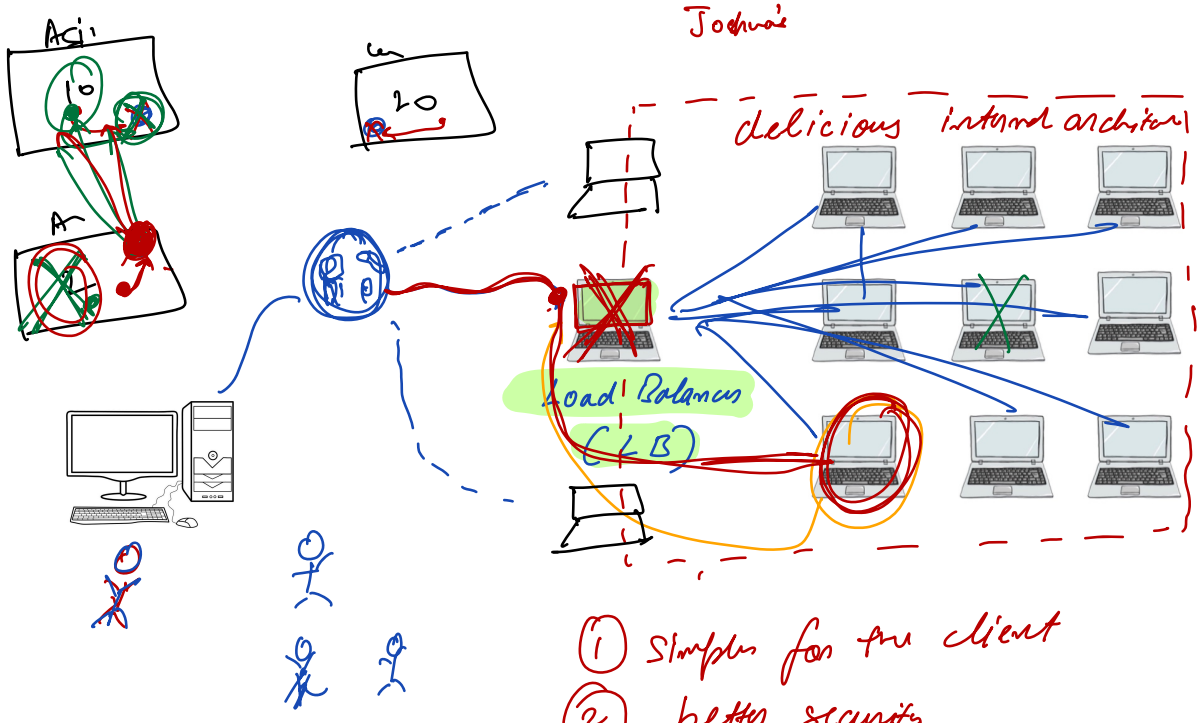
≈ 133 laptops.

Ocean all space
40GB * 133
≈ 5 TB

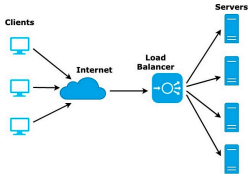
Scale-up Vertical	Scale-out Horizontal
– simple	– difficult to manage
– hard limit to how much we can scale ① technology is not there ② Price will grow exponentially.	– unlimited Price grows sub-linearly ∵ economy of scale.
– first choice	– last choice
– AWS (0-12tb --- 112x large 109 $ per hour 1 million vcpu 448 CPU 12 TB of RAM 100 GBit n/w)	

10:55 → 11:00

? Quiz - which ip?



① simpler for the client
② better security
③ possible to change & scale internal architecture freely.



Load Balancing

https://docs.google.com/document/d/1DxQzLpu1XPe_mRWsewNWtKL6E4uwKHQhBp7GX6Sg7qI/edit

① Load Balancer!
Reroutes requests to the appropriate servers.
↳ distributes load evenly across servers.

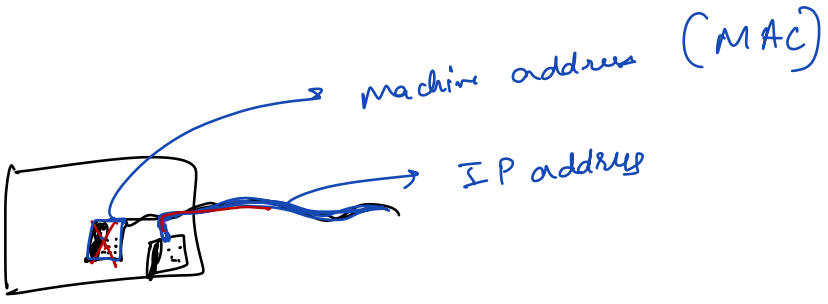
② Isn't the LB a single point of failure?! Not really ∵ easily scalable.

③ isn't the LB overloaded! → ① simple computation. → 10 million req/sec
② multiple load balancers & ip mappings in DNS.

? How does the Load Balancer track which servers are up & running?

Heartbeat (push based) → servers send a periodic signal to LB → "I am alive!!"

Health check (pull-based) → LB sends requests to each server periodically to check if it is alive.

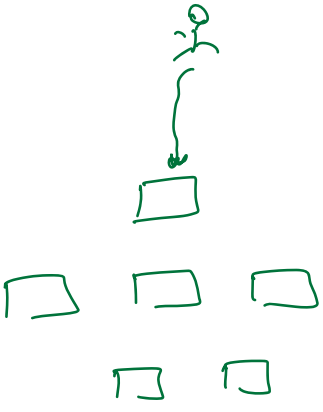

machine address (MAC)
IP address

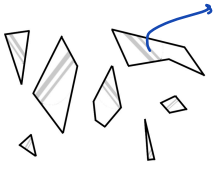
Further Challenges

? What if the Load Balancer goes down? ✓

? Which machine should we send the request to?
Routing Algorithm •

https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/





shard ∫ glass

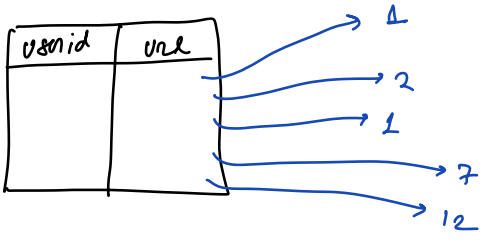
How to store the data?

? Can we store all data on 1 machine? X

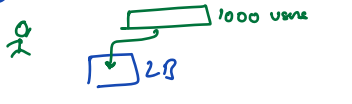
split the data across machines

Sharding

? Split randomly?



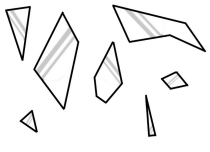
How will we retrieve it?



① store the data in a sharded manner

② also have to read it back.

does reading from all app servers for each request work? X bad.



Sharding

based on user_id

Simple mod ✗

10 users
5 servers
user-id % N

round-robin

U0 → S0	0	0	
U1 → S1	1	1	
U2 → S2	2	2	
U3 → S3	3	4	
U4 → S4	4	0	
U5 → S0	5	1	
U6 → S1	0	2	
U7 → S2	1	4	
U8 → S3	2	0	
U9 → S4	3	1	

① does LB now know which req → which server?
↳ Server crash. (# of servers change)

add a new server ⇒ # servers (N) = 6
↳ every server's data has to be changed.

Server crashes ⇒ S3 down want
↳ only users going to S3 get affected.

Mapping Table

Inside Load balancer

user-id	server-id
	6 ~~3~~
	6 ~~3~~ 6

a new server ✓
server crashes ✓

Cons?? this mapping table can get very large.

Range

10 servers.

U1 — U100 → S1
U101 — U200 → ~~S2~~
⋮
U901 → U1000 → S10
U1001 → U1100 S11

U → S ✓
Server crashes ✓
only the users going to that server are affected

new server gets added.
load on existing servers keeps increasing.

user_id % N = 5

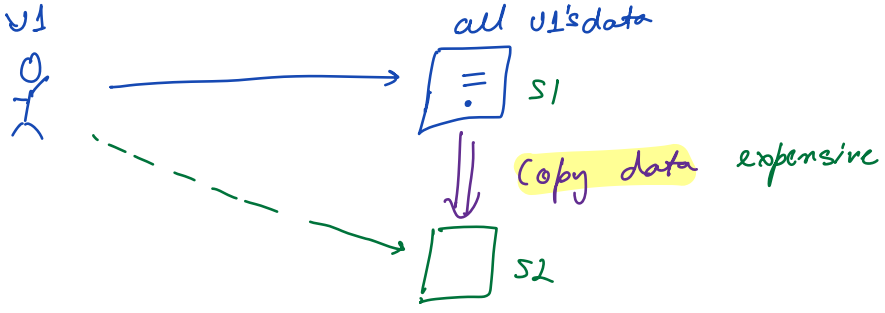
A (0)	B (1)	C (2)	D (3)	E (4)
0	1	2	3	4
5	6	7	8	9

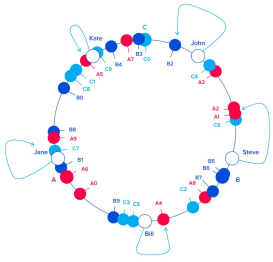
A (0)	B (1)	C (2)	~~D~~	E (3)

user_id % N = 4

0	1	2	3
4	5	6	7
8	9		

Servers go down
- ① which req goes to which server.
- ② restore data (to prevent data loss)
  - data (replication & redundancy)





Consistent Hashing

hash both user-id & the server-ids

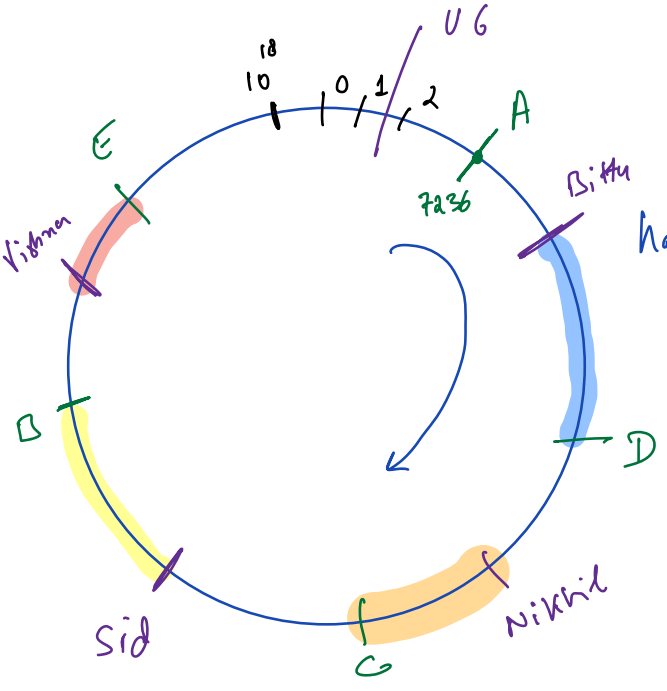
hash-function (user-id / server-id) → [0 ——— 10^{18}]

sha256
md5

long int

$hash(x) = (x^3 + x^2 - 1) \% 10^{18}$

Pass all server ids through the hash function.



A	B	C	D	E
172.168.1.2	—	—	—	—

hash(172.168.1.2)
7236

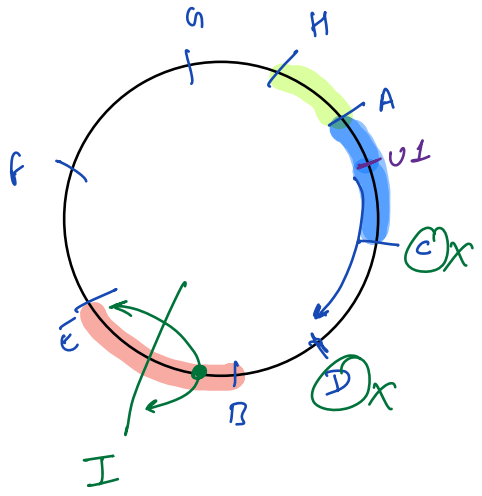
Sid	Bittu	Vish	Nikhil
↓	↓	↓	↓
B	D	E	C

1000 servers

10^{18}

probability of two servers same hash

$1000/10^{18} = 10^{-15}$



① is load equally distributed?

NO!

② Server C crashes.

all of C's load gets pushed to server D

D will also crash

Cascading failure!!

key → userid → server id

mapping should be consistent & easy to compute.

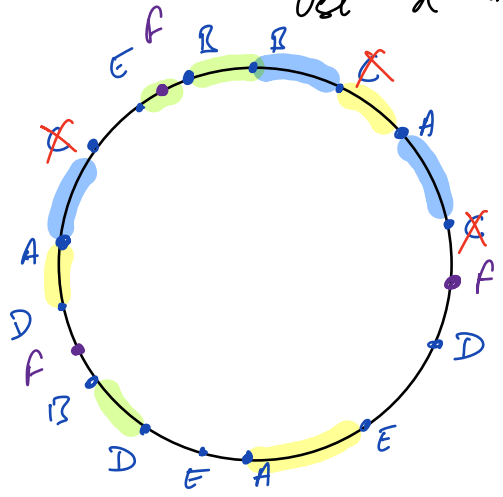
✓ ① new server gets added.

automatically some load is given to that server.

✓ ② server goes down

Instead of using 1 hash function for servers

Use k hash functions. $k = 3$



5 servers

A	B	C	D	E	F
h1	h1				
h2	h2				
h3	h3				

① is load equally divided?

② new server gets added.

load for multiple servers is reduced.

(3) Server crashes
the load gets dist
across multiple servers.

(1) Delicious

(1) HLD 101 & Consistent Hashing
(2) Caching
(3) Caching contd.
(4) CAP theorem & Master Slave
(5) SQL vs NoSQL
(6) NoSQL
(7) Case study — Type ahead
(8) Muti-master
(9) Case study — Messaging. Notification.
(10) Zookeeper + Kafka
(11) Case stg — Elastic Search
(12) Nearest neighbor — Quad Trees
file storage (GS)
(13) Case stg — Uber
(14) Popular Interview Questions
(15) Hotstar
(16) Micro services

theoritical.

(2)
(3) ⟹ Andrew
(4)
(5)
(6) ⇒ Mudit

Implement Consistent Hashing.

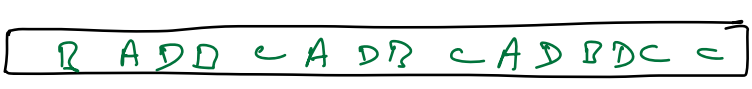
① multiple hash functions.

$h1(x) = (x^2 + x) \% 10^{18}$

$h2(x) = (x^3 - x^2 + 2x) \% 10^{18}$

$h3(x) = (\sqrt{x}) \% 10^{18}$

$hbad(x) = 10$

② 

m servers

k hashes (16, 32)

③ Req ⟶ hash(user_id) ⟶ v

1000 servers

16 hashes

16,000 values

$\log_2(16000) \approx 16$ steps

(17) Consistency — IRCTC concurrency. (7)

Cheat Sheets → when to choose what Database
↳ Common Cloud Provider

8+ years ↑ experience for SDE3 role.

Chat GPT → GPT 3.5 architecture

GPT 3 + RL pipeline
↳ LLM (transformers)
conversation safe

175 billion parameters
4 byte float
space = 700 GB
↑ space
GPU memory

user upto 4000 tokens ↑ history/context
↳ sub-word

every req can take 1 s of CPU computation time
↳ 0.04 $ per request.

100 million req/day.

OpenAI was burning 4M $ per day.

Git / docker

+91 7351769231

pragy@scaler.com

support@scaler.com

MERN → Express → Node
↳ React
↳ Mongo

SpringBoot → Java.
Django → Python

Backend

Spring / Dja

Full stack.

Full stack

MERN

Data Engg