High Level Design - 101

Friday, 19 November 2021 8:35 PM

pragy@scaler.com

SAT
└ mandatory
└ by tomorrow night

2.5 hours

LLD - 101 class happens tomorrow @ 9.00 pm (Saturday)

Agenda
1. What & Why of HLD
2. Case Study - Del.icio.us
3. Consistent Hashing - V.V. imp.

What is HLD

Code ← input, ↓ output — 1 machine

≤ 15% Netflix / AWS / Google / NFT firms / CDN / Social media / Banks / Youtube / games

code - 1 machine - simple
└ 100,000 machines - complex

HLD interview Q
└ given a list of names, sort it. } 50 PB of data.!

50 × 1000 × 1000 × 1000 × 1000 × 1000 bytes

20 Tb

2.5 × 1000 × larger

Study of large scale systems from 50 feet view
architecture
└ Scaling

SDE - 1
- 2+ positions

HLD is a must!!

I'm working as XYZ dev.

Scale - no!
~ 4,000 →
~ millions

Video streaming → delegate → give the task to someone else

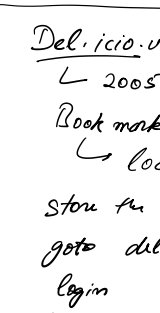
Code judge → ~30s to check 1 submission

Facebook / Google / Netflix / LinkedIn --
is learning the concepts going to be enough?

1 Tb/s bitrate @ Hotstar

① Jargon
② Reasoning behind them

Del.icio.us
└ 2005/6
Bookmarking website
└ local to your machine
Store the bookmarks on website
goto delicious
login



The laptop

① machine
② Internet connection
③ Run a server on your machine
└ kept it running

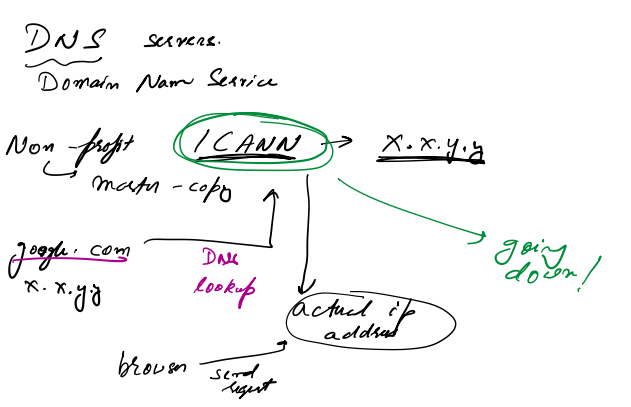
Each machine has an address
└ IP address - unique to each machine
IPv4 address
207.165.7.23 → never use ip addresses.

domain names

http:// www.google.com ↵
⇓
(ip-address)

domain → ip address | global map: domain → ip-address

DNS servers.
Domain Name Service



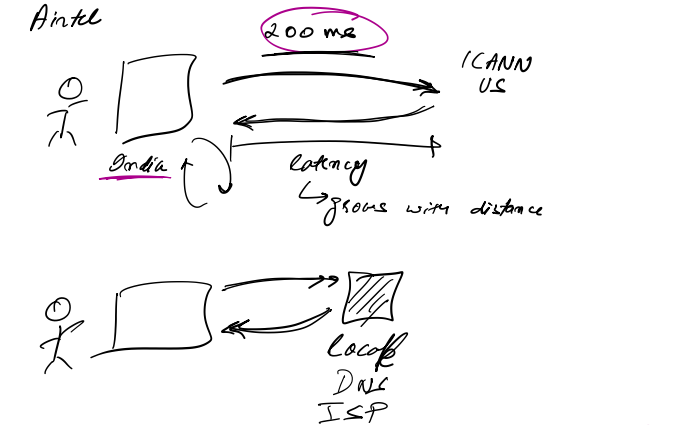
Non-profit
ICANN
x.x.y.y
master-copy
going down!
google.com
x.x.yy
DNS lookup
actual ip address
browser
send request

Most large organization
└ Companies
└ govts
└ ISP

google → DNS 8.8.8.8 / 8.8.4.4 } google DNS server

own DNS servers ⇓ have a copy of the ICANN mapping.

Airtel



200 ms
ICANN
US
India
latency
└ grows with distance
Local
DNS
ISP
└ Internet Service Provider

ISP → DNS server
default DNS server
System → override the DNS
8.8.8

---

① Laptop → run website
└ sleep
└ off → website down
heavy tasks → website slow

another laptop → always up & connected
dedicated machine to run the website
↓
server → get a special connection from the ISP

② 10 GB of space remaining
20 GB / 40 GB

2007 → 256 MB RAM & 4 GB HDD / CD Floppy disks └ 16 Kbps connection

RDMS
user-bookmarks

user-id	url

4 byte int — 1000 characters ~1000 bytes — 1 Kb of space

1 million new bookmarks each day
1/second
1 KB

1M records
1M × 1KB = 1 GB

---

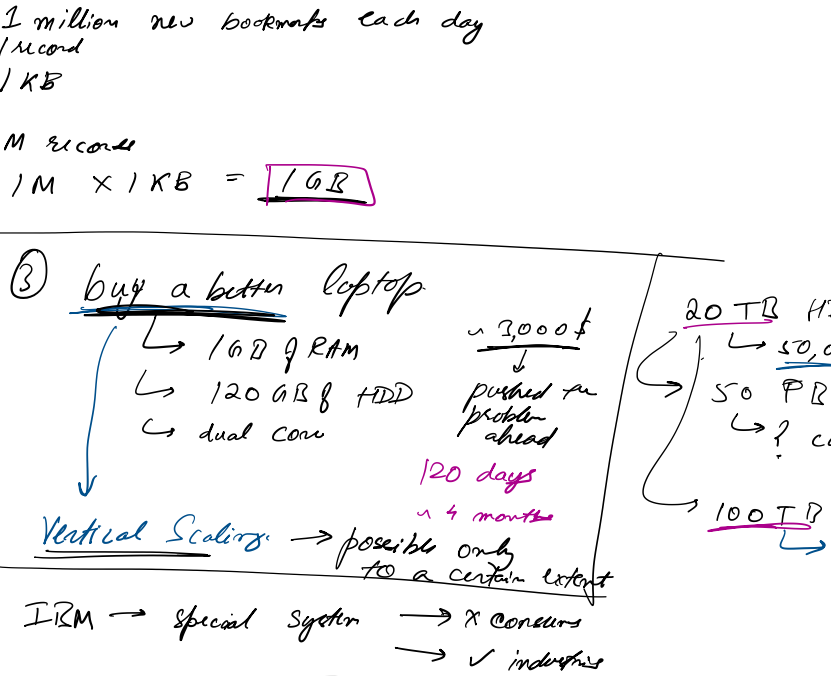
③ buy a better laptop
└ 16 GB of RAM
└ 120 GB of HDD
└ dual core

~3000 $ ↓ pushed the problem ahead
120 days
~4 months

20 TB HDD
└ 50,000 Rs
50 PB HDD ✗ no company is producing it
└ ? cost??
100 TB HDD
└ 50 Lakh Rs

Vertical Scaling → possible only to a certain extent

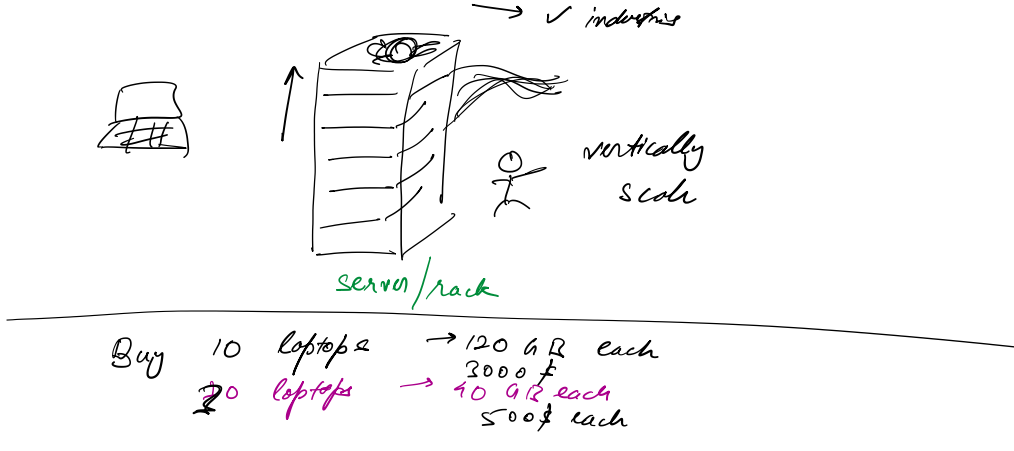
IBM → special system → ✗ consumers, ✓ industries

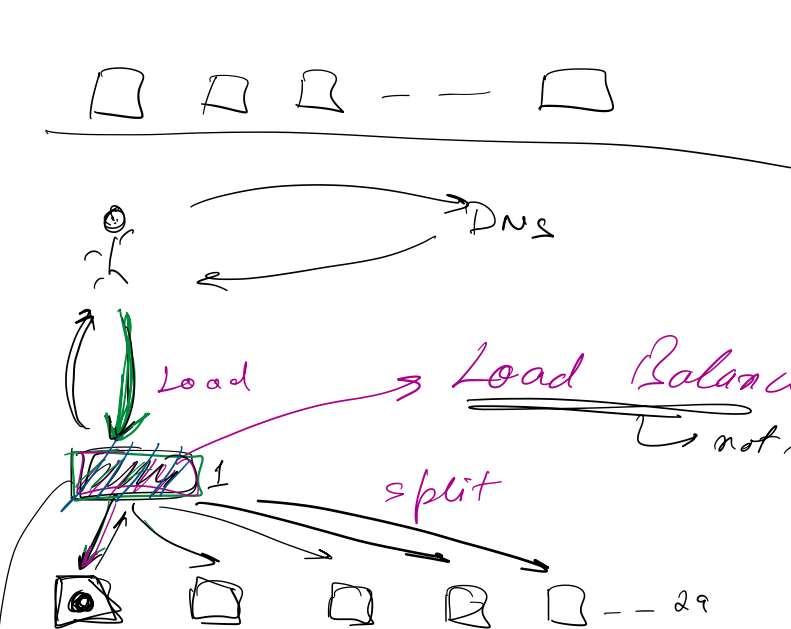


vertically scale
server/rack

Buy 10 laptops → 120 GB each, 3000 $
20 laptops → 40 GB each, 500 $ each

buy a lot of cheap machines





DNS
Load
1
split
storage
Servers
29

Load Balancer → only forwards requests
└ not really
└ doesn't do any fancy compute
doesn't store data

fancy CPU → quad core / 16 GB HDD, CPU / Powerful n/w card

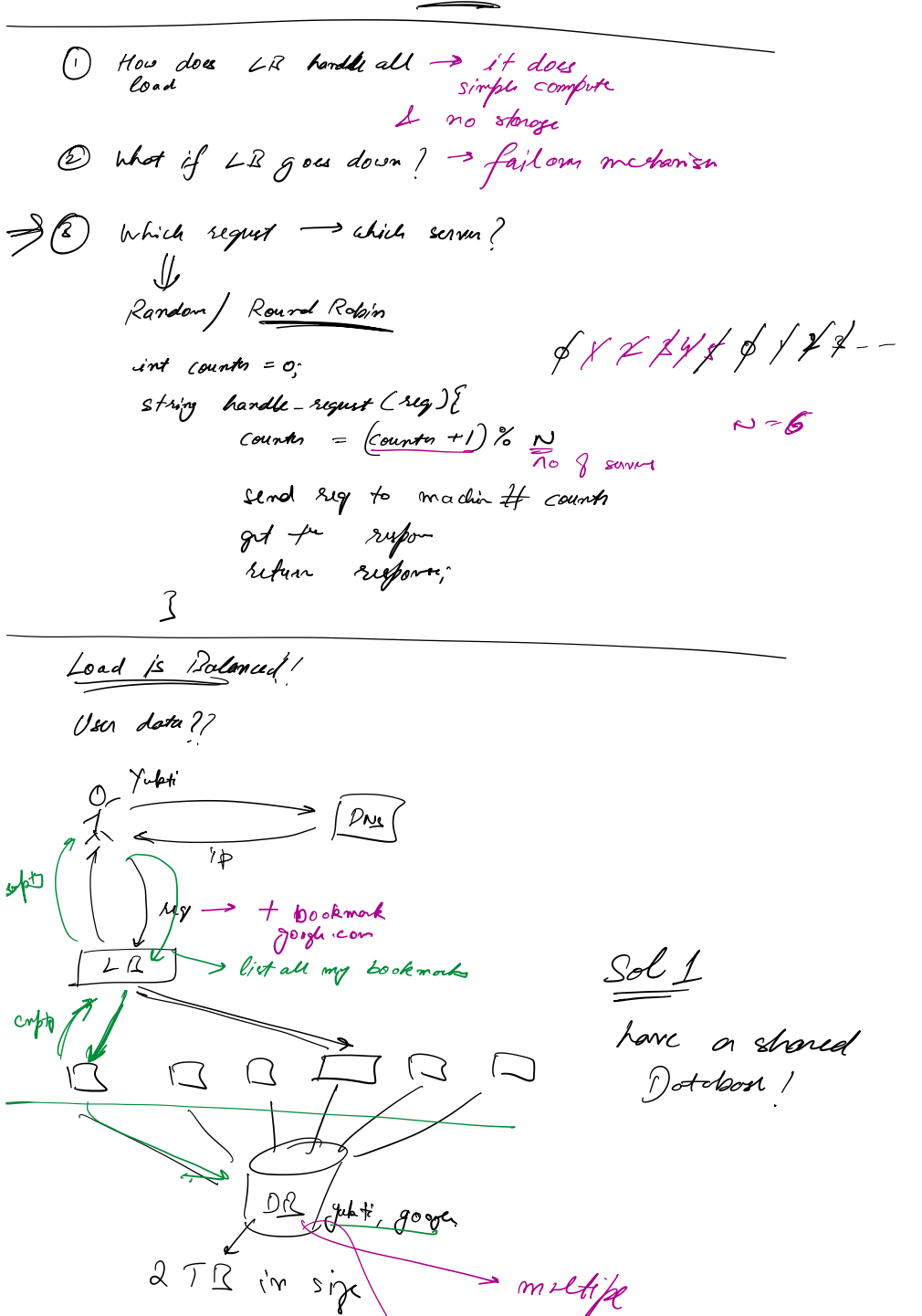
---

① How does LB handle all load → it does simple compute & no storage
② what if LB goes down? → failover mechanism
③ Which request → which server?
⇓
Random / Round Robin

```
int counter = 0;
string handle_request(req) {
    counter = (counter + 1) % N   // no of servers
    send req to machine # counter
    get the response
    return response;
}
```

N = 6

Load is Balanced!
User data??



Yukti
DNS
ip
req
+ bookmark google.com
list all my bookmarks
LB
copy
DB
Yukti, google
multiple machines

Sol 1
have a shared Database!

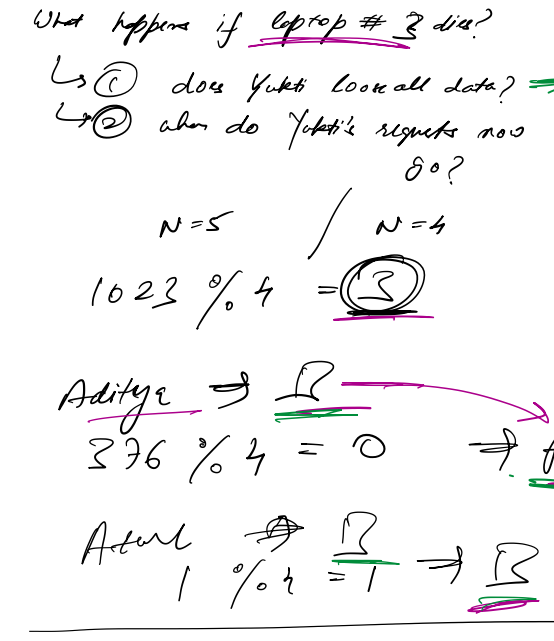
Same problem here!

2 TB in size
40 GB type

---

instead of round-robin
└ non-deterministic
unpredictable

Somehow Yukti → always goes to laptop 3 ✓



DNS
req → user-id
LB
Yukti, google

```
void handle_request(req) {
    user_id = req.user_id;
    machine_id = (user_id) % N;
    delegate req to machine # machine
    return response
}
```

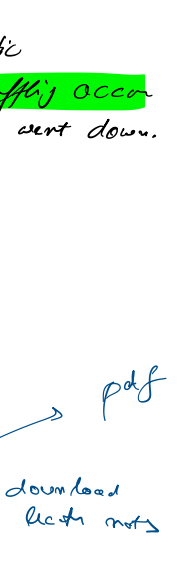
Yukti → 1023 → ③ N = 5 (1023 % 5 = 3)
Aditya → 376 → 1
Atul → 1 → 1

What happens if laptop # 3 dies?
└ ① does Yukti lose all data?
└ ② where do Yukti's requests now go?

N = 5 / N = 4

1023 % 4 = ③

Aditya 376 % 4 = 0 → A
Atul 1 % 4 = 1 → B



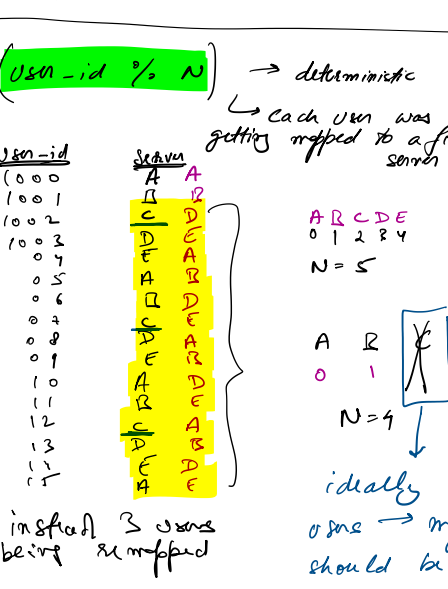
0 1 2 3 4
A B C D E
Aditya
backups safe!
40 GB
copy all data → expensive operation
HDD speeds
└ 30 MBps
40 GB
└ 1200 secon
20 mins.
0 1 2 3
A B C E
Adity
avoid?
No / server went down

was this change necessary?
✗

---

① Round Robin → non-deterministic
② userid % N → deterministic
a lot of shuffling occurs when a server went down.
③ Consistent Hashing

10:28 → 10:40
~11:30



Class
All Class
Live C
Ass
Mo
Watch now
pdf
download lecture notes
pdf view
dark mode

---

user_id % N → deterministic
└ each user was getting mapped to a fixed server

user-id	server	
1000	A	B
1001	B	C
1002	C	D
1003	D	E
04	E	A
05	A	B
06	B	D
07	C	E
08	D	A
09	E	B
10	A	D
11	B	E
12	C	A
13	D	B
14	E	D
15	A	E

A B C D E
0 1 2 3 4
N = 5

A B ~~C~~ D E
0 1 2 3
N = 4

ideally users mapped to C should be remapped

instead of 3 users being remapped
14 users are being remapped

Lots of shuffling

server C went down

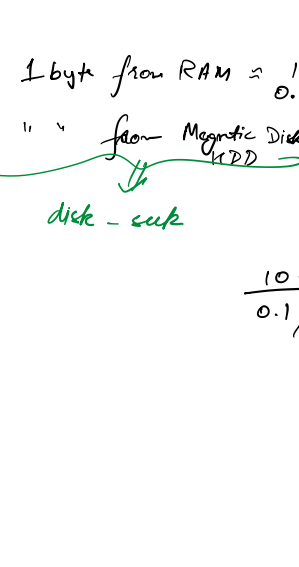
Memory of LB
└ maintained a mapping
store user-id → server-id

user-id	server	
1000	A	
1001	B	
1002	C	→ A
1003	D	
04	E	
05	A	
06	B	
07	C	→ B
08	D	
09	E	
10	A	
11	B	
12	C	→ D
13	D	
14	E	
15	A	

# of users

maintain this mapping!

how do you do if S_x goes down what users are affected?



(user id + server id) * # of users

4 bytes 32 bit int — 4 bytes 32 bit = 8 bytes per mapping

1 billion users → 8 GB of RAM

8 GB
HDD ✓
bottleneck

Read 1 byte from RAM ≈ 100 ns 0.1 μs
" " " from Magnetic Disk HDD ≈ disk - seek ≈ 10 ms

10 ms / 0.1 μs = 100,000 ×

most users → do the compute
for some → store mapping
user-id ⇒ map ✓
user id % N ?

user id % 4	
0	→ 0
1	→ 1
2	→ 2
3	→ 3
4	→ 0
5	→ 1
6	→ 2
7	→ 3
8	→ 0
9	→ 1
10	→ 2

2 goes down

AWS → 10,000 static ips
elastic ip → AWS id instead com

aid	ph

Consistent - Hashing

① Put users & servers on the same number line
② Move in 1 direction
③ go crazy

user-id, server id → live in the same space.

N → M

function
f(input) → output
domain: Real no — range: 0/1, 0 - 10

Hash function

f(x) = x²
① if x ≥ 0 the this function can be reversed!
② output is not bounded.

f(―) = 100, x=? = 10
f(10) = 100
$f(10^{100})$ = 1

f(x) = x²
$f^{-1}(y) = \sqrt{y}$
f(10) = 100

① it should be irreversible (uniform)
② Output should be bounded / finite

∞ possibilit → finite possibilities 100

output is bounded/finite

user id % N
(% 5) → {0 ... 4}
1000 % 5 = 0
1005 % 5 = 0

f(0) → 10
f(1) → 20
f(2) → 30
f(3) → 40
f(4) → 50
f(5) → 20

5 diff values at max

many → one
by nature irreversible!

ans = 20
x? → ✗ no ans.

Hash (――) → (―) finite amount of possibilities

% 5, % 6, % N
Hash functions.
(x*2 + 3) % 7

sin (x)

user-id → H (user-id) ⇒ {0 - 5}
server-id → H (server-id) ⇒ {0 - 5}

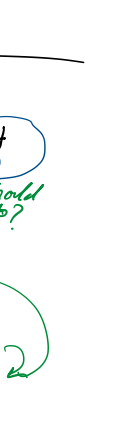
1000 ... 1010

users & servers → same no line
output of hash(―) ↓ output range has to be the same.

H(x) = x % 10 → users ⇒ 0 - 9
G(y) = (y % 5) * 2 → servers ⇒ 0 - 8

x	H(x)
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	0
11	1
12	2
13	3
14	4
15	5
16	6
17	7
18	8
19	9

y	G(y)
0	0
1	2
2	4
3	6
4	8

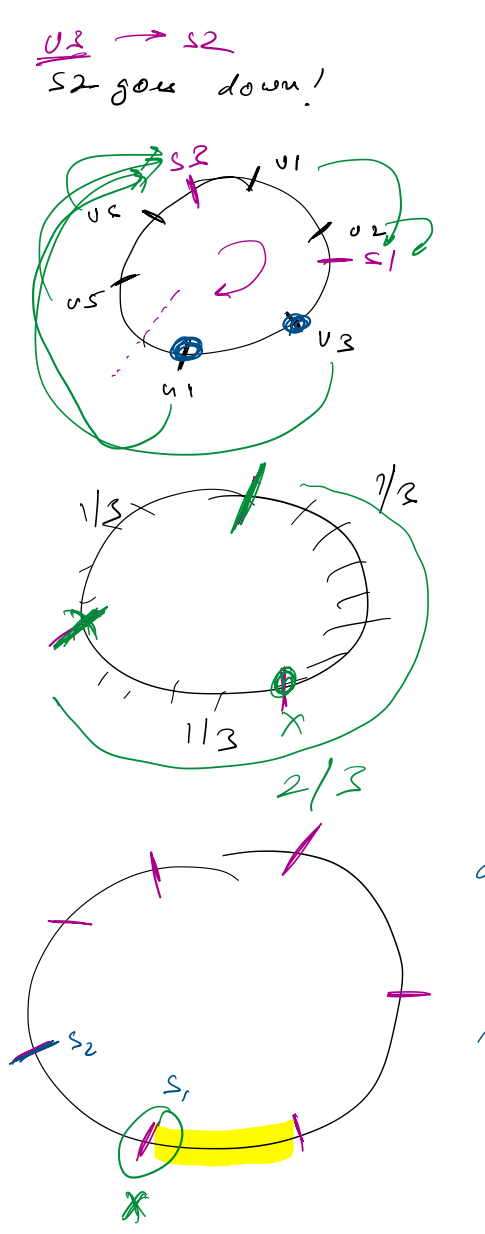


0 1 2 9
0 1 2 3 4 5 6 7 8 9
10 11 12 19
S_1 S_2 S_3 S_4 S_5

---

for a given user → what server should they be sent to?

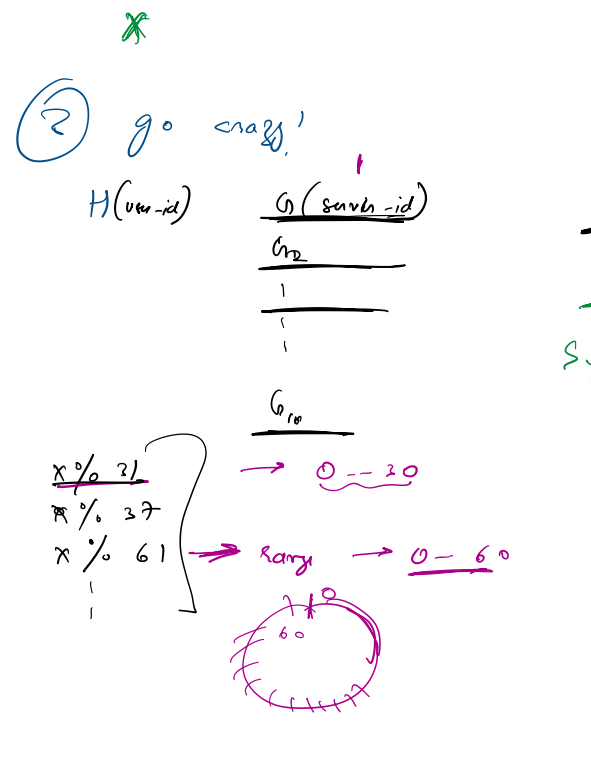
move right



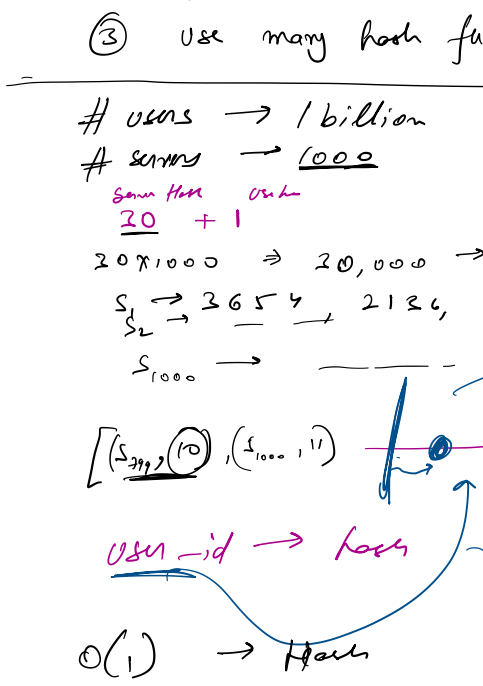
S2
U1
U2
S1
U3
U4
U5
U6
U2 → S2
S2 goes down!

solved → if S_x goes down only users mapped earlier to S_x will get remapped!!

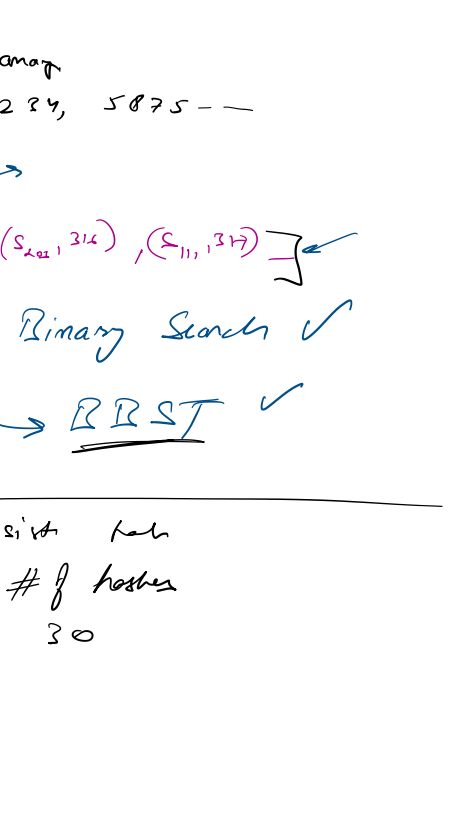
issue → all the remappings go to 1 poor server..



Cascading failure



desired: yellow load shouldn't all go to S2
it should be equally distributed



S1 S2 S3 S4 S5 S6 S7
equally dist.

② go crazy!

H(user-id) — G (server-id), G₂, ..., G₁₀

x % 71 → 0 - 70
x % 37
x % 61 → range → 0 - 60

① put servers & clients on the same circle
② go clockwise
③ use many hash function for each server.

---

users → 1 billion
servers → 1000
30 server hash + 1 user hash

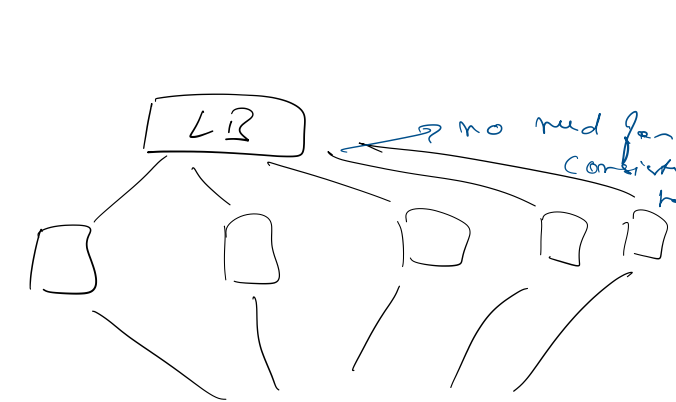
30 × 1000 ⇒ 30,000 → array
S_1 → 3654, 2136, 1234, 5875 ---
S_2 →
S_{1000} →

[(S_{89}, 10), (1, ..., 11) ... (S_{889}, 314), (S_{11}, 387)]

user_id → hash → Binary Search ✓
BBST ✓

O(1) → Hash
O(log n) → Hash + consist hash
└ # of servers * # of hashes
1000 × 30
= 30,000

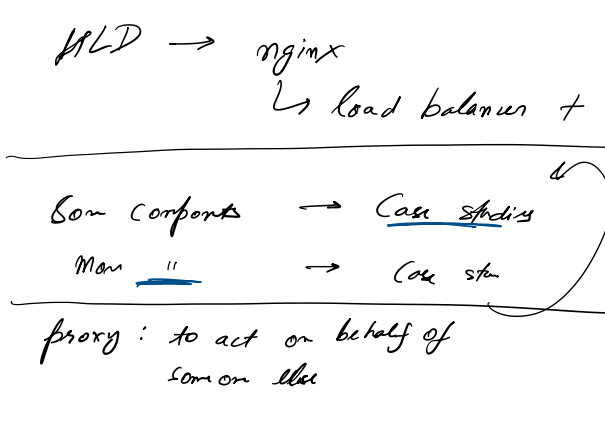
$\log_2(30,000)$ ≈ 15
2^{16} = 65536
2^{15} ≈ 30K



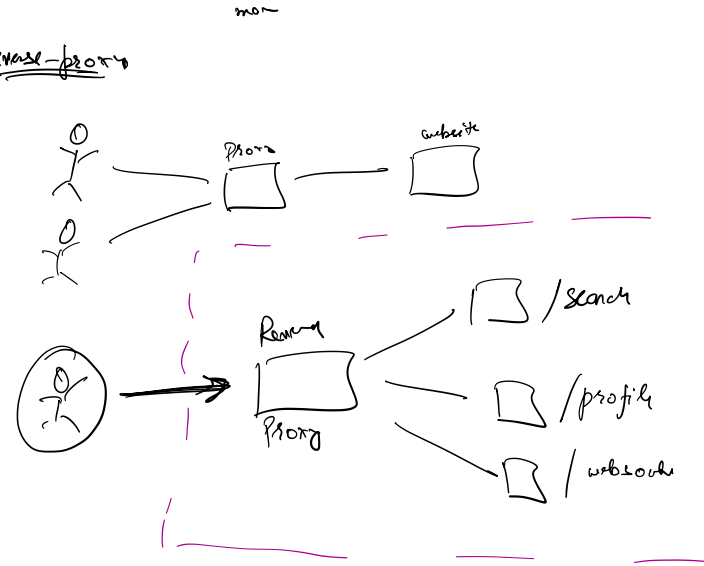
detect → goes down → polling / Heartbeat
↓
process copy

all data is backed up somewhere

① Put users & servers on a circle
② move clockwise to allot server to user
③ Have multiple spots for each server



S_{10}
S_{10}
S_{10}
S_{10}

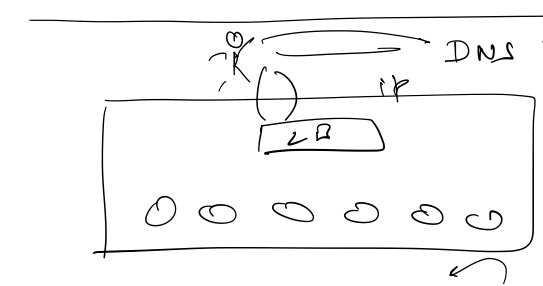


LB
no need for config & hash
Config file

HLD → nginx
└ load balancer + reverse proxy

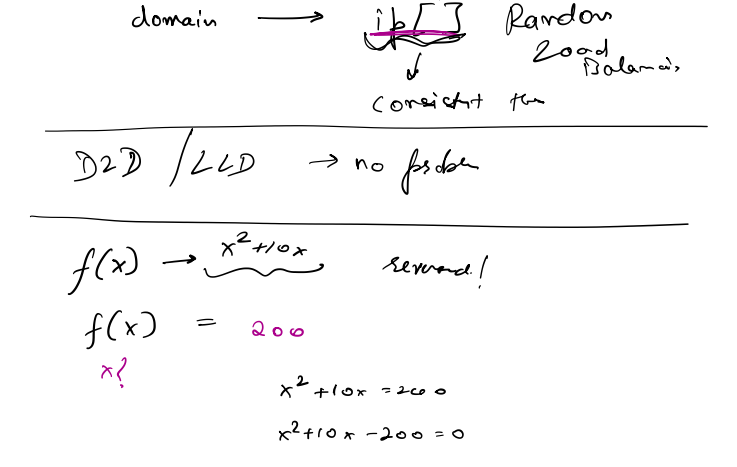
Gen components → Case studies
More " → Case st

proxy : to act on behalf of someone else



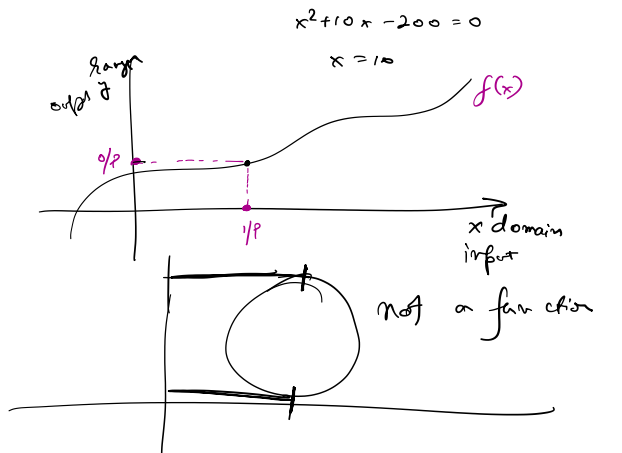
proxy server
proxy
middle man

Reverse-proxy



proxy
website
Reverse Proxy
/search
/profile
/website

nginx → load balancer / reverse proxy



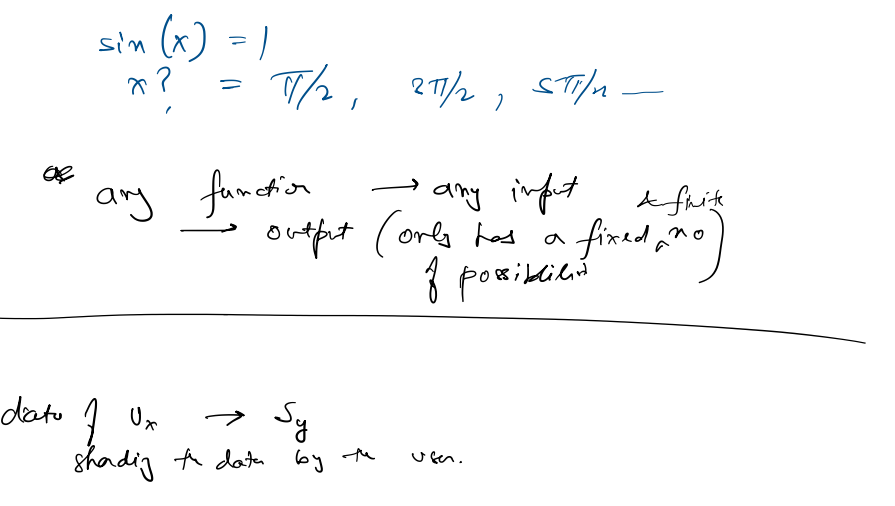
DNS Server
ip
LB

DNS server → itself act as a LB

domain → ip → Random Load Balancing, consistent hash

D2D / LLD → no follow

f(x) → x² + 10x reversed!
f(x) = 200
x?
x² + 10x = 200
x² + 10x - 200 = 0
x = 10

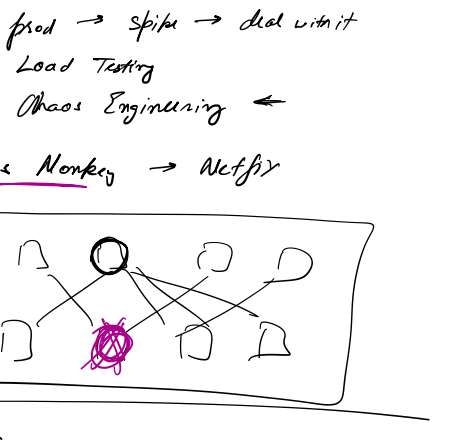


sin (x) = 1
x? = π/2, 2π/2, 5π/2 —

any function → any input
→ output (only has a fixed & finite no of possibilities)

---

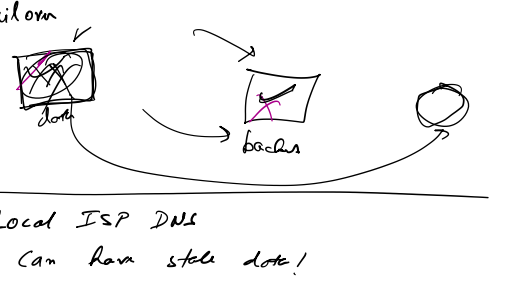
data of U_x → S_y
sharding the data by the user.



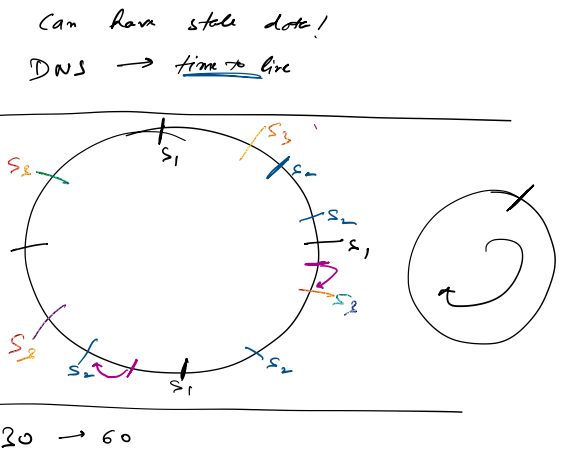
time
req
failed
resp
Time out
S_x

Testing Scalability
① prod → spikes → deal with it
② Load Testing
③ Chaos Engineering ←

Chaos Monkey → Netflix

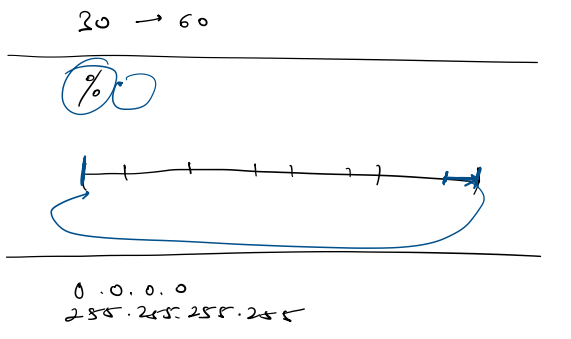


failover



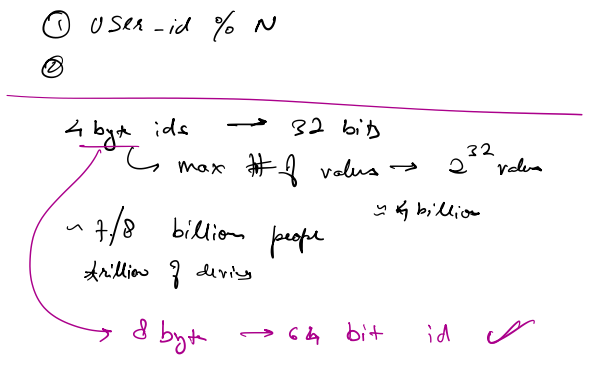
dead
backup

Local ISP DNS
can have stale data!
DNS → time to live

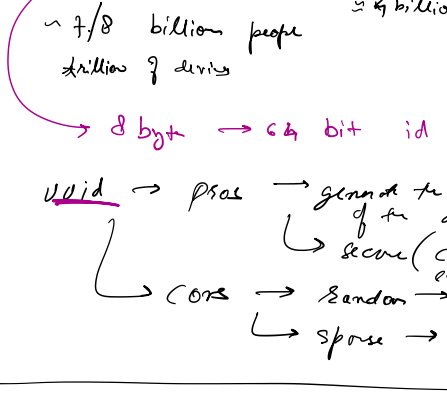


30 → 60

%



0.0.0.0
255.255.255.255



collision avoidance
① hash function → prime no
f(x) = x % p (7 digit)

① user-id % N
②

4 byte ids → 32 bits
└ max # of values → 2^{32} values ≈ 4 billion
~7/8 billion people
trillion of devices
→ 8 byte → 64 bit id ✓

uuid → pros → generated independently of the database
└ secure (cannot be guessed)
└ cons → random → indexes are slow
└ sparse → 128 bits