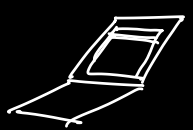
- ✓ DSA → Optimal, accurate code
- ✓ LLD → How to structure the code
- ✓ HLD → How to scale your system.
  - OS ✓
  - DBMS ✓
  - CN ✓
  - DSA ✓

2005/2006

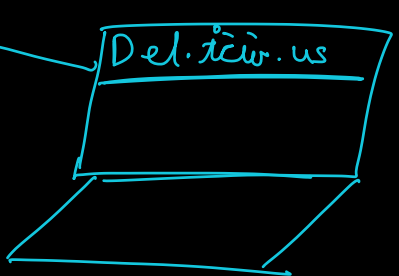


→ Create account

→ Save Urls

DEL.ICIO.US

- → CreateAccount
- → addBookMark(UserId, site_url);
- → getAllBookMarks(UserId);



CN : DNS

Narsinghpur (MP)

497001

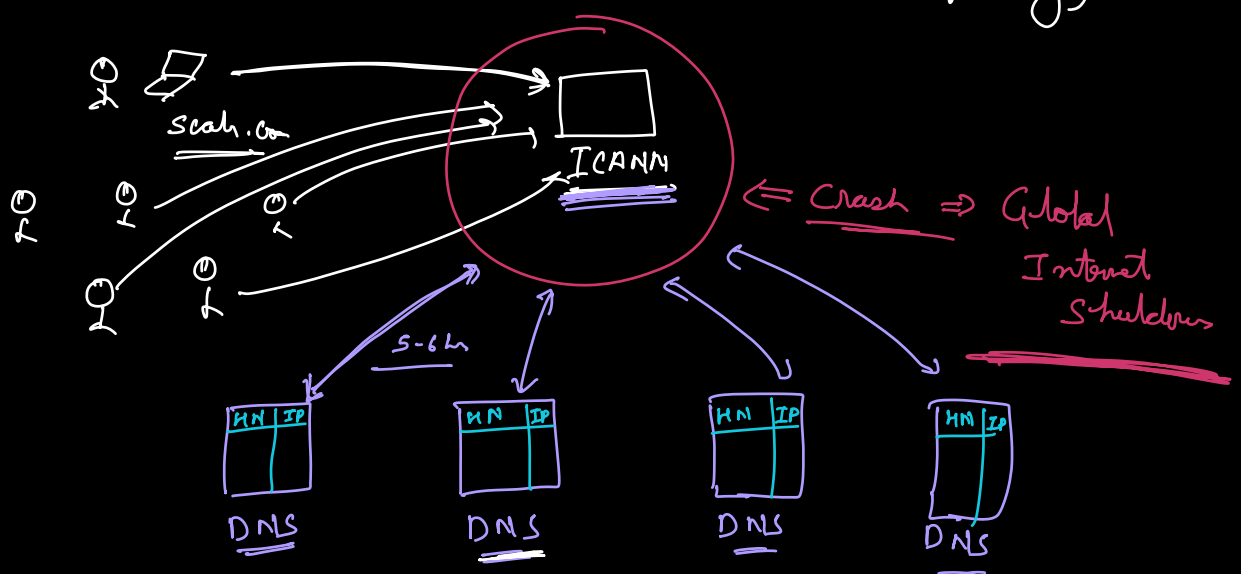
→ Ambikapur Chali

Host Name	IP
www.scaler.com	121.56.41.85
fb.com	58.56.120.11

Centralized authority

ICANN

(Non profit org)



Who maintains DNS ?

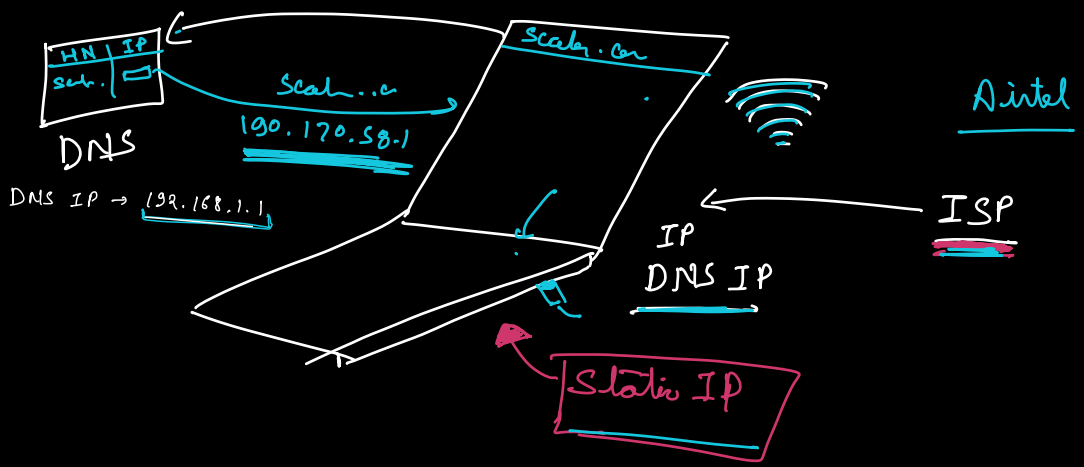
→ Internet service provider (ISP)

Airtel, TATA Docomo, Jio

→ Google, Facebook

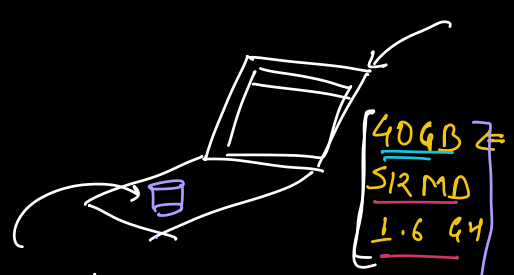
→ 8.8.8.8

→ 8.8.4.4



Please DO NOT write while I am speaking.

2005/2006



ICANN

Delicious	10.24.28.10

User Id	site_url
10	scaler.com
10	fb.com
11	cowin.

1 record → 1000 byte ≈ 1 KB

if 1 million req / day. (add bookmark

then memory req to save records / day

1 KB × 1 million

1000 B × 10^6

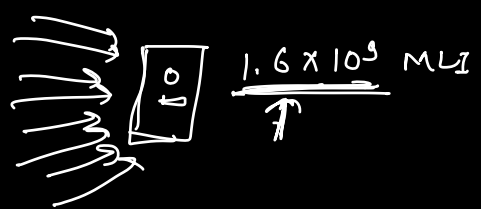
10^9 B ⇒ 1 GB

1 GB records getting saved every day.

& 40 GB ⟶ < 40 days

Constraints

- Will run out of memory within 40 days
- 1.6 GHz
- 512 RAM



(1) Buy a better laptop

(500 GB, 4 GB RAM) → 500 days.

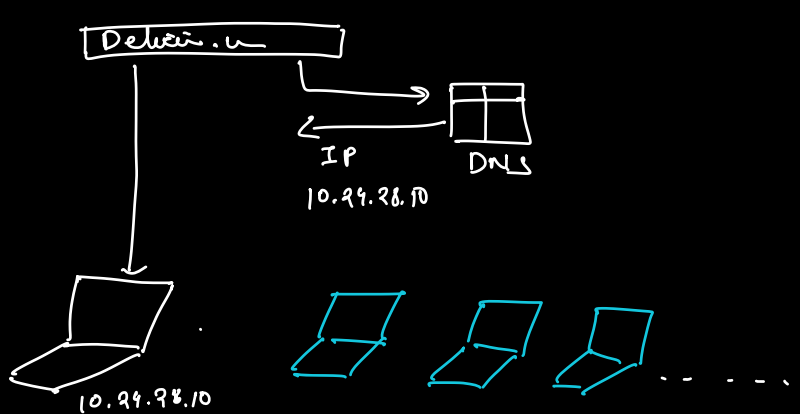
Vertical Scaling ↑ 500GB, 40GB

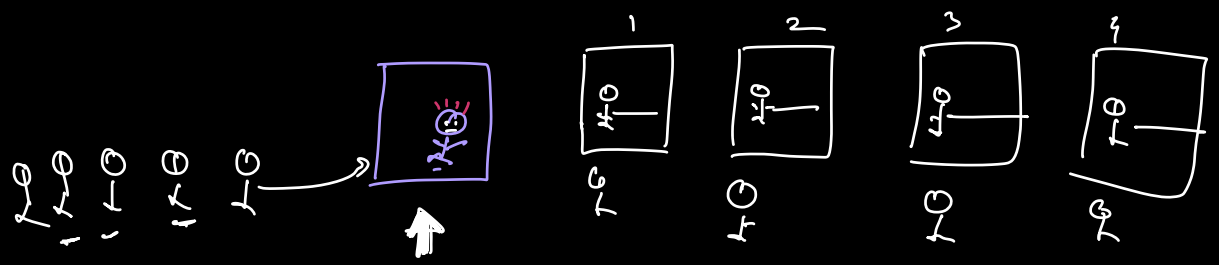
(2) Buy more laptops (of same config)

20 × 40 GB

800 GB ?

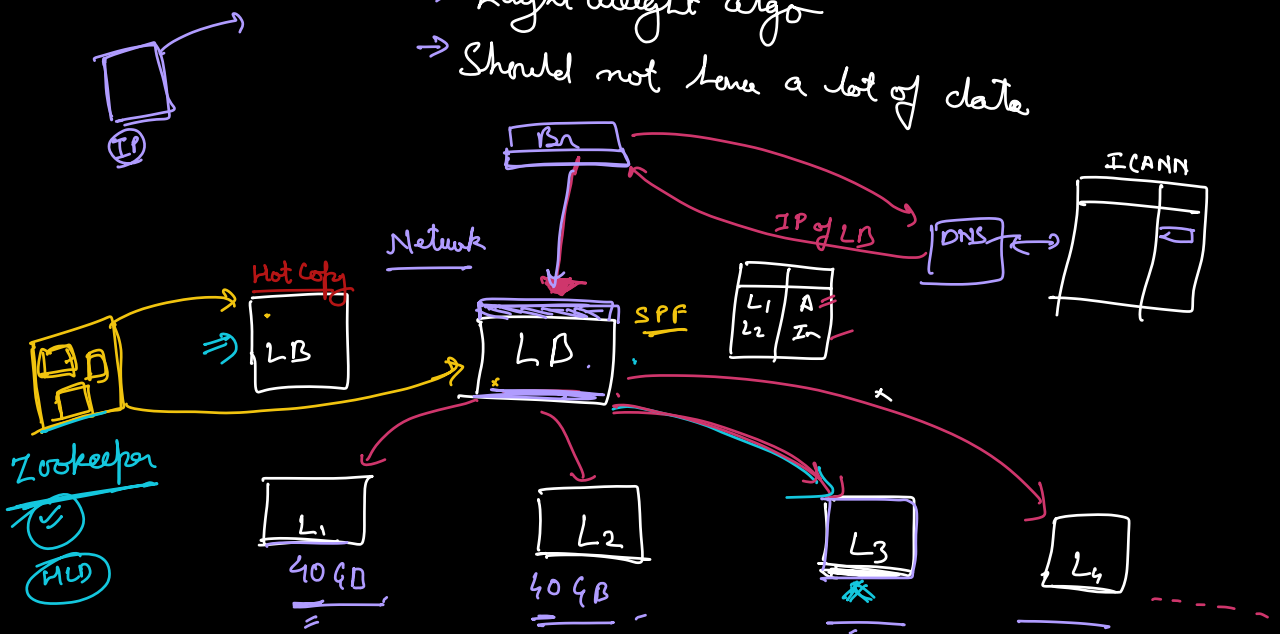
Horizontal Scaling





Distributing the load

- → Light weight algo
- → Should not have a lot of data



1. Health - Check
2. Heart - Beat

- → Round Robin
- → Least connection first
- → Least response time first

Data Partitioning / Sharding

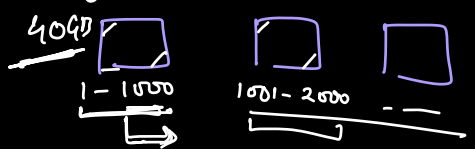
RAM → < 1 ms
HD → 10 - 100 ms
Network → > 1 s

To ans one query → Request as less servers as possible.

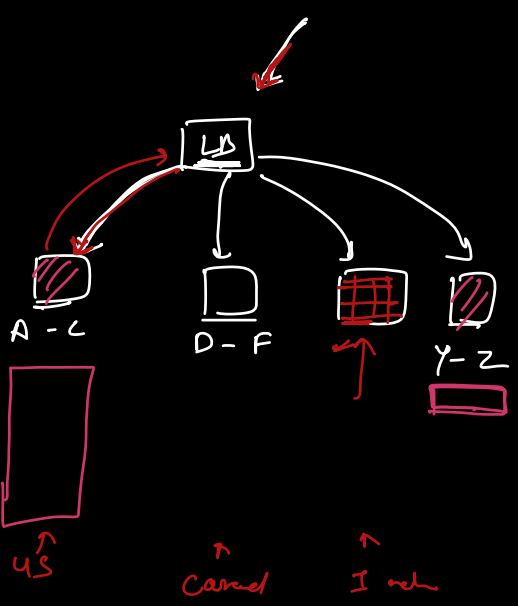
1. Register / log-in (UserId)
2. AddBookMark (UserId, URL)
3. Get All BookMark (UserId)

Sharding Key ⟹ UserId

1. Range based ✗



2. Alphabetically ✗



3. Geographically ✗
4. Hash function (mod) ✗
5. Round Robin ✗

Hash function

$f(x) = y$

$h(x) \Rightarrow y$

% $\Rightarrow$ Mod

$a \% b \Rightarrow$ Remainder of $a \div b$

$10 \% 4 \Rightarrow 2$

$11 \% 3 \Rightarrow 2$

$15 \% 5 \Rightarrow 0$

$N \% 4 \Rightarrow [0, 3]$

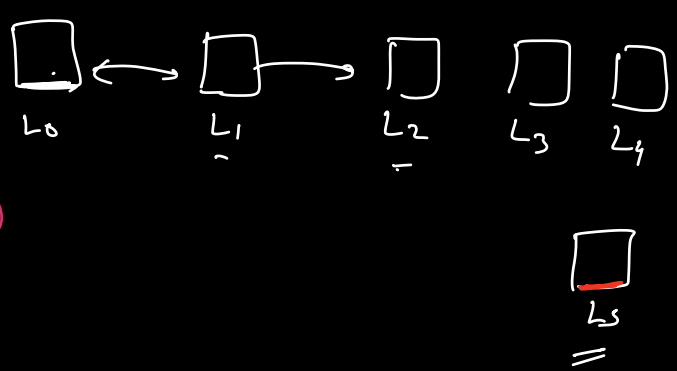
$\Uparrow$

Remainder

$A \% n \Rightarrow [0, n-1]$

UserId % 5 $\Rightarrow$ UserId % 6

UserId	N=5	N=6
12	L2	L0
13	L3	L1
15	L0	L3
18	L3	L0

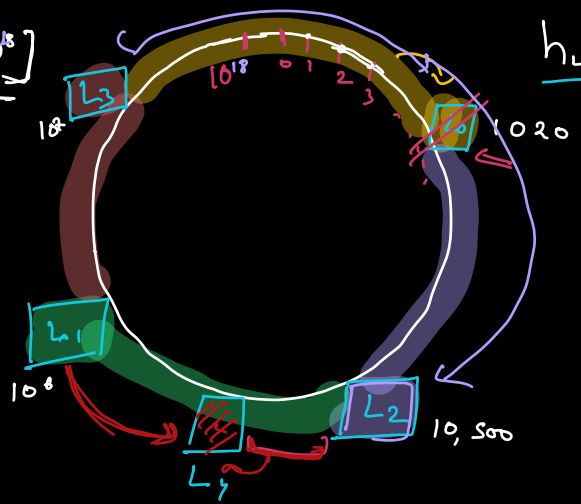


Consistent Hashing

$h(\text{userId}) \Rightarrow [0, 10^{18}]$

(Randomly)

$P(x) = 1/10^{18}$



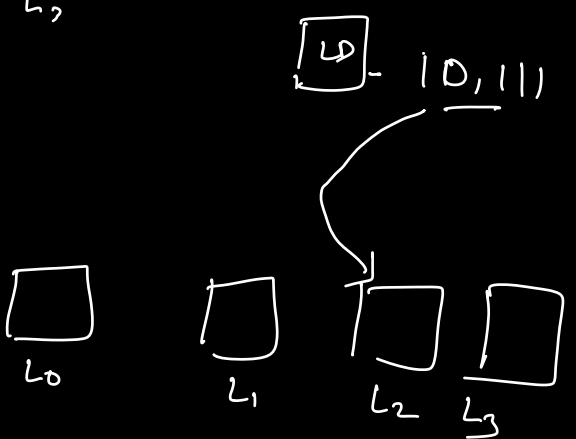
$h_L(L_{Id}) \Rightarrow [0, 10^{18}]$

$L_4 \rightarrow 15,000$

1020	10,500	10^6	10^7
L_0	L_2	L_1	L_3

$h(500) \Rightarrow 10,111$

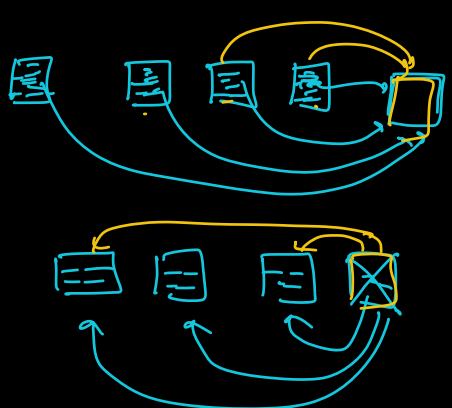
user 500



Ideally

⇒ After addition of new server, load of every server should be reduced.

⇒ When a server goes down it should equally distribute its load in all other servers.

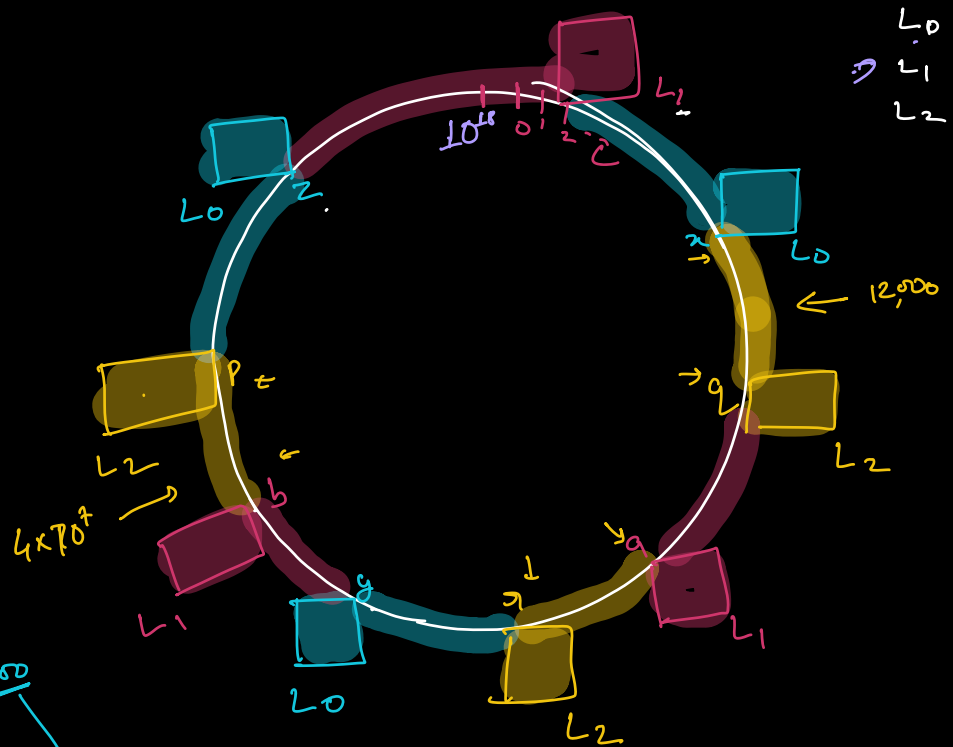


We need diff hash fns.

	$h_1()$	$h_2()$	$h_3()$
L_0	x	y	z
L_1	a	b	c
L_2	p	q	r

$h(L_0) \Rightarrow x$

$h(L_1) \Rightarrow$ ~~x~~



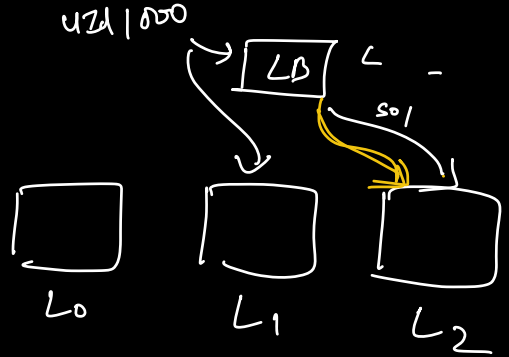
500

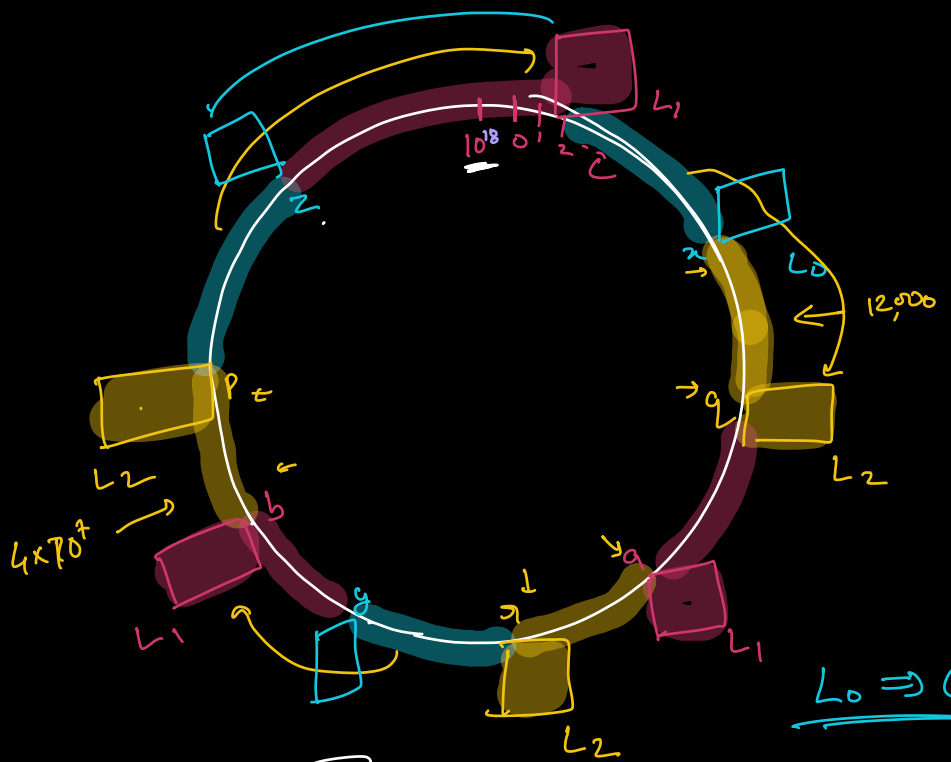
c	x	a	a	r	g	b	p	z
10	1000	50,000	10,00000	10^6	8×10^6	10^7	5×10^7	9×10^7
L_1	L_0	L_2	L_1	L_2	L_0	L_1	L_2	L_0

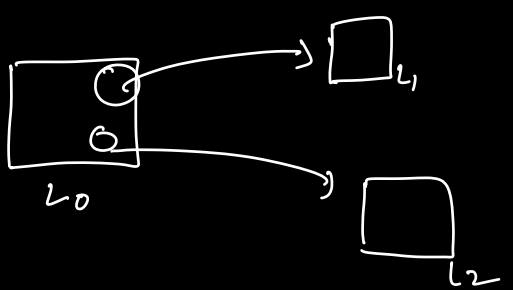
$h(501) \Rightarrow 12000$

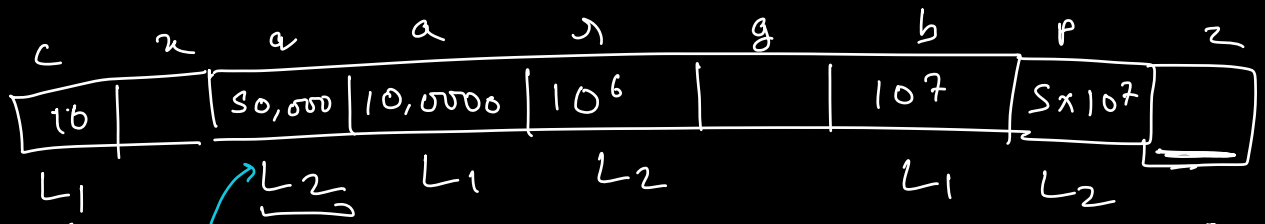
$h(1000) \Rightarrow 10^8$

$h(5) \Rightarrow 4 \times 10^7$









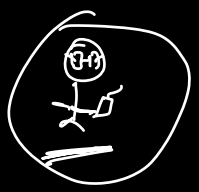
In case of collision, DON'T run the hash fn again.
Just leave it. One of the server will have one less copy.

$h(L_{id}) \Rightarrow [0 - 10^{18}]$ — Practically Impossible

$h(L_0) \Rightarrow 10,101$

$h(L_1)$

$P(h(L_1) \Rightarrow 10,101) = \frac{1}{10^{18}}$



$h(x) = x \% 10 \Rightarrow [0-9]$

$P(11) = 0$

$P(10) = 0$

$h(x) = (3x^2) \% 10$

$h(1) = 3 \% 10$

$h(-1) = 3 \% 10$