CN → HLD

Joshua

2005 - 2006

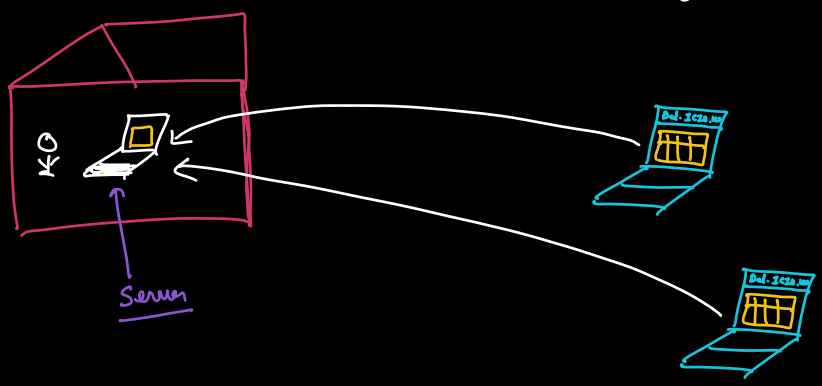
DEL.ICIO.US

Bookmark

→ Create Account

→ Save Bookmarks.

→ Get All Bookmarks

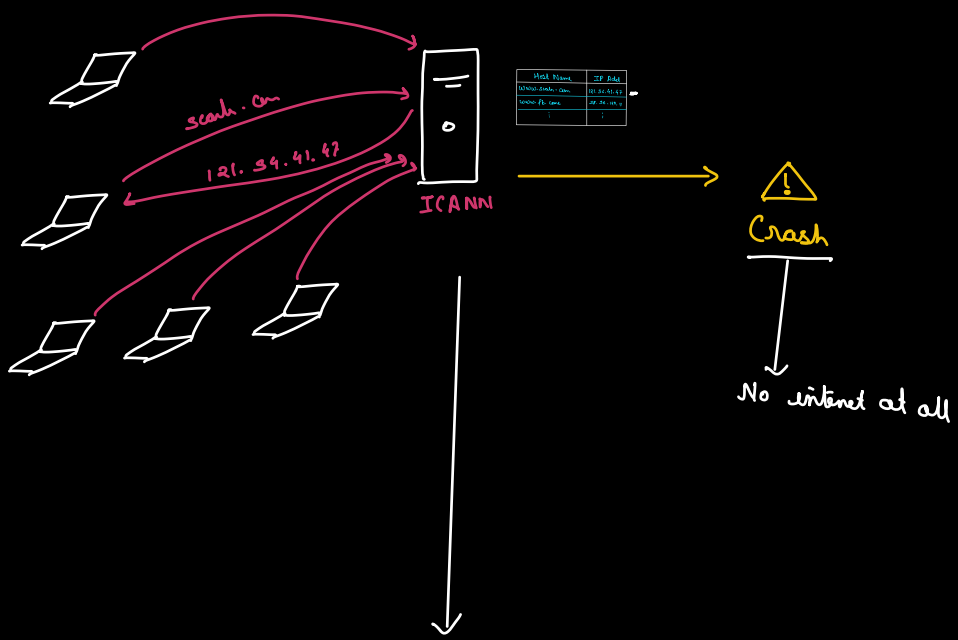


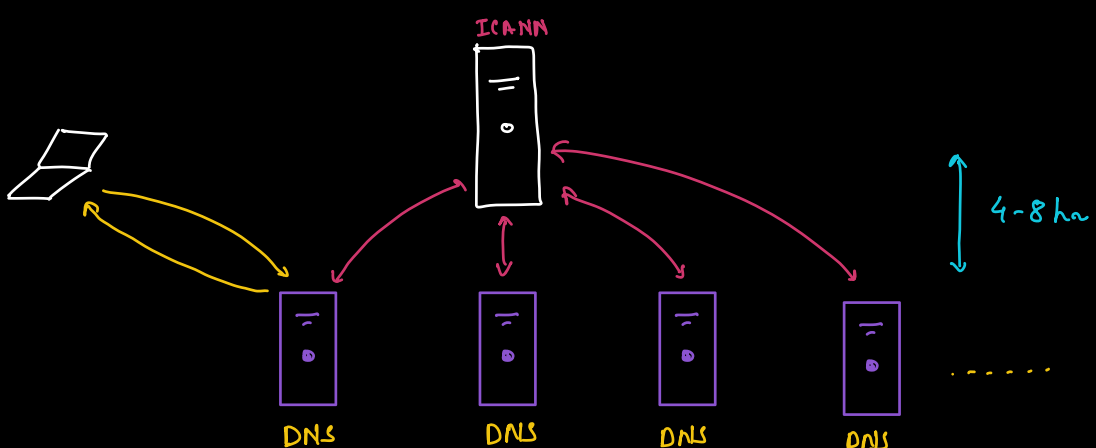
To be continued ....

Computer Network

Host Name	IP Add
www.scaler.com	121.56.41.47
www.fb.come	58.56.120.11
⋮	⋮

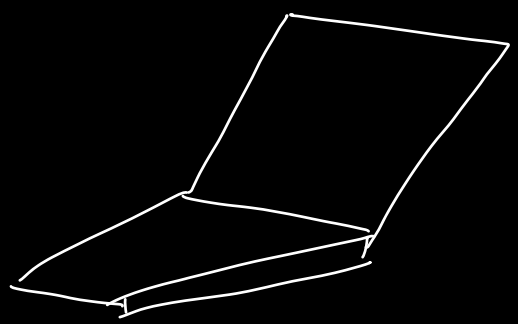
ICANN
(Single source of truth for the mapping)
(Non profit Org)





Who maintain DNS?

- ISP (Airtel, jio, tata docomo ...)
- Google & facebook



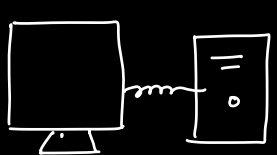
Airtel

IP

DNS IP

Joshua

(2005 - 2006)



RAM : 512 MB

Hard Disk : 40GB

Clock Speed : 1.6 GHz

User ID	URL
10	scaler.com
20201	google.com
10	Interwin Bit.com

1 KB

Code

Database

Other process

If

1 record takes 1 KB ⟶ 1000 Byte (approx)

& he gets 1 million requests per day
(actual bookmark)

Memory required to save records created per day

1 KB × 1 million

≈ 1000 B × 10^6

≈ 10^9 B

≈ 1 GB

If 1 GB records getting stored per day

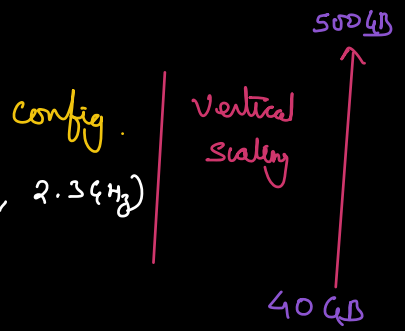
40 GB ⟶ ≤ 40 days

Scaling Challenge

Solutions

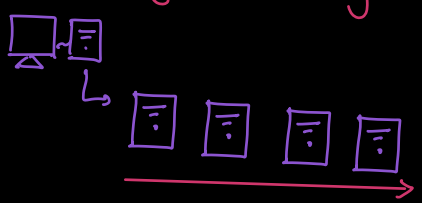
① Buy a better system with higher config.
(500GB, 4GB, 2.3GHz)

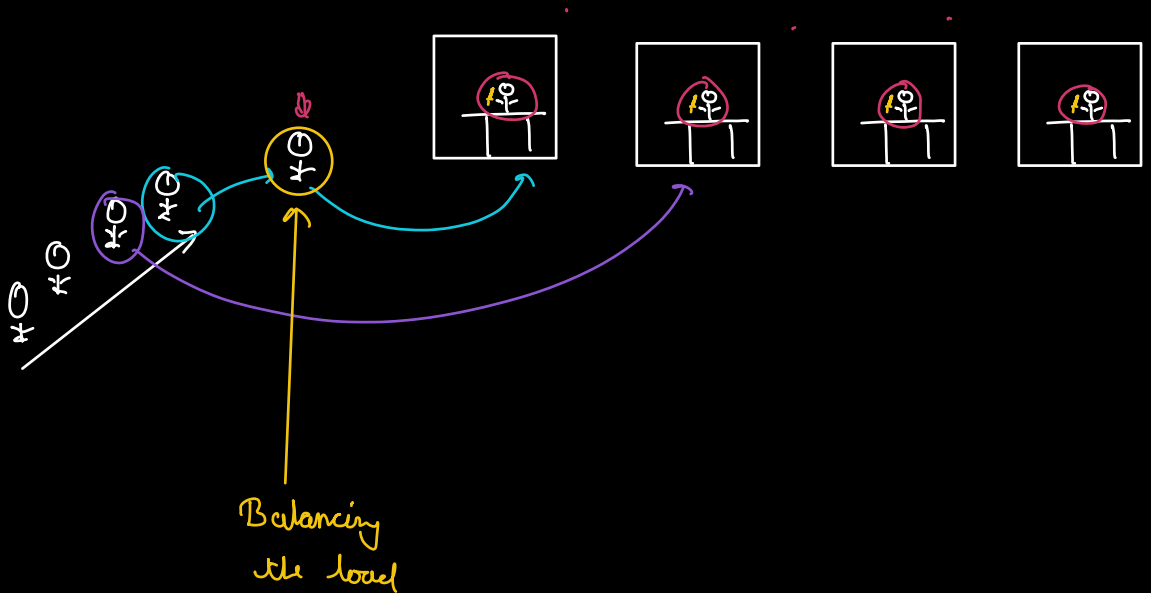
Vertical scaling



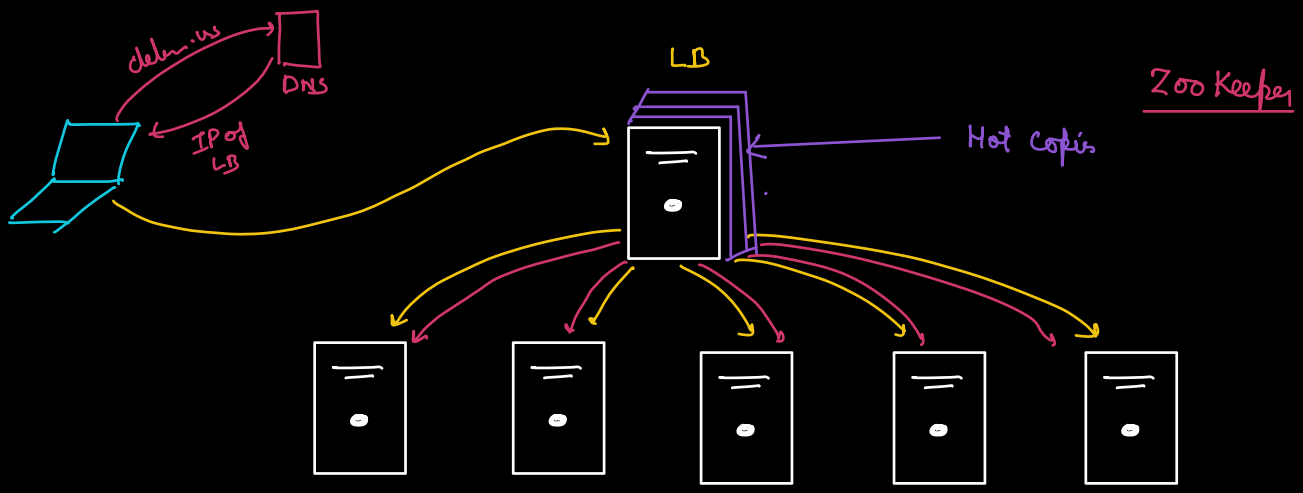
② Buy more systems of similar config

Horizontal scaling

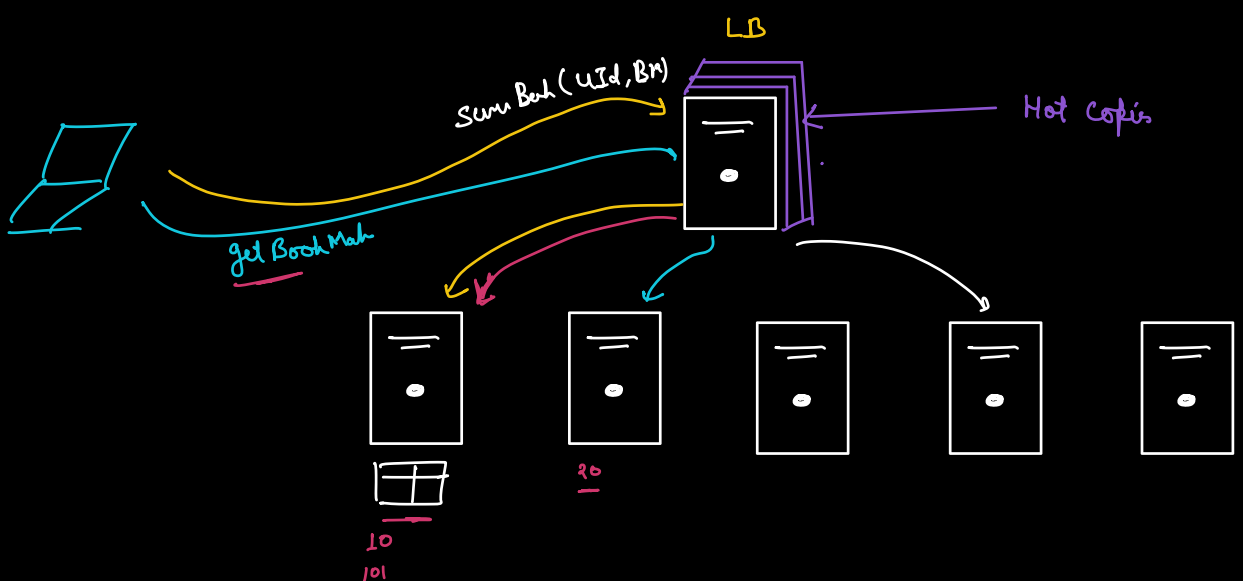




Balancing the load



LB can use
- Round Robin ✗
- Least connection first ✗
- Least Response time first ✗



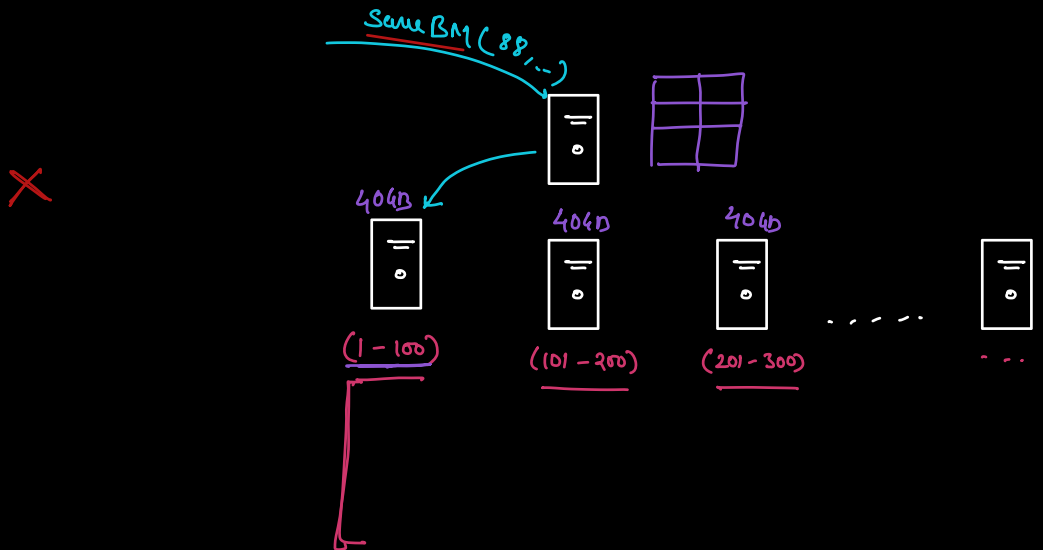
Data of a specific person should be stored in a single server.

Data distribution Strategies (Sharding)

- Ranges (on UserId)
- Region / Geography
- Hash function / Modulus

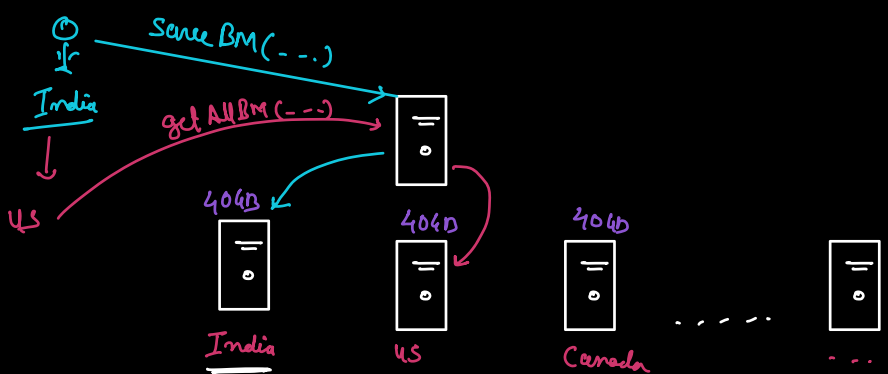
LB has to be
- light weight
- should not store a lot of data

Ranges



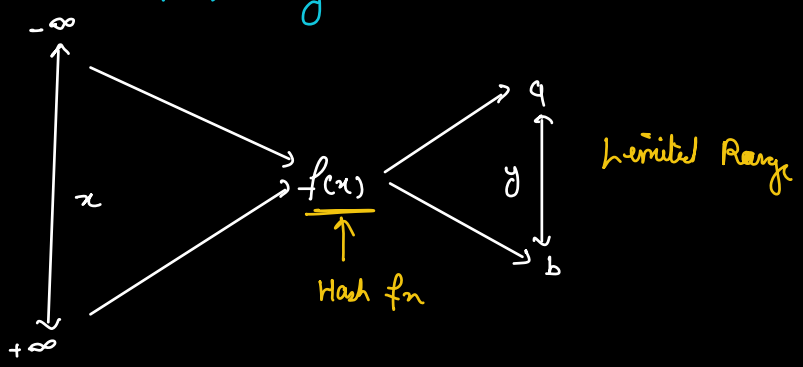
Region

- Uneven distribution
- Consistency



Hash Function

$f(x) = y$



% ⇒ Modular

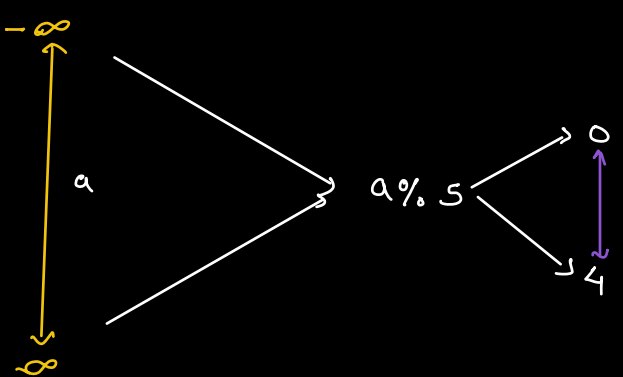
a % b → The remainds of a ÷ b

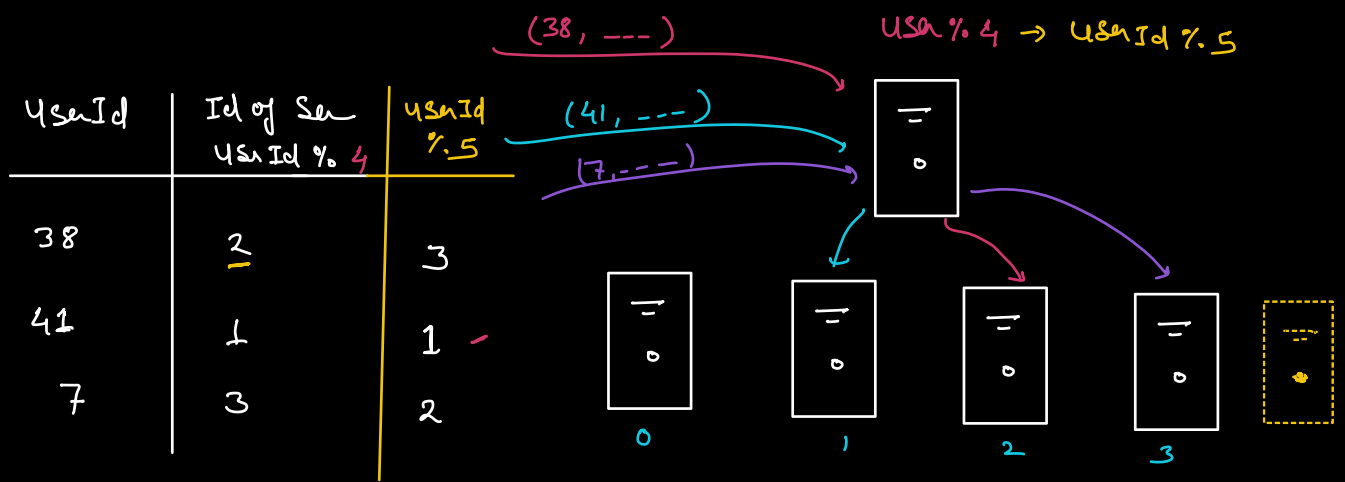
$f(a, b) \Rightarrow a \% b$ → 0 to b-1

$f(112, 10) \longrightarrow 112 \% 10 \Rightarrow 2$

$f(105, 10) \longrightarrow 105 \% 10 \Rightarrow 5$

$f(20, 10) \longrightarrow 20 \% 10 \Rightarrow 0$





UserId	Id of Ser UserId % 4	UserId % 5
38	2	3
41	1	1
7	3	2

Inconsistency

Rechristribute the complete data to maintain consistency.

Break till 10.55 pm

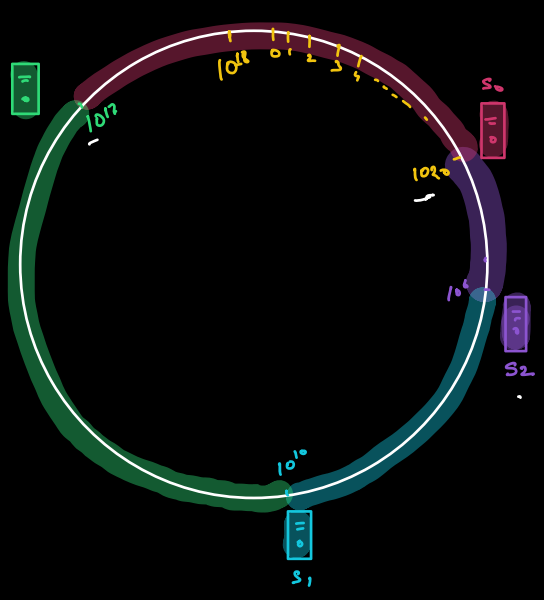
Consistent Hashing

$h_4(UID) = [0, 10^{18}]$

$h_4(500) = 10,111$

$h_4(101) = 10^{18}$

$h_4(512) = 1000000007$



Hash fn

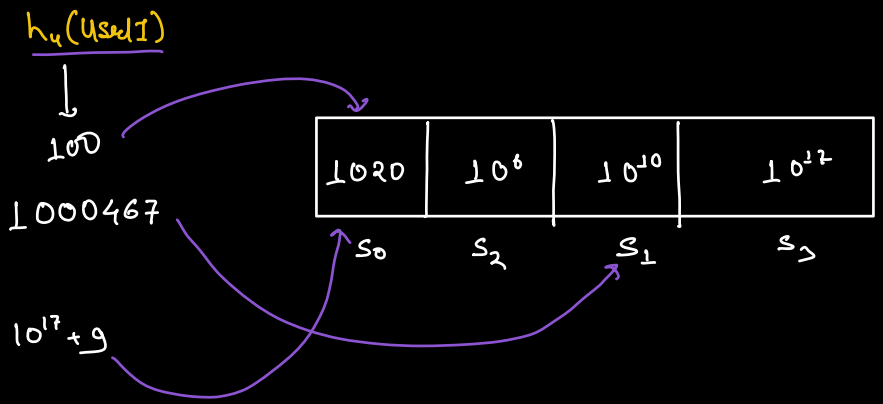
$h_s(S_{ID}) \Rightarrow [0, 10^{18}]$

$h_s(S_0) = 1020$

$h_s(S_1) = 10^{10}$

$h_s(S_2) = 10^{6}$

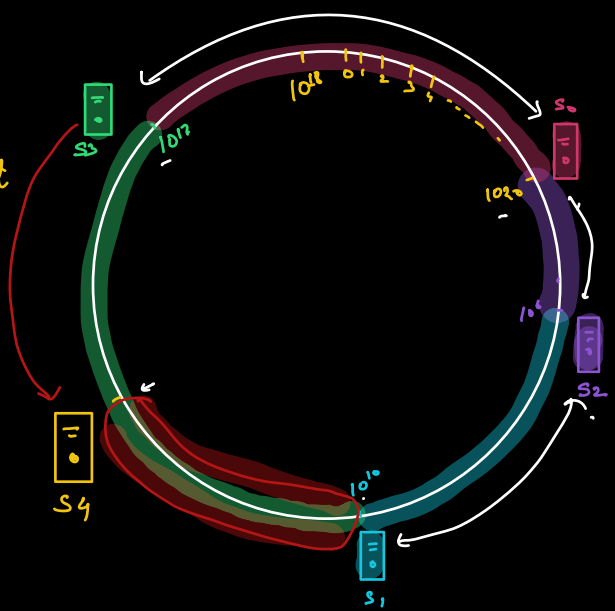
$h_s(S_3) = 10^{17}$



Add a new Server

Only data movement

$S_3 \rightarrow S_4$



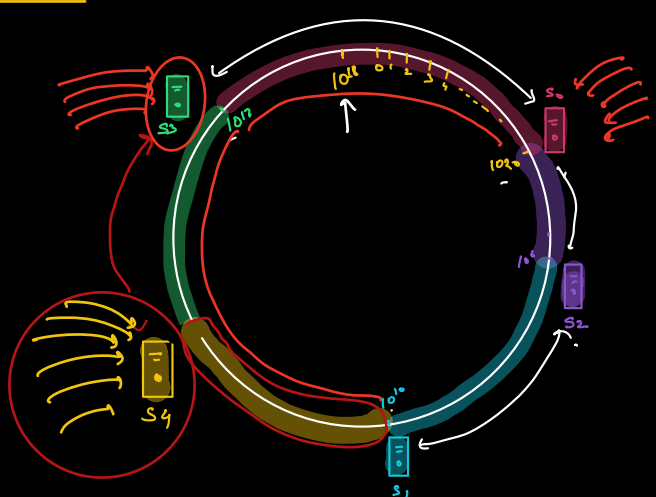
If a server goes down

$S_4 \rightarrow S_3$

$S_3 \rightarrow S_0$

$S_0 \rightarrow S_2$

$S_2 \rightarrow S_1$

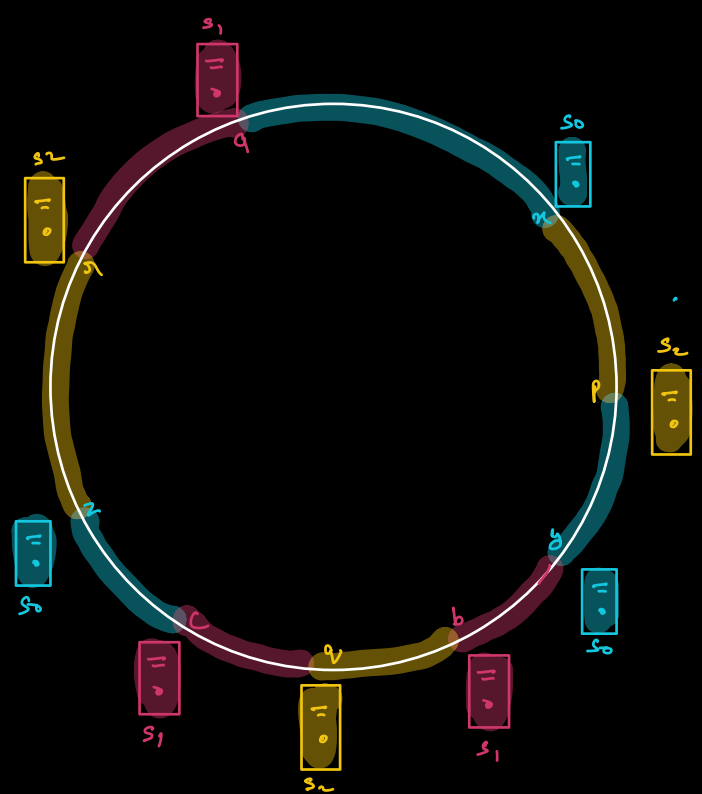


Whole System fails

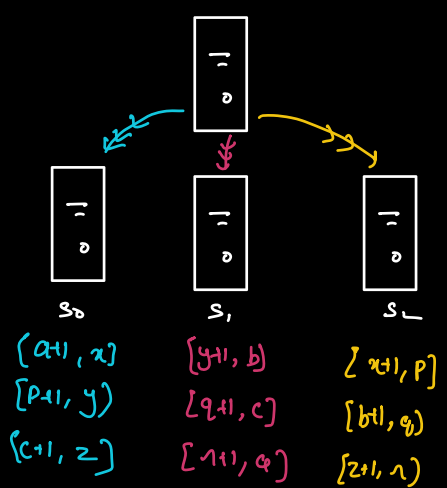
Ideally

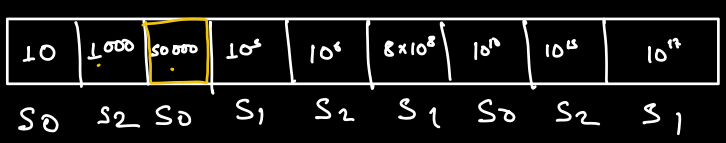
① After adding a server => Reduce load of all servers equally

② When a server goes down => the load should get distributed amongst all servers equally.



	$h_1()$	$h_2()$	$h_3()$
S_0	x	y	z
S_1	q	b	c
S_2	p	q	r





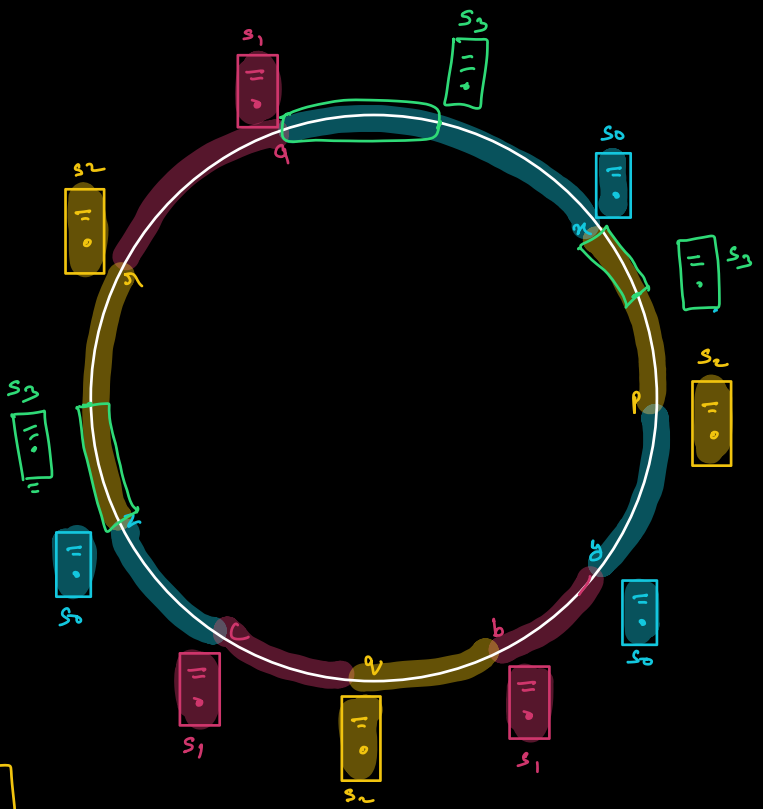
User Id	h_u (User Id)	Server
500	12000	S_0
100	10^8	S_1
5	10^{11}	S_1
200	10^{12}	S_0

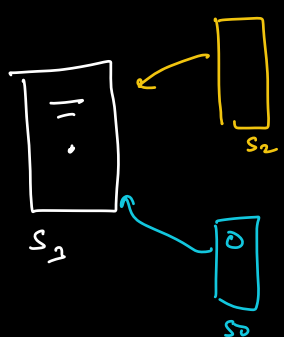
Add a new Server

$S_0 \rightarrow S_3$

$S_2 \rightarrow S_3$

$S_2 \rightarrow S_3$

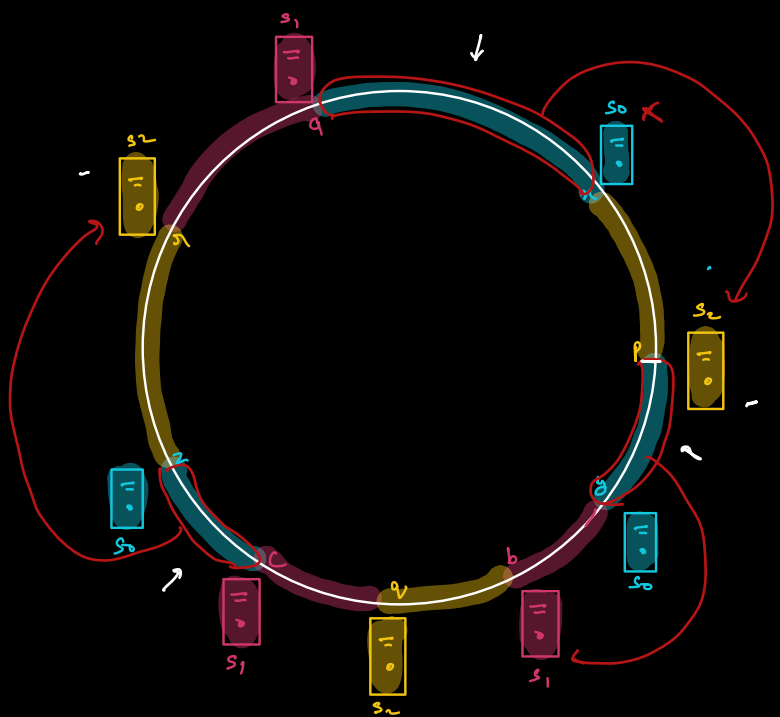




If a server goes down

$S_0 \rightarrow$ down

$S_0 \rightarrow S_2$
$S_0 \rightarrow S_2$
$S_0 \rightarrow S_1$



$h_1(S_0) = h_1(S_1)$

$h_1(S_1) = h_2(S_0)$

$\Rightarrow h_1(S_0) = 100$
$h_1(S_1) = 100$

$h_i(S_i) \rightarrow [0, 10^{18}]$

Probability of getting same value for 2 servers.

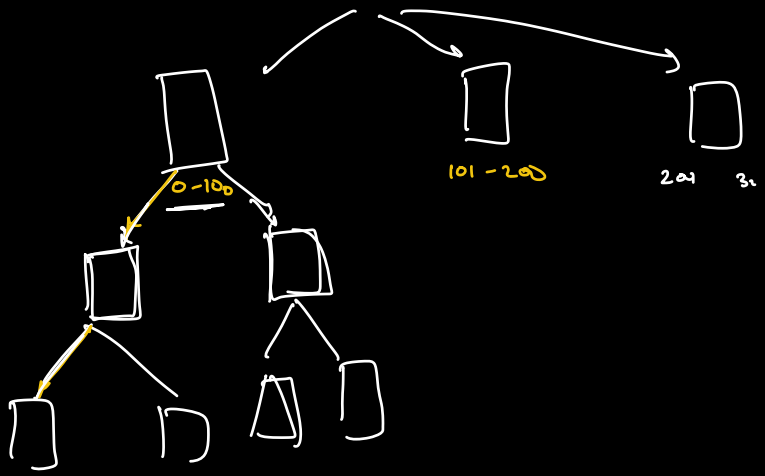
$= \frac{1}{10^{18}}$

0.000000000000000001

Del. icio. us $\rightarrow$ Yahoo

$h_1(S_0) = h_1(S_1)$

$\frac{1}{10^{18}} \times \frac{1}{10^{18}} \Rightarrow \frac{1}{10^{36}} \rightarrow 0$


0-100
101 - 200