Agenda

Today

1. Vertical v/s Horizontal Scaling
2. Load Balancers
   - ↳ Journey
3. Intro to Sharding
   - ↳ Hashing
   - ↳ Range Based

Next Class

- ↳ Cons Hashing

If not attended prev class

"System Design and CN 101"

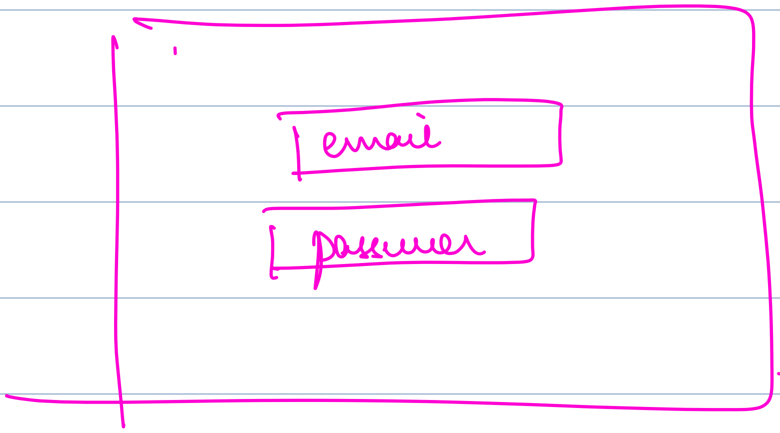
- ↳ leave this class
- ↳ watch recording.

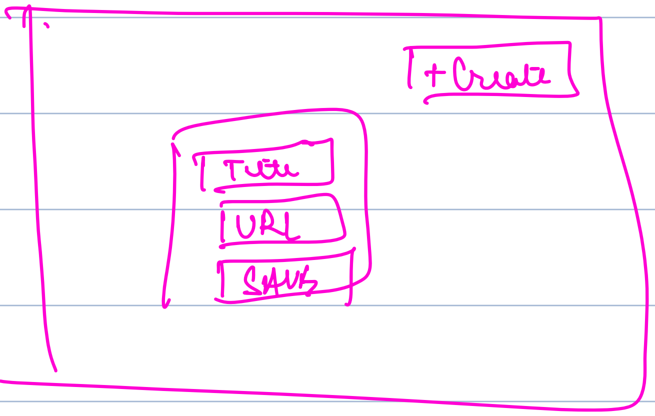
① Patient

Recap of last Class

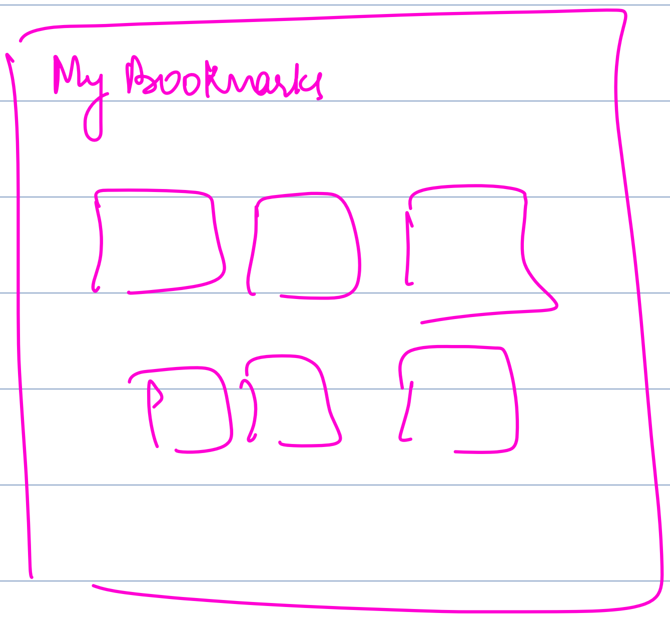
Case Study of Del.icio.us

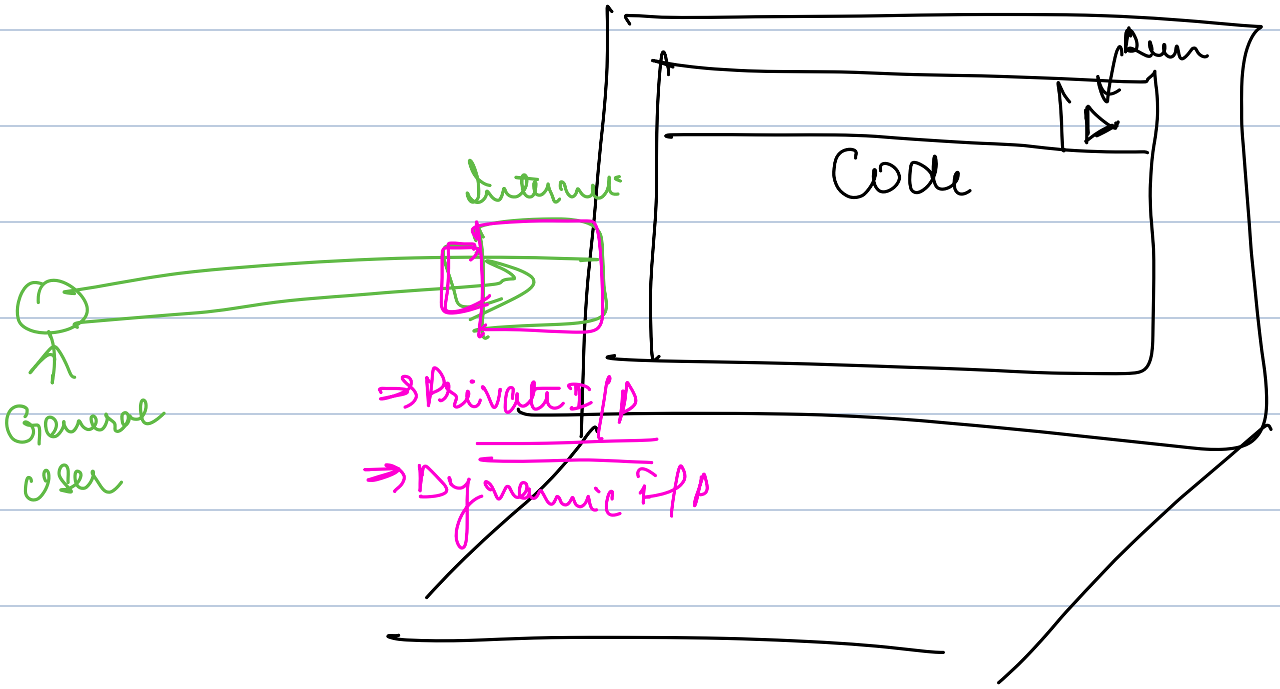
(2003)





→ College Student

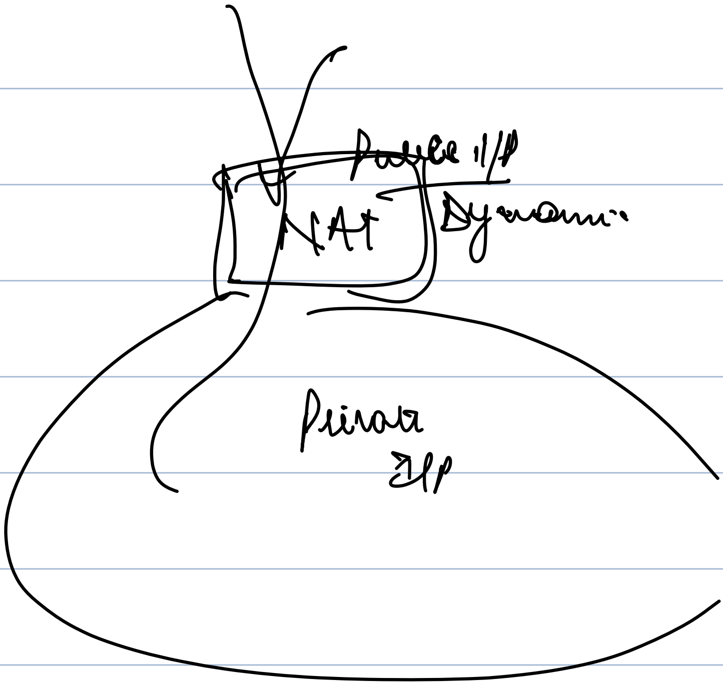


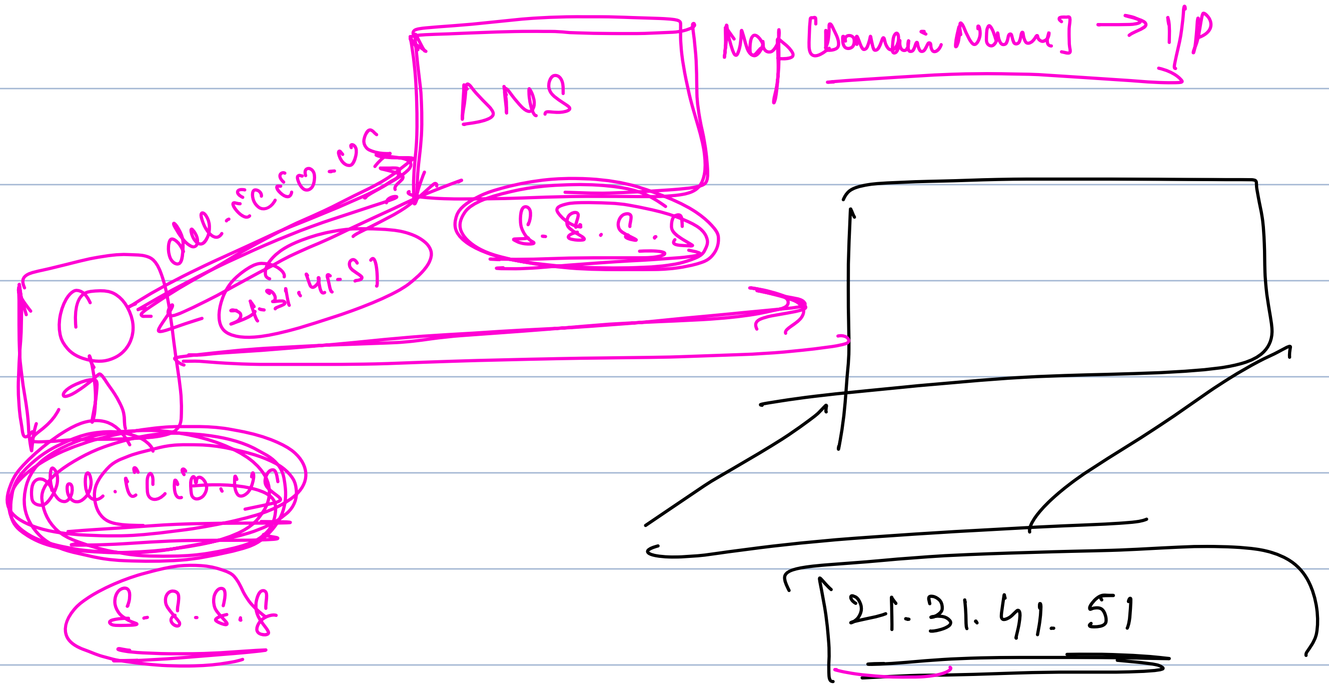


→ Private Dynamic I/P

→ Public Static I/P (to host website)

Pay extra I/P





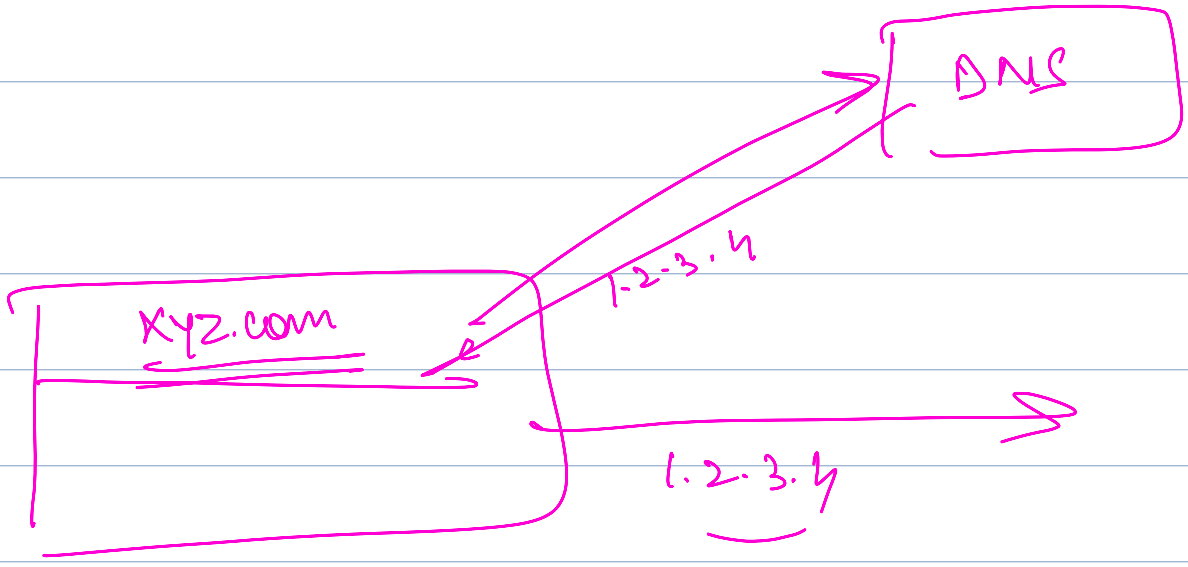
Map [Domain Name] → I/P

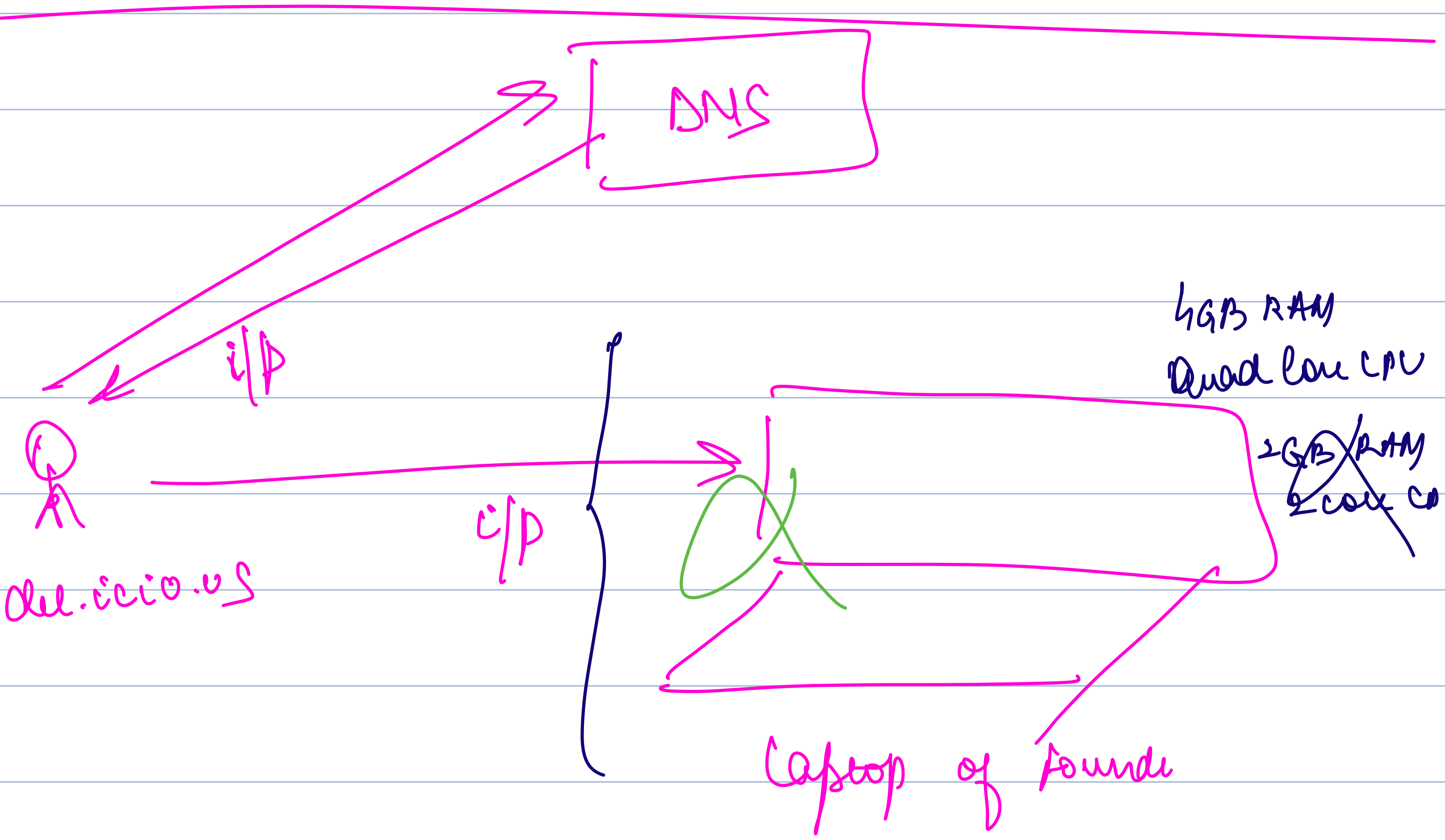
People like to type Names

21.31.41.51

domain name : google.com
fb.com

1. Bought a domain name (godaddy.com)
2. Got a public static ip connection
3. Wrote code
4. DNS updated the i/p of del.icio.us to the public static i/p





⇒ As website becomes more popular the laptop won't be able to handle all load / might crash.

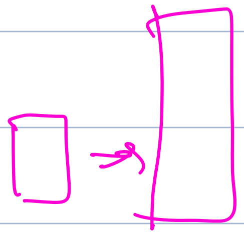
Scaling : ability of a website to handle more users

2 ways

↓

Vertical Scaling

Still same amount of hardware But better hardware



Pro

1.) Simpler

Cons

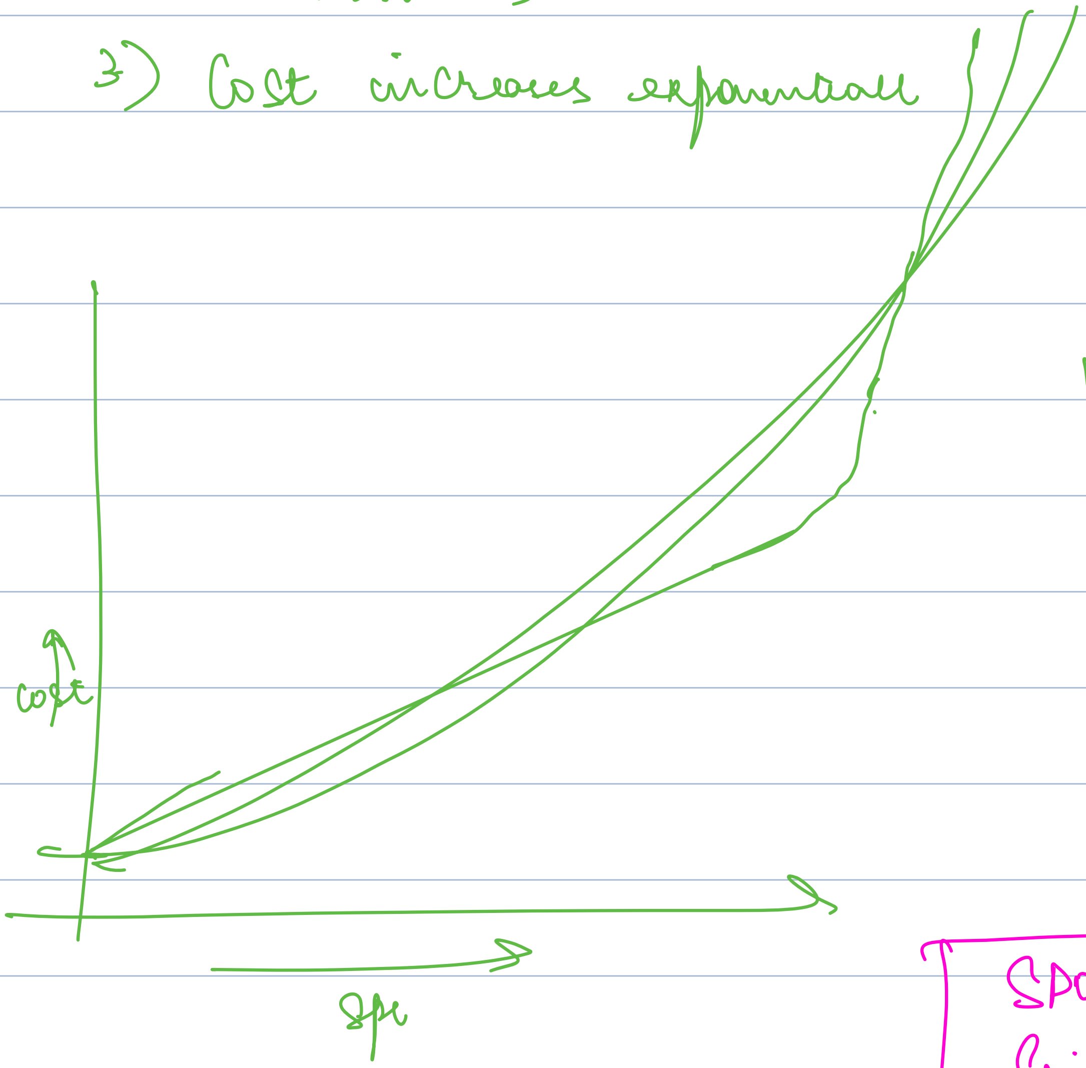
1.) SPOF

2.) Not infinitely ?

Scalable }

(Can't scale above a threshold)

3) Cost increases exponentiall



64GB X

128GB - 5/

SPOF

Single Point of Failure

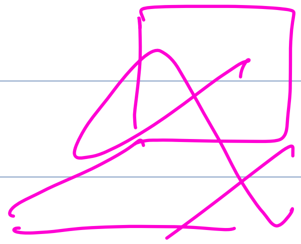
② Horizontal Scaling

⇒ By getting more laptops, instead of only single

Pro :

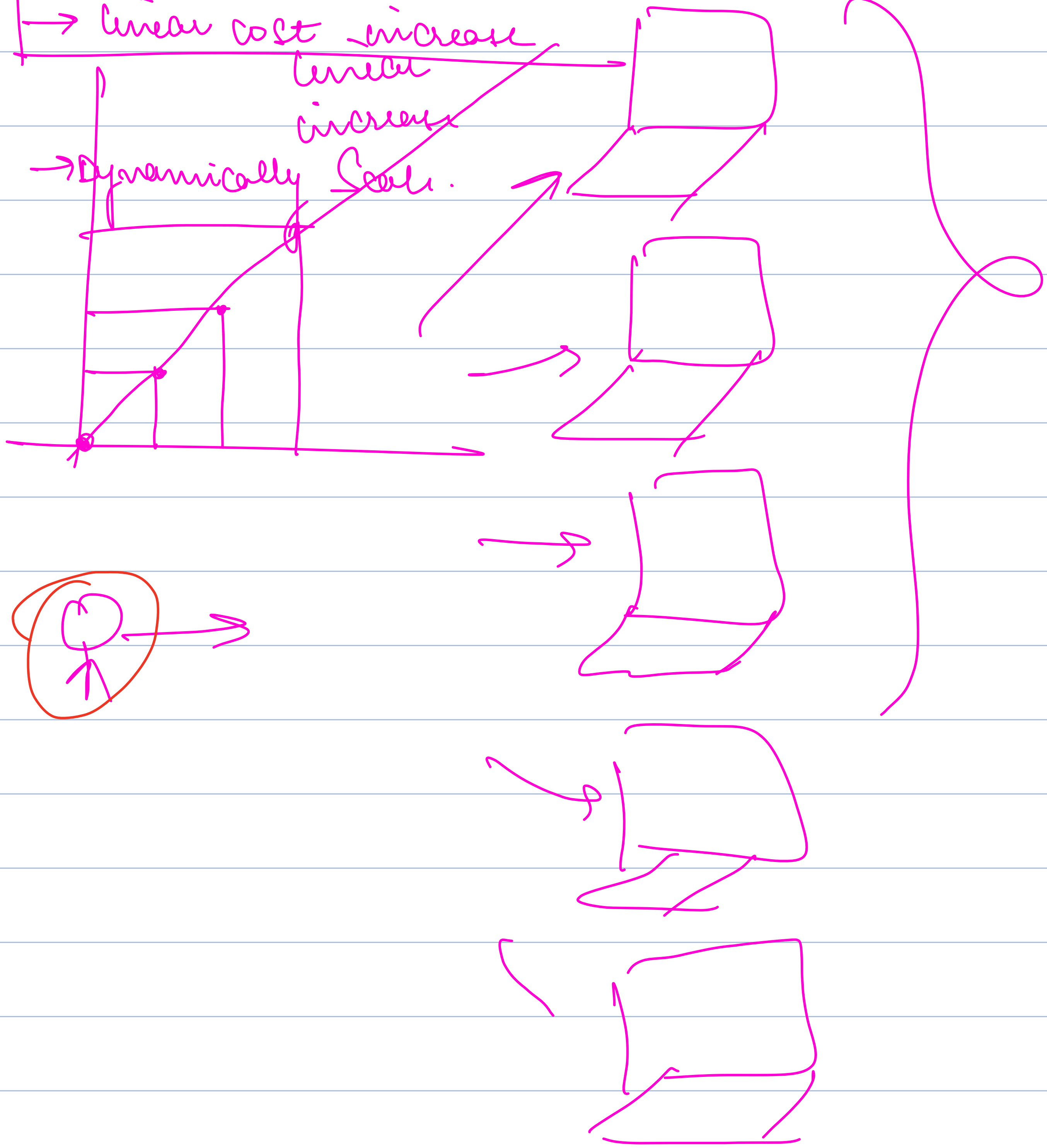
→ Infinitely Scalable.

→ No SPOF



→ Linear cost increase

→ Dynamically Scale.

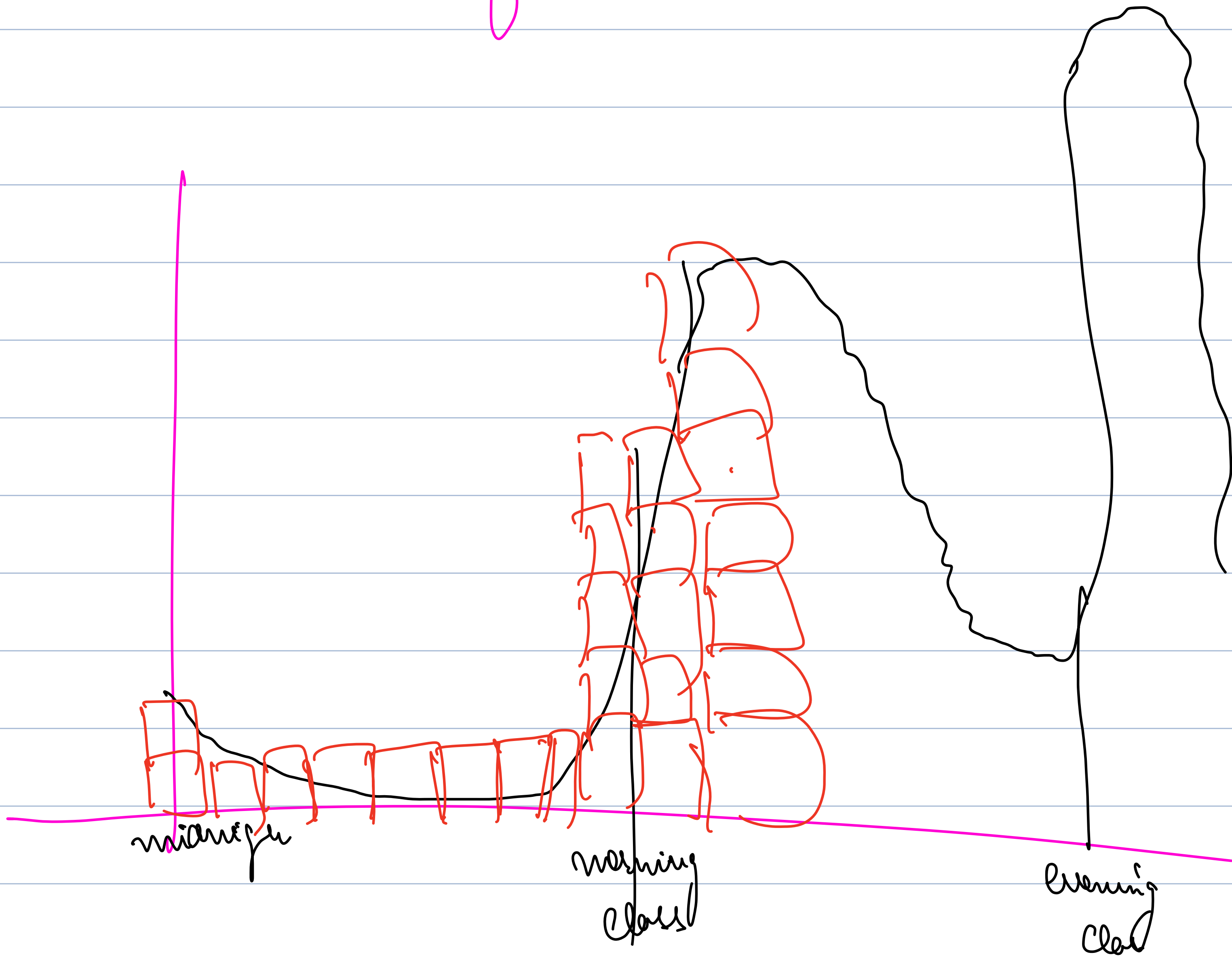


Cons

1.) Complexity of management

↳ monitor multiple machines

↳ how to decide where to forward a req.

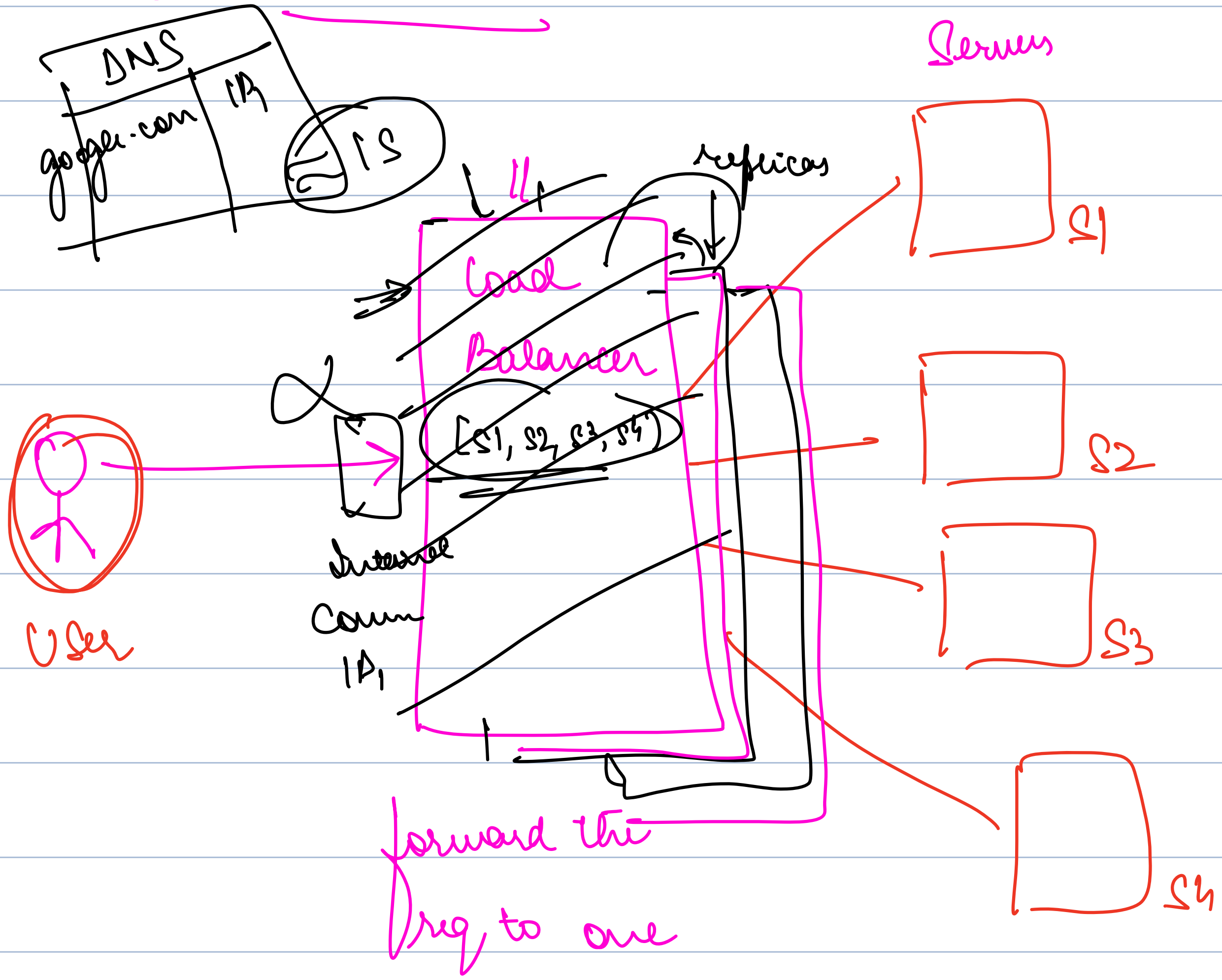


KISS

Keep it simple, stupi'

How does horizontal scaling work in reality

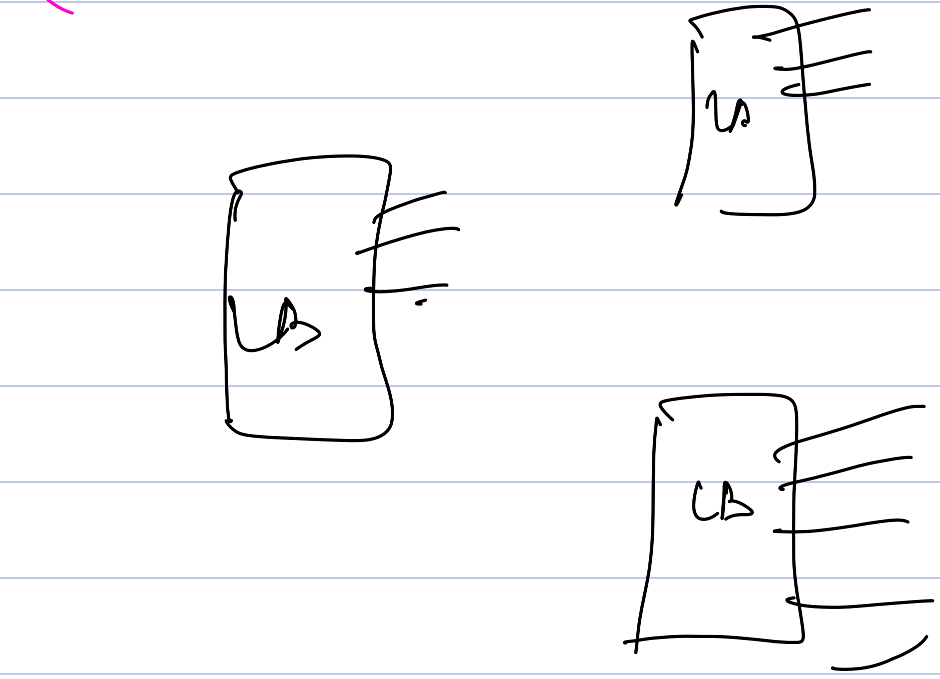
Load Balancer

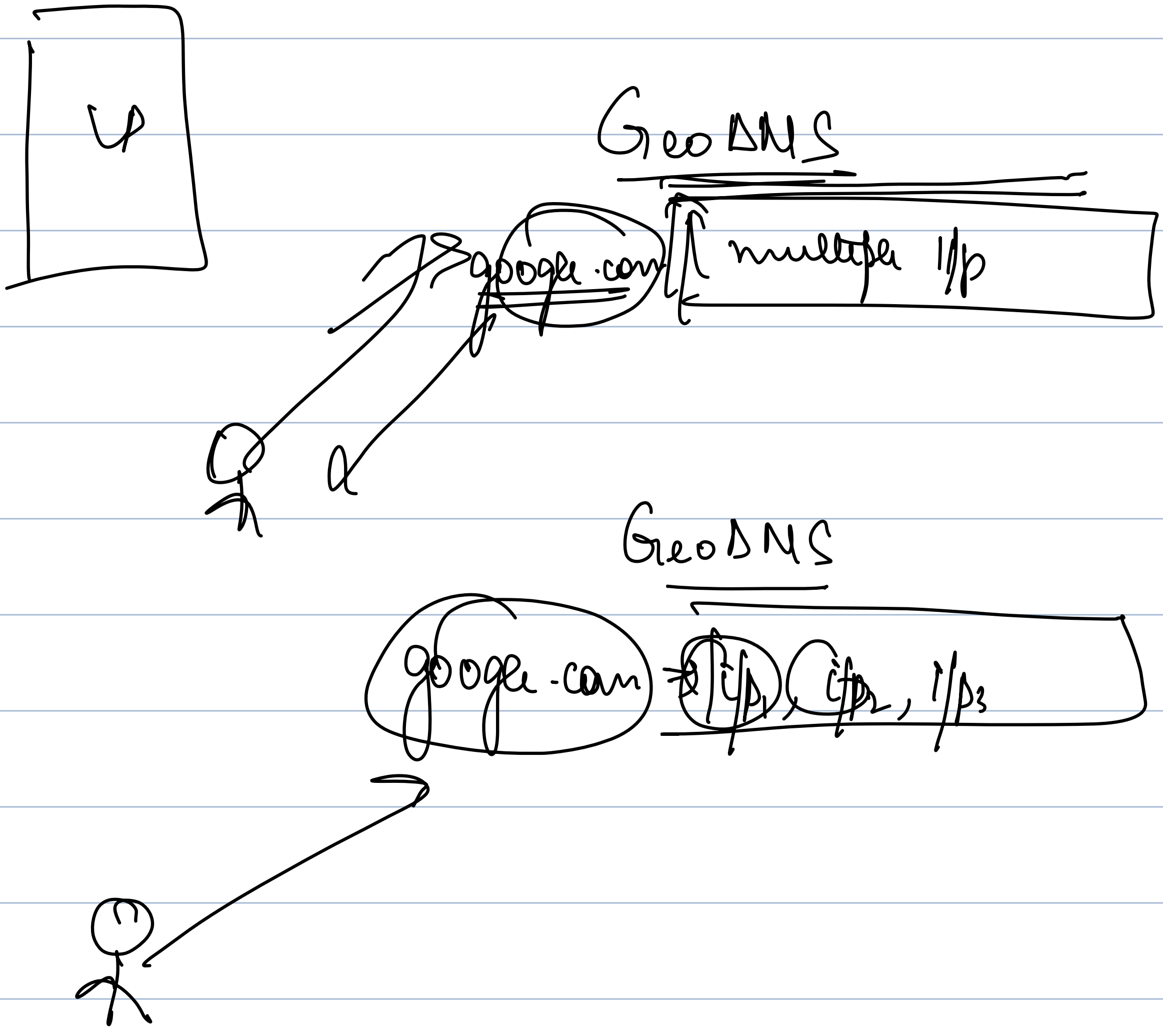


forward the req to one of servers BTS

LB: :distribute the req amongst servers in a uniform way

(So each server is almost equally loaded)

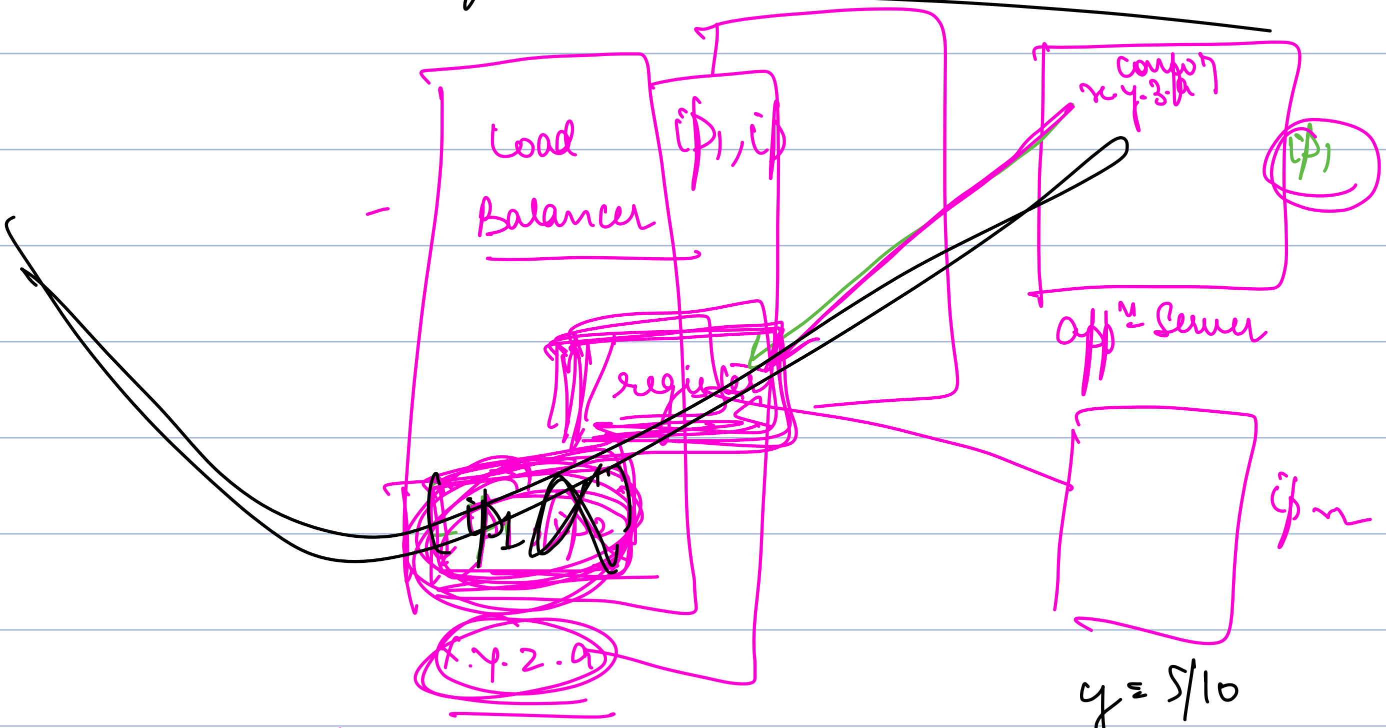




How will LB work

- → How will LB get to know servers
- → How will LB know of server dead
- → How will LB decide where to forward req.

How will LB get to know a server



y = 5/10

y = 2.4

x.y.z.a / register

How will LB know that app^n server has died

1.) Healthcheck

2.) Heart Beat ~~deregister~~

Normal Days

Parents care of your health

They calling you

Health check: every "x" sec LB will send a req to every server asking if it's okay

(i) if Not okay → deregister

(ii) if server doesn't reply for 3 times

→ assume it has died

→ remove it

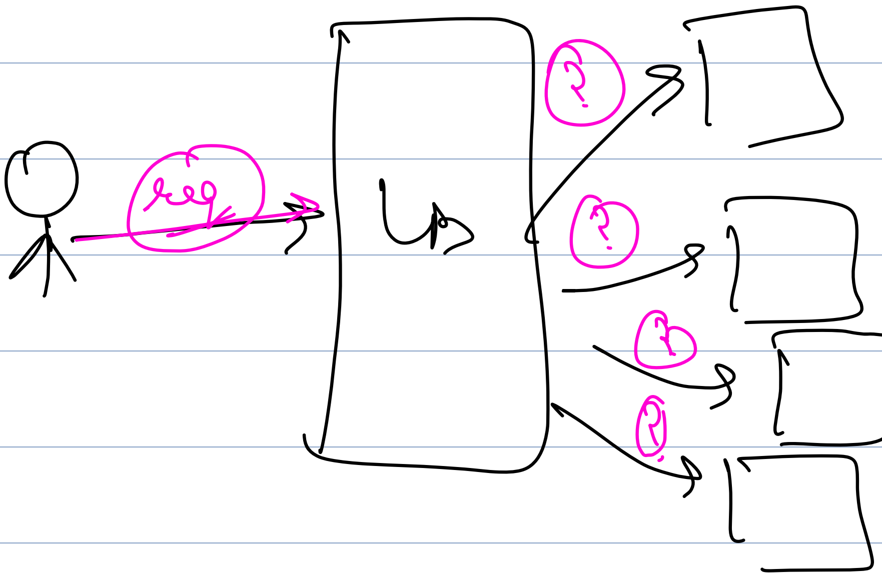
/health

Heart Beat :

every server has to send an "ok" to LB every "x" sec (10 sec) 3 sec

if LB doesnt get consecutive ok, it will assume dead

How will LB forward Req



Virus from 3 IDIOTS "as expected"

approach 1

~~LB asks each server about their current CPU/RAM usage forwards to least one.~~

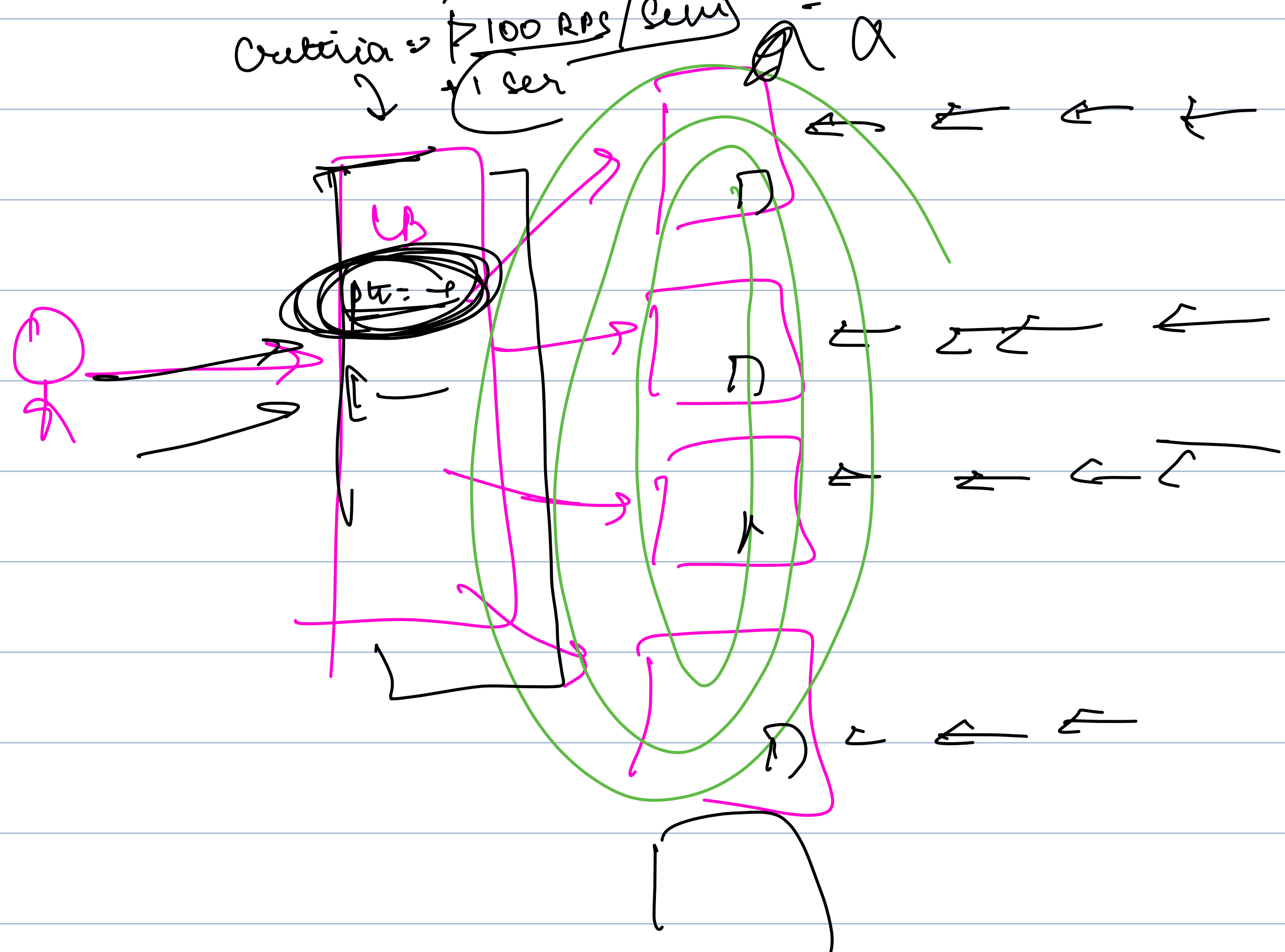
→ Terrible.

→ Make the req slow

approach 2: Round Robin

(principle of probability)

Criteria ≥ 100 RPS/server
1 ser

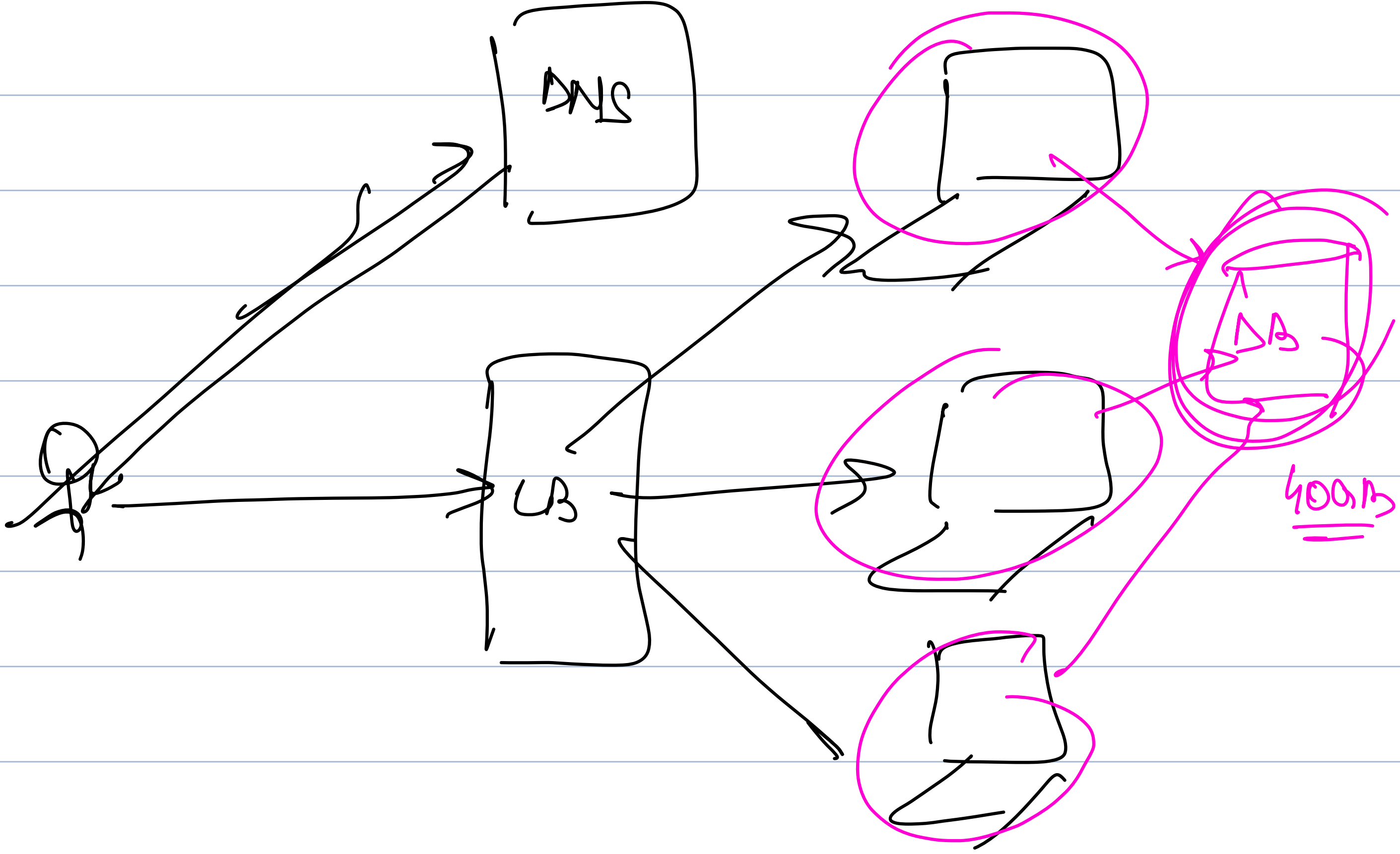


Birth	→	Registering
Death	→	Health Check
		HeartBeat
Forward	→	RR

RPS (Requests Per Sec)

av. CPU usage

av RAM usage



2003

HDD => 40 GB

2004

72GB HDD

400M users

every user ≈ 20 BM

=> Total BM: $400 \times 20 \times 10^6$

→ 8×10^9 → 8B BM

bookmarks

id	url	title	user_id
8B	50 B	50B	8B

← 100B

⇒ Total Size

→ $8 \times 10^9 \times 100$ Bytes

→ 800×10^9 Bytes

⇒ 800 GB

⇒ Unfortunately 1 machine can't have all data

⇒ Split Data

⇒ Sharding