Agenda (HLD Design)

→ Requirements
  - Functional
  - Non Functional

→ Facebook
→ Instagram
→ Twitter
→ WhatsApp
→ Tinder / Uber / Swiggy

→ System Architecture

→ Deep Dive
  - → How a post is going to be made
  - → How newsfeed is generated
  - → How to see nearby friends

Naman Bedla

60% ⇒ First MC

40% ⇒ Old people

System Design of a Social Media App^n

UI / UX Design

66% ⇒ Yes 34% ⇒ No

Plan

Software System

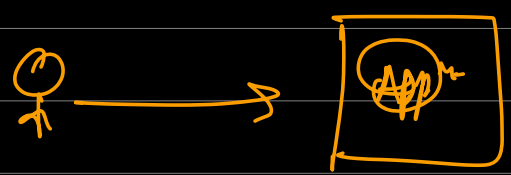
⇒ Requirement Gathering

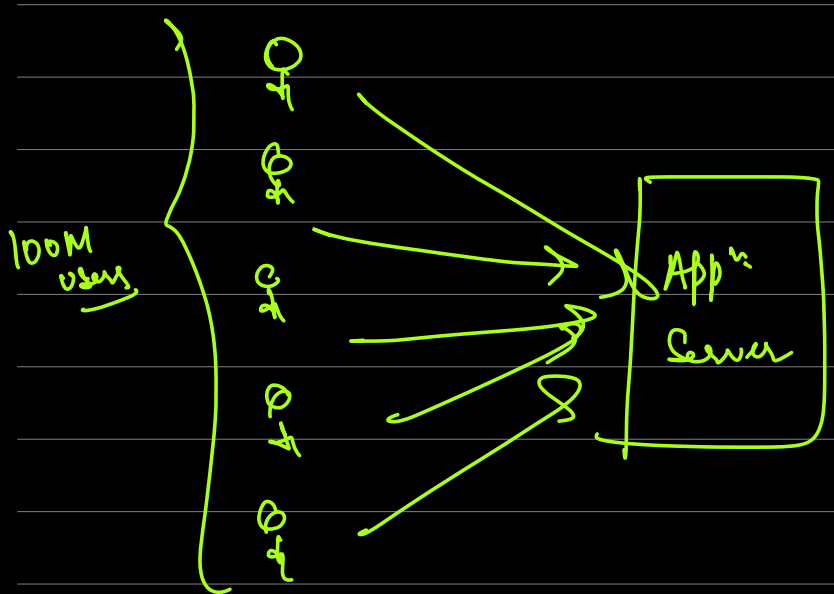
⇒ HLD of the app^n

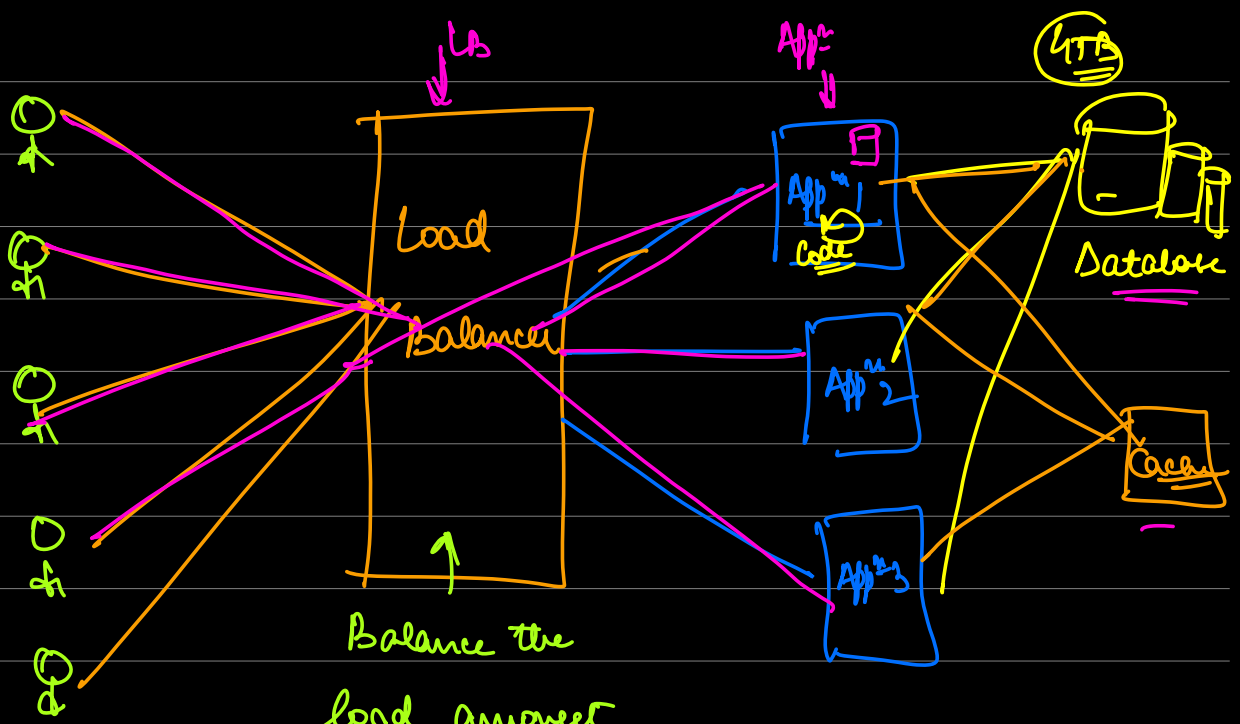
⇒ LLD of the app^n

⇒ Code

System Design

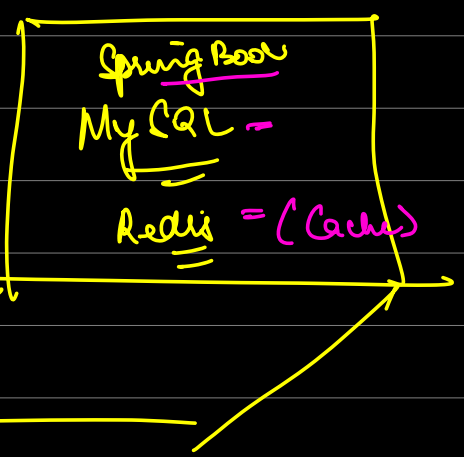






High level Design: Design of how different infra layers are going to work together to serve an application at desired efficiency

How your system behaves is decided by software that lives on machine



→ LLD: How to write good S/W

LLD ⇒ How to write good software

HLD ⇒ How to write scalable software

HLD of a social media app

→ Readable
→ Understandable
→ Modular
→ Reusable

How to approach HLD Problems in an interview

HLD Interview → SDE 2+ Top Startup (2+ yrs)

≥ 2 yrs

2+ yrs MNC (FB L4) ⇒ 1 HLD

5+ yrs ⇒ Google L5 ⇒ 2 HLD

Steps to approach HLD Problems

⇒ Understand the problem statement well (DON'T MAKE ANY ASSUMPTIONS)

ASS + U + MR ⇒ ask clarifying questions

① Functional Requirements ⇒ Features that need to be supported

② Non functional requirements ⇒ Qualities / Behaviours

FR ⇒ Ability to send friend requests. (Feature)

NFR ⇒ Latency should be less (Quality)

II Back of Envelope calculation → Guestimates

Metrics around resource usage

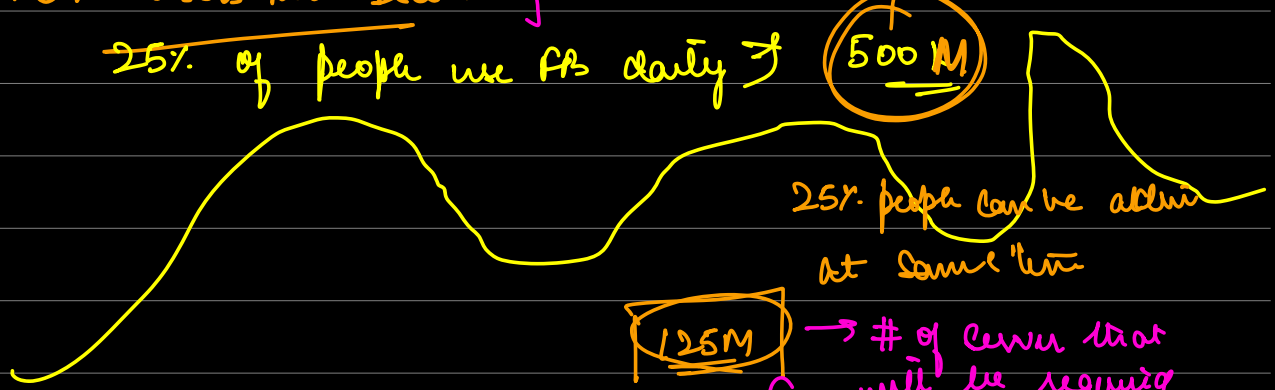
- → QPS (Queries per second)
- → Amount of storage used
- → Network Bandwidth

2B users

FB has 2B monthly users

Peak Users Per Second

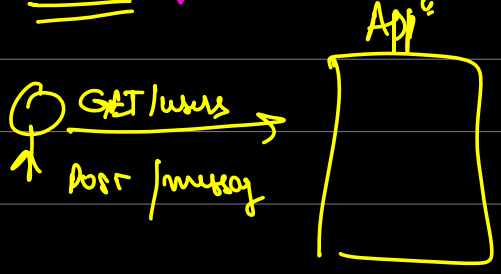
~~25%~~ of people use FB daily → 500M



→ # of servers that will be required

→ Amount of storage that will be req

III API Design ⇒ How will a client talk to my app?



IV Architecture ↻ Iterative

V Database Design

VI Deep Dive : When we go into details of architecture. How a particular sub-part of design will work

45 - 50 mins

DESIGN A SOCIAL MEDIA APPLICATION

↳ Facebook

Gathering Requirements

Functional

~~X → Ability to login and logout~~

→ Newsfeed

→ Chat

→ Like / Comment / Share

Always lead the design interview

→ Ability to make posts
↳ text/media
→ Filter
→ Search
→ User Profile
→ Follow Users
→ Block / Mute
→ Nearby Friends → Tinder, Uber, Swiggy, Chat
(Location Based Appⁿ)
⇒ People can make friends

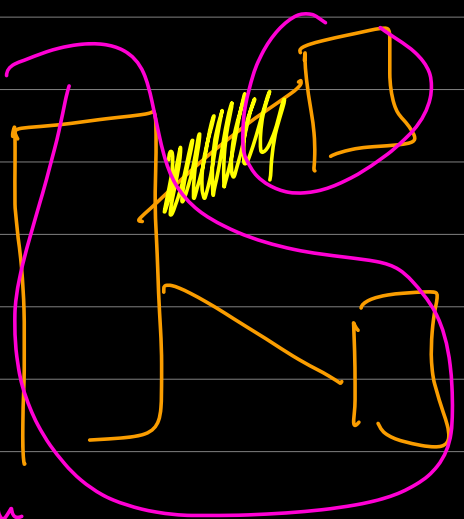
BeReal
Nearby Friends

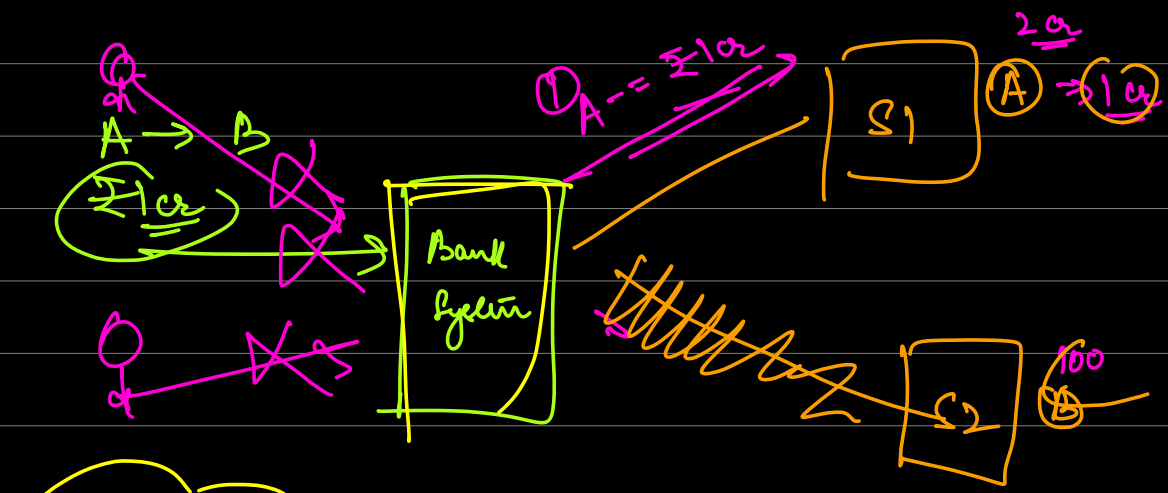
NON FUNCTIONAL Requirement

→ High Availability
→ Low Latency

CAP Theorem

C → Consistency
A → Availability
P → Partition Tolerance
↳ When there are 2 parts of the system that are unable to talk to each other (N/W Problem/Restart)



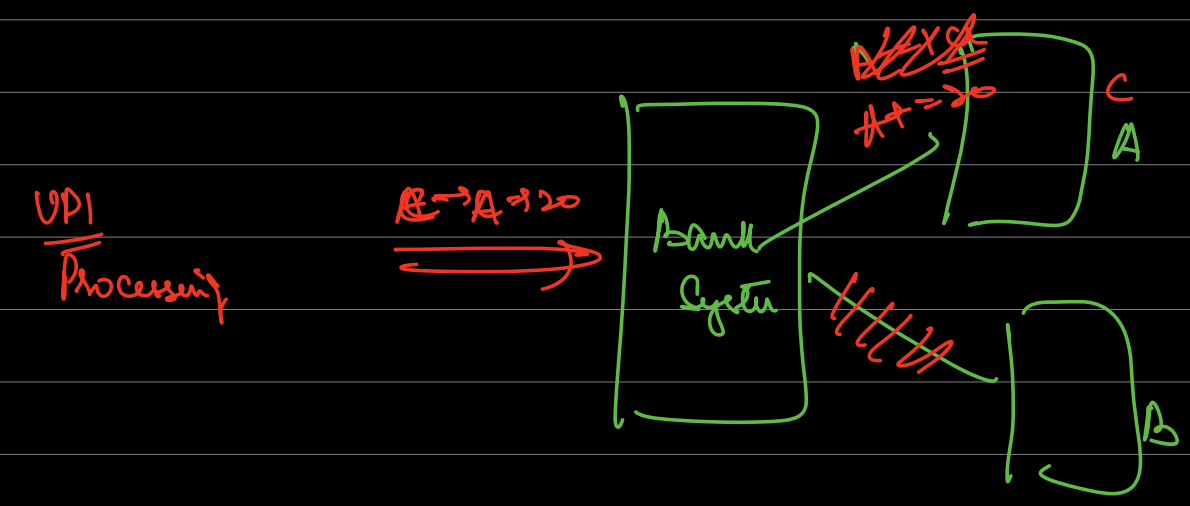


Consistent

OR

Available

XNOT BOTH



FB

→ Availability

Back of Envelope Calculations

- Storage
- Bandwidth
- # of Requests
- # of users

Newsfeed

How many posts

→ 500 M people use per day

90% → read → 50 M people create / day

10% → create → 2 posts / person

→ 100 M posts / day ⇐

70% of posts have media → 70 M posts / day

Size / media → (5MB) → 350×10^6 MB per day

$\xrightarrow{\times 10^3}$ 350 GB per day

→ 350 TB / day ⇐

100 M posts / day × 100 → 10B reactions / day

$\frac{10B}{25 \times 60 \times 60} \Rightarrow \frac{10 \times 10^9}{25 \times 4000} = \frac{10^{10}}{100000} = \frac{10^{10}}{10^5}$

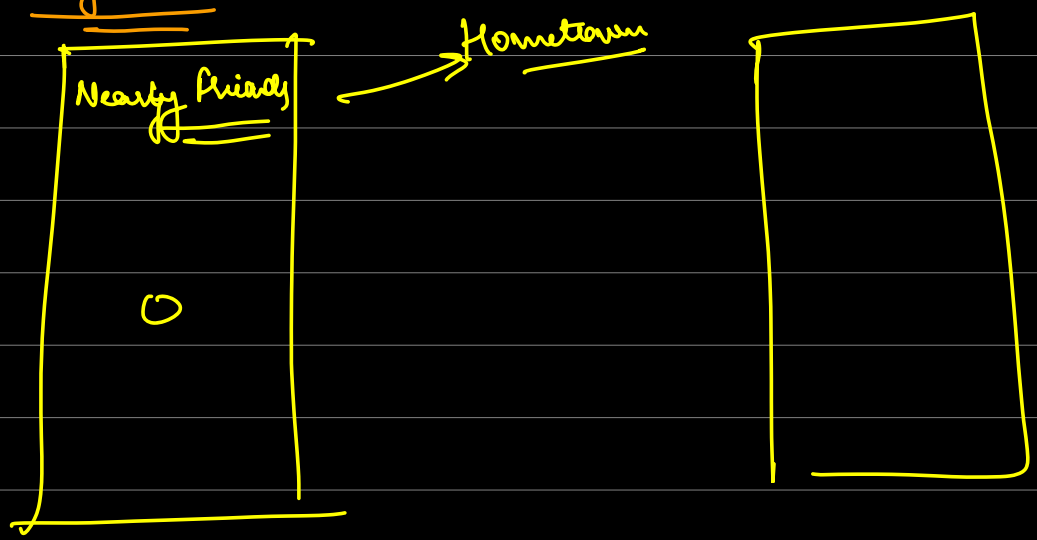
3600

→ 10^5 rps

0.1 M reactions / day

→ 1 lakh reactions / sec

Nearby feature



→ 500 M user / day

10X user nearby friend

⇒ 50 M use / day

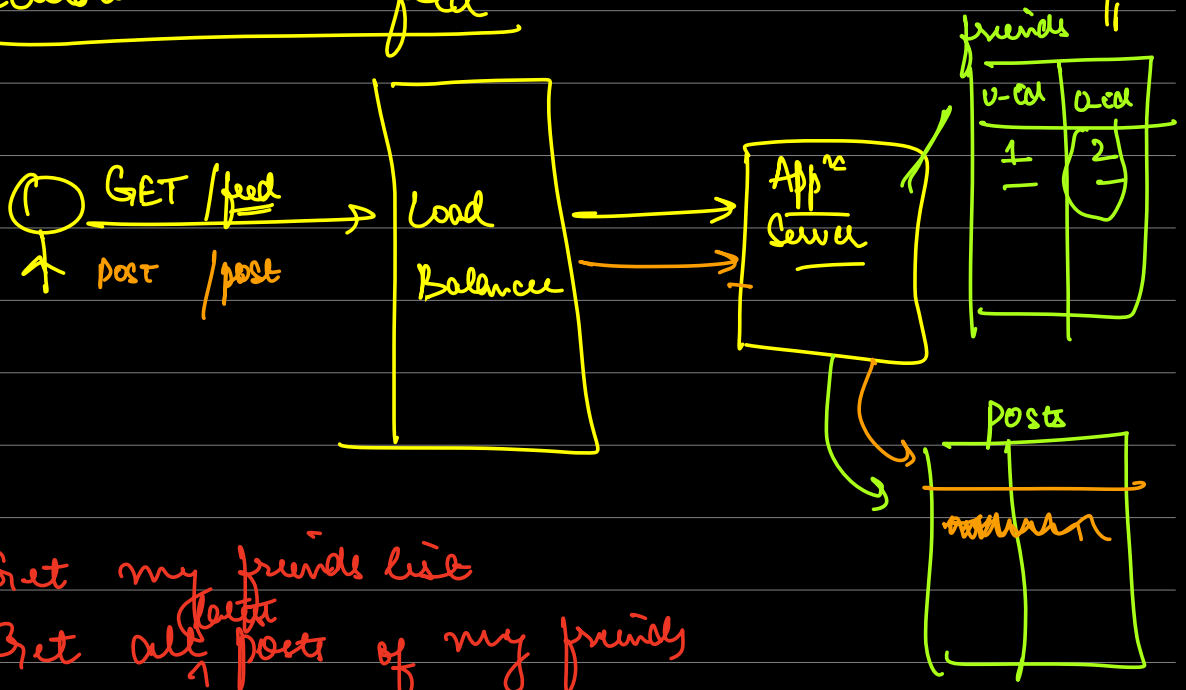
⇒ 5M active at a time ⇐

≈ 30%

System Architecture

Newsfeed / Ability to make post

Calculate the news feed



1. Get my friends list
2. Get all posts of my friends
3. Merge them and return the feed.

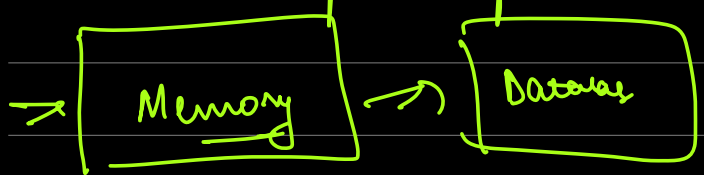
Easy ⇒ very very slow (5s)

Pros ⇒ Creating a new post is easy

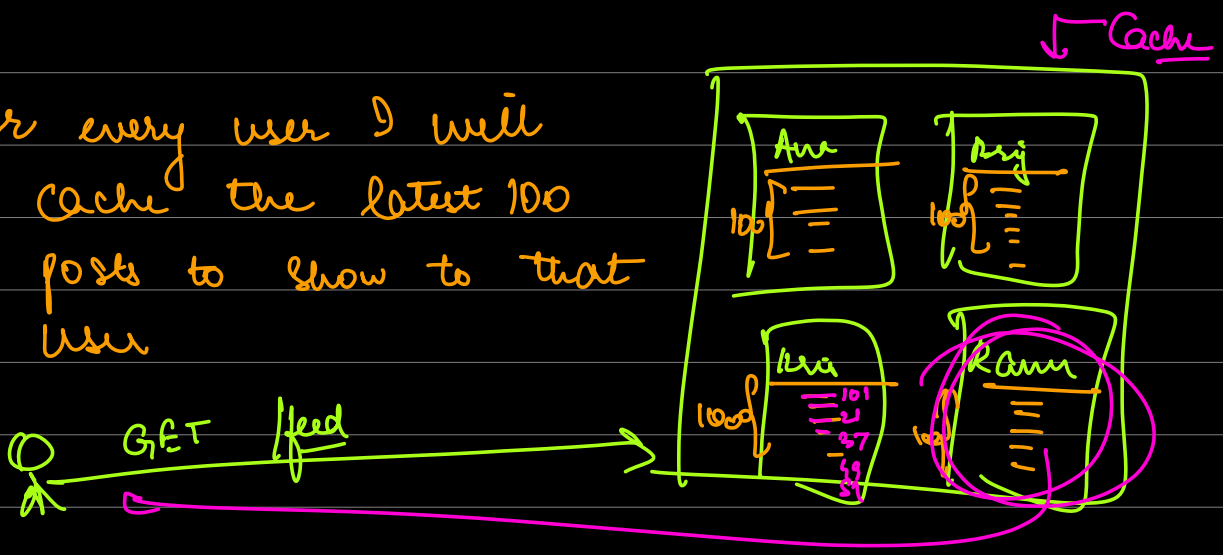
(A) ⇒ easier for you to use cache

low latency ⇒ pre computation

⇒ Can I precompute newsfeed of every user



For every user I will cache the latest 100 posts to show to that user



For every user, compute the feed in the cache (feed till for it 100). When a user requests their feed just give it

ID = long long int

$2B \times 100 \times 64\,Bit$

$200B \times 64\,Bit \approx 200\,B \times 100\,B$

$\approx 200 \times 10^9 \times 100$ bits

$\approx 200 \times 10^{11}$ bits

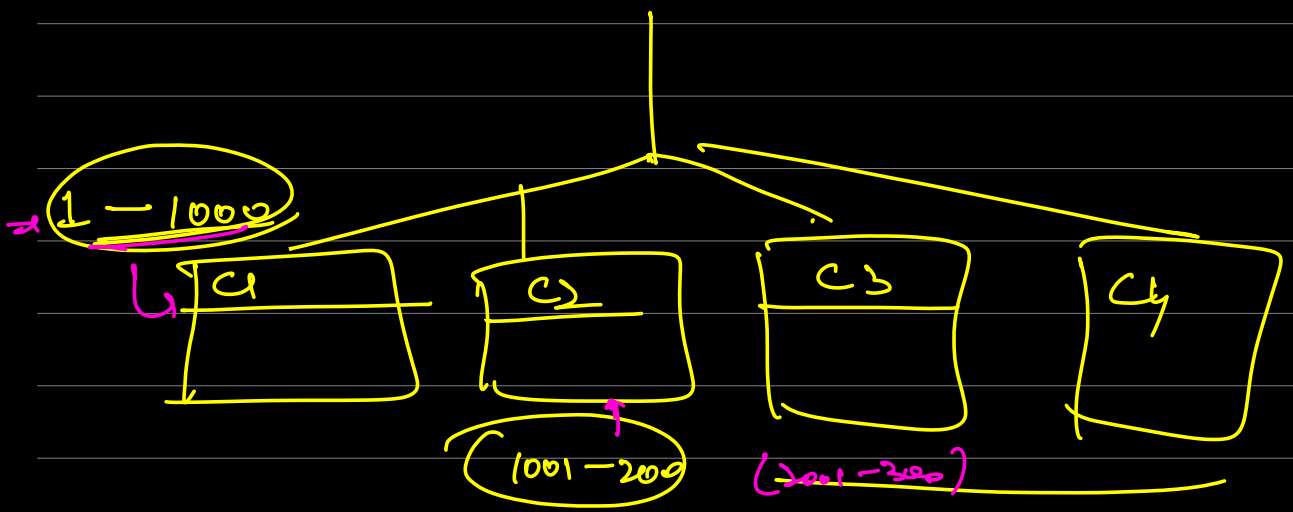
$\approx 200 \times 10^5$ gigabits

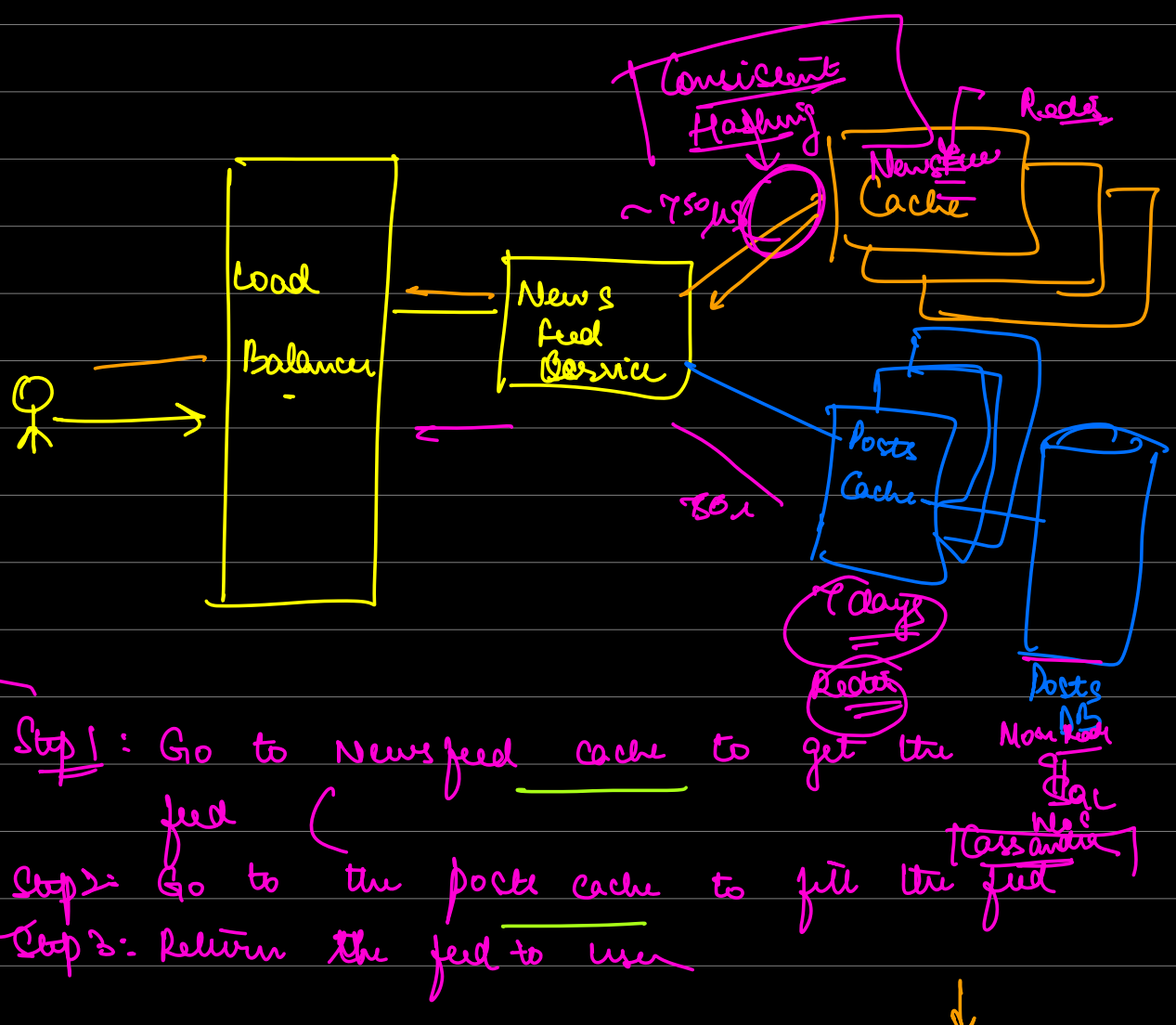
$\approx 200 \times 10^2$ gigabits

$\approx$ (20 TB) $= \approx 10\,GB$

$20 \times 10^2\,GB$

10 GB

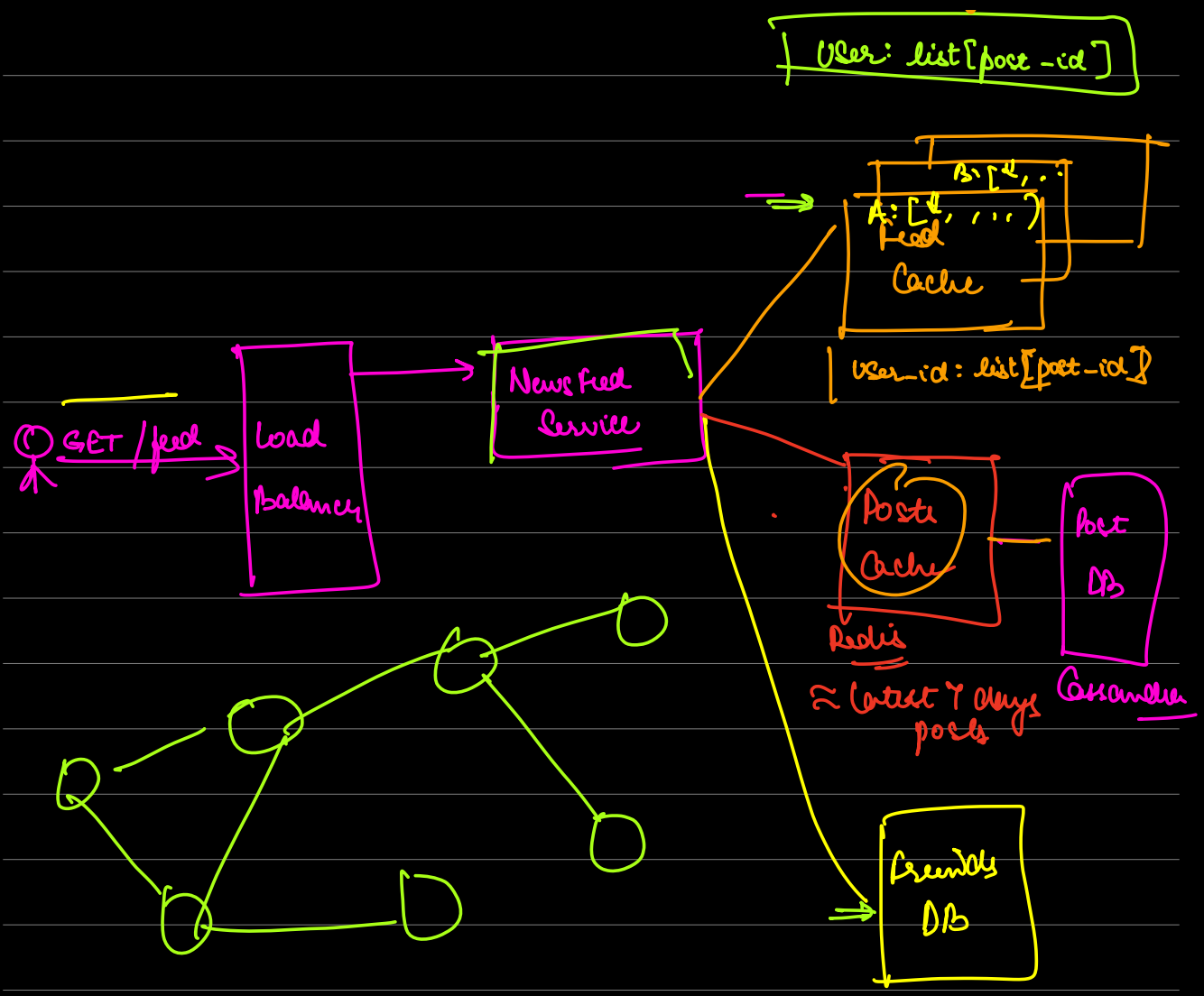




Step 1: Go to Newsfeed cache to get the feed

Step 2: Go to the posts cache to fill the feed

Step 3: Return the feed to user



FEED LOGIC

→ Go to the feed cache

→ If feed is found:

→ get post details from posts cache

→ done

→ else:

go to the friends db

find all friends of user

go to the posts db

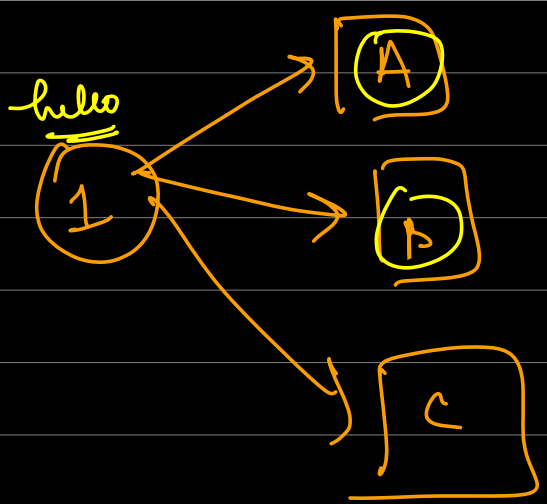
find their posts

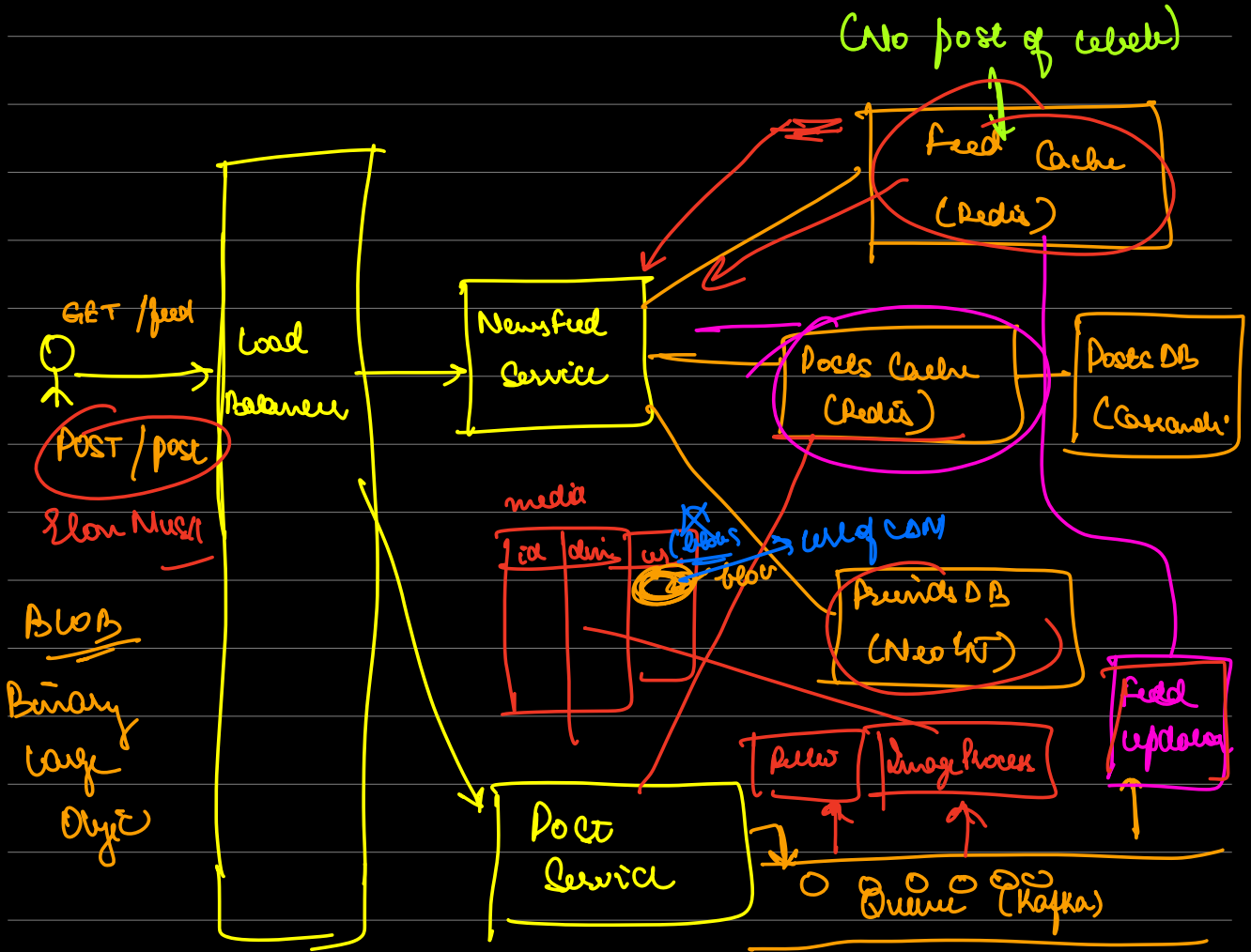
return and add to cache

HOW WILL FEED CACHE BE COMPUTED

→ News feed has to be updated only when a new post from one of my friends

→ When a person makes a post, I get a list of their friends, go to their news feed cache and update that

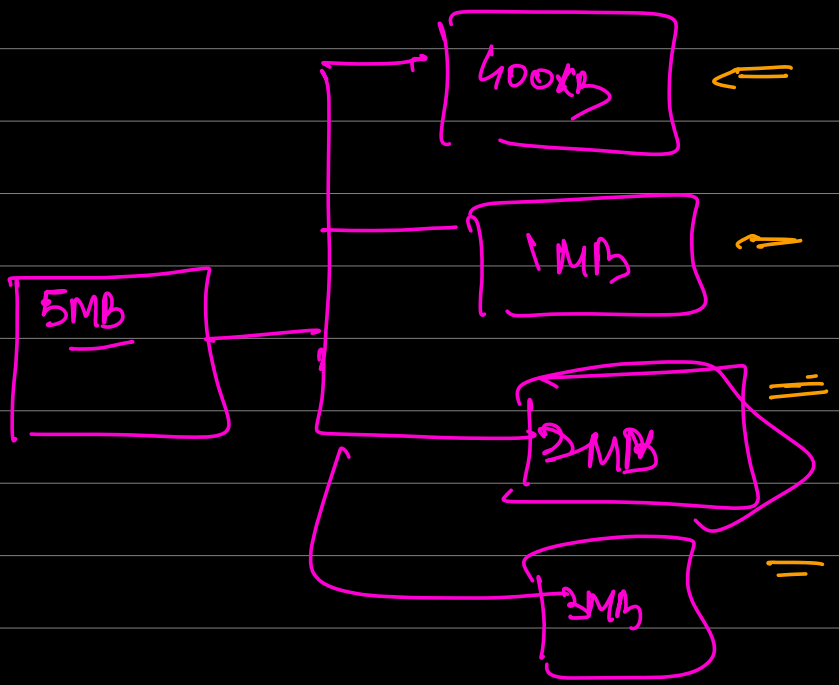




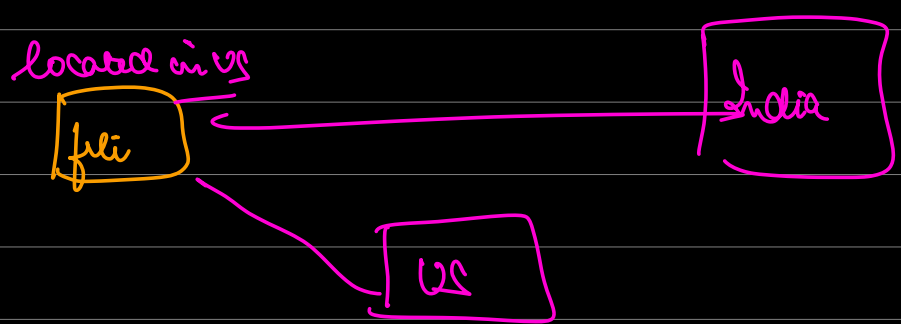
When a user creates a post

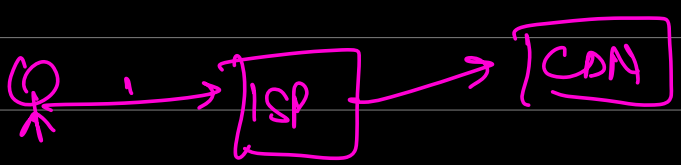
1. update posts cache and DB and return success
2. Asynchronously update the feed of every friend in the cache

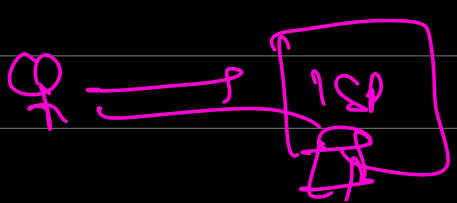
Messaging Queues

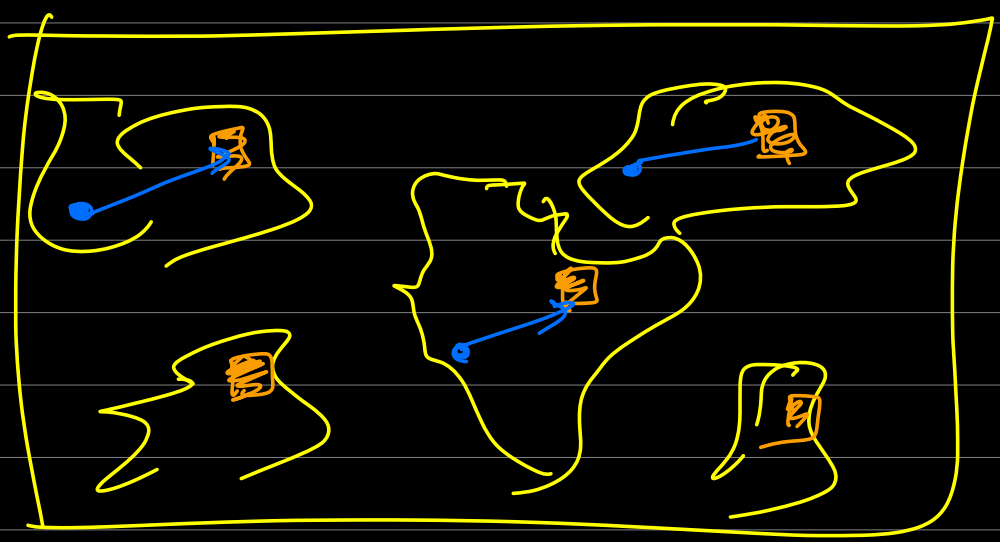


CDN ⇒ Content Delivery Network ⇒ distribute file across globe









Design of News feed (FB/ Twitter/ YT)

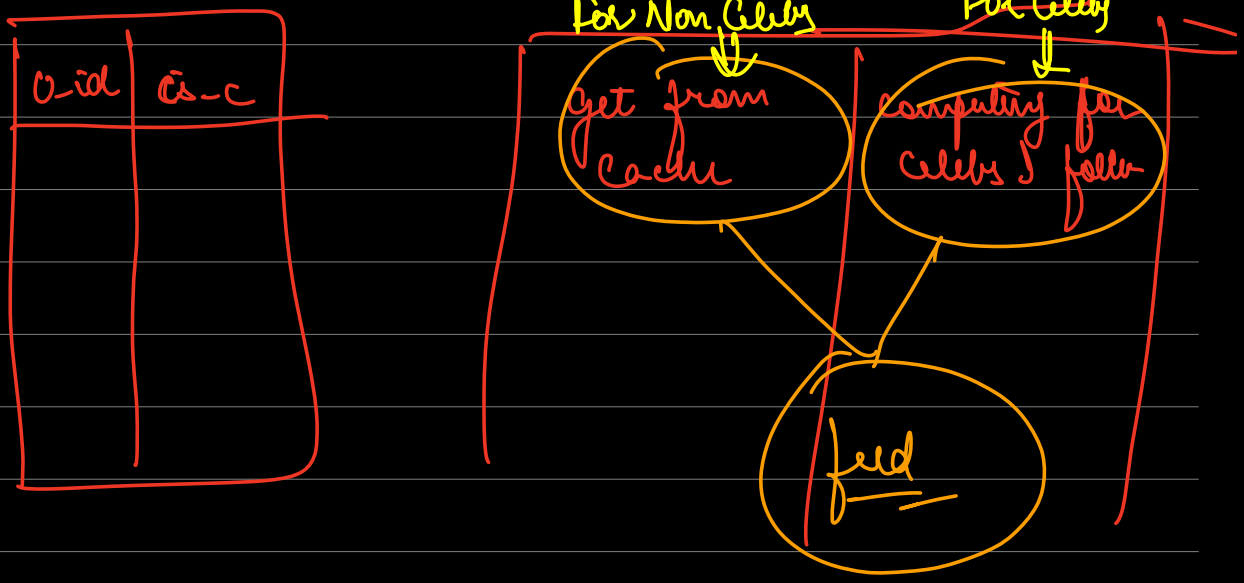
FB → 5000 friends

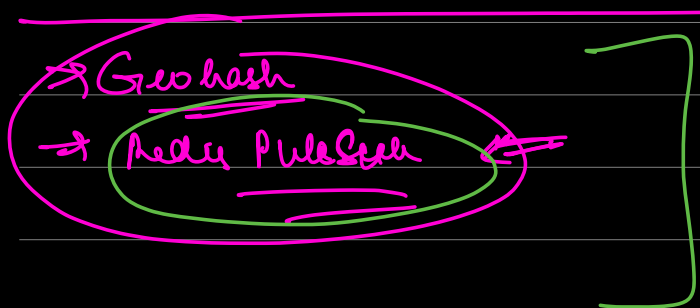
followers → ∞

Split logic

→ Get the normal feed via cache

→ for celeb users, compute the feed


U_id
Cs_C
For Non Celebs
For Celebs
Get from cache
computing for celebs & follo
feed


Geo hash
Redis PubSub

Pub - Sub