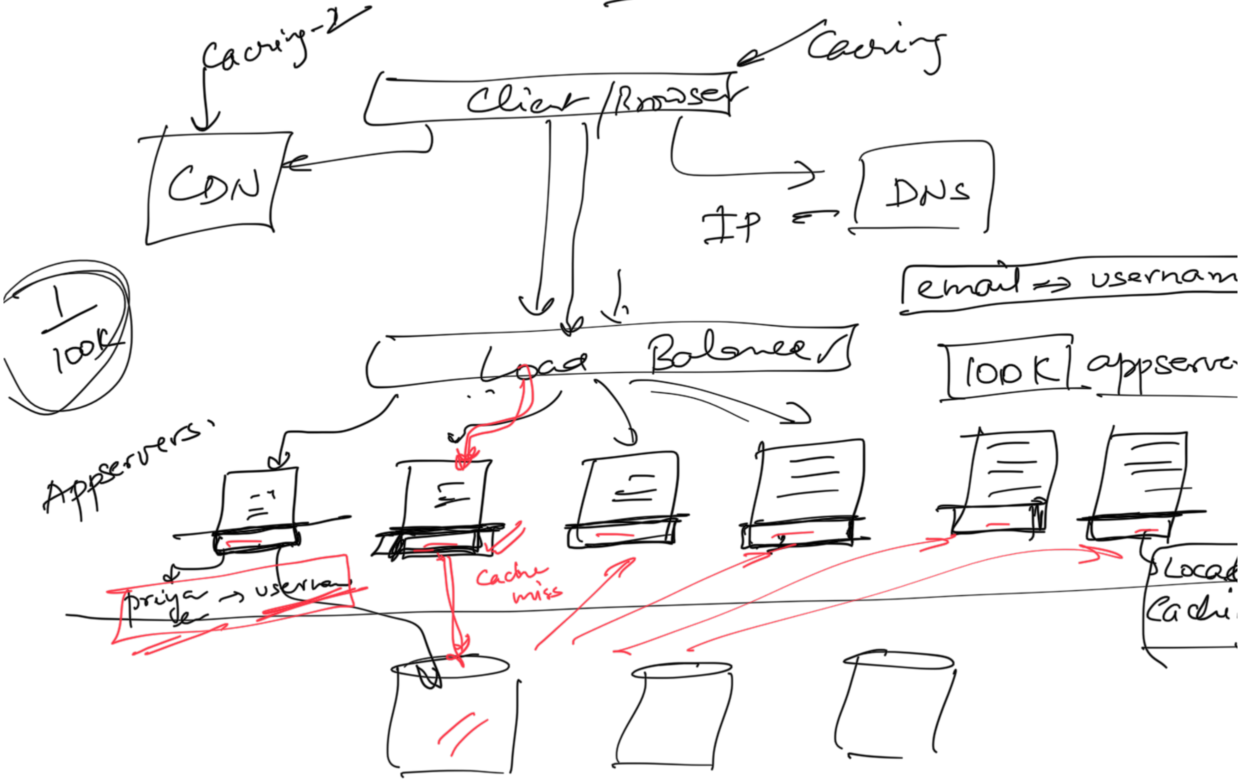
# CACHING - II

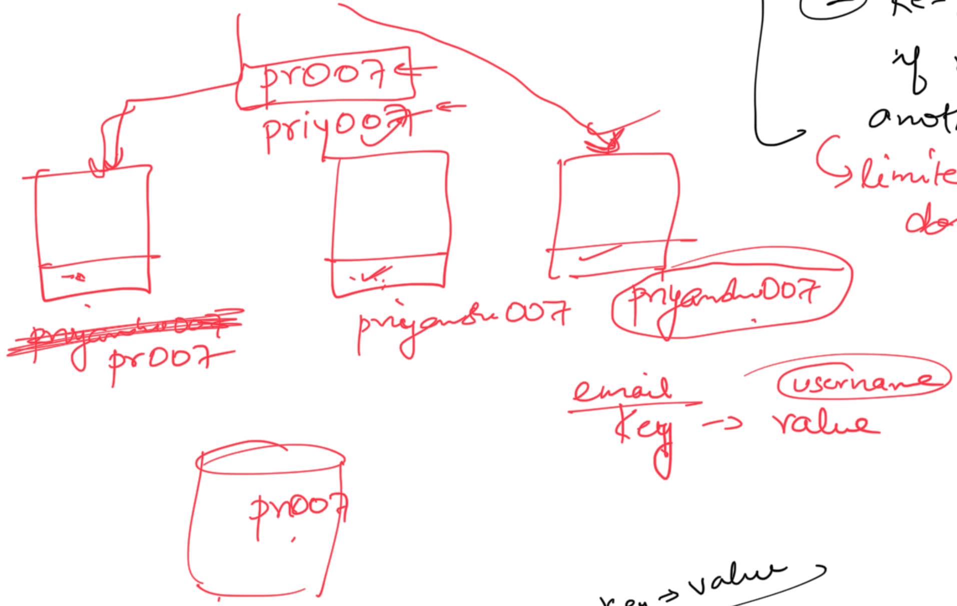


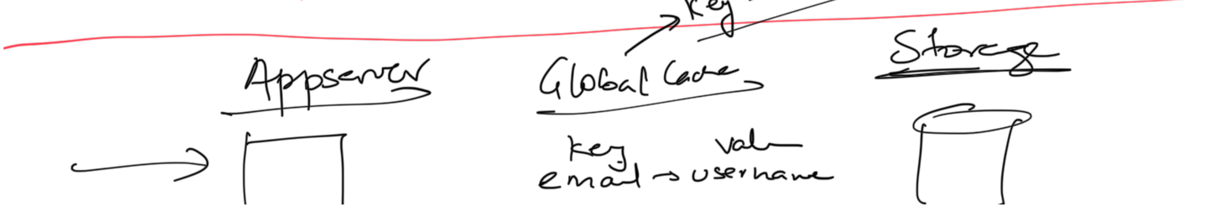
100K machines

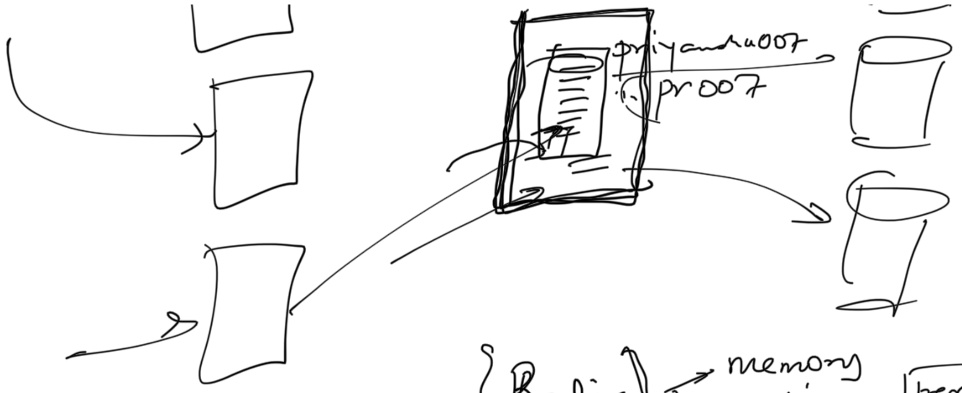
↳ very expensive

① Old data
↳ cache inv

② Re-fetching dat
if req goes to
another appser
↳ limited amount of
data you can
cache.







Cache eviction
↳ LRU

{Redis} → memory
↳ option → persist entries

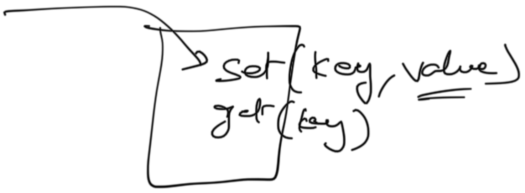
{Memcache} → memory only.

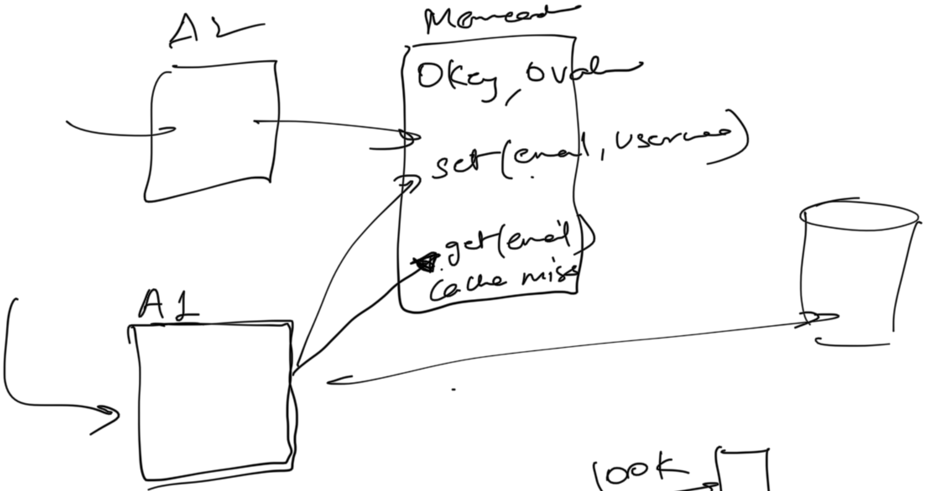
In memory
↳ RAM

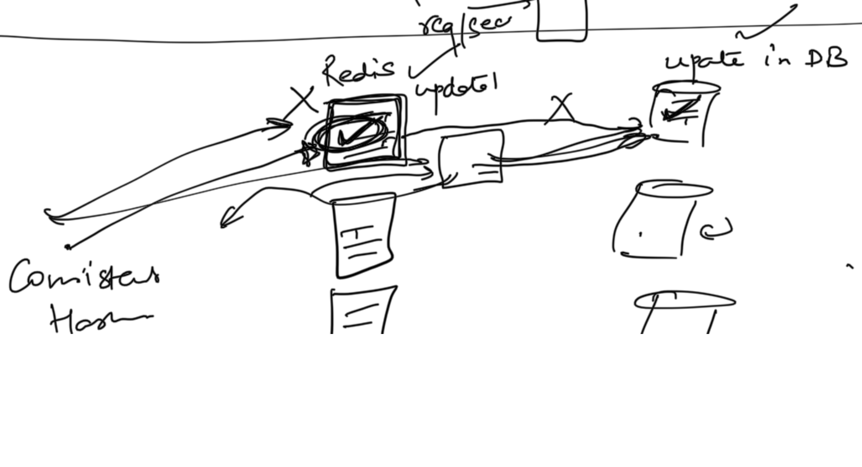
→ Cache → optimise reads RAM

① Most likely to be fetched → popular data

② Derived information







① Write through cache, → Highly consistent

① write latency.

② Machine dies

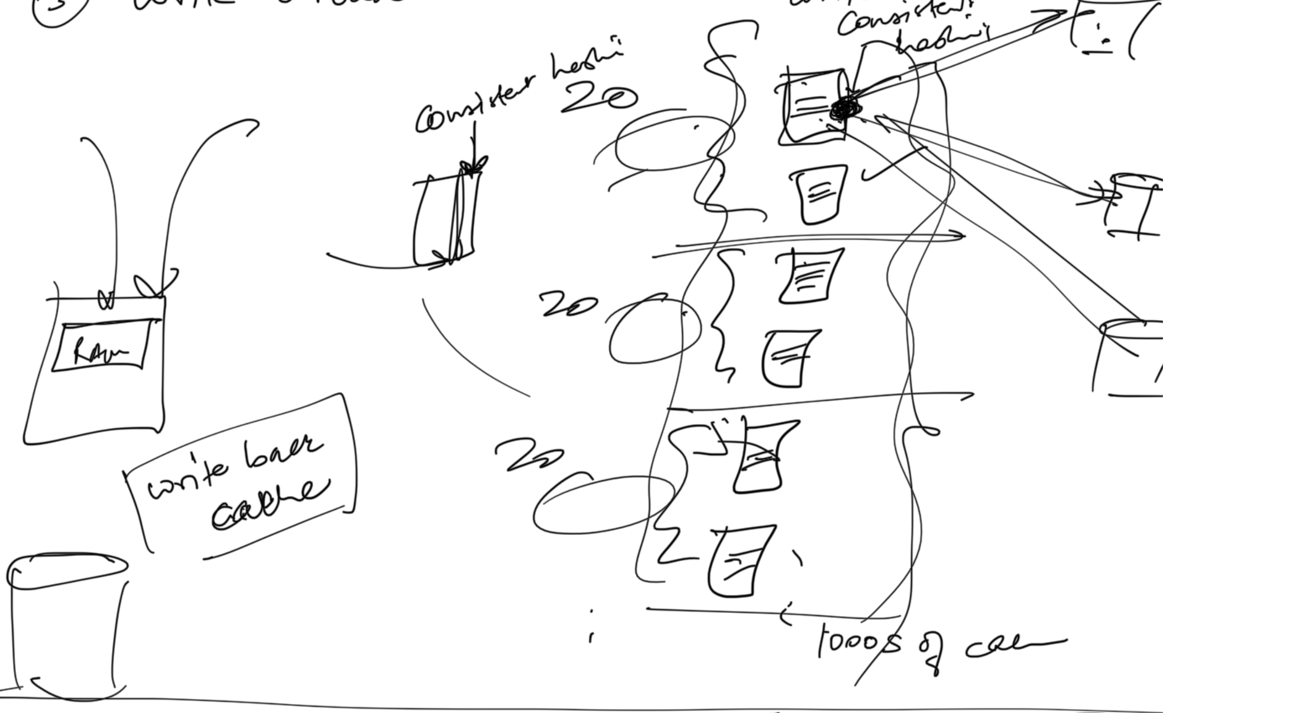
↳ High laten

100ms 200ms 200ms.

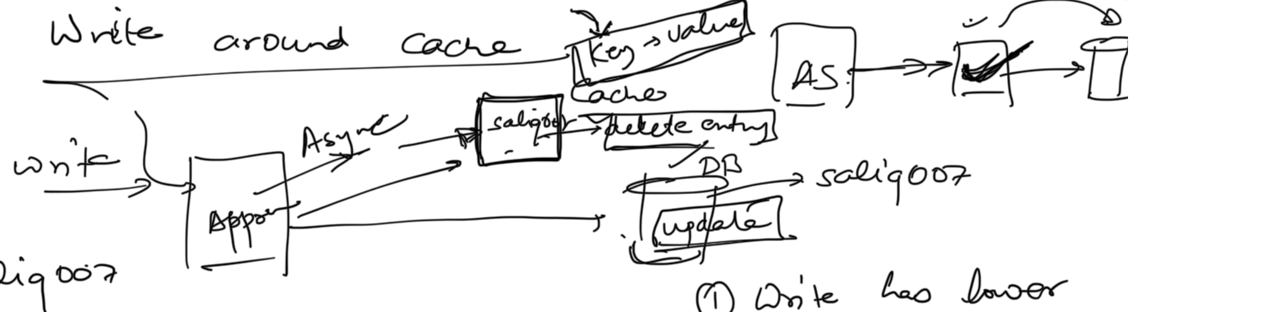
↳300ms

② Write back cache

③ Write around cache



Write around cache



① Write has lower latency

② Temp inconsistent

first read will lead to cache miss

Read

Go to Appserve

~~Fr~~ Read from cache

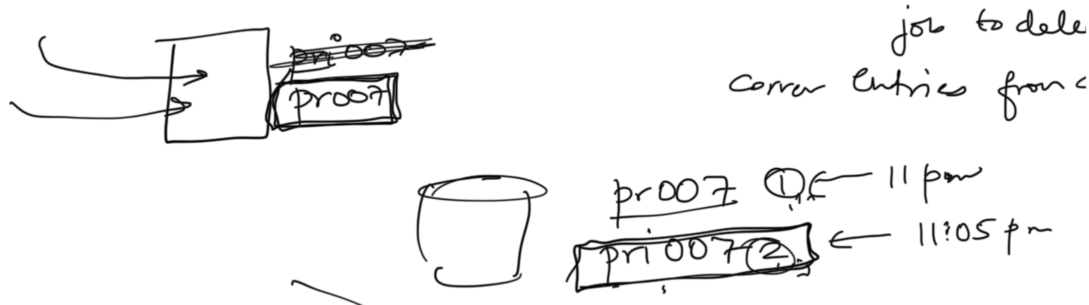
↳ if entry is found, then
return

If not, then you find entry in DB
update cache with entry

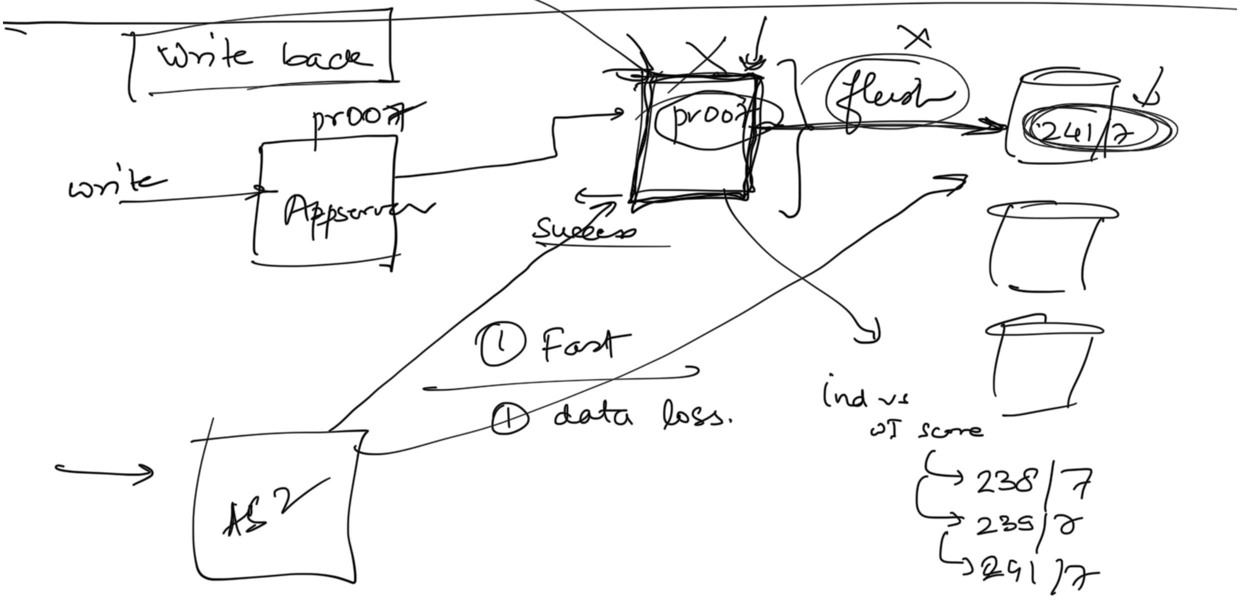
Write_Around

① Appserver
↳ write to D
↳ retu

② schedule a
job to dele
corres entries from c



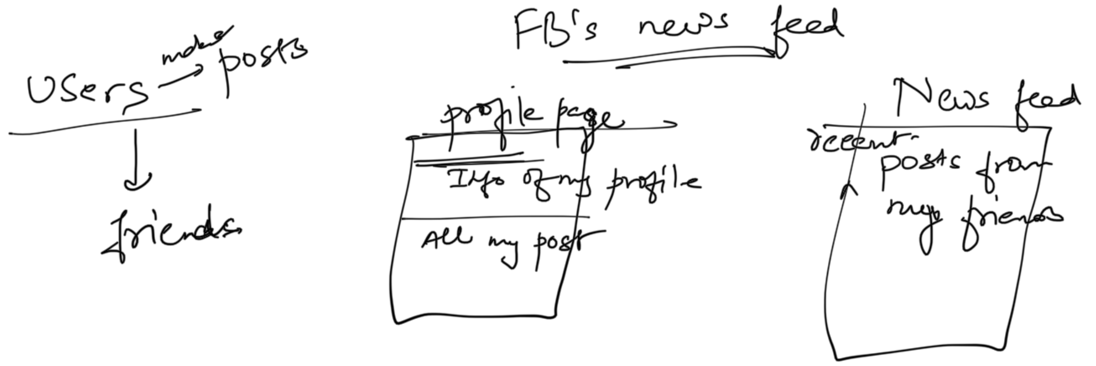
Write back



① Fast

② data loss.

FB's news feed



User_s

| id | username | email | profile |
|---|---|---|---|
| | | | |

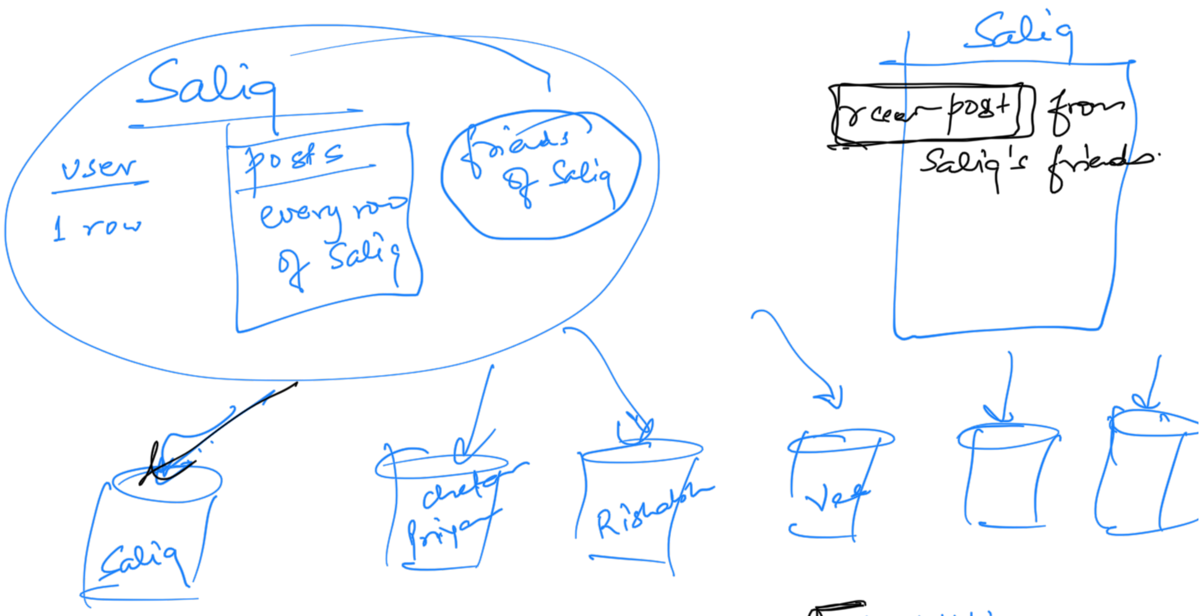
Posts

| id | user_id | content | attach | ts |
|---|---|---|---|---|
| | | | | |

User_friend

| userid | friend_id |
|---|---|
| 10 | 100 |
| ~~100~~ | ~~100~~ |
| 100 | 200 |

RAM



28 bytes post

| ts | userid | content | path |
|---|---|---|---|
| | 8 bytes | 100 bytes | 100 bytes |

200 bytes

1 billion registered users

200 million DAU

40 million posts every day

40 million * 200 bytes

40 * 1000 * 1000 * 200 byte

1000 → MB, 1000 → KB

= 40 * 200 MB = 8000MB

= 8GB

8 * 30 = 240 GB

derived Recent posts:

Last 30 days of all posts made

Post

| post-id | creatorid | ts | content | attach_path |
|---|---|---|---|---|

SELECT * FROM
posts
where user_id IN

① Go to Saliq's DB

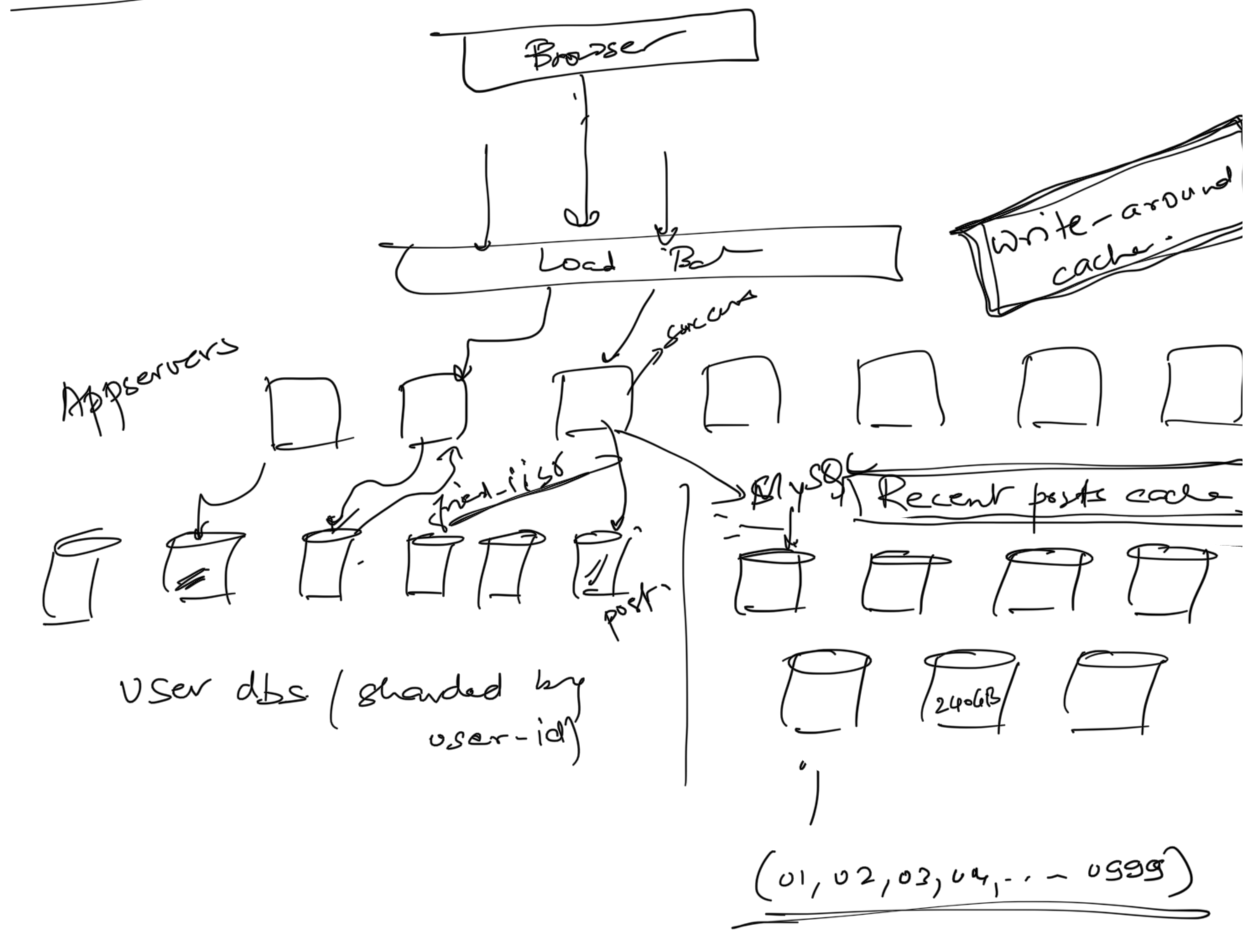
w

( _ step1 _ )

to fetch list of Saliq's friends,

ORDER BY ts DESC ②

LIMIT 20, offset 0



SEL

WHERE (user_id) IN
( - ——— )

~~AND~~
ORDER BY ts DESC