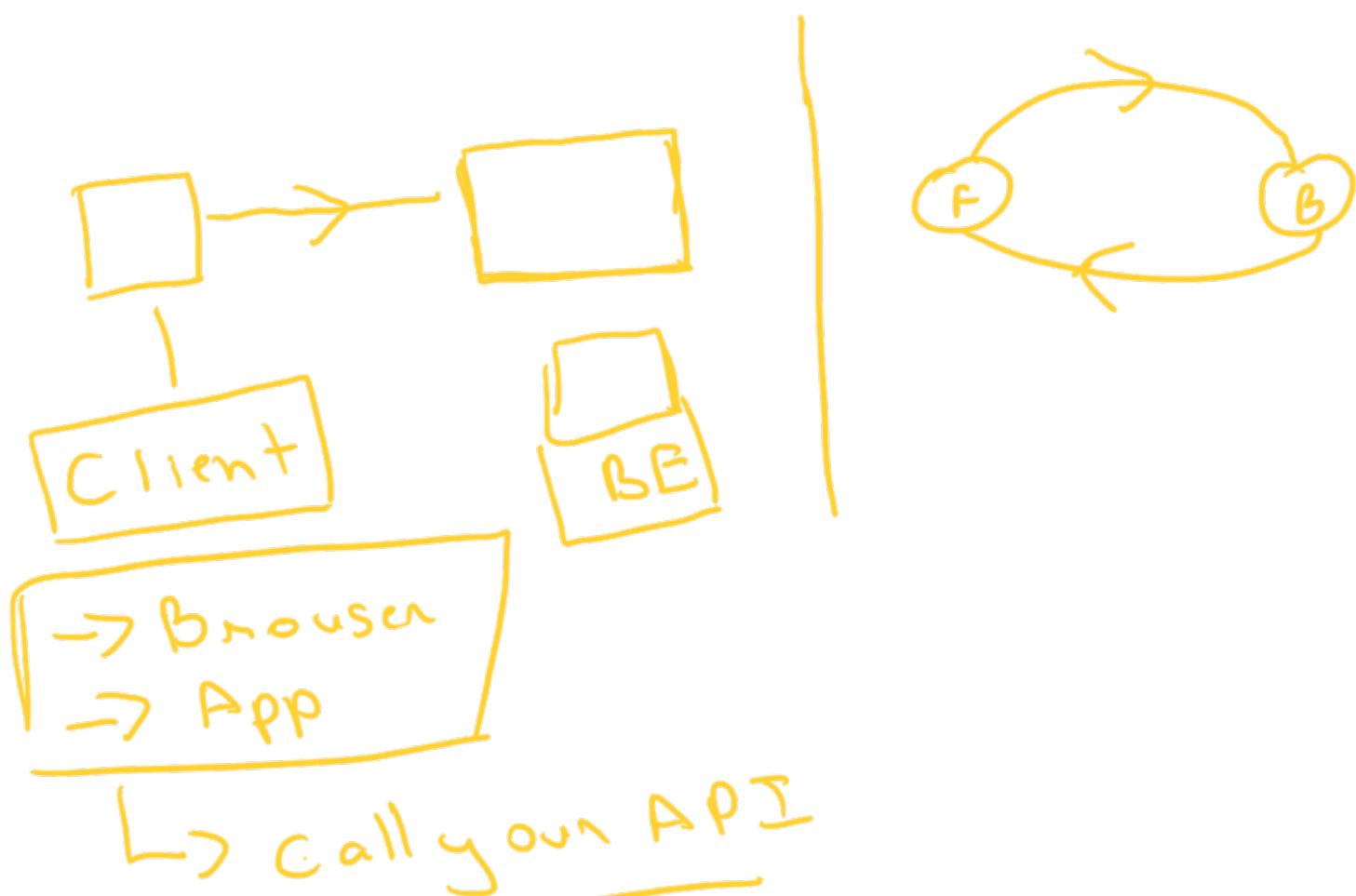


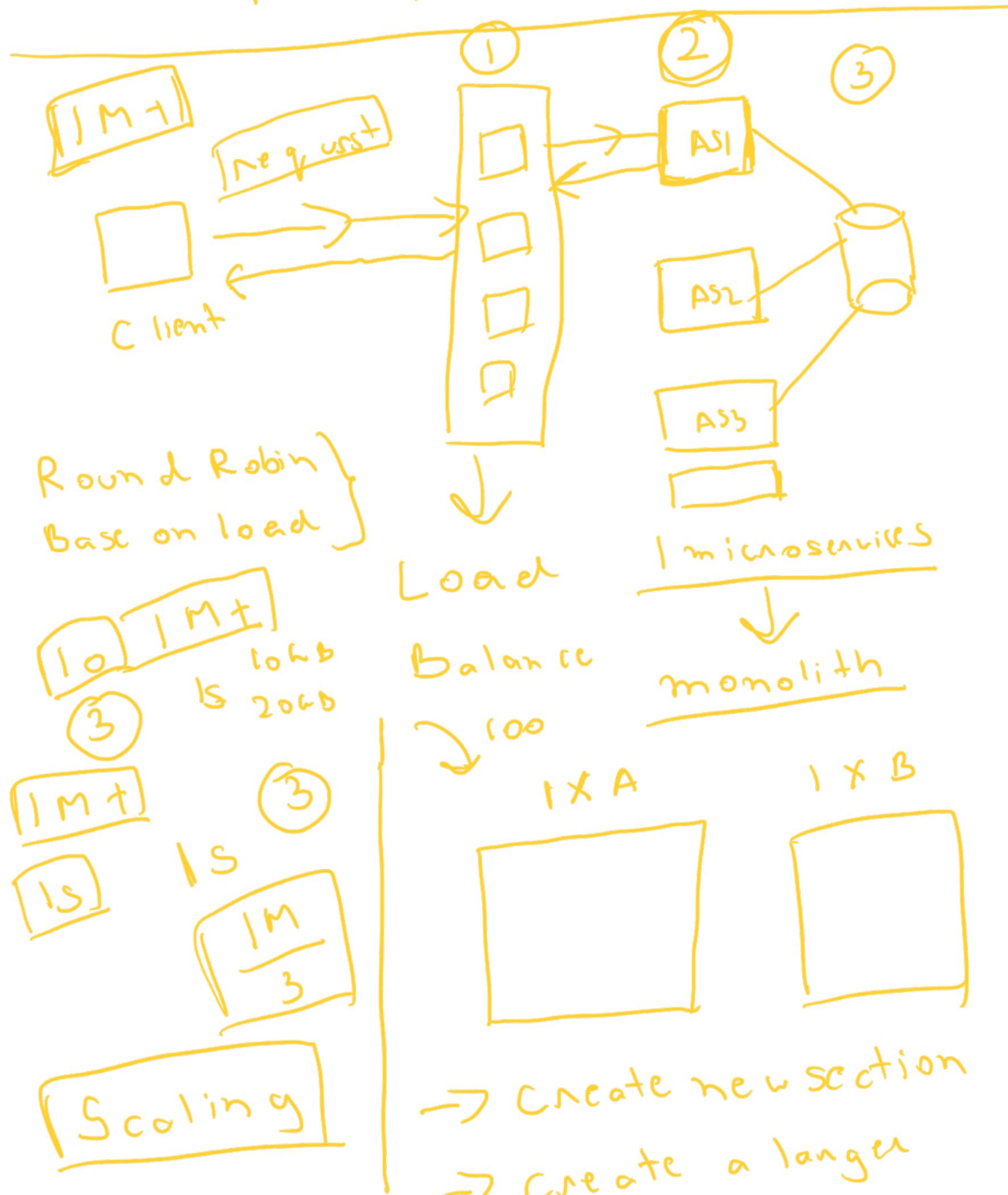
Design a distributed cache

- Caching
 - motivation
 - Types of caching
- Distributed Cache
 - motivation
- Strategies of caching
 - read / write
 - eviction
- Design in OOP
a distributed cache

Motivation



FE HA Project



Scaling

↳ vertical scaling
→ increase capacity of the machine

→ 100 now 200

$\rightarrow 200 \rightarrow 1$
 $300 \rightarrow 1$

↳ horizontal scaling

↳ creating more classrooms

↳ 100 more charges }
→ load →

→ disadvantage

→ expensive

→ distributed



— 1 machine

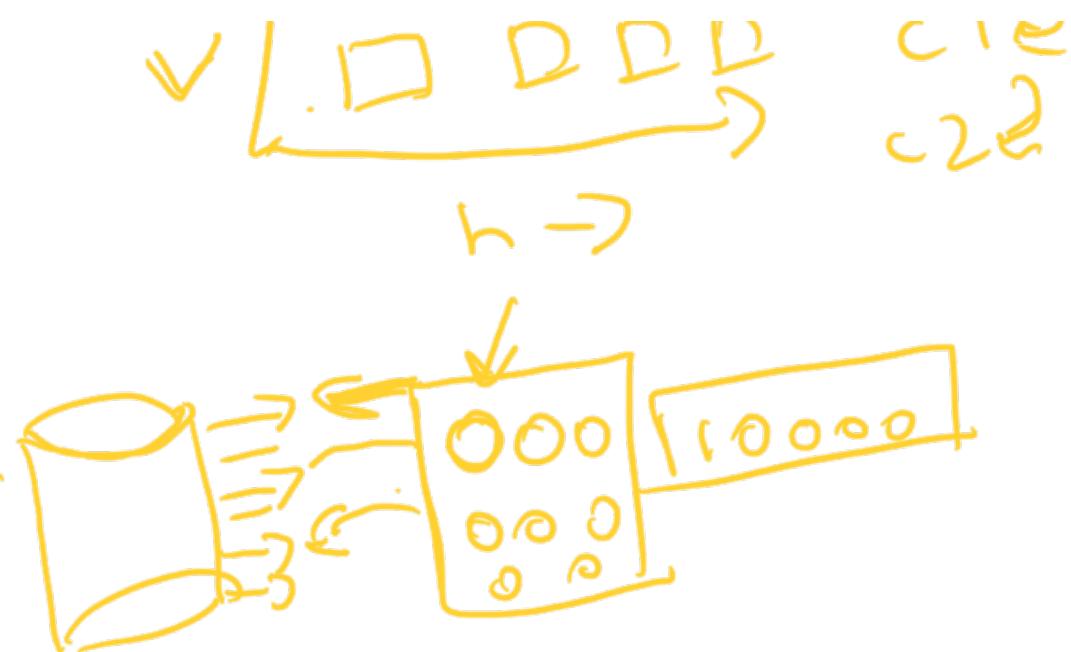


10 GB
+ 10

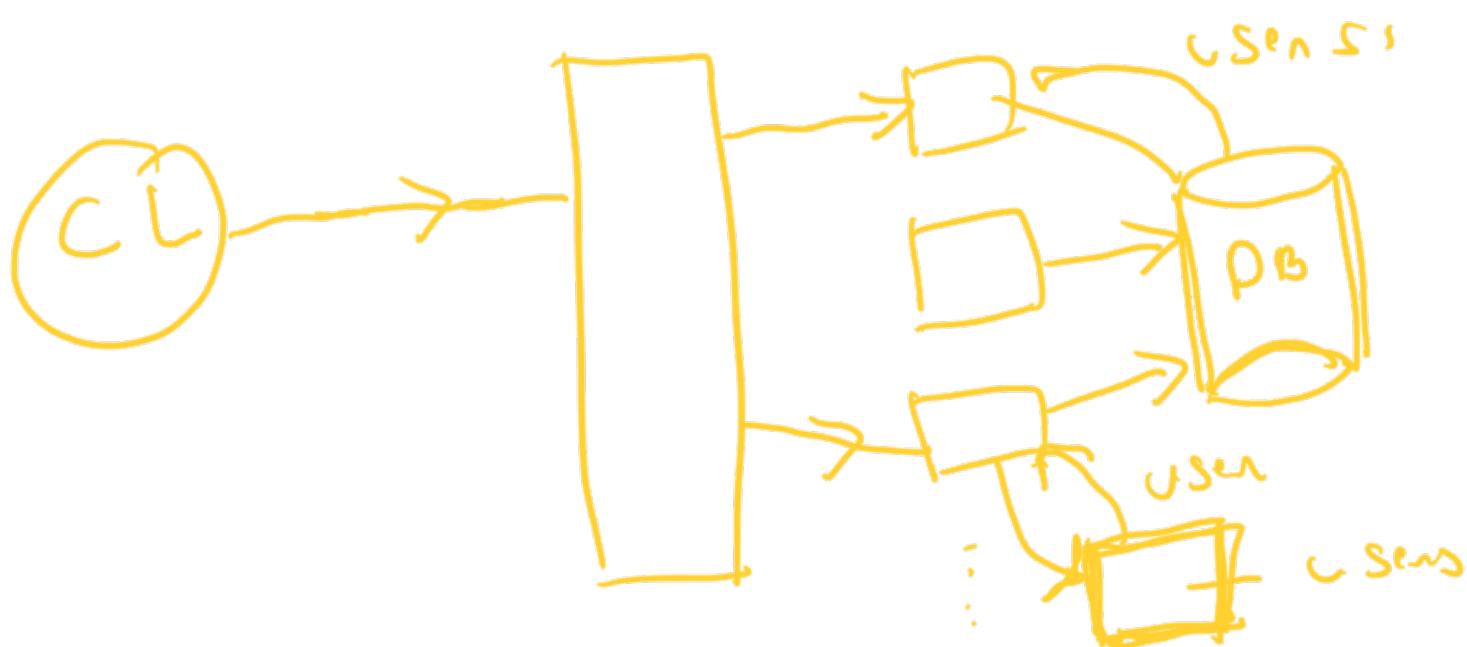
vertical

Diagonal





→ shard
→ replicate



Access Pattern

- which data is being loaded
- very frequent
- which API is called a lot
- query is taking most time

1 million

- 80% of **GET USER**

→ **replication** | sharing

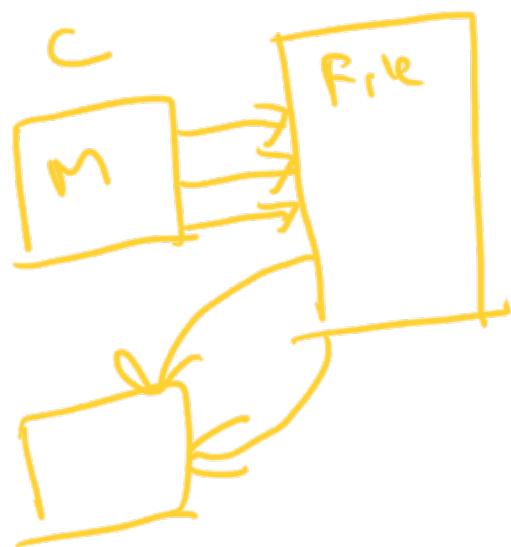


Cache vs DB

- access time
- Faster at Fetching data

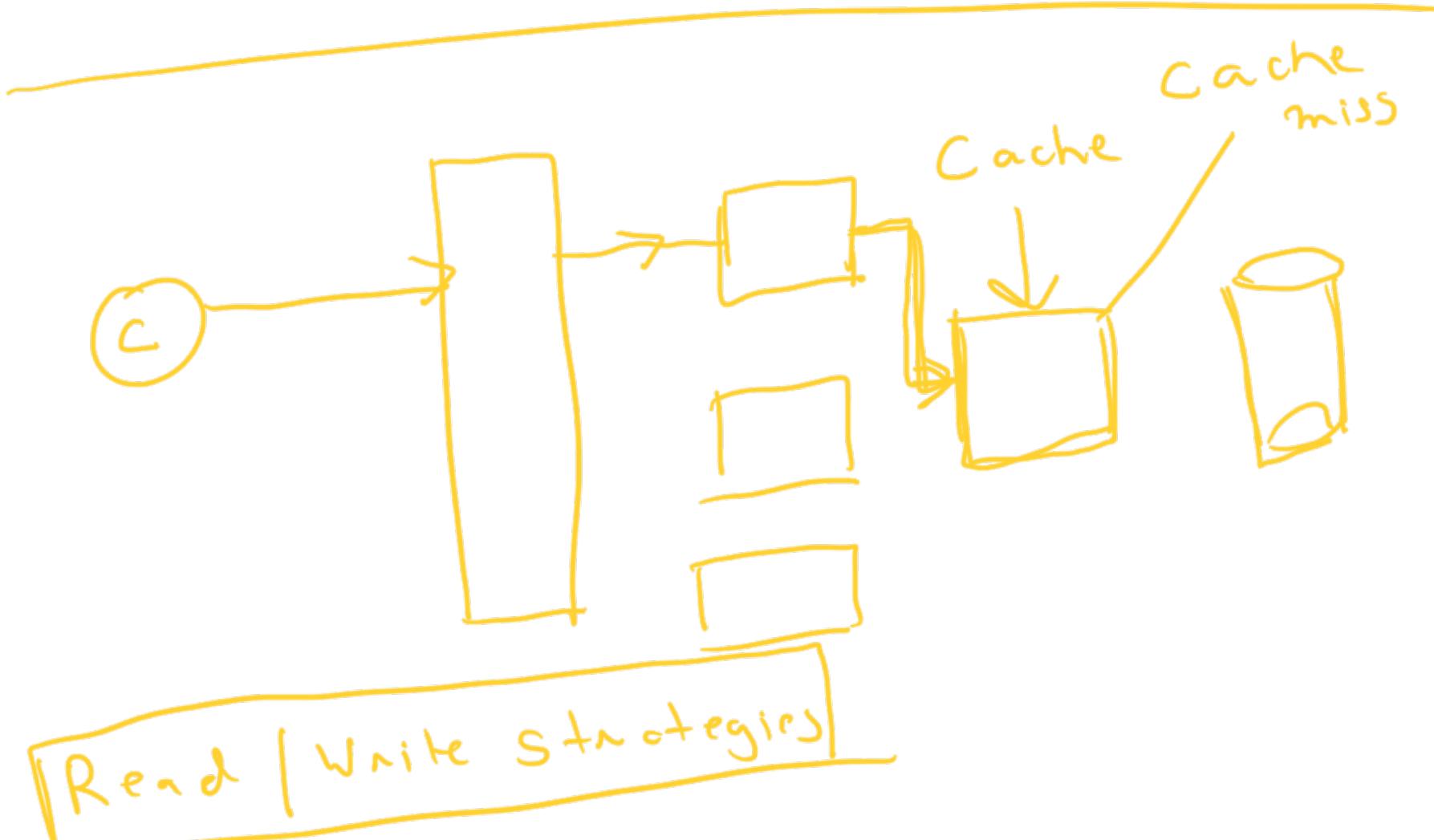
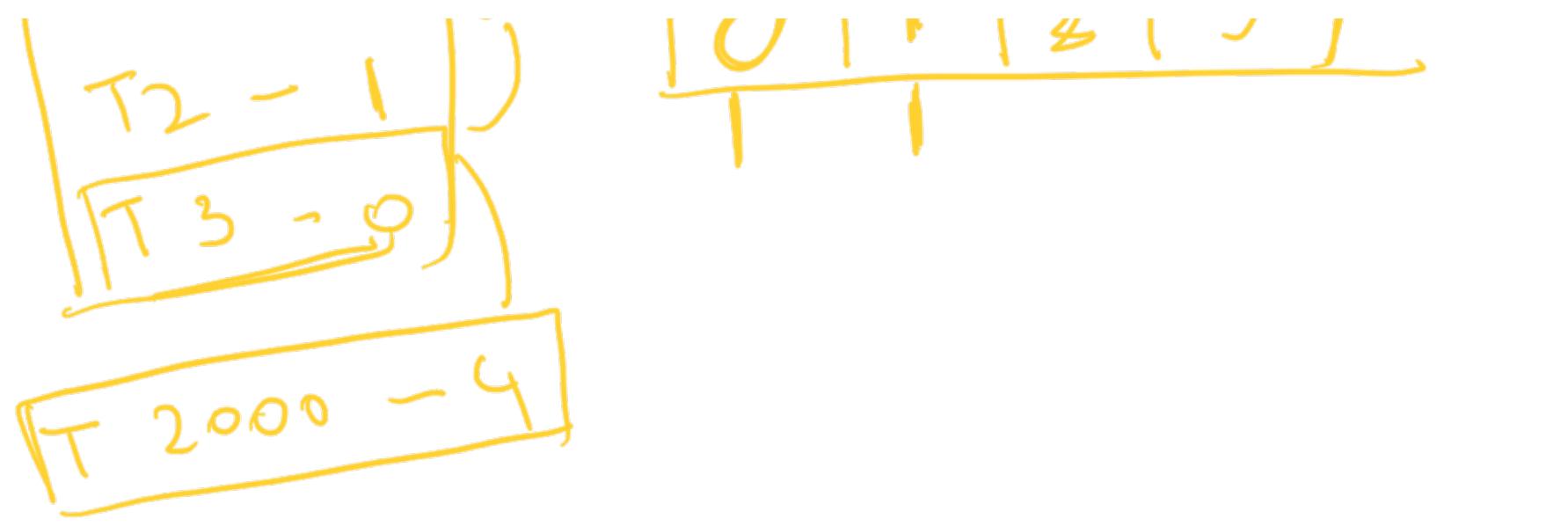
Persistence

- no
- disk based

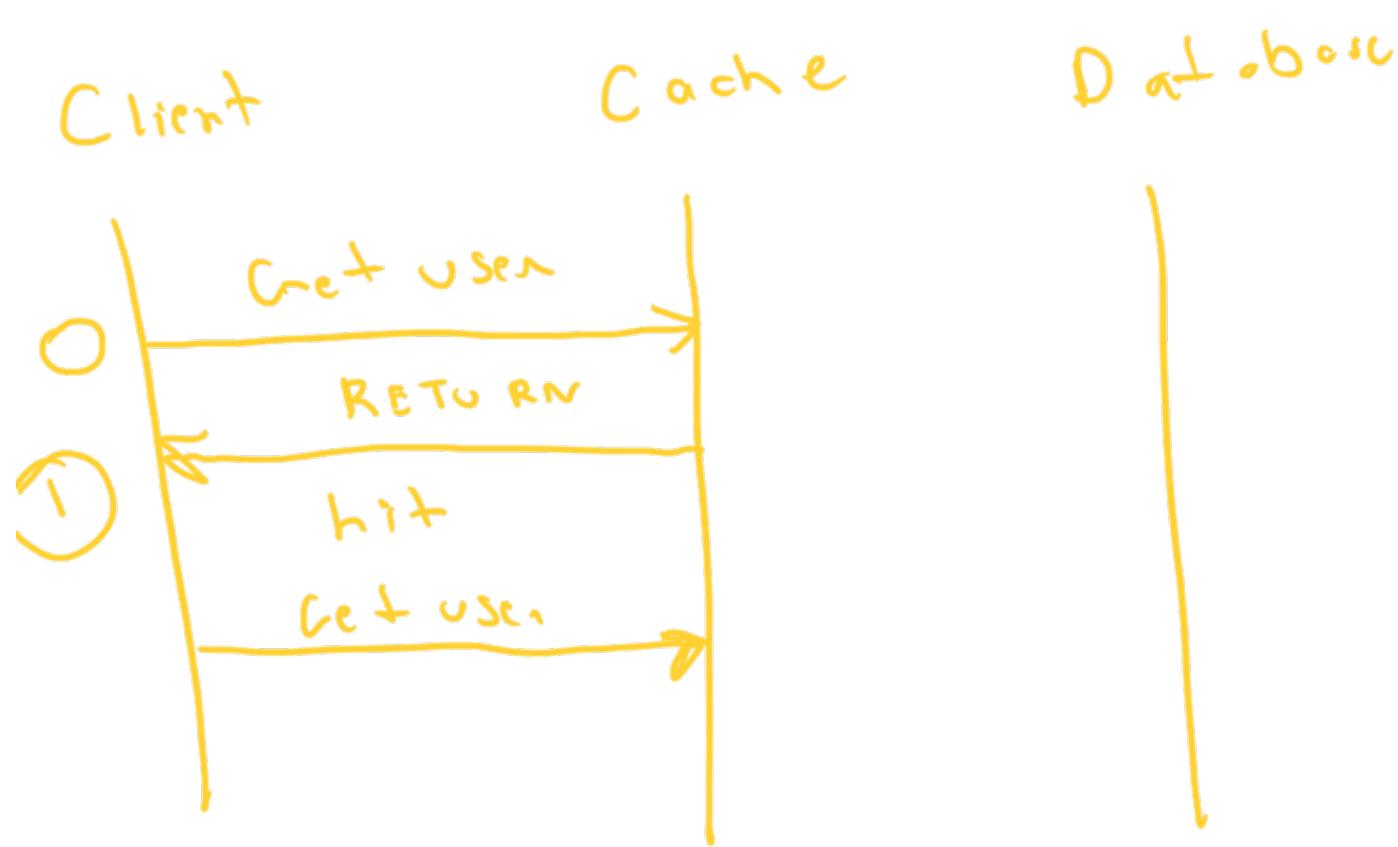


→ Locality of reference





→ Cache-aside

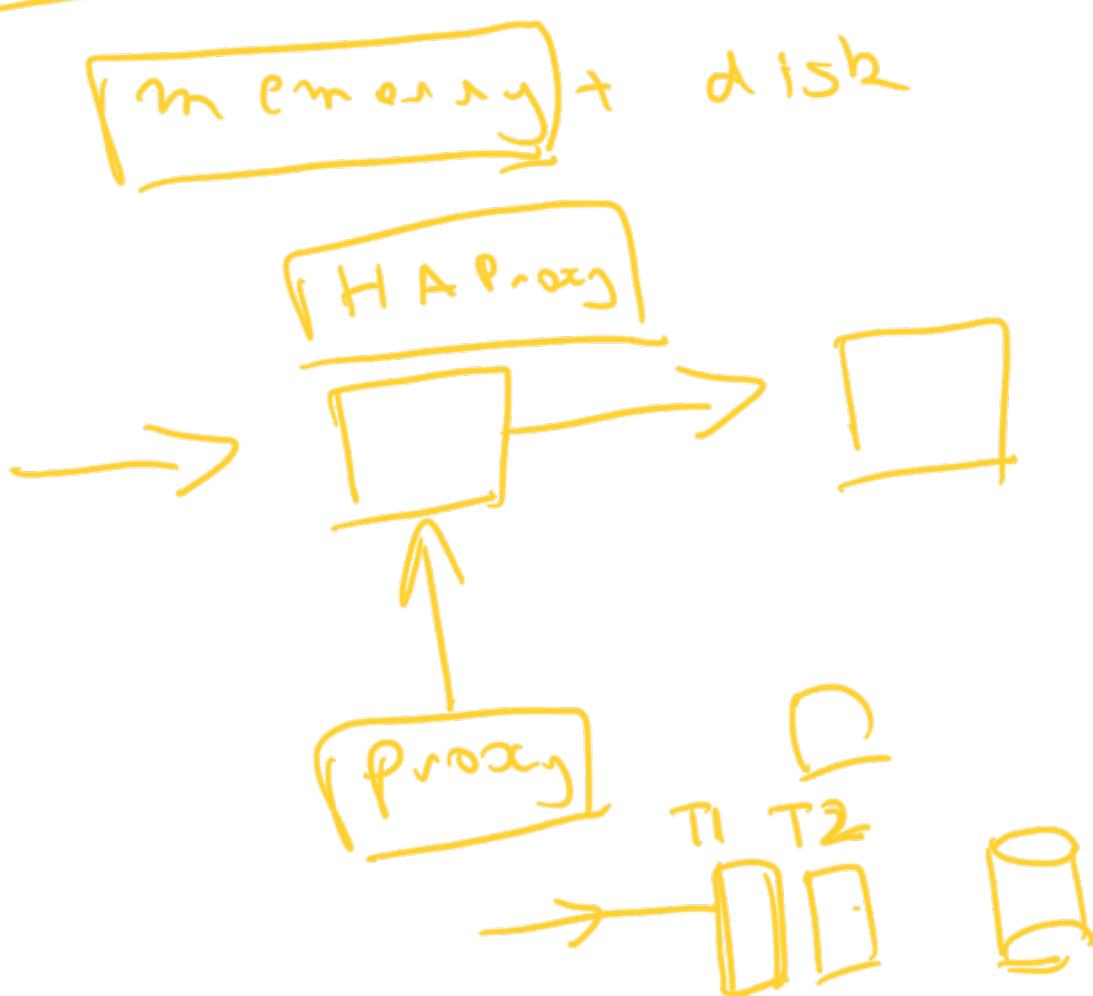


application server

- gets the data from the UD
- and stores in the Cache

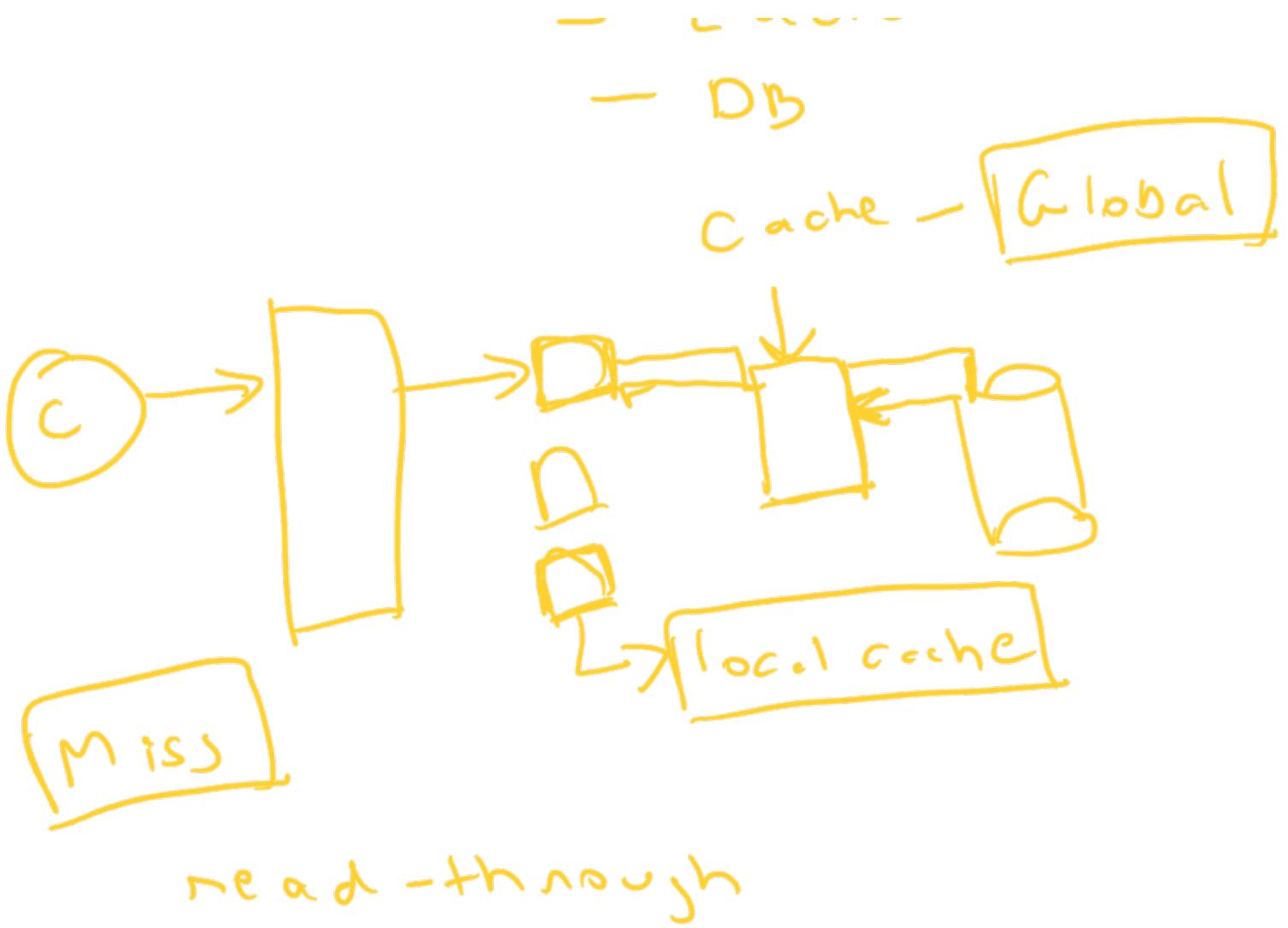


cache aside

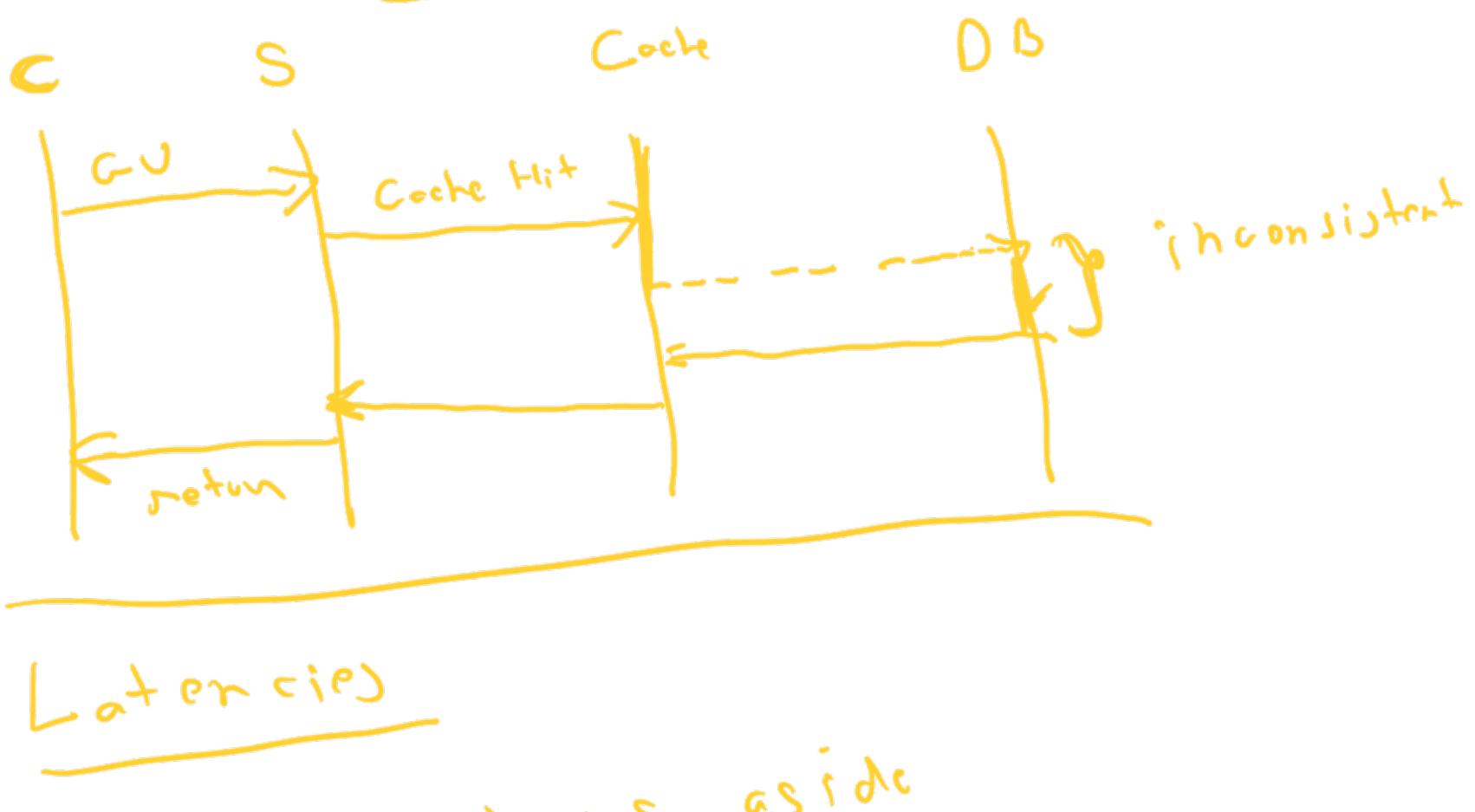
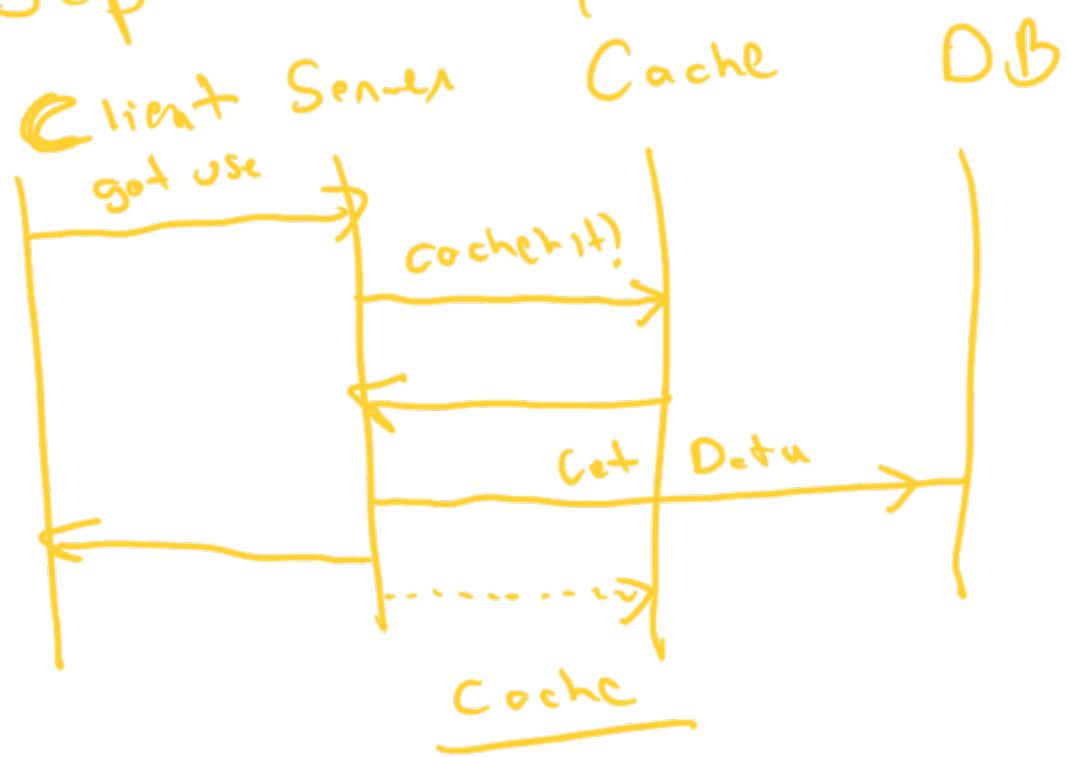


App Server

- cache invalidation
- eviction
- multiple data store
- cache



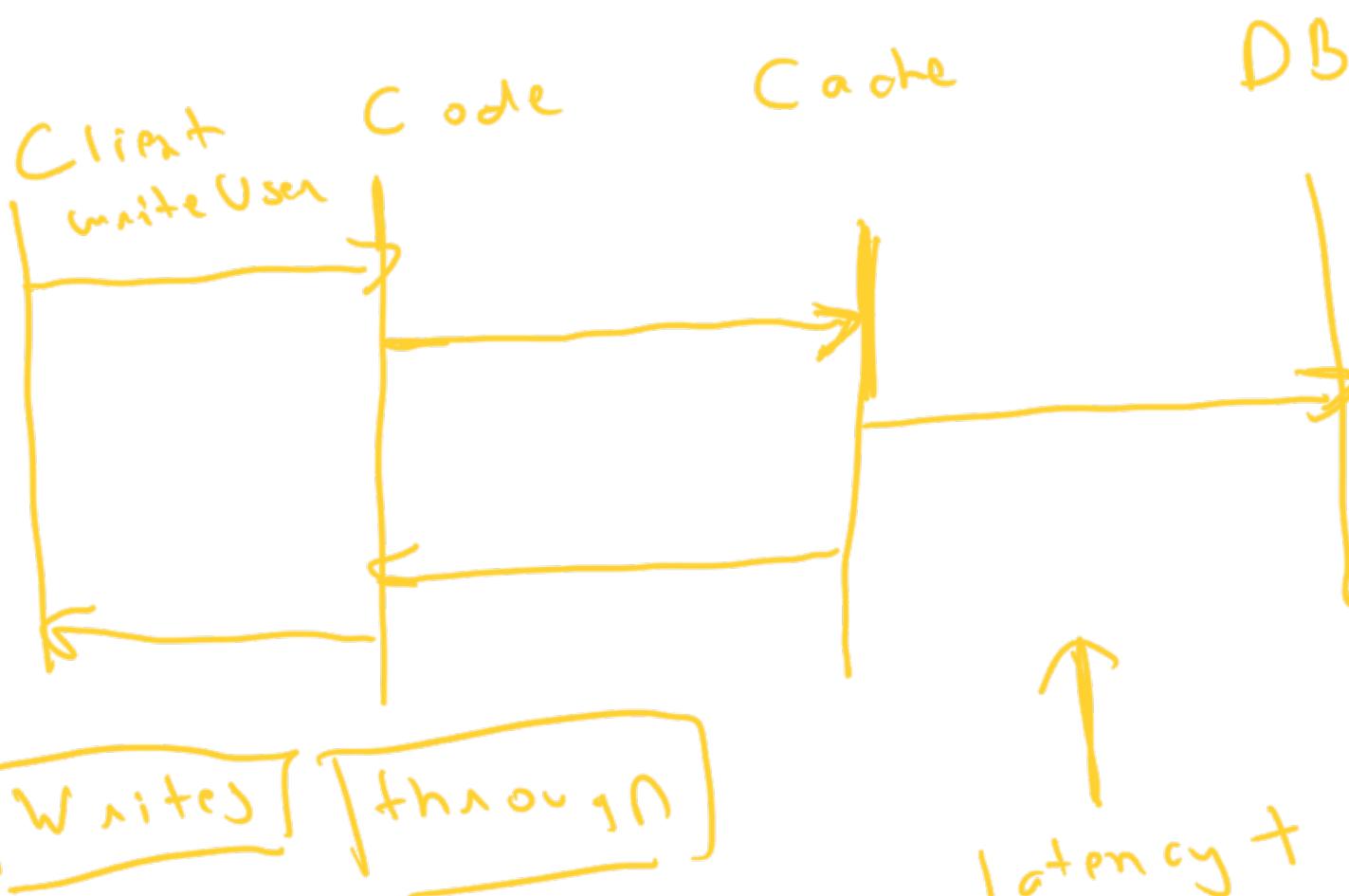
Separation of concerns



- through vs
- smart vs dumb caches
- latency → request has to be within on the DB

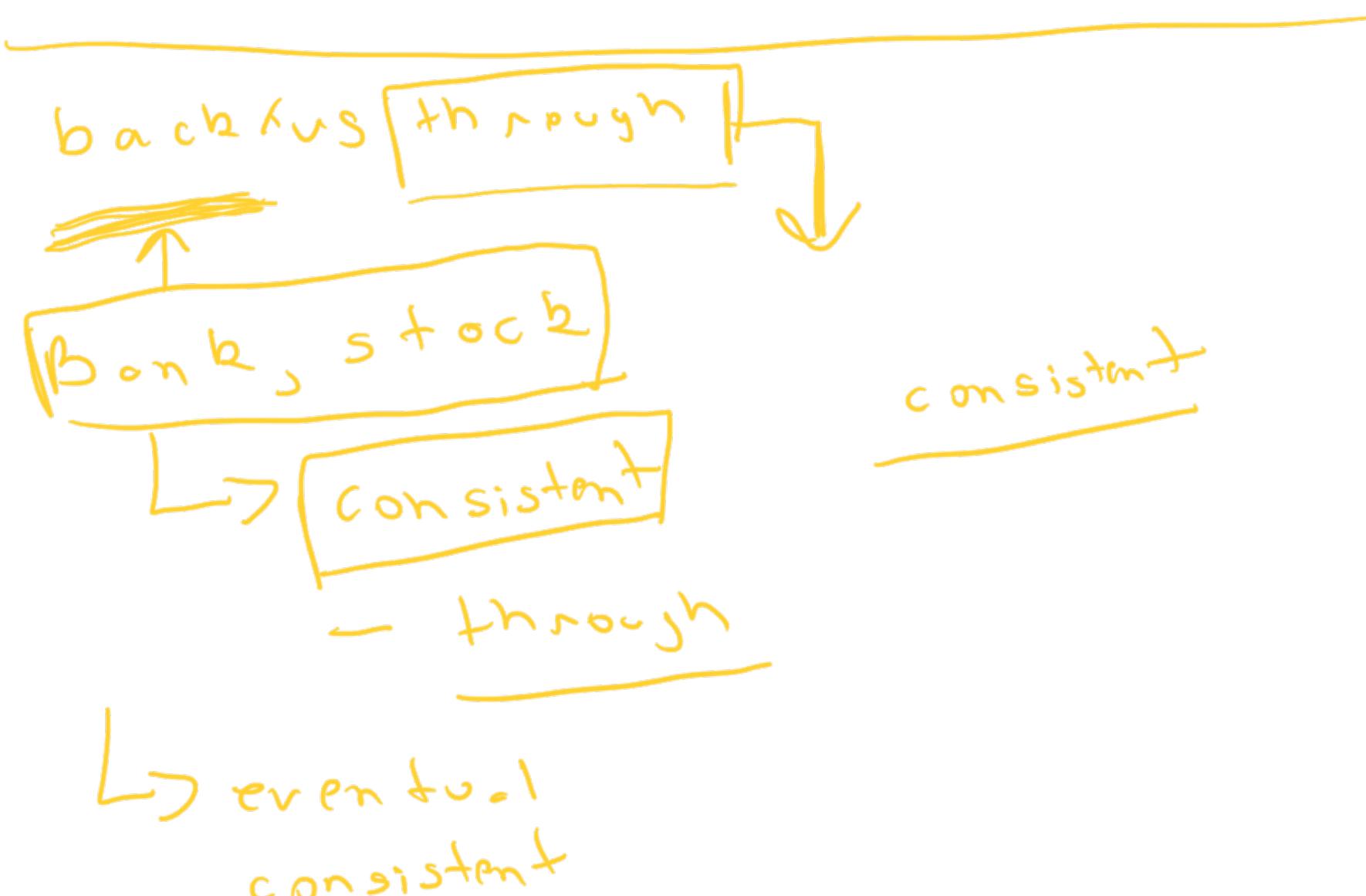
Ban b's

Can we write to the cache later



Can the cache store the data in DB



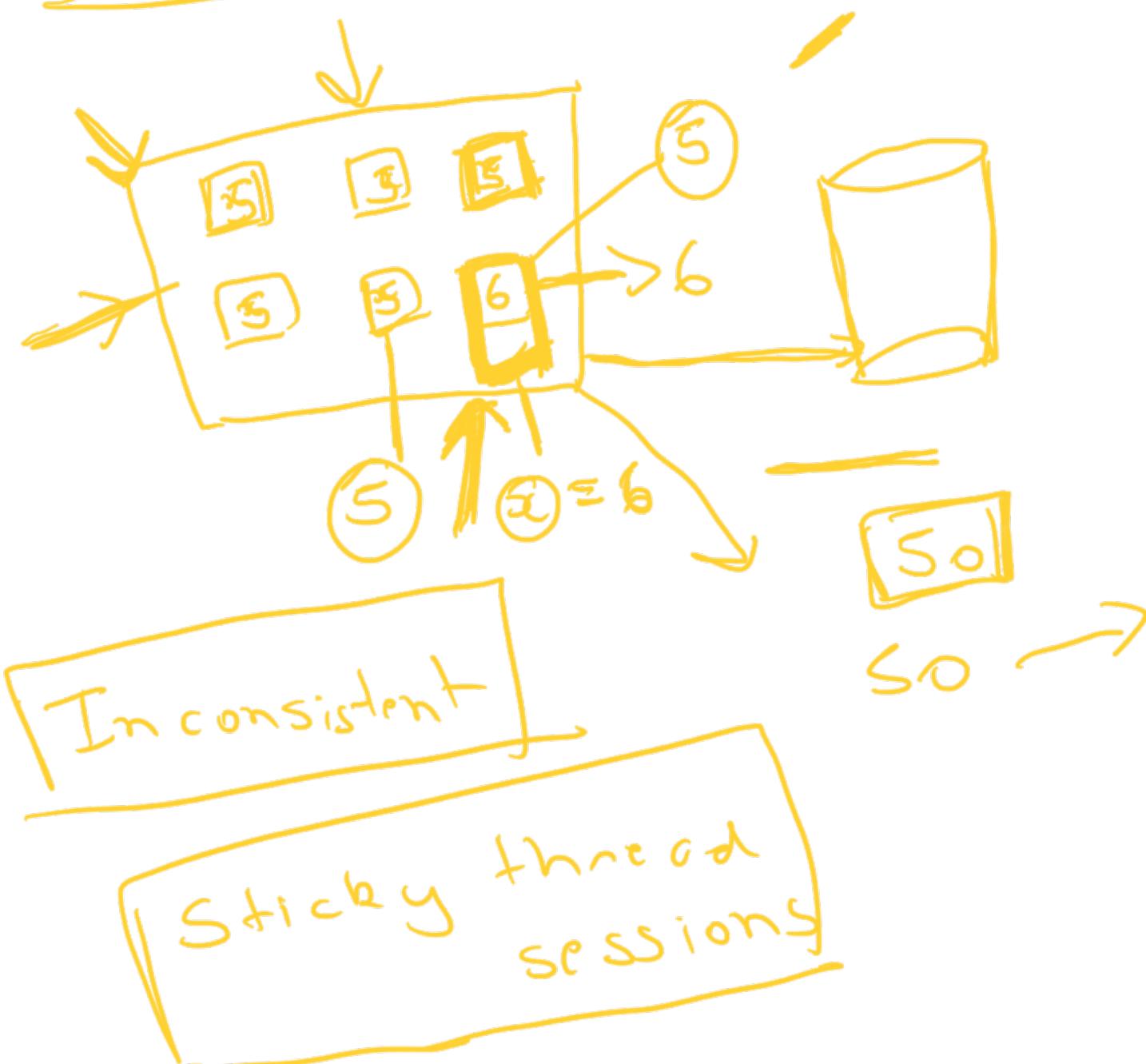


5:58 | 6:03

- Cache-aside
- n/w through
- n/w back
- **Distributed**
- consistency back
- ↳ rotation

--> EVICTION

→ OOP design for cache



- Read Your Own Write

- Consistent Hashing

Cache =



Cache =



(1) LRU - Least Recently used

LFU = Least frequently used

- Frequency
1 = 10 min

2 = 100 min

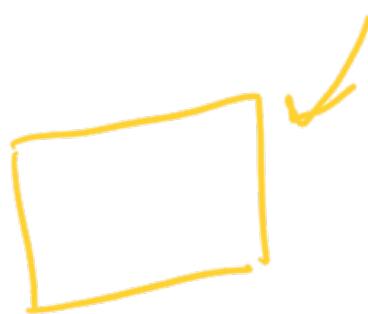
3 = 1000 min

4 = 10000 min

10000

Data - 10000

Distributed



→ users

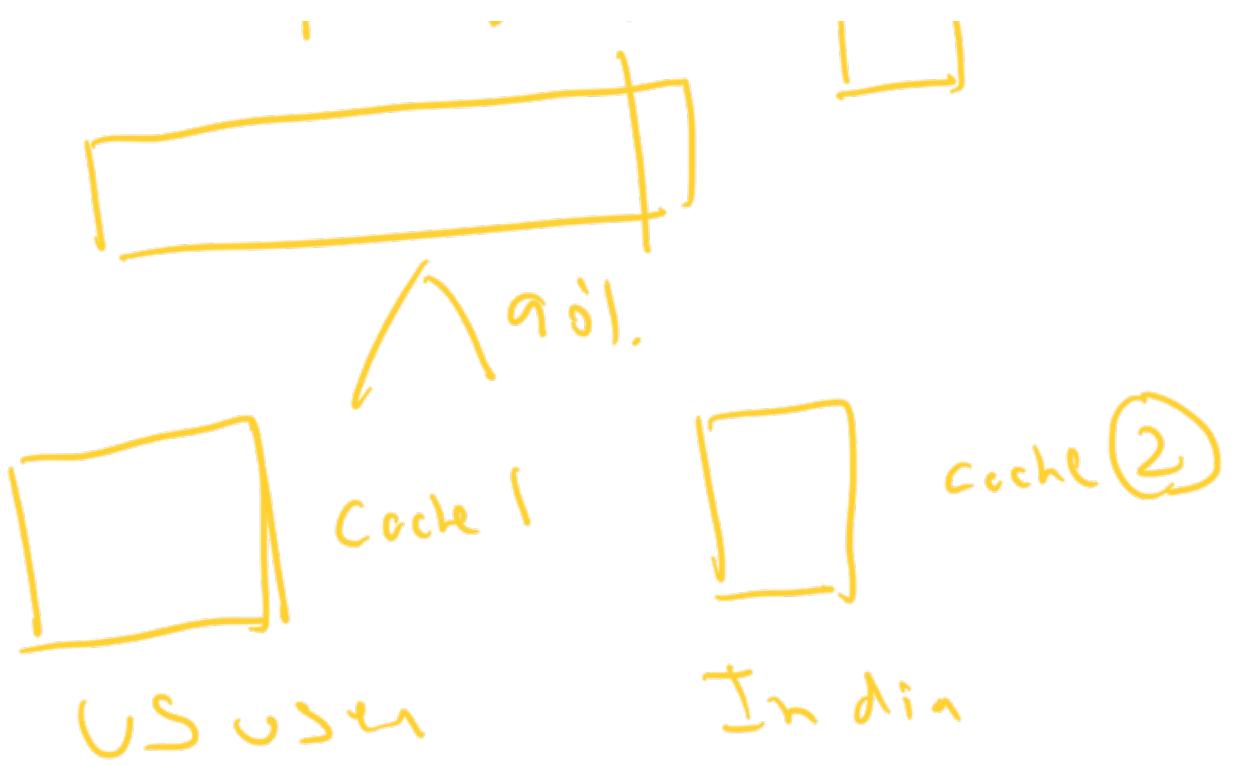
→ moves

→ events

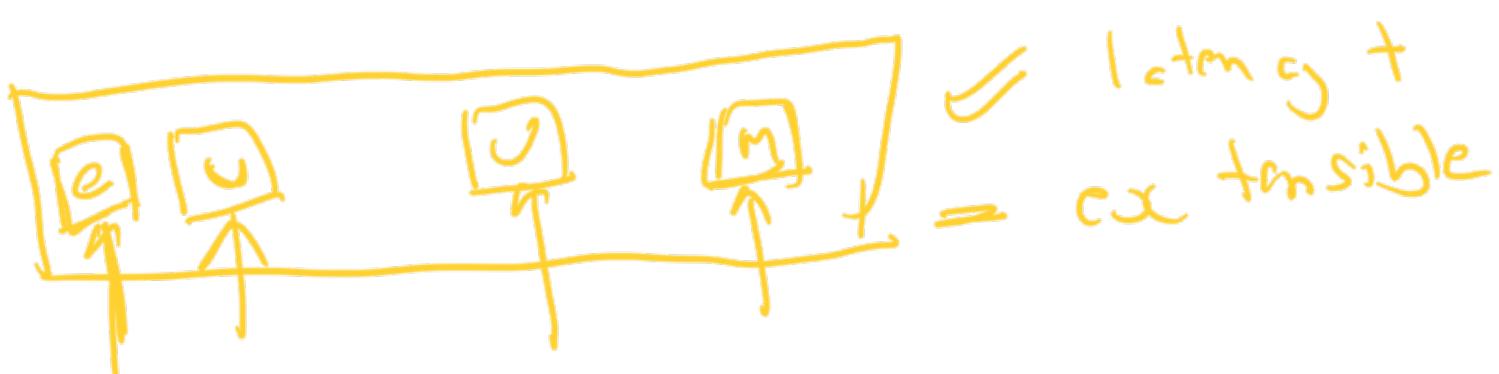
Sharding

Divide our data

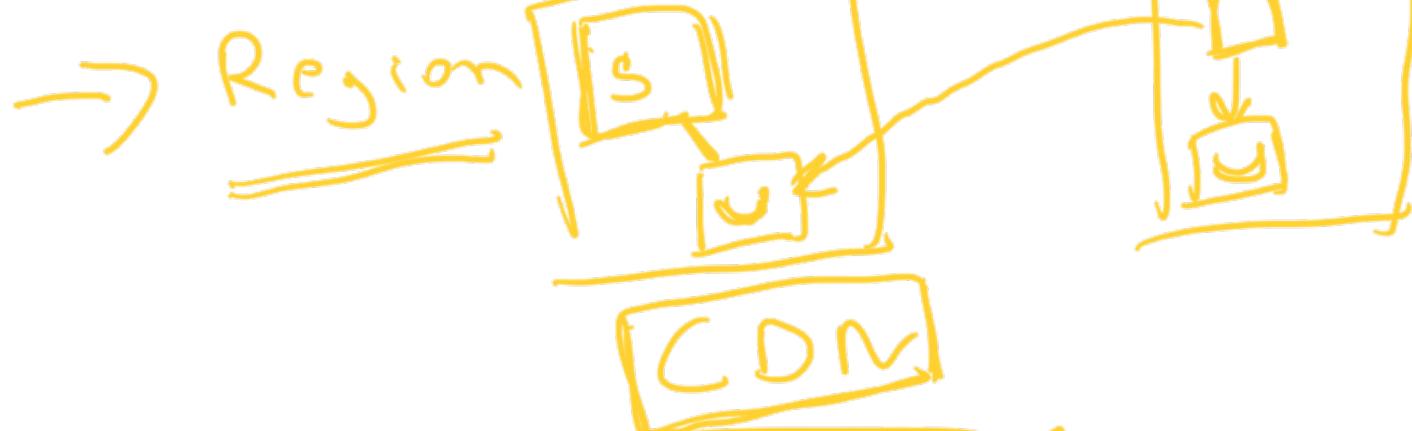




each cache will be served for a hit



→ Large amount of data



→ Number of request

IM — I cache

- Share your data

- replicate cache

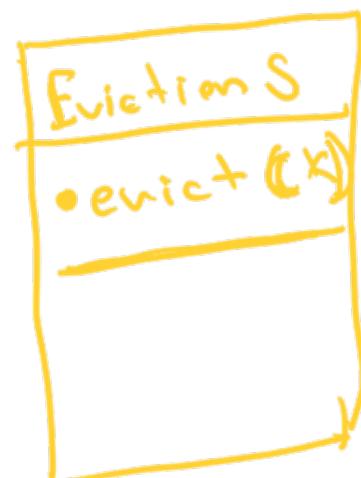


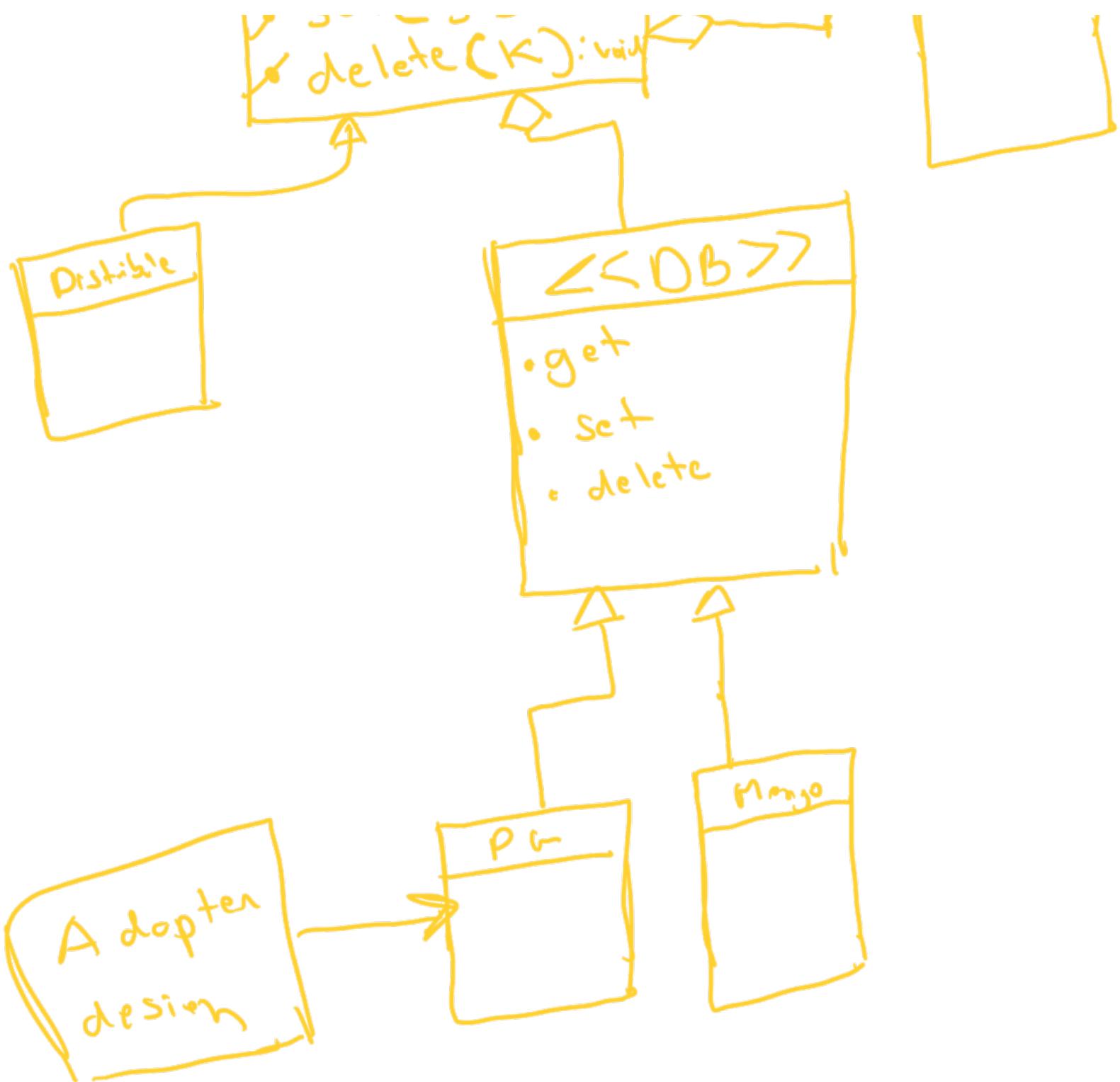
Class Diagram of a Distribute Cache

→ Requirements
— classifications

→ Distributed cache

- = Support multiple write
- = Support multiple evictions
- = Support no locks
- = low latency
- = Partitioned / sharded

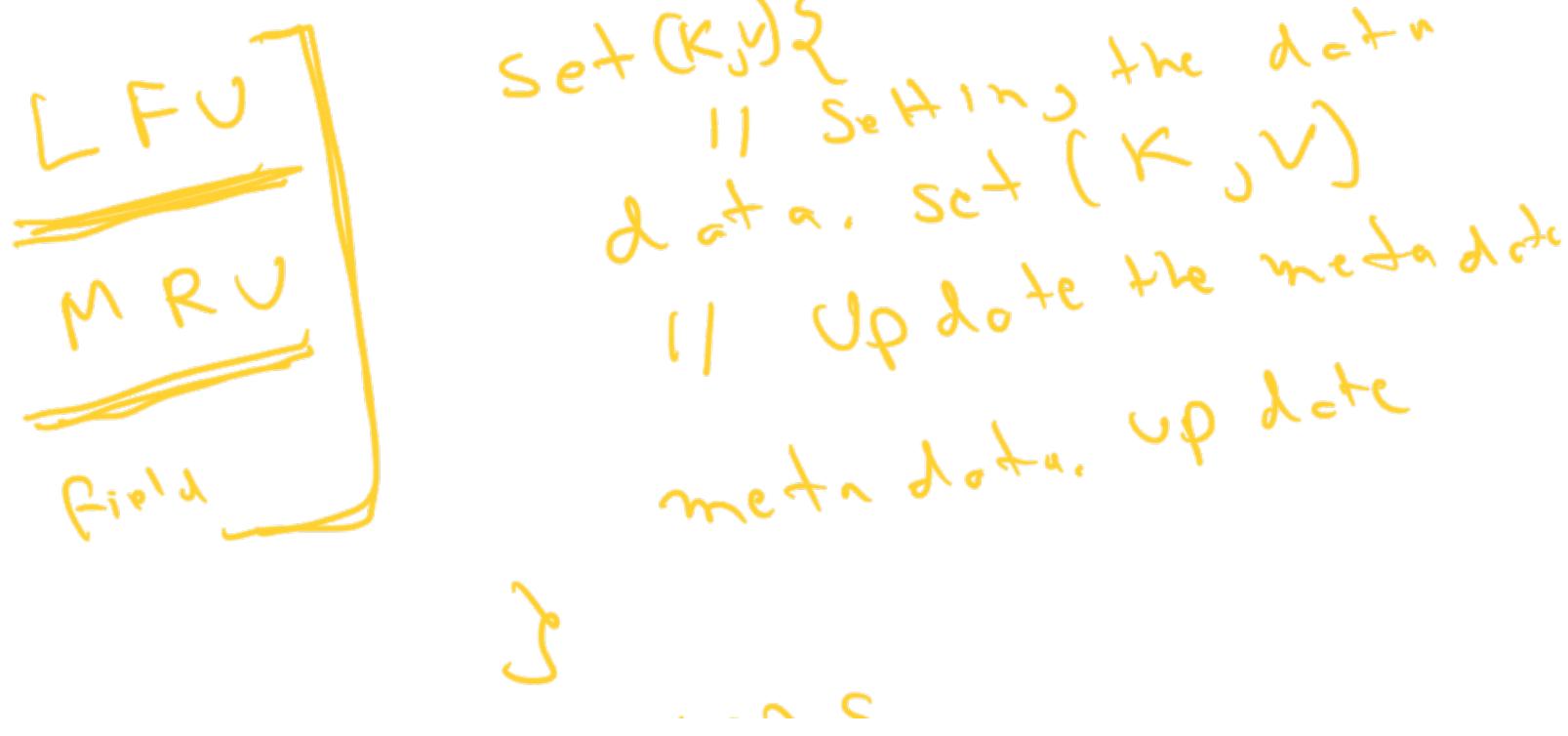




Q

"Key": "Value", Created, Updated
Access At

"Key": "Value"

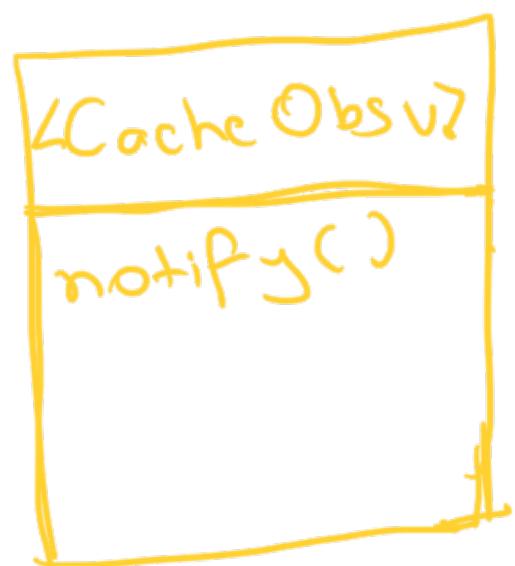


get(k) 2
|| Dataset

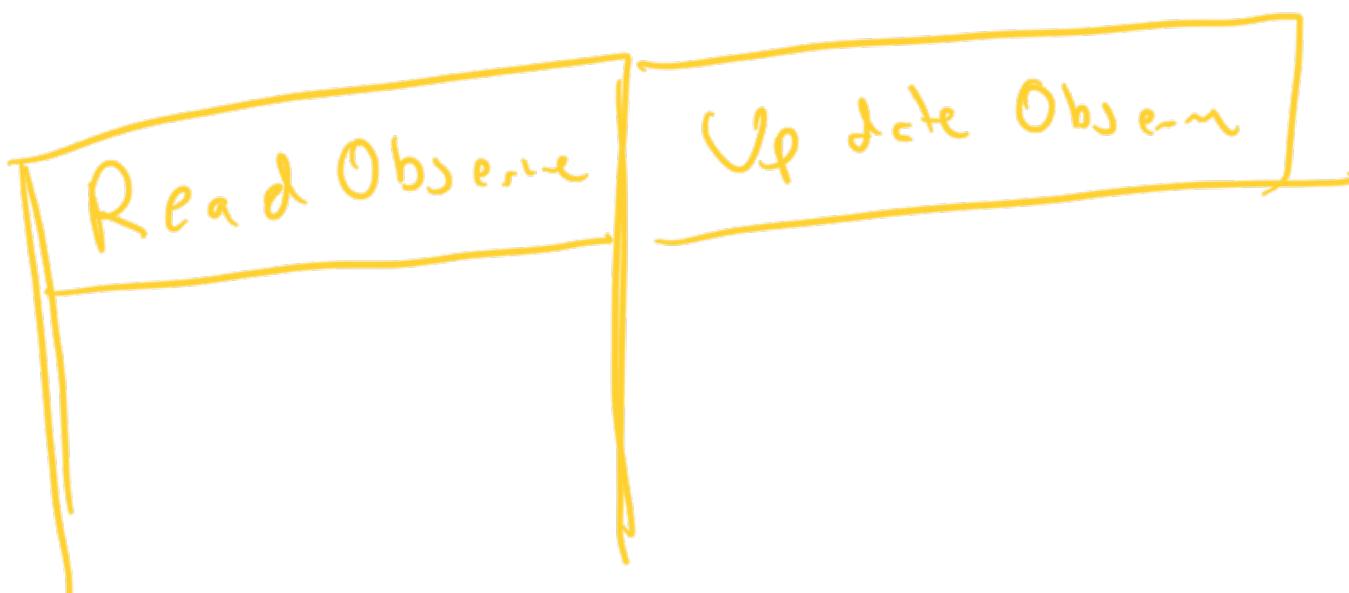
|| Metadate set

3

I want to update
the metadata



→ notified when
a change
perform a metadate
update



Association

is-a
has-a
...