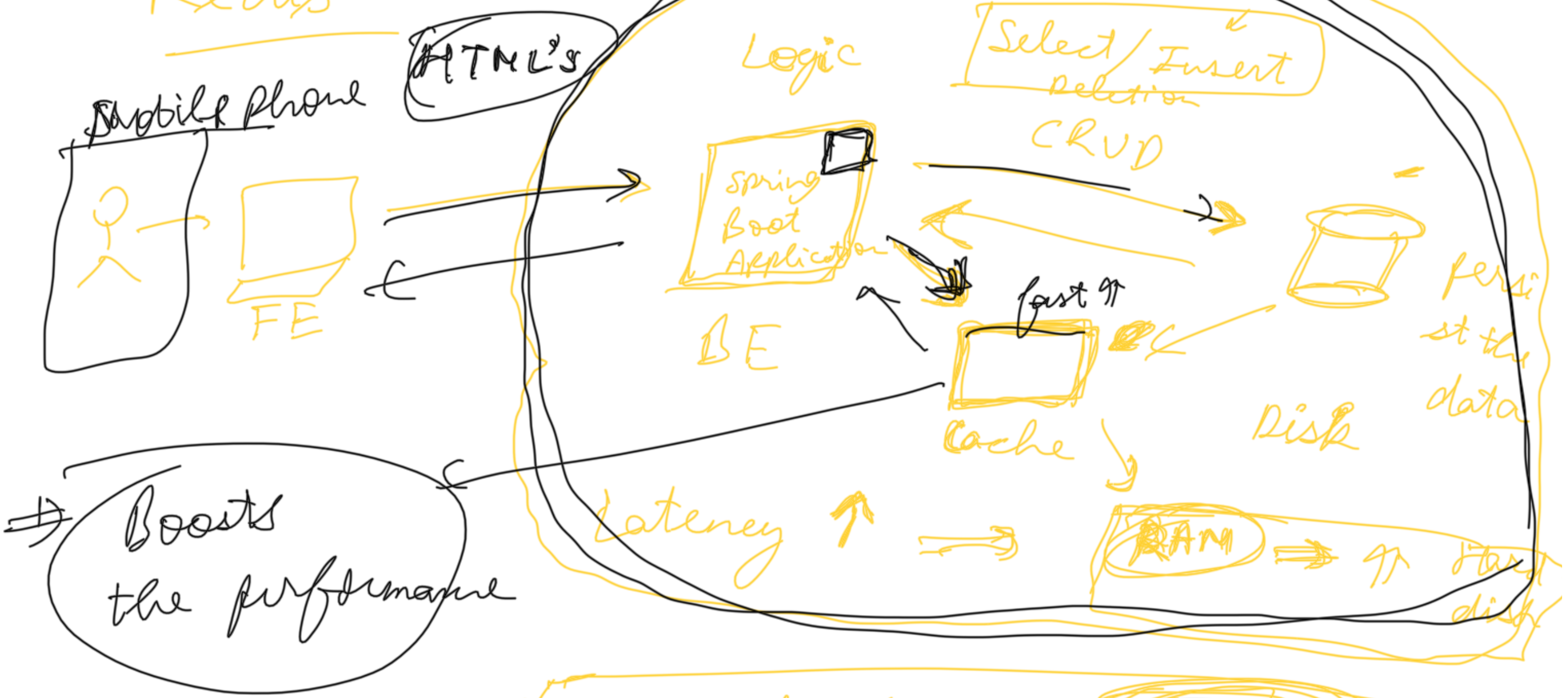
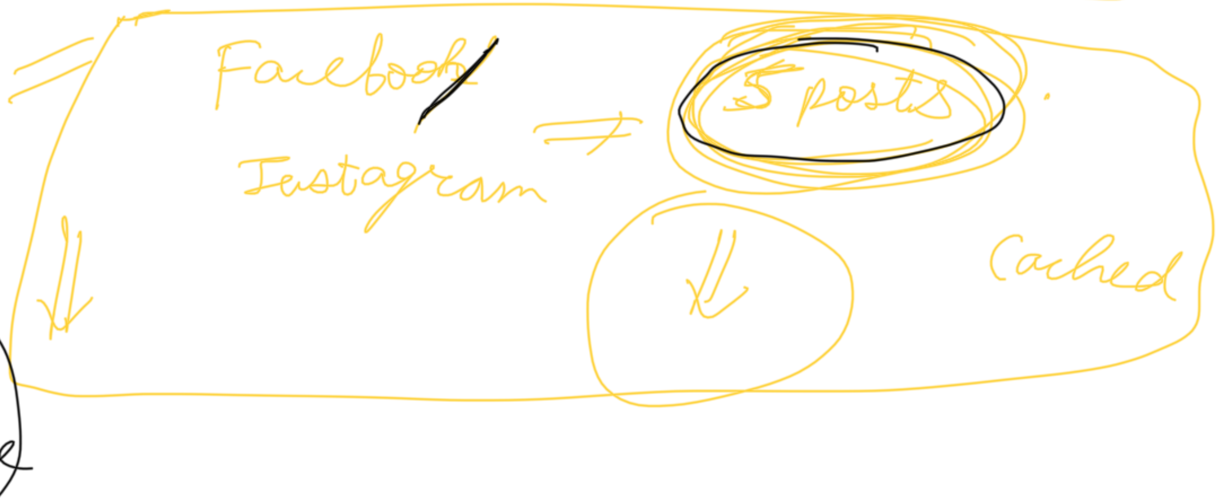
Redis →



⇒ Boosts the performance

⇒ Reduces latency

⇒ Better customer experience



~~At~~ ① Populating the cache →

② Cache eviction (FIFO, LR

⇒ Persistence is very important
↳ DB is reqd.

We might loose out on info.

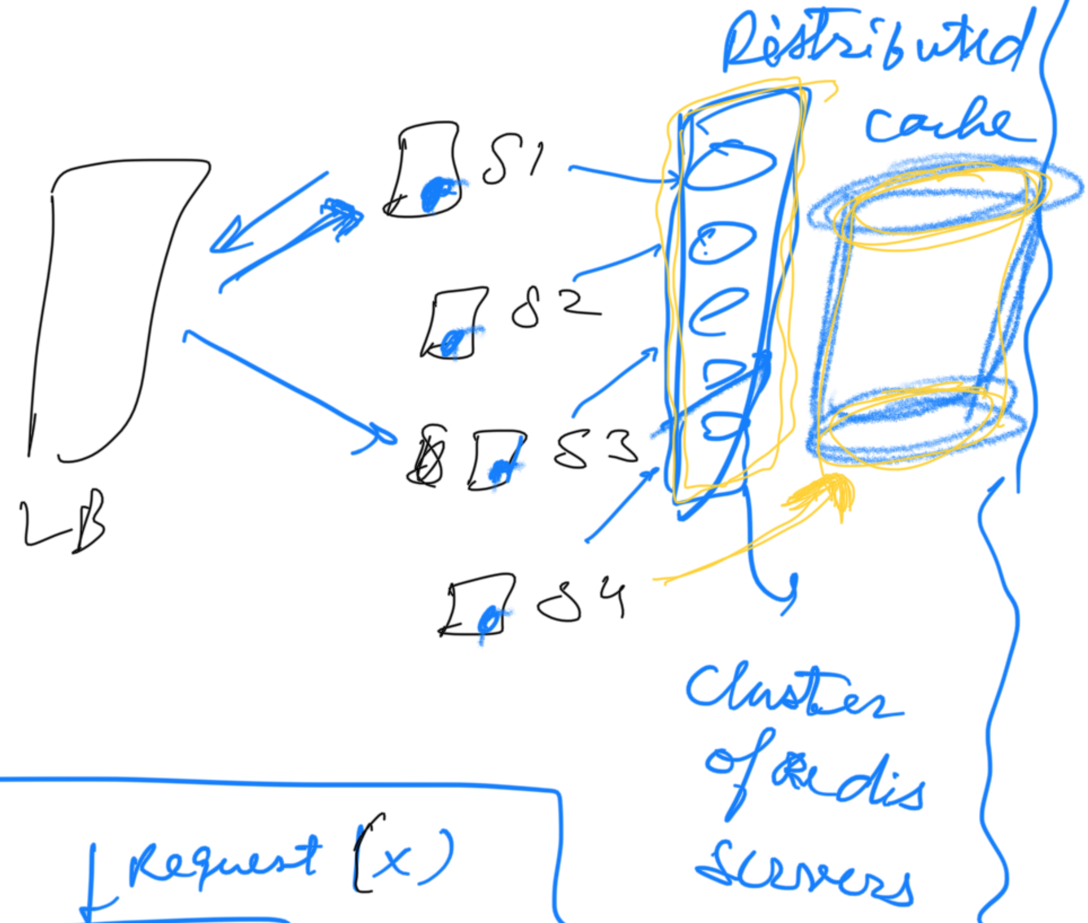
⇒ But interaction with DB is very slow

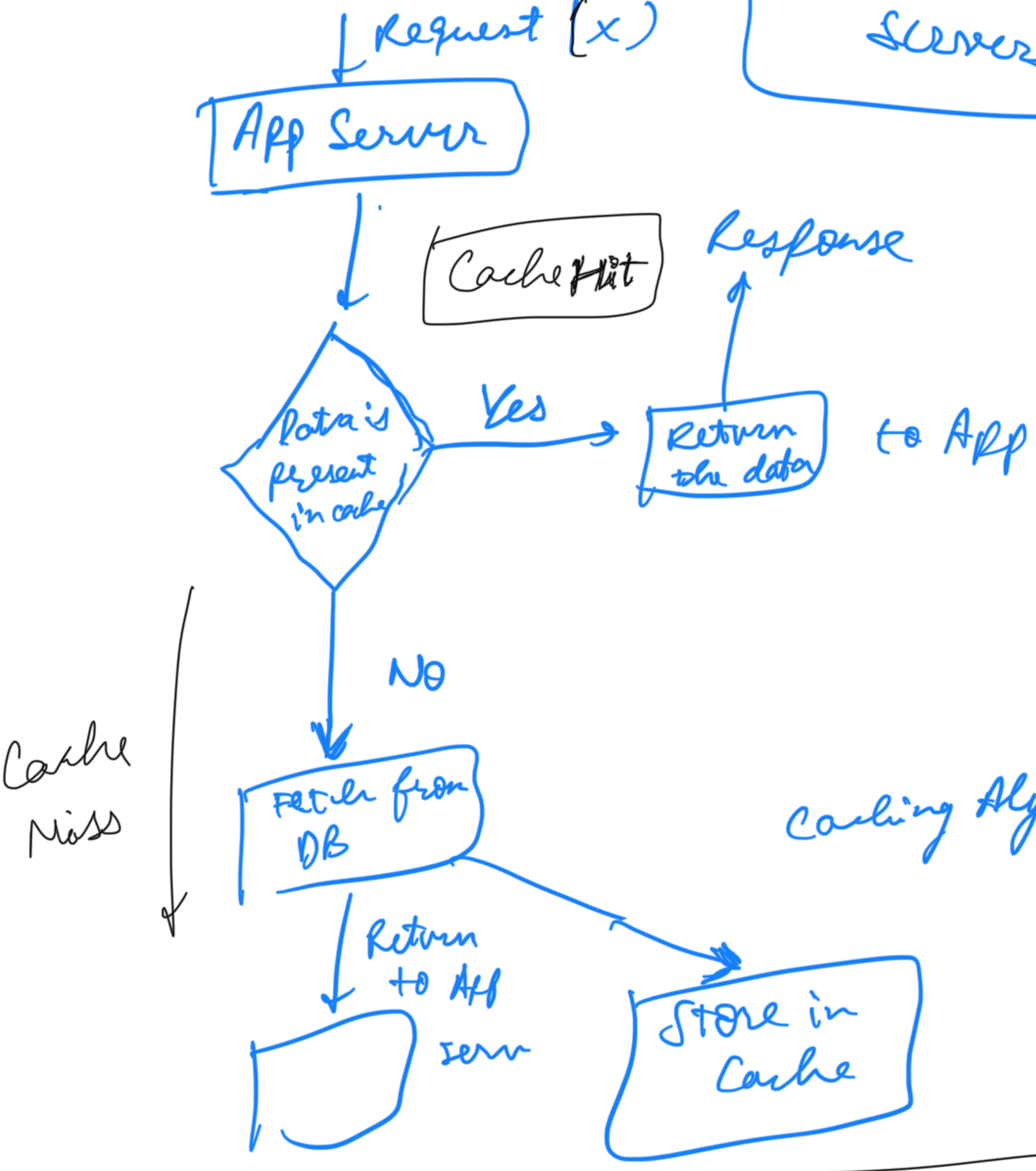
we can optimise using indexing, normal-isation.

But still there will be some latency.

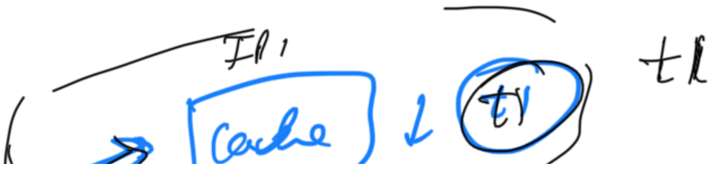
In Memory Cache

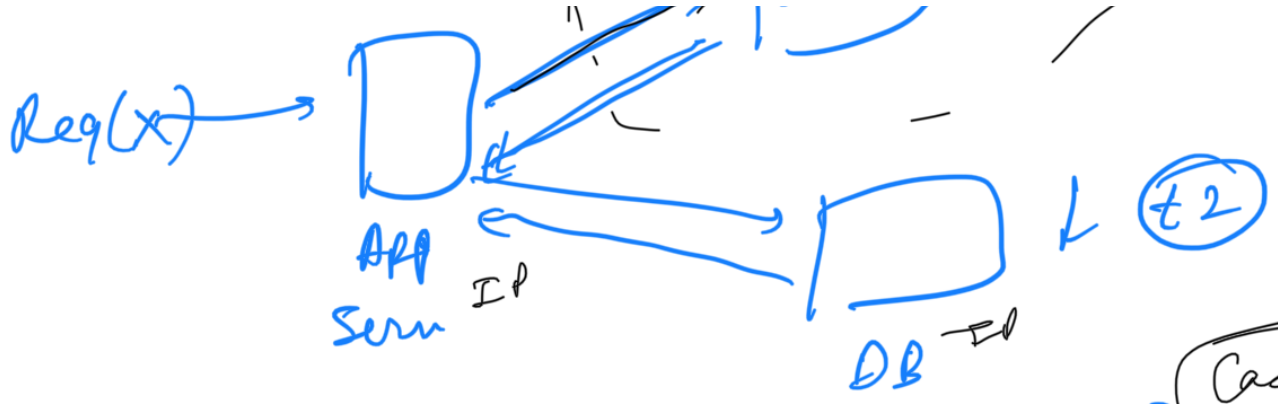
→ Distributed Cache → Redis Cluster





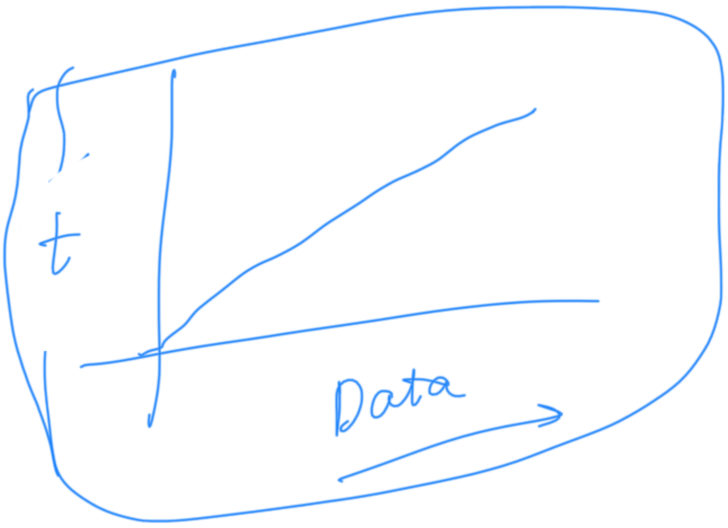
Caching Algo's

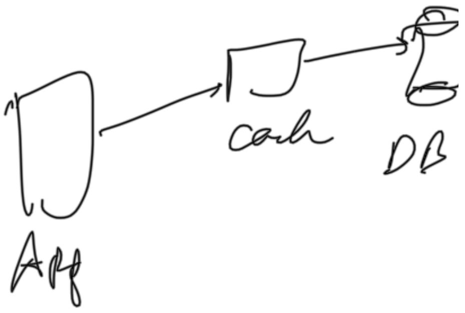




Cache Hit

$t2 >> t1$





Cache Hit → $t1$

Cache Miss → $t1 + t2$

$t2 \rightarrow T_{ret\,DB} + T_{save\,cache}$

Data is not present in cache and it's been retrieved from DB

$t1$

$t1+t2$

$T_{ret\,DB} = t2 - T_{save\,cache}$

$<t2$

$T_{ret\,DB} < t2$

$t2>>t1$

70% of Cache hits → its beneficial to have caches

to store frequently used

Caches → Used to

info in a faster manner.

→ Improves overall system performance.

→ They are limited in size

↳ full → We need to erase some data for newer data

Cache eviction strategies

LRU,
LFU,
FIFO,
⋮

Cache's are not helpful always →

1. Cache Hit needs to be > 70%
2. Balance

Cached
transaction

Consistency > Responsiveness

stores information

Cookie ~> file that is

that identifies an

HTTP Cookies and cache

↓ To enhance ~~customer~~ user experience and streamline the web loading process

Cookies → key value pair stored on FE ?!

FE can pass this to BE in API calls so that BE can use that

Redis → Type of NoSQL DB

But it has some limitations

↓

→ open source

→ in memory datastore

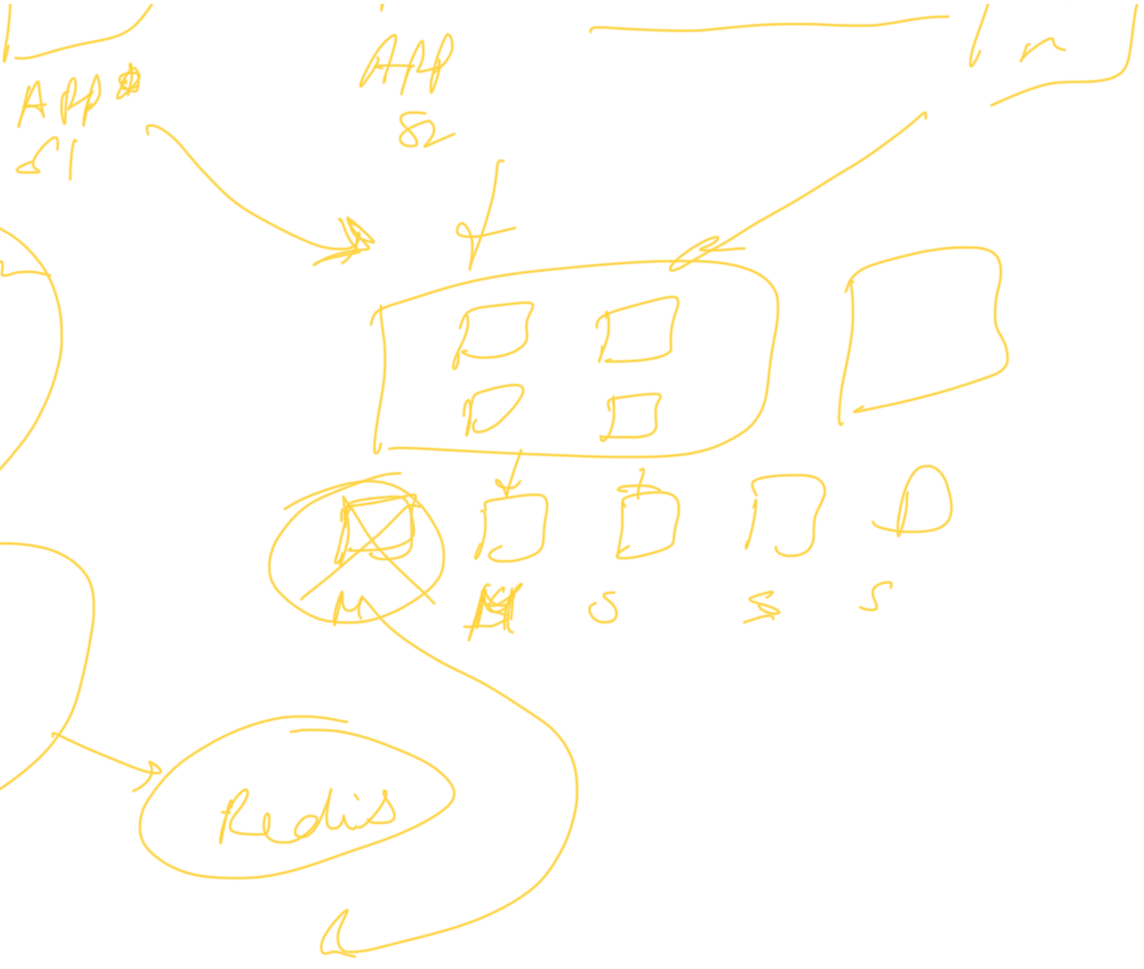
→ caching

Distributed Cache + persistence → ~~Redis~~ Redis Cluster

IP1
IP2
IPn
APP

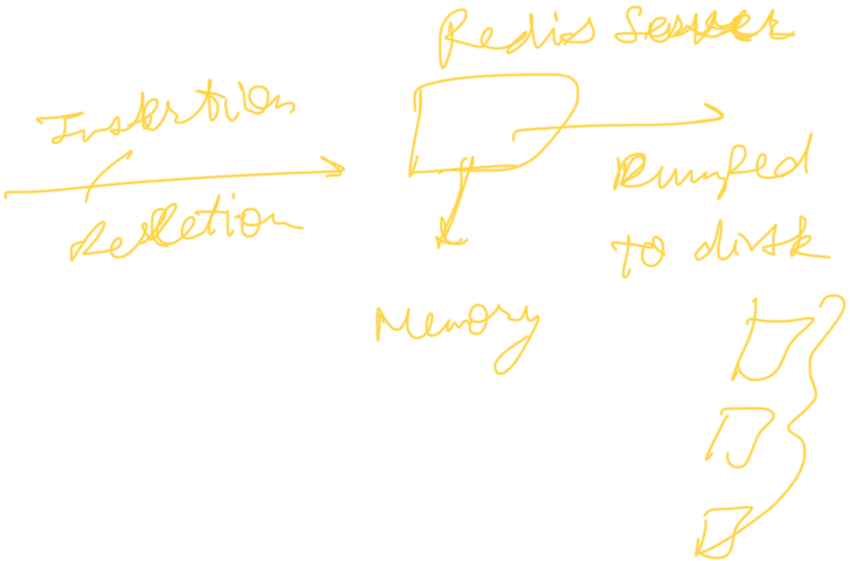
Centralisation + Replication

Distributed Cache → Redis

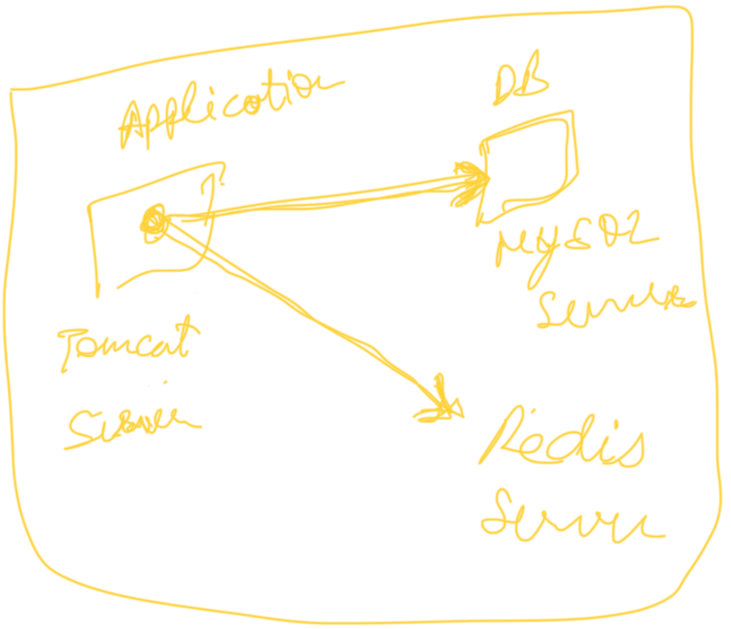


(S1)
will be lost

Redis Client



Redis Cluster / Redis



Key-Value Store

Strings
~~Strin~~
Strings
List

tation and machine

① Own implementation...

② Use Managed Services → Redis Labs

→ lifo

→ fifo

MySQL

MySQL-root ⟷ redis-cli

Redis

"," string

Key — Value

String

→ string

→ list [s1, s2, s3 - -]

→ set [s1, s2, s3, - -]

→ hashes : { "name": _ _ }

Student
String name;
String rollNo;

↑ Stringify

JSON

{ "name" : "Aakash"

Rollno : 2K16/COE/03