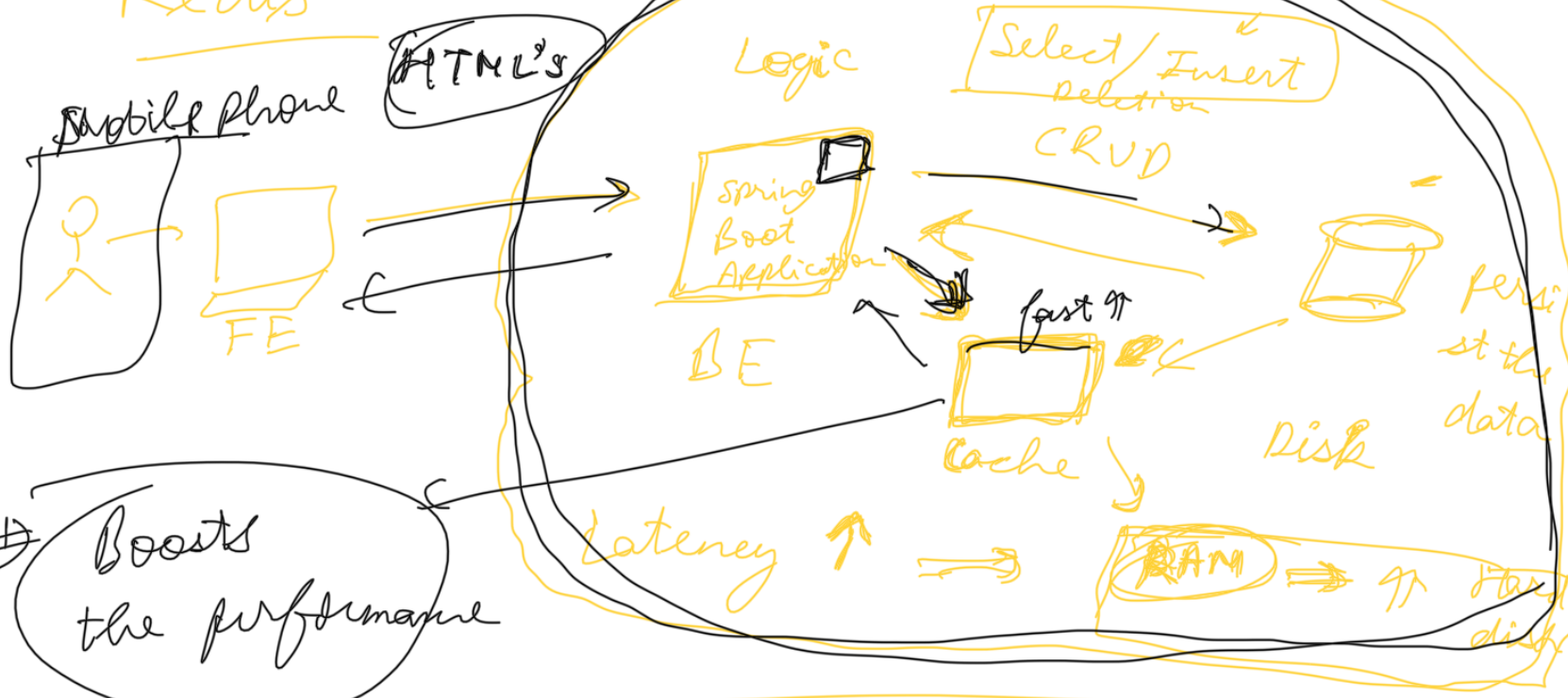
Redis →



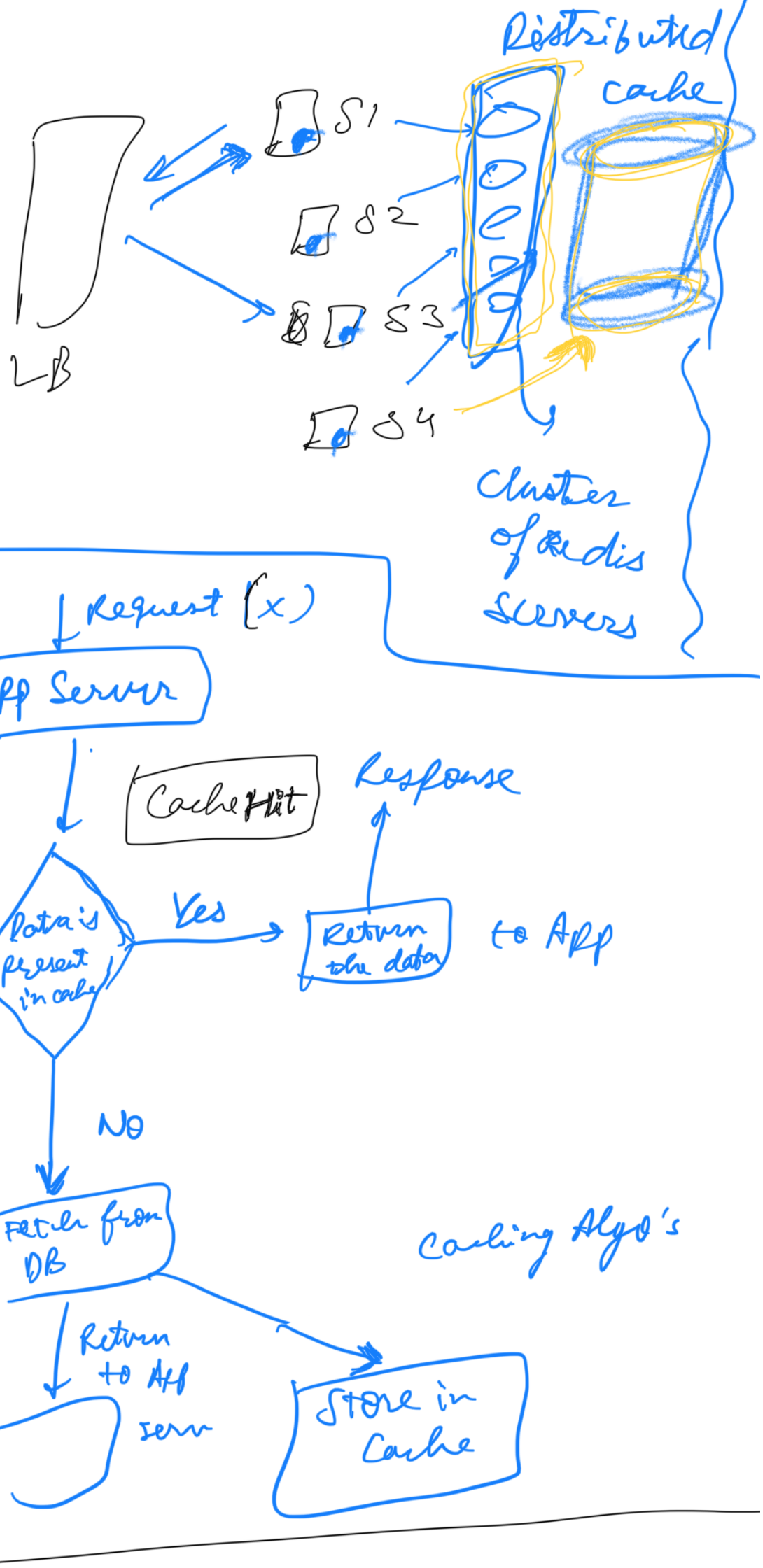
② Cache eviction (FIFO, LR
→

⇒ 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



Request (x)

App Server

Cache Hit

Response

Data is present in cache

Yes

Return the data to App

No

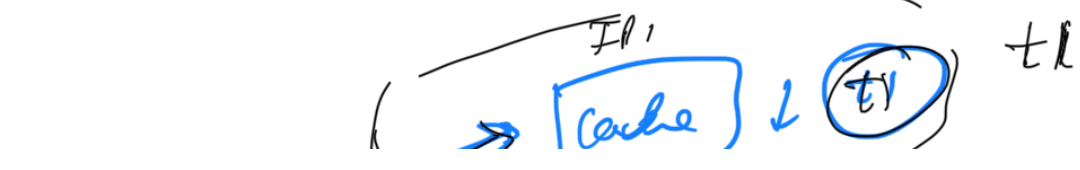
Cache Miss

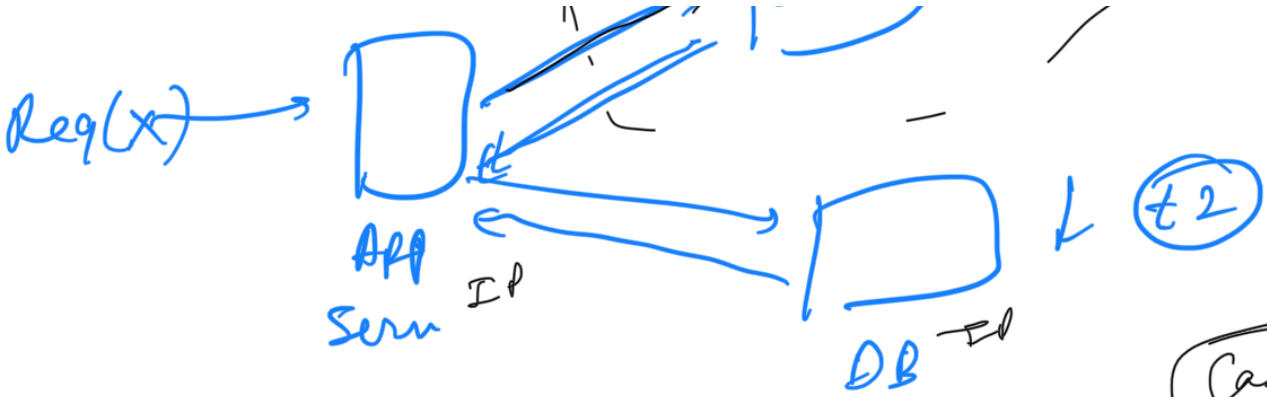
Fetch from DB

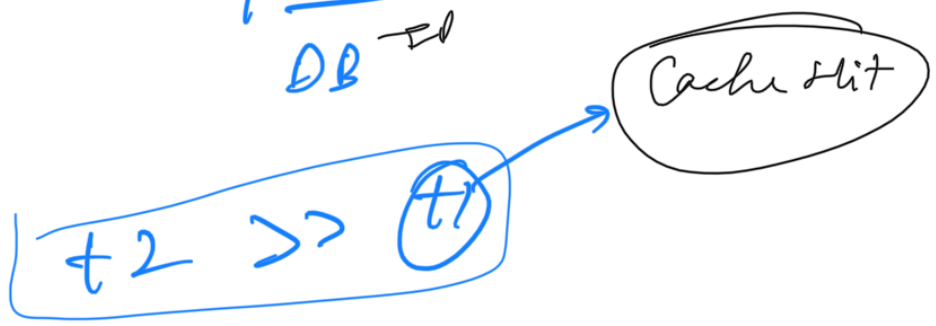
Caching Algo's

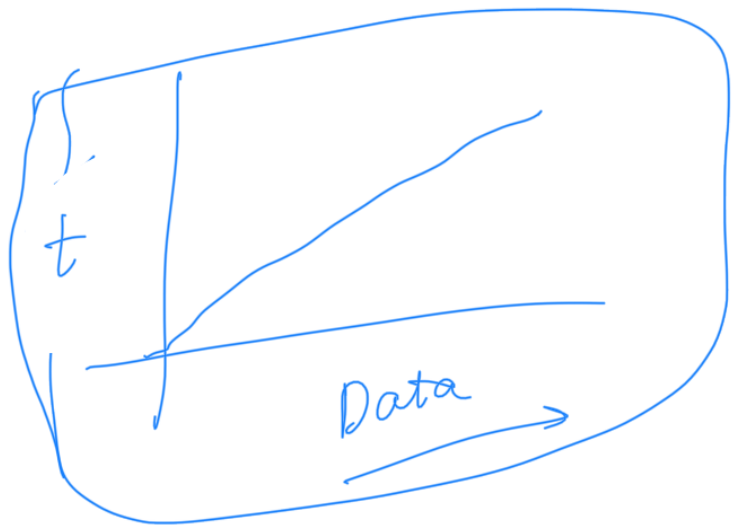
Return to App

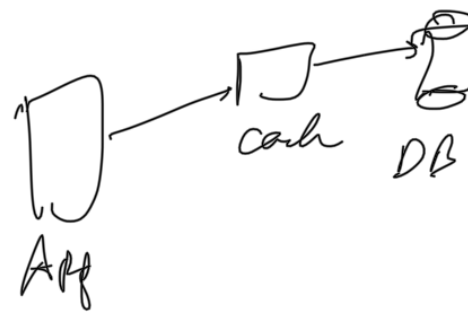
Store in Cache

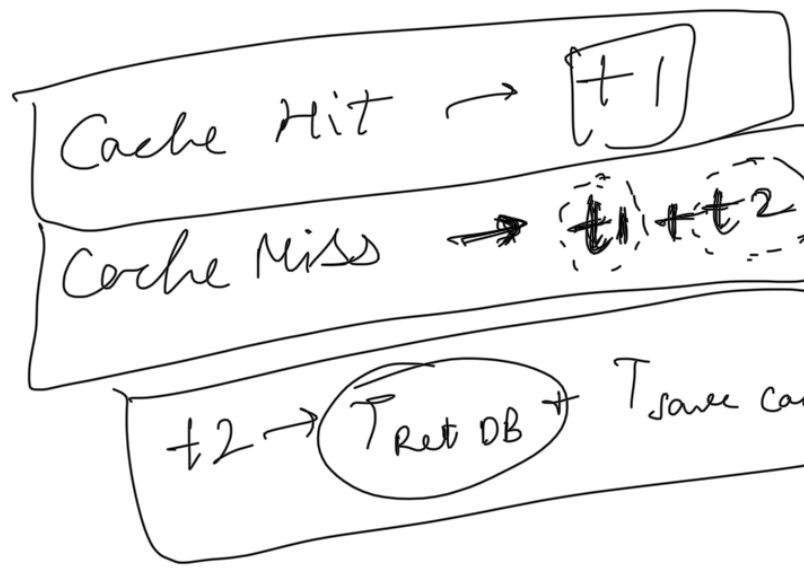












Data is not present in cache and it's been retrieved from DB

t1 + t2

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

$T_{ret\ DB} < t2$

< t2

t2 >> t1

70% of Cache hits → it's beneficial to have caches

to store frequently used

Caches → need to [illegible] 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, (

→ [illegible]

Cache's are not helpful always →

① Cache Hit needs to be > 70%

② Balance

Cached

[illegible]

Consistency > Responsiveness

[illegible] stores information

Cookie → file that stores 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

App S1

App S2

Centralisation + Replication

Distributed Cache

Redis

will be lost

(S1)

Redis Client

Insertion

Deletion

Redis Server

Memory

Dumped to disk

Redis Cluster / Redis

Key - Value Store

Strings

Strings

Strings

and machine

① Own implementation

② Use Managed Services → Redis Labs

→ lifo
→ fifo

MySQL MySQL - root ⟷ redis-cli

Redis

Key — String — Value

→ String
→ list [s1, s2, s3 - - -]
→ set [s1, s2, s3, - - -]
→ Hashes : { "name" : - - - }

Student
String name ;
String rollNo ;

↑ Stringify
JSON

{
"name" : "Akash"

Roll No : 2K16/COE/03

}