Agenda.

↳ REDIS : Cache

Redis (REmote DIctionary Server) is an open-source, in-memory data store that's used as a database and cache. It's known for its speed, reliability, and performance

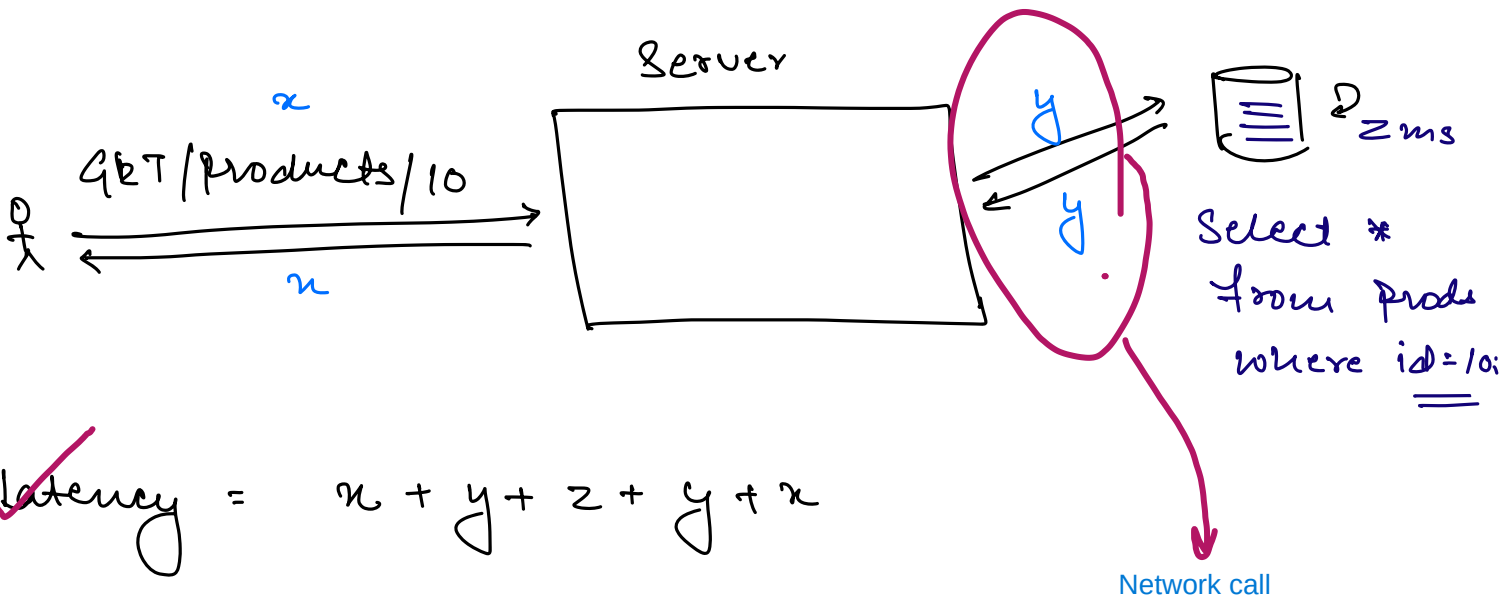
→ What is Caching ?.

→ Why caching is required ?

→ Where we can integrate cache in our Product Service ?.

⇒ Search API.

getProductById(id)

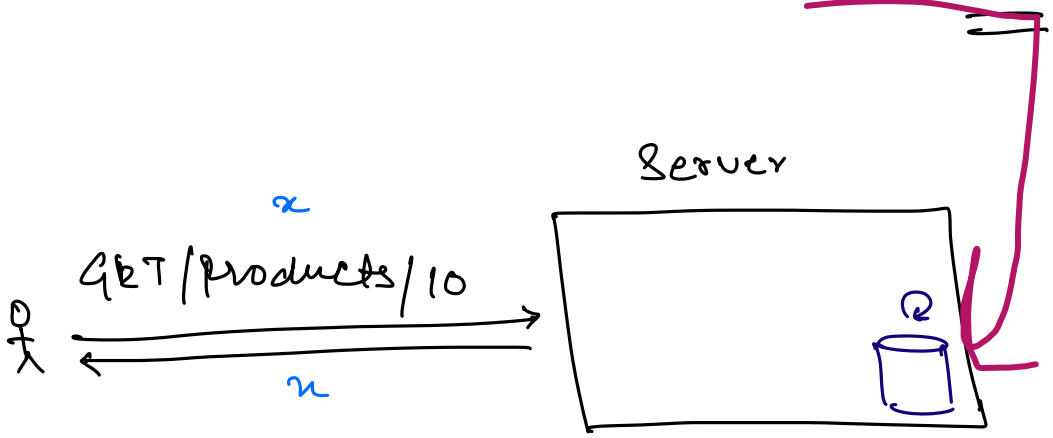


latency = x + y + z + y + x

= (2x + 2y + 2) ms.

For get request, what is the latency time, it takes x time from user to server and from server to db it takes y time, z time while applying the search query in the db.

=> If we install DB within the server, we'll be able to reduce the N/w call.



Latency = x + 2 + x

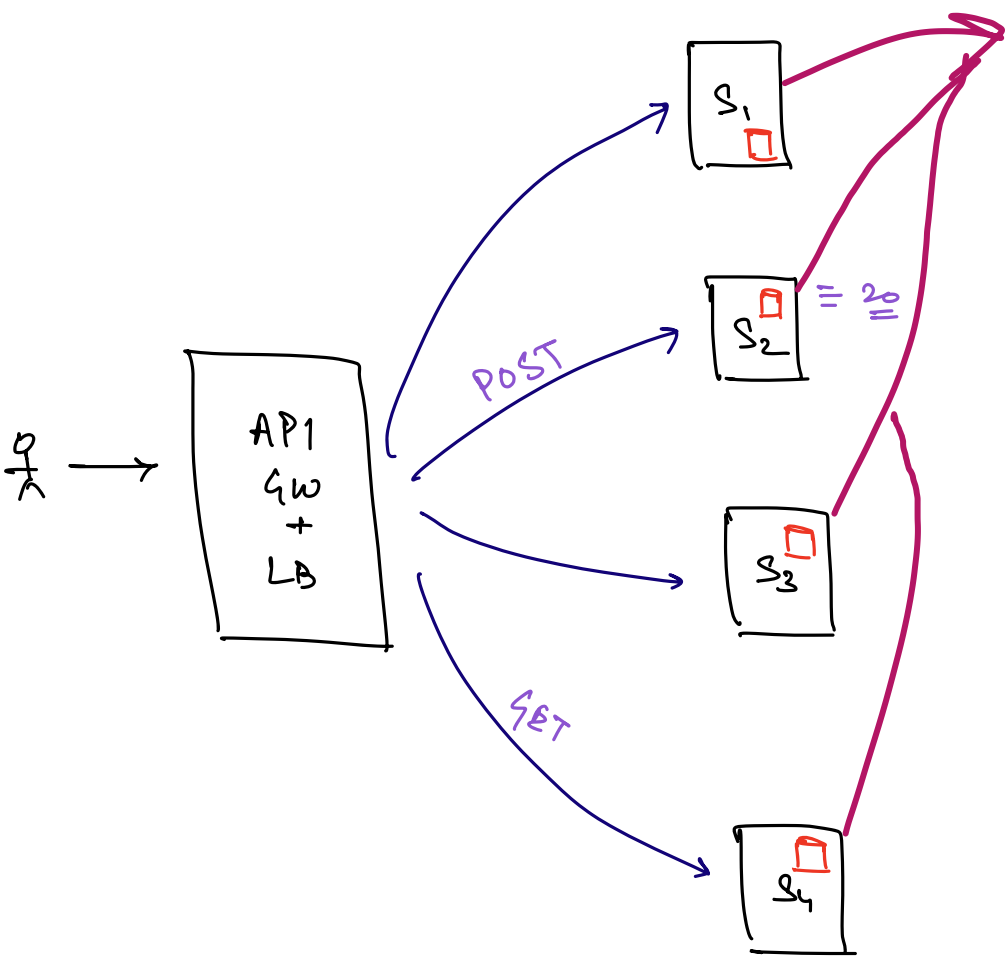
= 2x + 2.

We can reduce the latency by just moving the db machine in the server itself. But if we store db into server, db may carry 4TB, 8Tb of data coz of which your application server might become heavy, so if your app server goes down then your db also goes down -> so there is a tight coupling b/w both of them. If some other code is running at the background, with server down, your code or other services will also go down

=> Distributed.

Database sharding is a technique that splits a large database into smaller parts, called shards, and stores them across multiple servers. This allows for more efficient storage and processing of large amounts of data.

So Db is not present at one place, it many be distributed -> sharding so the question is, should we place all the shards on the server -> this is practically not possible



Application server

There are many app servers behind LB, so suppose I made a POST req and product got created in S2 server (db inside it) and a GET req so is it possible that this request can go to any server lets suppose to S4, since product was created in S2, so I might get NULL value in return

Sync issue.

⇒ Inconsistency.

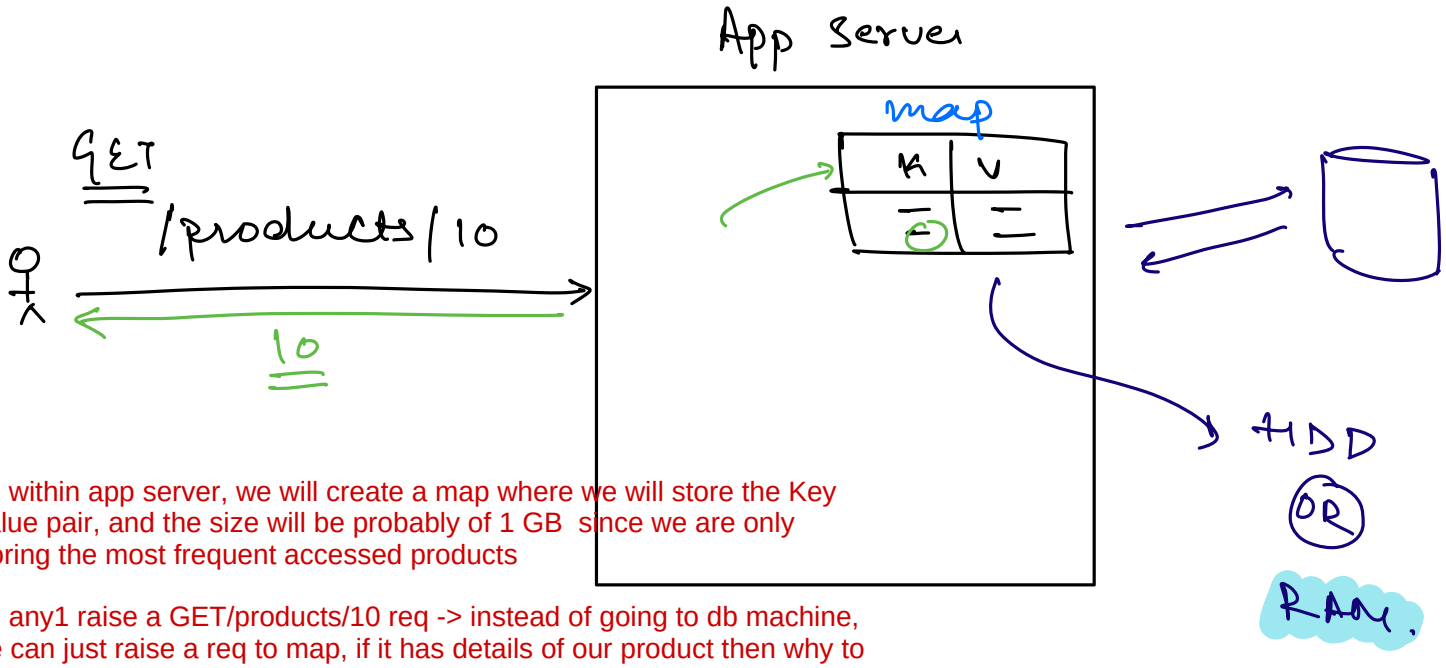
Will need synchronization

Stateful (Vs) Stateless.

DB within server. Server is also aware of the state of the data

Your Db is completely different from your app server . in distributed envt you cannot control #of db machines, caches, message queues, everything should be separate from each other

Instead of storing the whole DB within the App server, We can just store some of the most frequently accessed products.



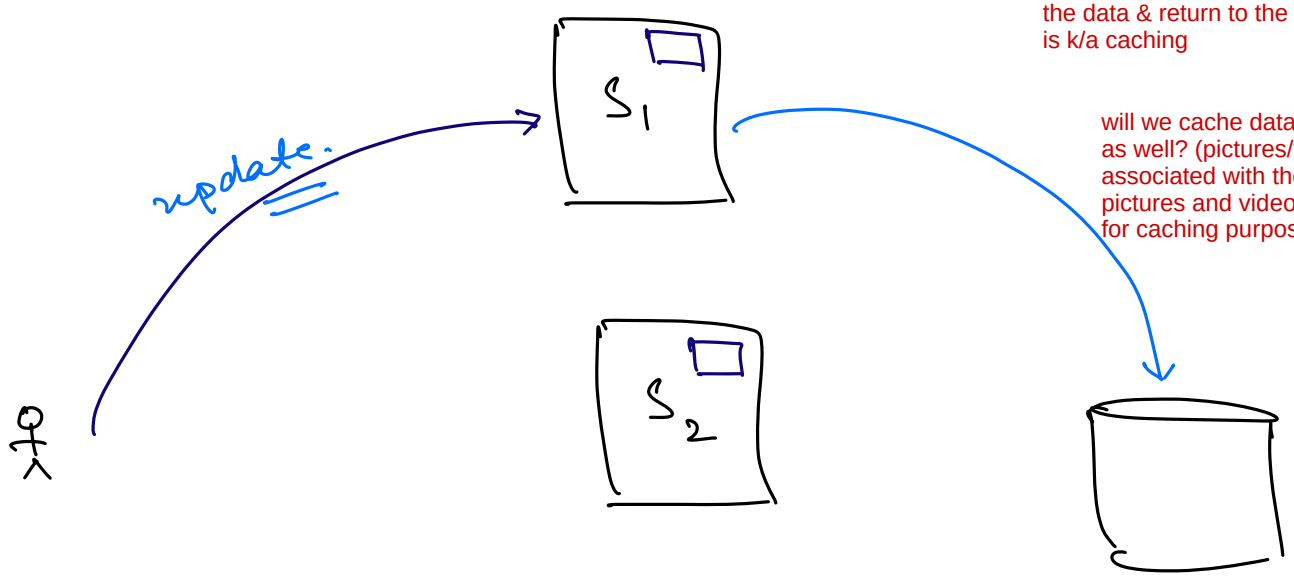
So within app server, we will create a map where we will store the Key Value pair, and the size will be probably of 1 GB since we are only storing the most frequent accessed products

So any1 raise a GET/products/10 req -> instead of going to db machine, we can just raise a req to map, if it has details of our product then why to go to db, just return the product with id =10 from the map itself

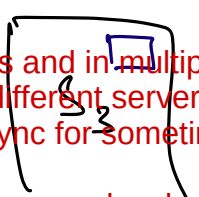
should this map be stored on a Hard disk(HDD) or RAM? RAM coz the speed of data access through RAM is much higher through RAm than HDD.

⇒ Caching

So when you store frequently accessed data near to the app server so that you donot have to make a db call to fetch the data & return to the user ->concept is k/a caching

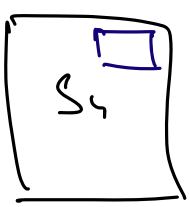


will we cache data from file store as well? (pictures/videos) associated with the product-> for pictures and videos we use CDN for caching purpose

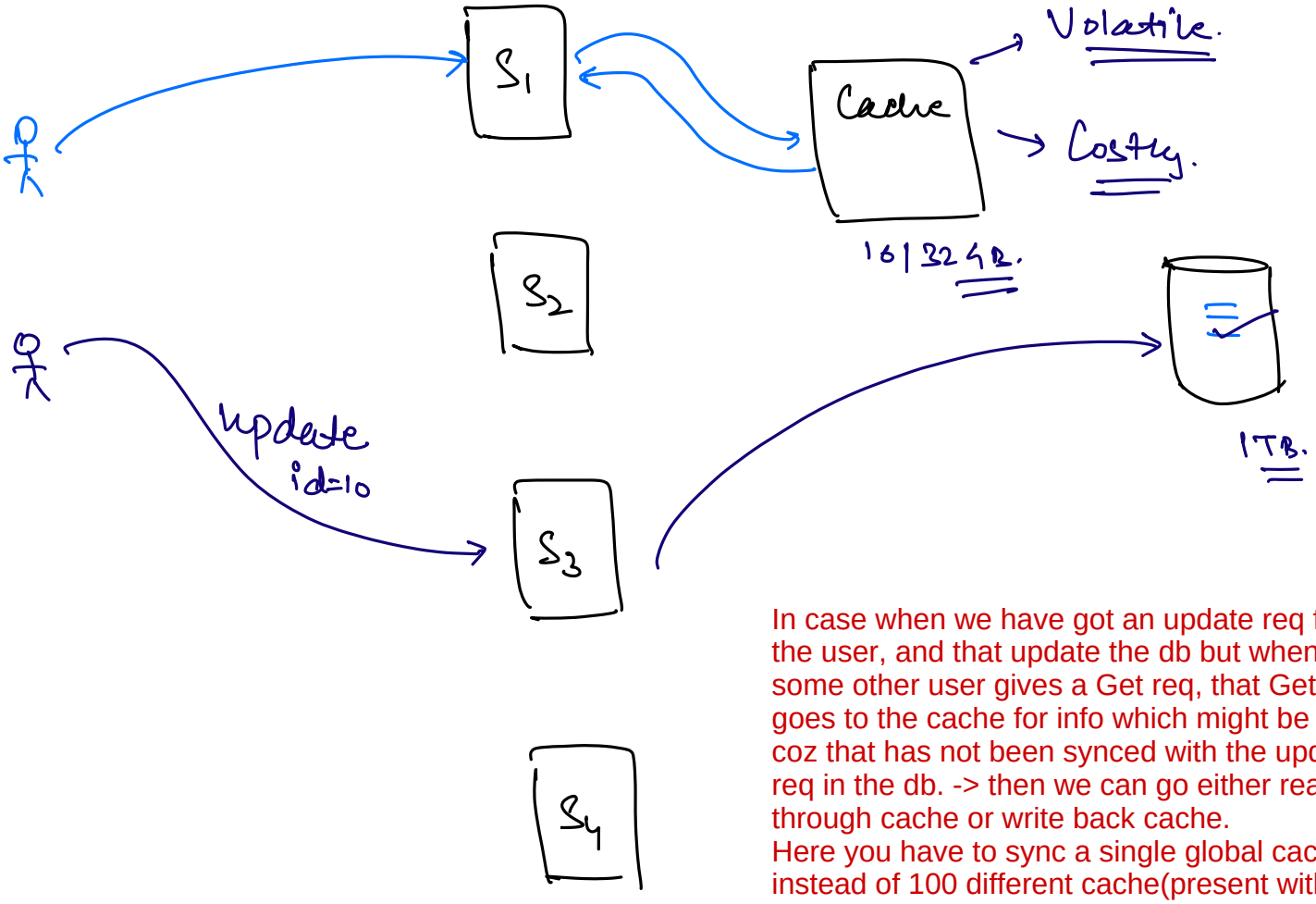


so there will be multiple servers and in multiple servers, there will be catching mechanism but storing same data across different servers is not a good scenario plus it may happen that the cache may go out of sync for sometime so you might see inconsistency in data

so instead of keeping cache at server level, we can keep centralized cache with cache server



⇒ Global Cache.



In case when we have got an update req from the user, and that update the db but when some other user gives a Get req, that Get req goes to the cache for info which might be old coz that has not been synced with the updated req in the db. -> then we can go either read through cache or write back cache.
Here you have to sync a single global cache instead of 100 different cache(present within the server)

- → Write through
- → Write back
- → TTL
- → Read through

HW.

getProductById(id)

Introduce caching in our productservice -> through getProductById method

```
Product p = cache.get(id);
if (p != NULL) return p;
Product p = ProductRepo.findByIdId(id);
Cache.put(id, p)
return p;
```

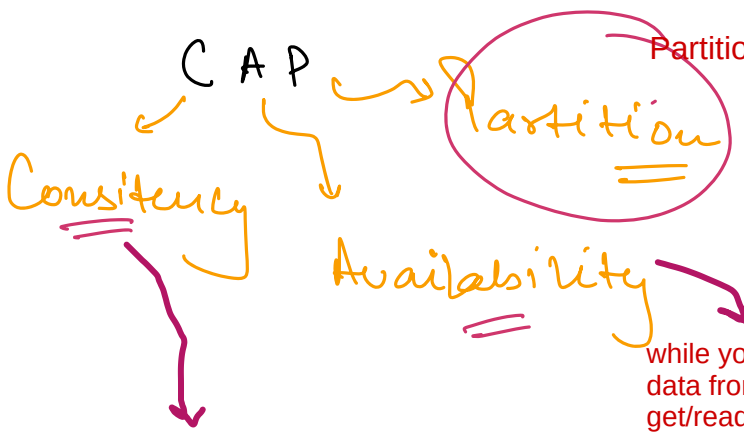
3

when you give a get req from cache and get the data -> cache hit

Cache HIT : When data is present inside the Cache.

Cache MISS : When data isn't present inside the Cache.

CAP theorem states that out of these 3 you can achieve 2 things at a moment. Partition/Network tolerance will be there you cannot do anything, now you ahve to choose b/w Consistency & Availability



Partition -> means Network Tolerance

1. Financial.

If your data is financial data, you cannot show older data, so you should follow consistencystrictly. Here you can comprise on availbility

2. Google Search / Viewers Count on YT / Hotstar

latest data is updated in db as well as in cache

while you are syncing data from db to cache, get/read req can be given -> availability is you are getting the data everytime you make a req

for google search or viewers count on youtube -

→ Browser. ✓

browser will also do some kind of caching

We cannot keep all db data in centralized cache coz cache memory is volatile memory, it means if your cache gets restarted -> all the data will be deleted