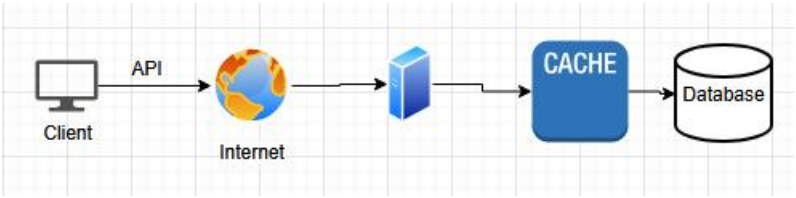
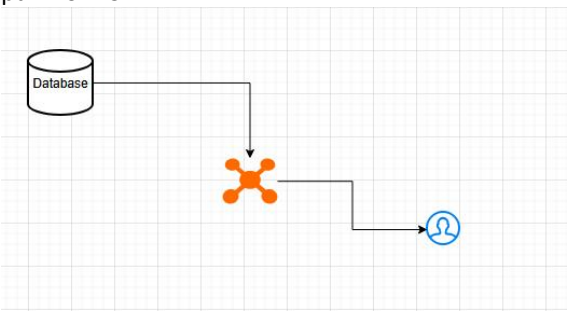
Understanding Cache

What's cache? Why do we need cache?
Say we have a client trying to reach Backend App code then a database, we want something in between to communicate faster. We need a latency-free kind of a system. For that we use cache. When we call our database for every query, then the latency will go up. Also, it latency goes up too much, the database might crash. To protect database, we use cache. There are different ways we can apply cache to our system.
Cache is a small unit in which we store the data, which is used repeatedly. We could have client-side cache (repeatedly used values in the browser, we can store on the client-side cache). Second option is to have a cache between database and Application-code, only if data is not there in the cache we will do a call to the database.



There is also CDN: Content-Delivery Network, stores static data
Static data like Images, Static text, Video
Sometimes, we store only certain length of the video in cache and only if user crosses that length, we pull from CDN



Lets say a new episode of a TV series is getting uploaded into main server, maybe the main-server is in France etc. Every CDN should get updated with the latest information.

There are different strategies that are applicable to cache, are applicable to CDN also, in some cases.
If one CDN server is down, then our request will be re-directed to another CDN
YouTube video

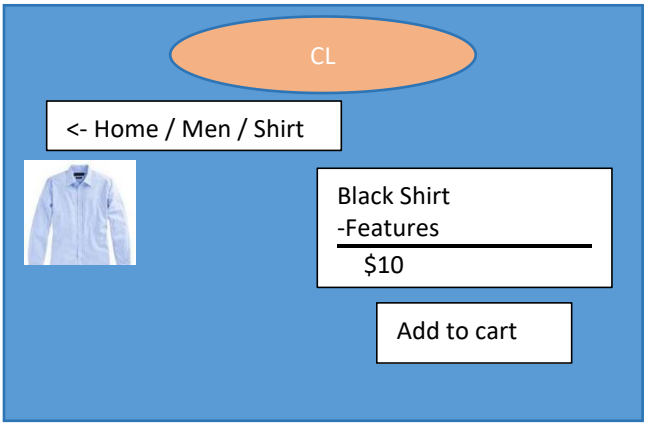
ID	Thumbnail	Text	Videos
1	CDN server URL (Image)	YouTube desc	CDN server URL

30:00

Cache stores data, CDN stores content. CDN is a data center

Cache can be implemented on the client-side or server-side

PDP - Product Display Page



Lets say, we have CL as the clothing brand
On this page, what are the things we want to cache and what not to cache. So, when UI talks to Backend, we are only getting from Cache not Database that means data in the UI will be static for sometime.
What are the things we currently have on the page?
Image --> is it going to rapidly change or not? NO then Yes it can be cached
Name --> Yes can be cached
Reviews --> No it should not be cached because Reviews keep on changing according to customers
Features --> Yes can be cached
Price --> No price cannot be cached
Couple search box --> No don't cache
Add to cart button --> it contains the link to cart so it can be cached
S3 bucket is usually cached

Cache: why do we need cache?
Fast storage

Caching strategies:
Cache terminologies:
1. Cache hit --> Whenever we try to find data in cache and if we find it then we don't need to go database and this is called as the cache hit.
Application --> AppCode --> Cache --> Database

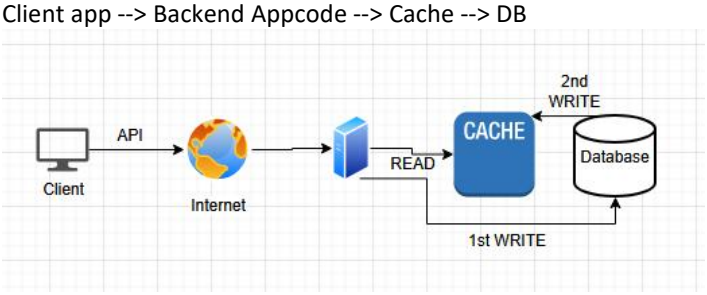
2. Cache miss --> Opposite of Cache hit, it application cannot find the data in cache then it needs to go fetch the data from database. That means, cache is missed. Not necessarily, data not present is a system failure, it could be a cache miss by design itself. For example, we are not storing Price or Reviews in cache by design.

3. TTL (Time To Live) --> As a System designer, we decide for how long the content should remain in the system and that time, then it should be evicted from the cache. Say we want to update the cache every one minute. Whenever we delete the data in the database, we should know how long our cache should hold that data. For a clothing brand, a campaign runs for 3 or 4 months, it is safe to say after 3 months the cache should get cleared.

4. Latency --> Whenever user is sending some request, how long our system takes to respond back. If our Backend is pulling data from cache then it can work fast

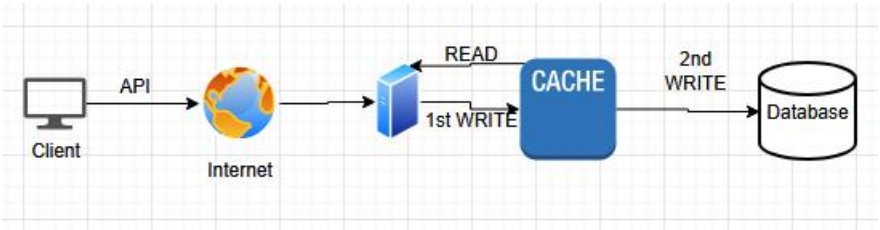
Caching eviction policies: when data will be deleted from cache
1. LRU (Least Recently Used)
The data that was not used recently will be removed.
2. MRU (Most Recently Used)
3. LFU (Least Frequently Used)
4. FIFO (First In First Out)
5. LIFO (Last In First Out)

Caching strategies:
1. ReadThroughCache (RTC):
Client app --> Backend Appcode --> Cache --> DB



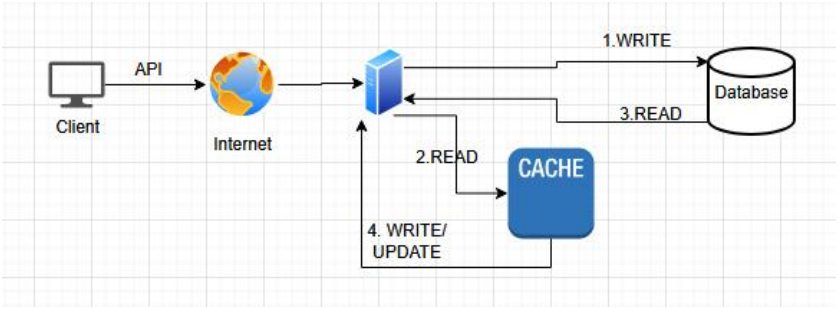
When we read, we read only from cache only. Whenever we write something, we write them first into Database then Database will update cache. If we are not getting data from cache, only at that point we read from Database.
Because it is reading from cache, that means our reads will be faster so mostly implemented in CDNs
When we write something, we write directly into database. Database will also update the cache because we are not reading from database. So write will eventually be bit slower because first we got to write into database then into cache second.

2. WriteThroughCache (WTC):
Whenever we are writing anything, we will first write into cache then into Database. In this, our cache always gets updated information. Read is always from cache. It always works from cache itself.



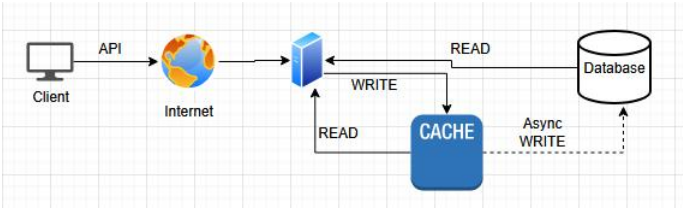
In Financial Apps, we want our transactions updated very rapidly.

3. WriteAroundCache (WAC):



First we Write into database, then Read from Cache, if it is not in Cache, we Read from DB, then we update Cache with that new info. We are responsible for managing cache

4. WriteBackCache (WBC):



First we write into cache and read from cache, however, we write into Database asynchronously

Performance metrics:
1. Cache Hit Ratio:
Cache hit number / Total calls = 90 / 100
2. Cache Miss Ratio:
Cache miss count / Total calls = cache miss + cache hit = 10 / 100

Homework
Strategies --> we can pick an example of a Website / App or Feature
We got to figure out which strategy applies to which feature, same which eviction policy applies to which feature