A Quick Overview of

Server Side Caching Strategy

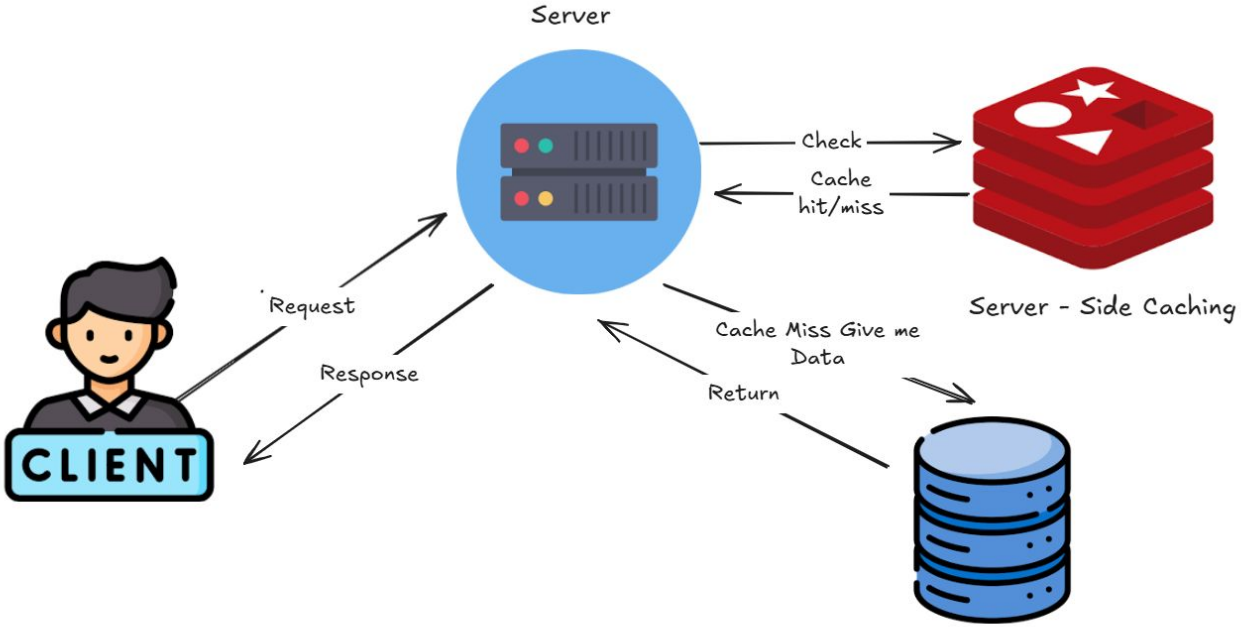
Sumonta Saha Mridul

Trainee Software Engineer

Cefalo Bangladesh LTD.


ME
SANZIDA
FATIN BHAI
imgflip.com

What is Server Side Caching?



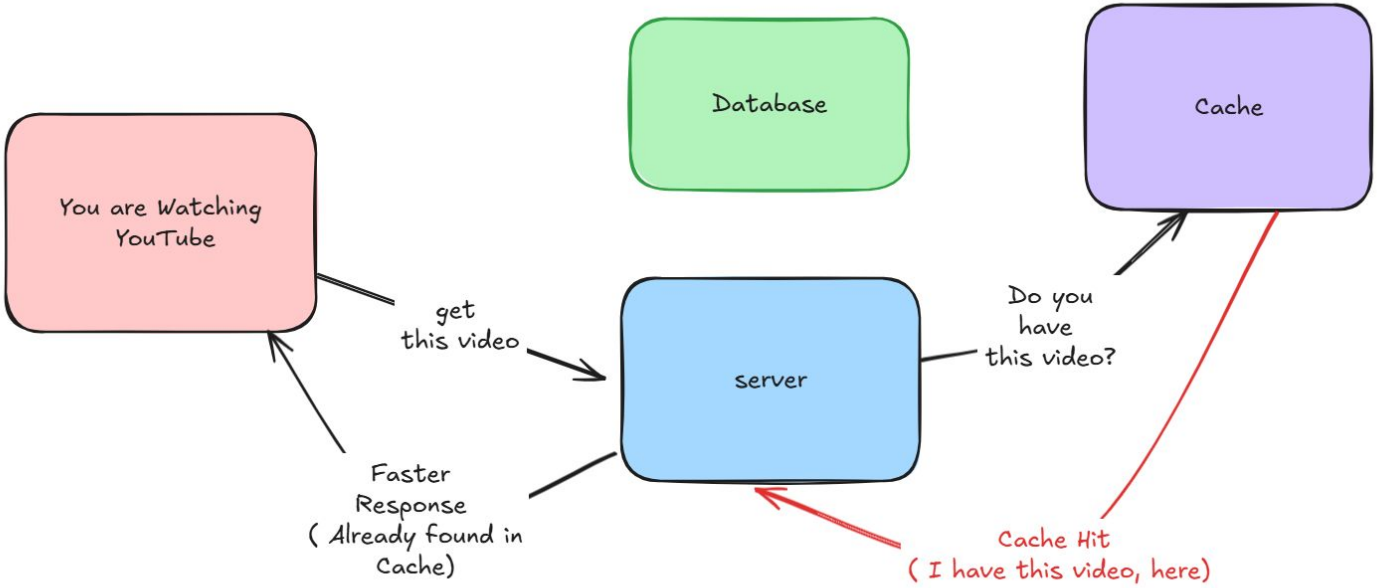
Server-side caching generally refers to the cache layer positioned between the server's application logic and the database.

What is Cache Hit & Cache Miss?

What is Cache Hit?

- Requested data is already stored in the cache
- The cache quickly returns the data without needing to fetch it from the original source, saving time and resources.

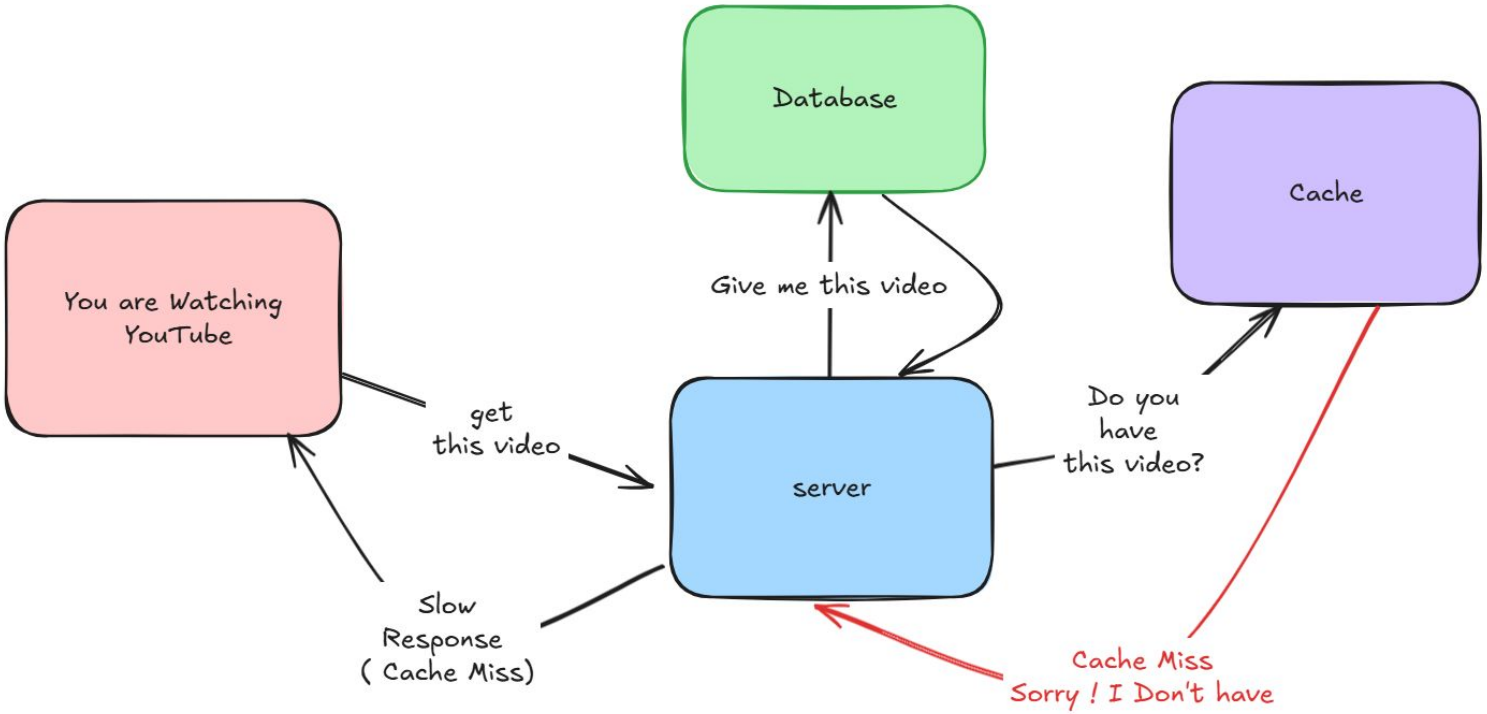
What is Cache Hit?



What is Cache Miss?

- Requested data **is not stored** in the cache
- The system fetches the data from the original source (like a database or server), which takes more time.

What is Cache Miss?

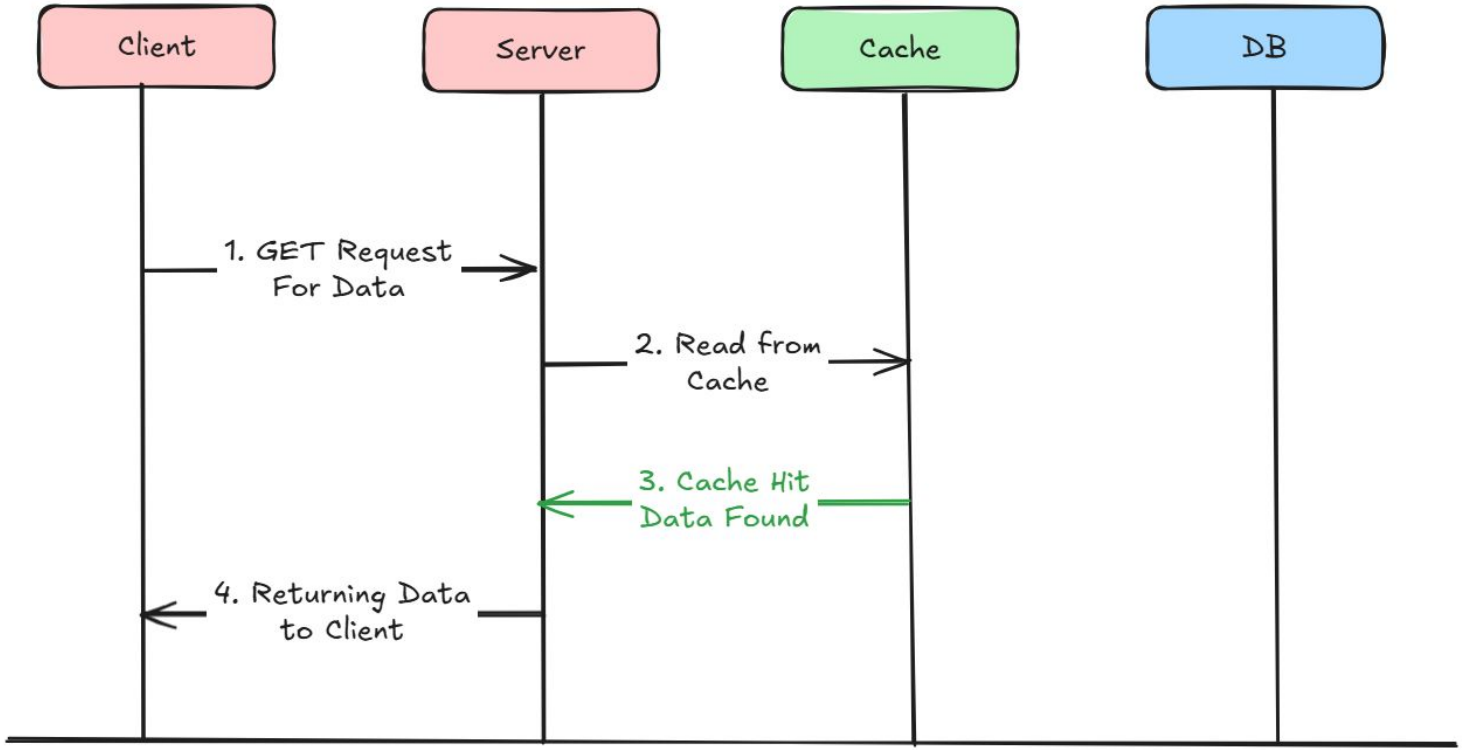


Cache Aside Caching Strategy

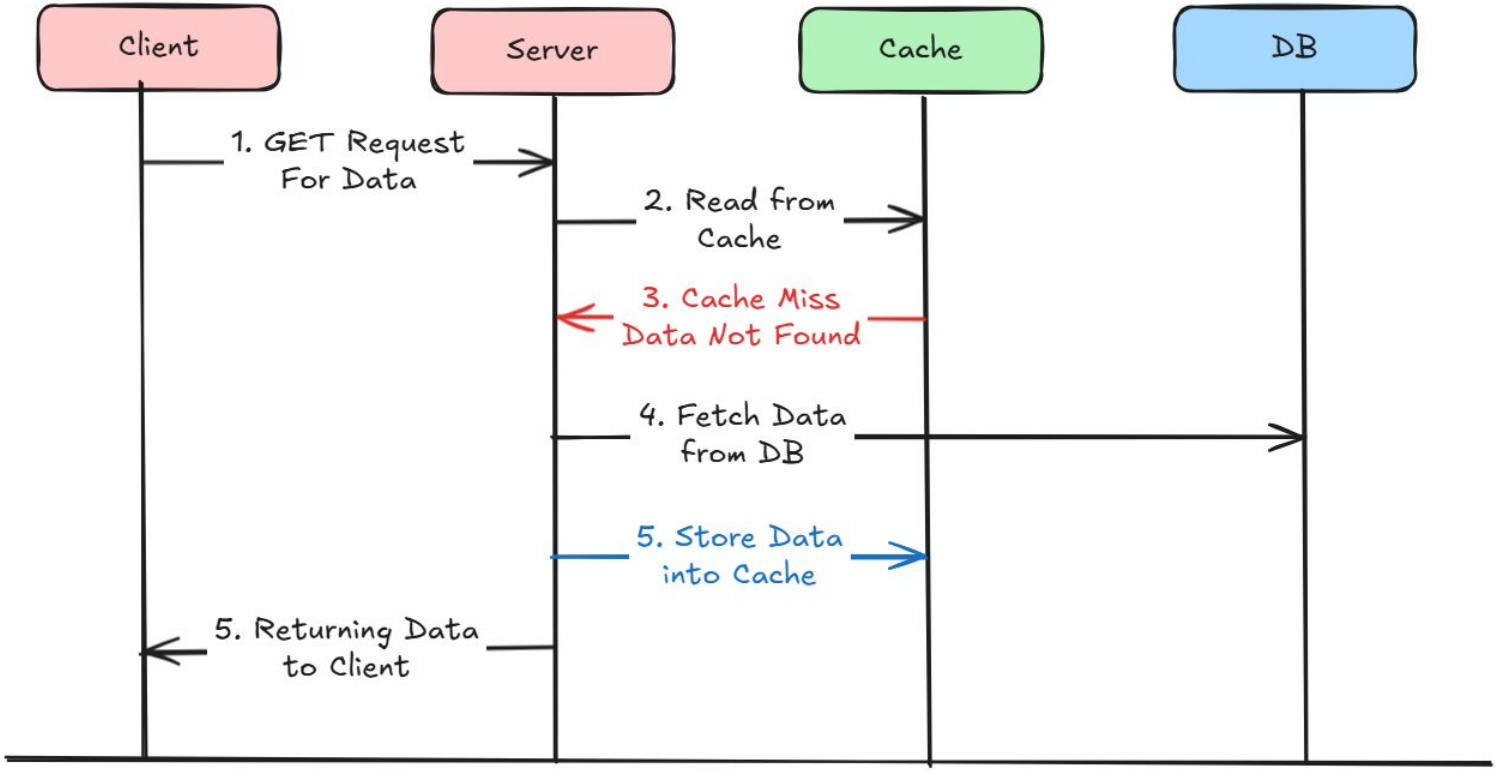
Cache Aside Strategy

- First check cache for data
- Cache Hit ! Return data to Client
- Cache Miss !
  - Server Fetch data from DB
  - Store data in Cache
  - Return data to client

What happens in Cache Hit?



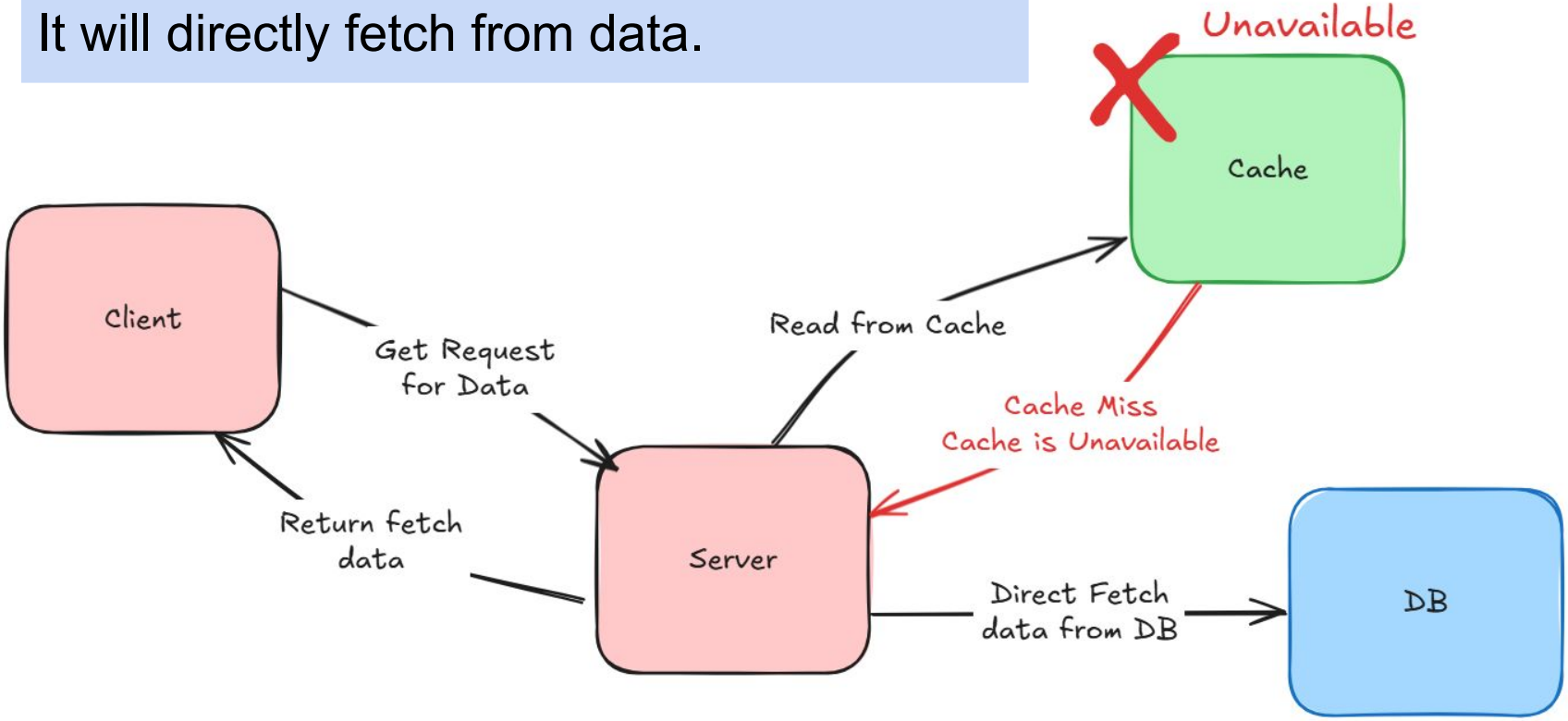
What happens in Cache Miss?



Cache Aside : Advantages

- Good Approach for Heavy Read applications
- Even Cache is down, request will not fail, It will directly fetch from data.
- Cache document data structure can be different that data present in DB

Even Cache is down, request will not fail,
It will directly fetch from data.



Cache document data structure can be different that data present in DB

```sql
CREATE TABLE Users (
    UserID INT PRIMARY KEY,
    Name VARCHAR(255),
    Email VARCHAR(255),
    Phone VARCHAR(15)
);
CREATE TABLE Posts (
    PostID INT PRIMARY KEY,
    UserID INT,
    Content TEXT,
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

Database (Structured for Storage)

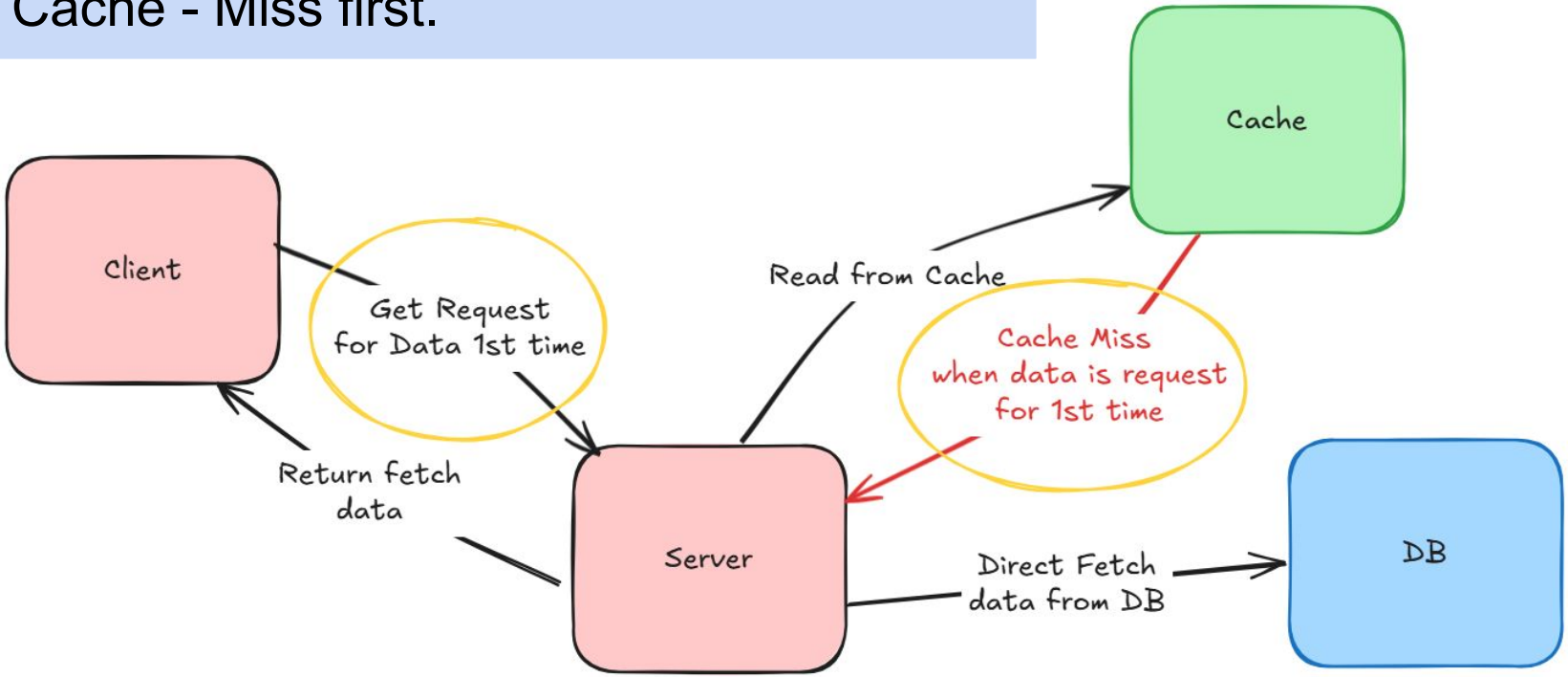
```json
{
  "UserID": 1,
  "Name": "John Doe",
  "Email": "john.doe@example.com",
  "RecentPosts": [
    {
      "PostID": 101,
      "Content": "Hello, world!"
    },
    {
      "PostID": 102,
      "Content": "Loving the sunny weather today!"
    }
  ]
}
```

Cache (Optimized for Access)

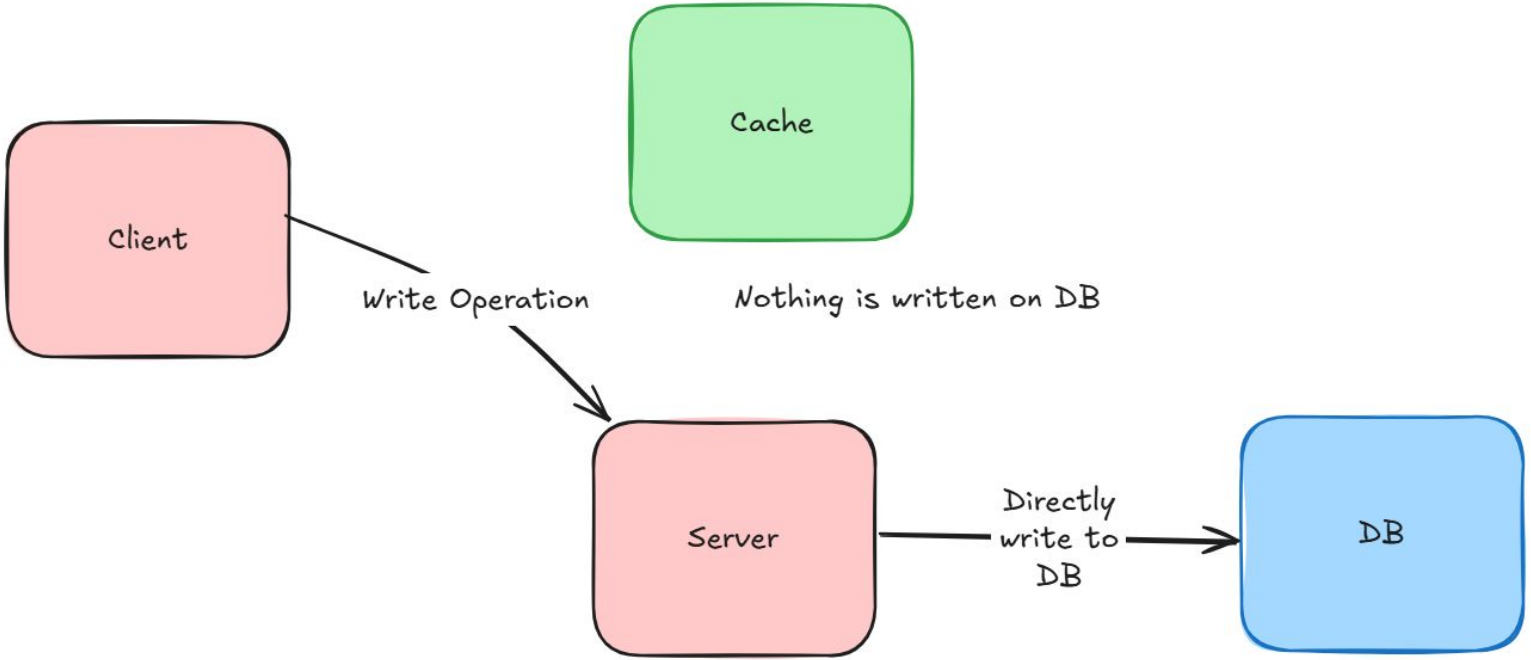
Cache Aside : Cons

- Write Only Method will always directly write to DB
- For new data read, there will always be Cache - Miss first.
- If appropriate caching is not used during write operation, there is a chance of inconsistency between Cache and DB

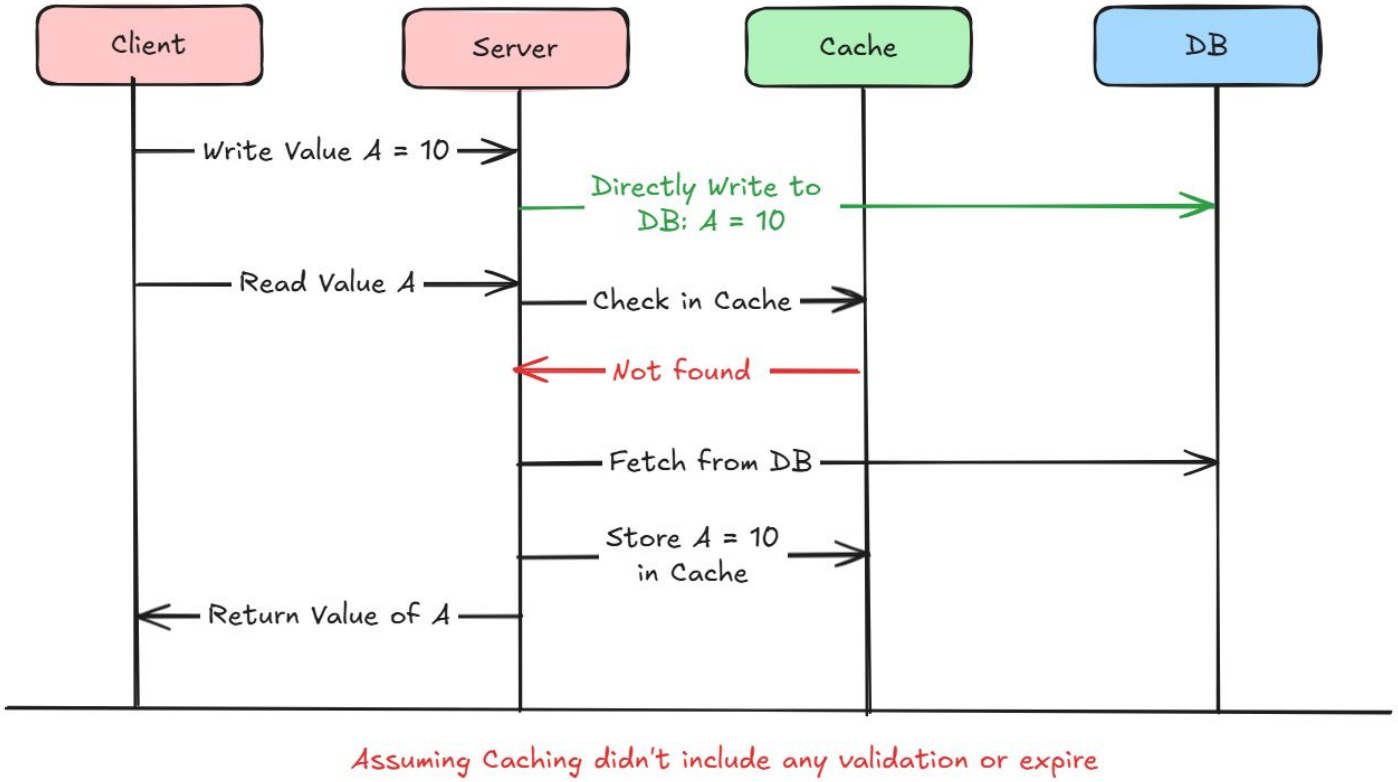
For new data read, there will always be Cache - Miss first.



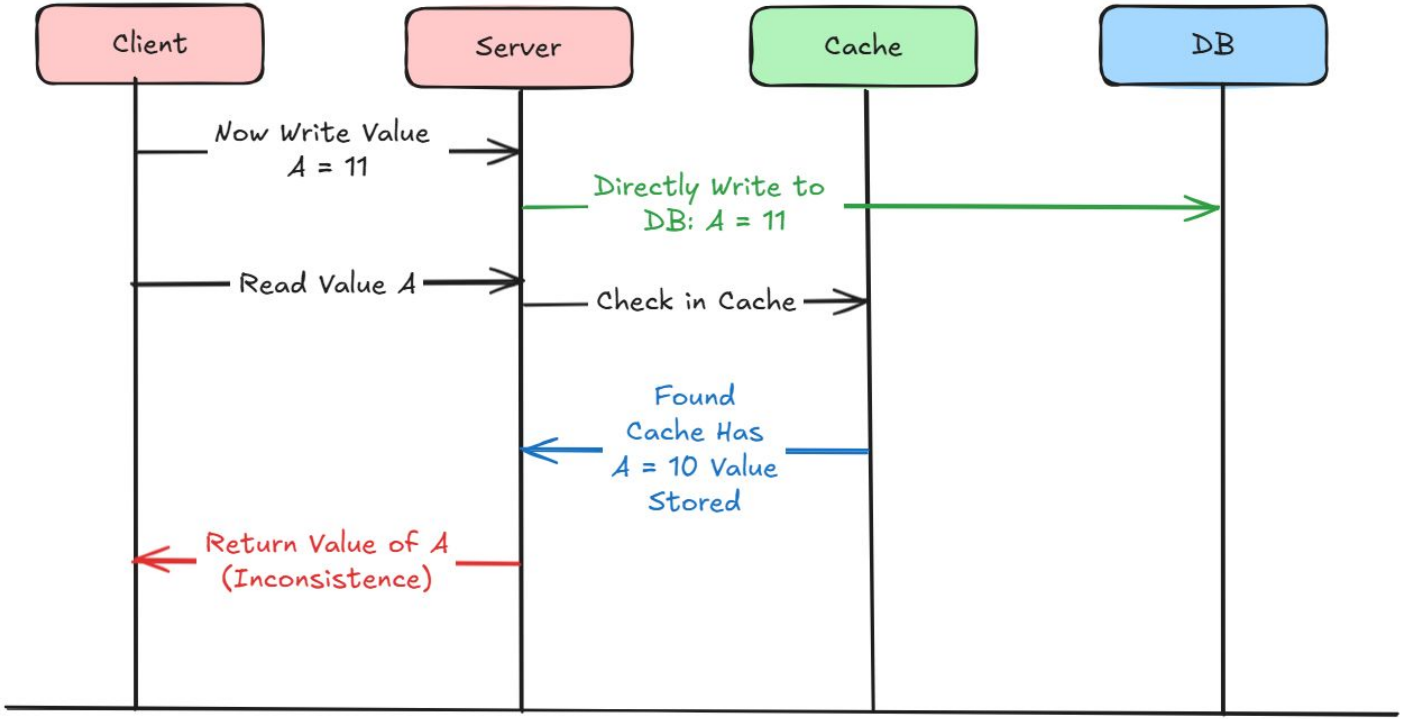
Write Only Method will always directly write to DB



If appropriate caching is not used during write operation, there is a chance of inconsistency between Cache and DB



If appropriate caching is not used during write operation, there is a chance of inconsistency between Cache and DB

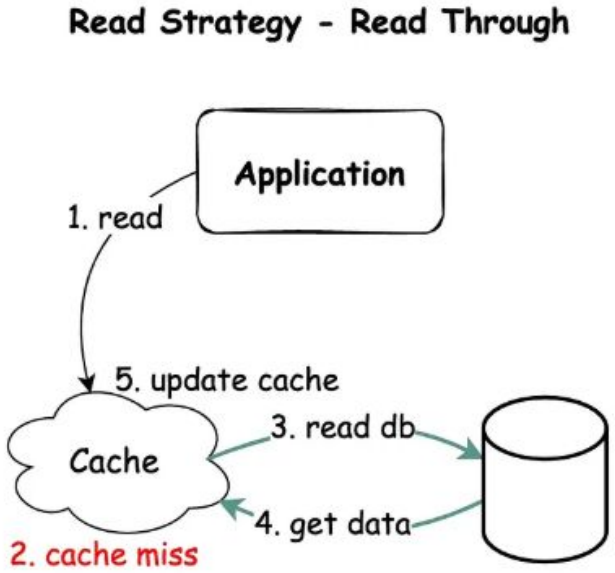


Assuming Caching didn't include any validation or expire

Read Through Caching Strategy

Read Through Strategy

- First check cache for data
- Cache Hit ! Return data to Client
- Cache Miss !
  - Fetch data from DB
  - Store data in Cache
  - Return data to server



What’s the difference then?

What's the difference then?

- If cache miss, **application or server** takes the responsibility
  - Fetch data from DB
  - Store data in Cache
  - Return it to Client

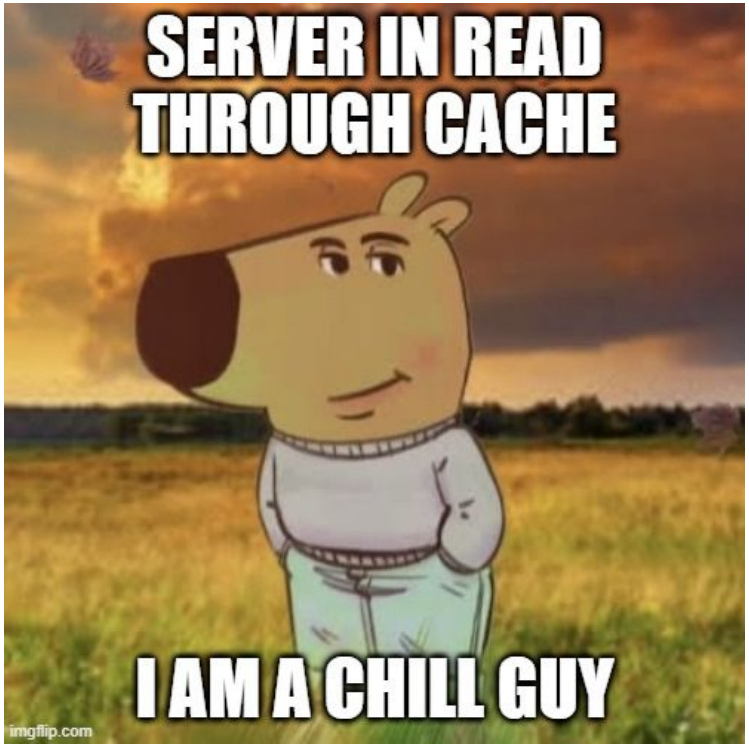
(Server has to do all of this task)

Cache Aside

- If cache miss, **cache library** takes the responsibility
  - Fetch data from DB
  - Store data in Cache
  - Return it to server

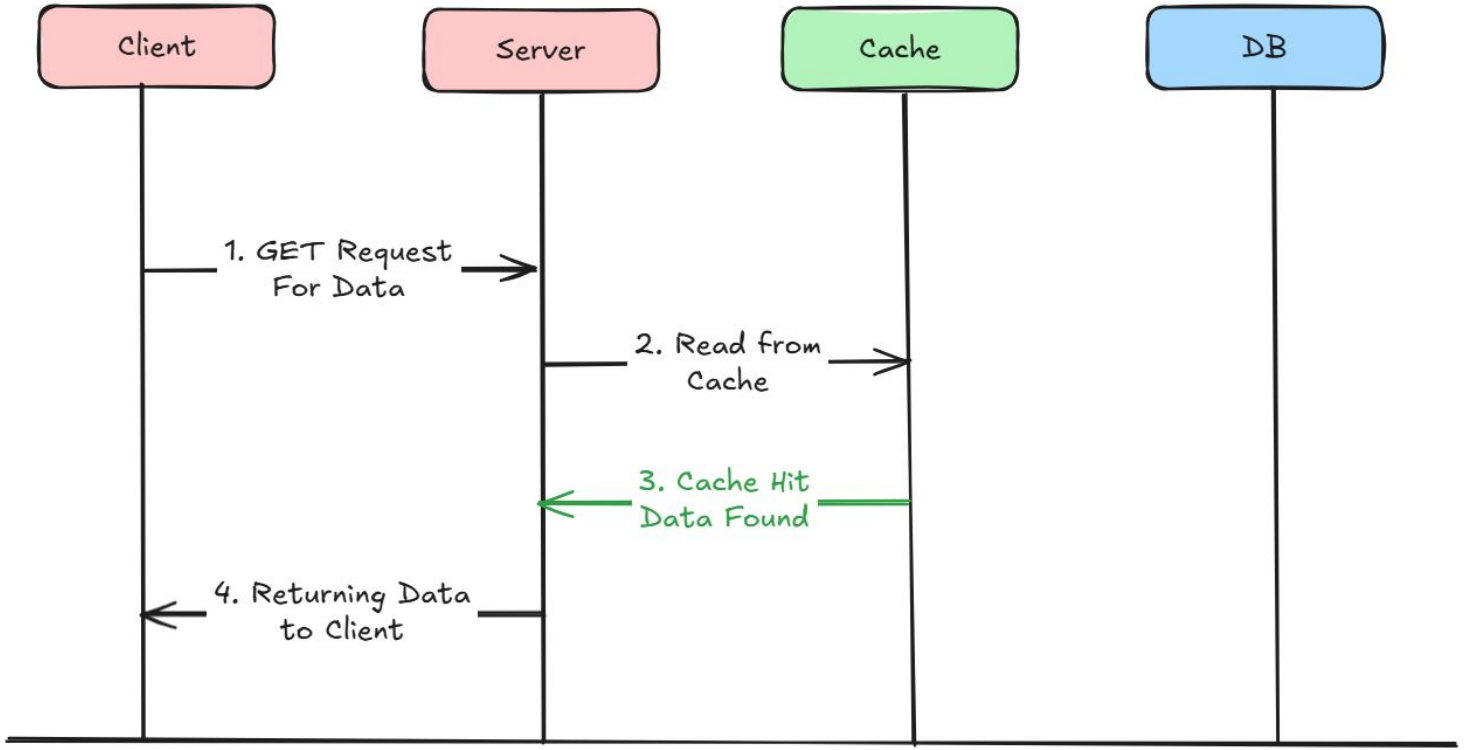
(Server has no tasks to do here)

Read Through

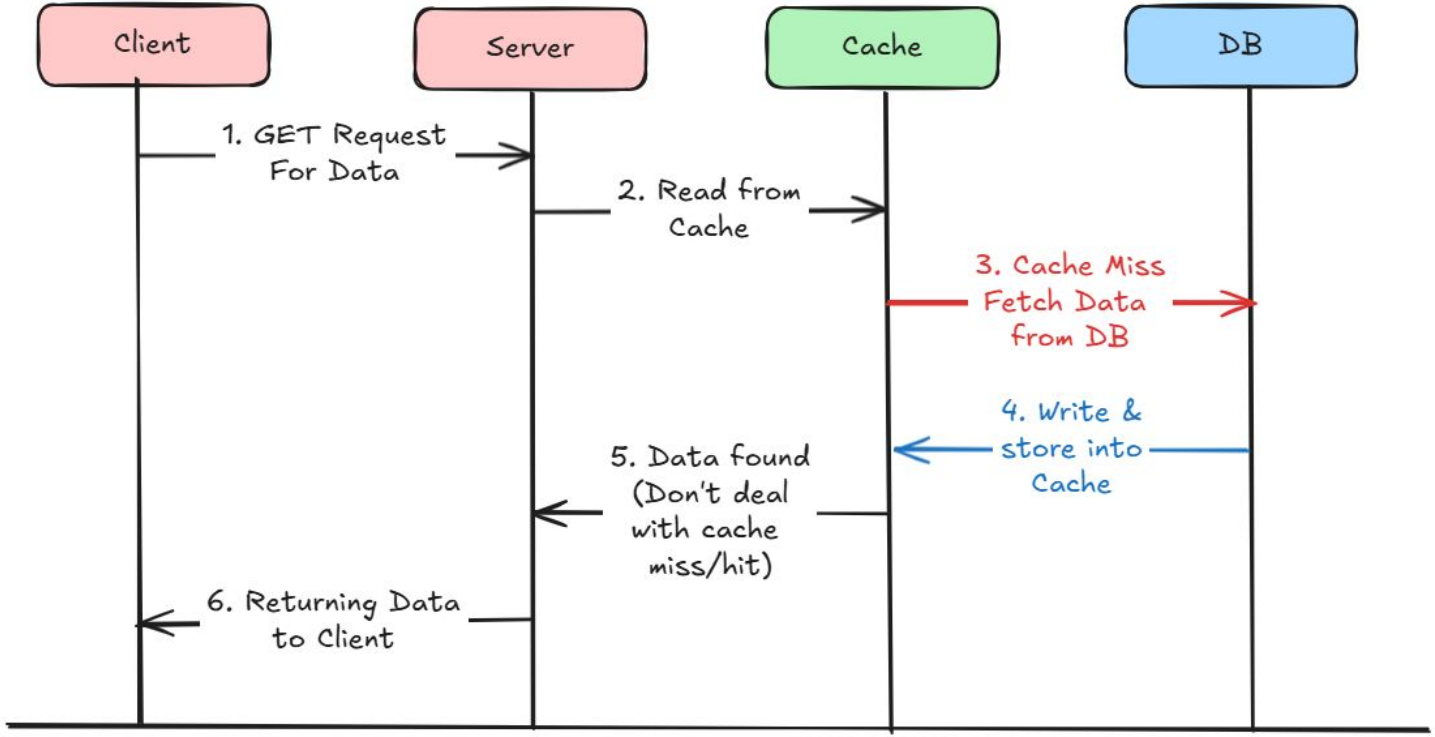

SERVER IN READ THROUGH CACHE
I AM A CHILL GUY
imgflip.com


CACHE MISS
SERVER IN CACHE ASIDE
imgflip.com

What happens in Cache Hit? (Same)



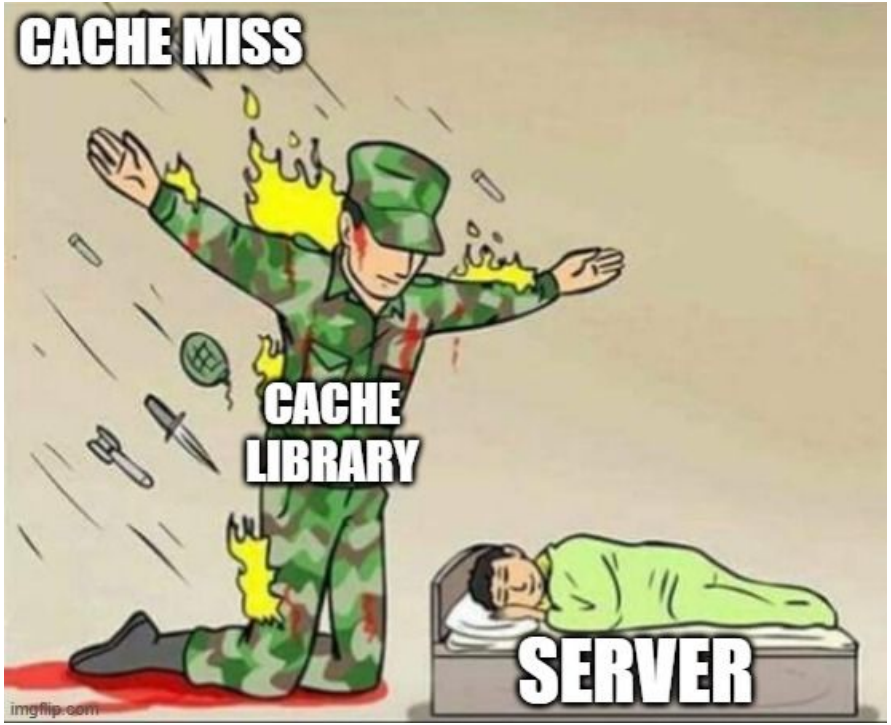
What happens in Cache Miss?



Read Through : Advantages

- Good Approach for Heavy Read applications
- Server don’t need to worry about whether it is a cache hit or miss
- Offloads the responsibility from the application, resulting in simpler application code.

Server don’t need to worry about whether it is a cache hit or miss



Read Through : Cons

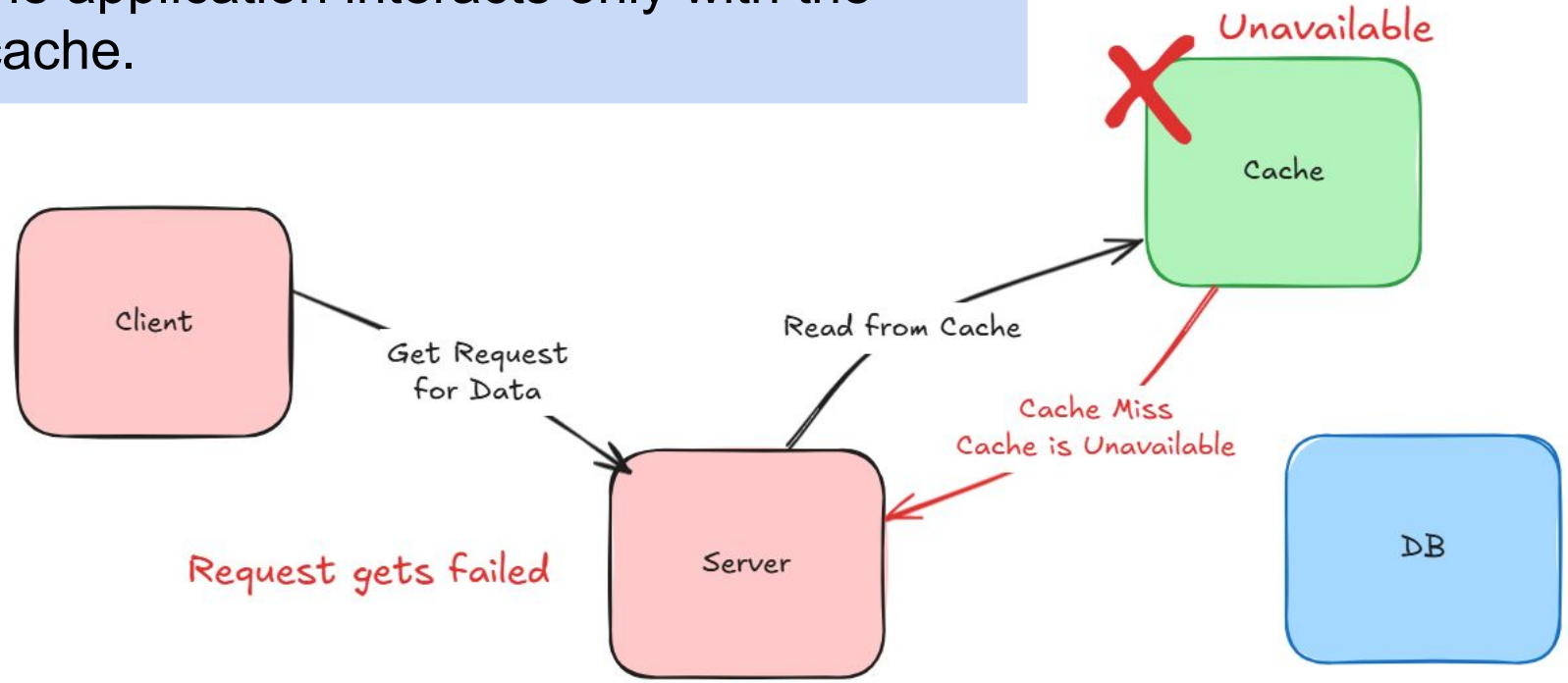
- Write Only Method will always directly write to DB
- For new data read, there will always be Cache - Miss first.
- If appropriate caching is not used during write operation, there is a chance of inconsistency between Cache and DB

Read Through : Cons

- Cache document data structure **can not be different** that data present in DB

- If Cache goes down, **request will fail**, since the application interacts only with the cache.

If Cache goes down, request will fail, since the application interacts only with the cache.



Cache document data structure **can not be different** that data present in DB

```sql
CREATE TABLE Users (
    UserID INT PRIMARY KEY,
    Name VARCHAR(255),
    Email VARCHAR(255),
    Phone VARCHAR(15)
);
CREATE TABLE Posts (
    PostID INT PRIMARY KEY,
    UserID INT,
    Content TEXT,
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

```json
{
  "UserID": 1,
  "Name": "John Doe",
  "Email": "john.doe@example.com",
  "RecentPosts": [
    {
      "PostID": 101,
      "Content": "Hello, world!"
    },
    {
      "PostID": 102,
      "Content": "Loving the sunny weather today!"
    }
  ]
}
```

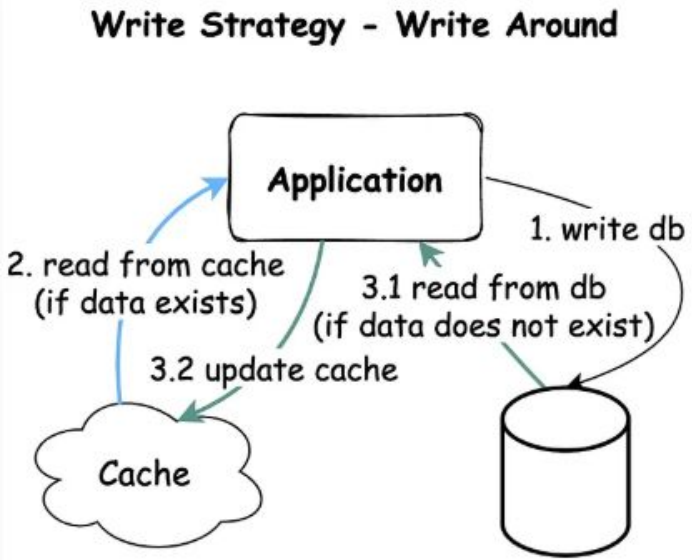
Database (Structured for Storage)

Cache (Optimized for Access)

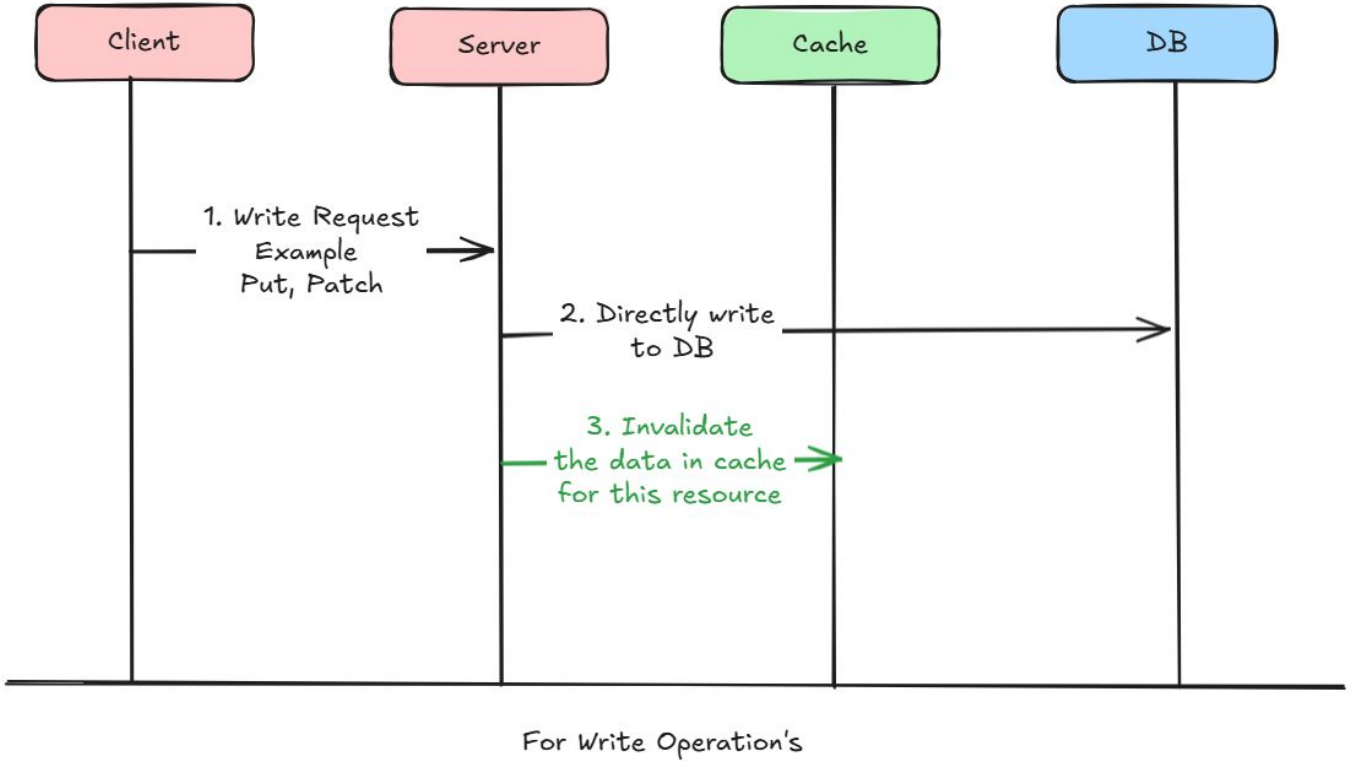
Write Around Caching Strategy

Write Around Strategy

- Directly writes data into the database
- For Read : Choose between
  - Cache Aside
  - Read Through

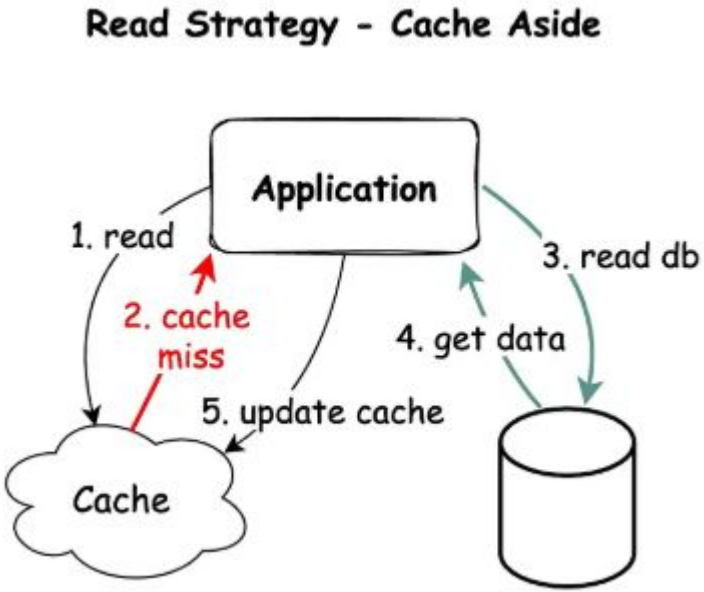


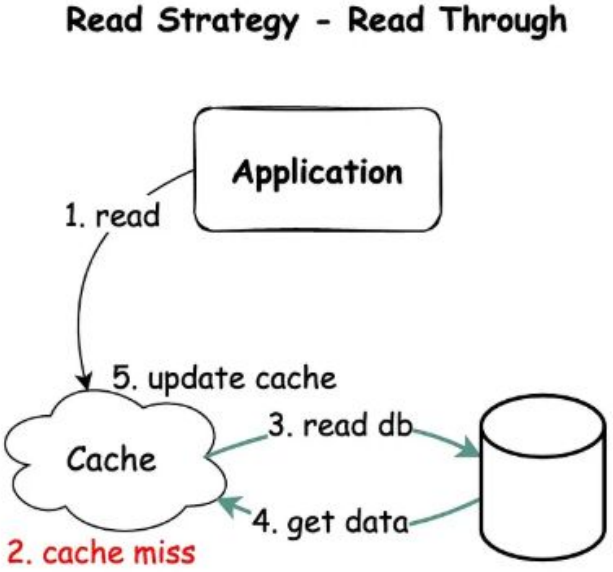
How write operation works?



For Write Operation's

What to do when read operations?

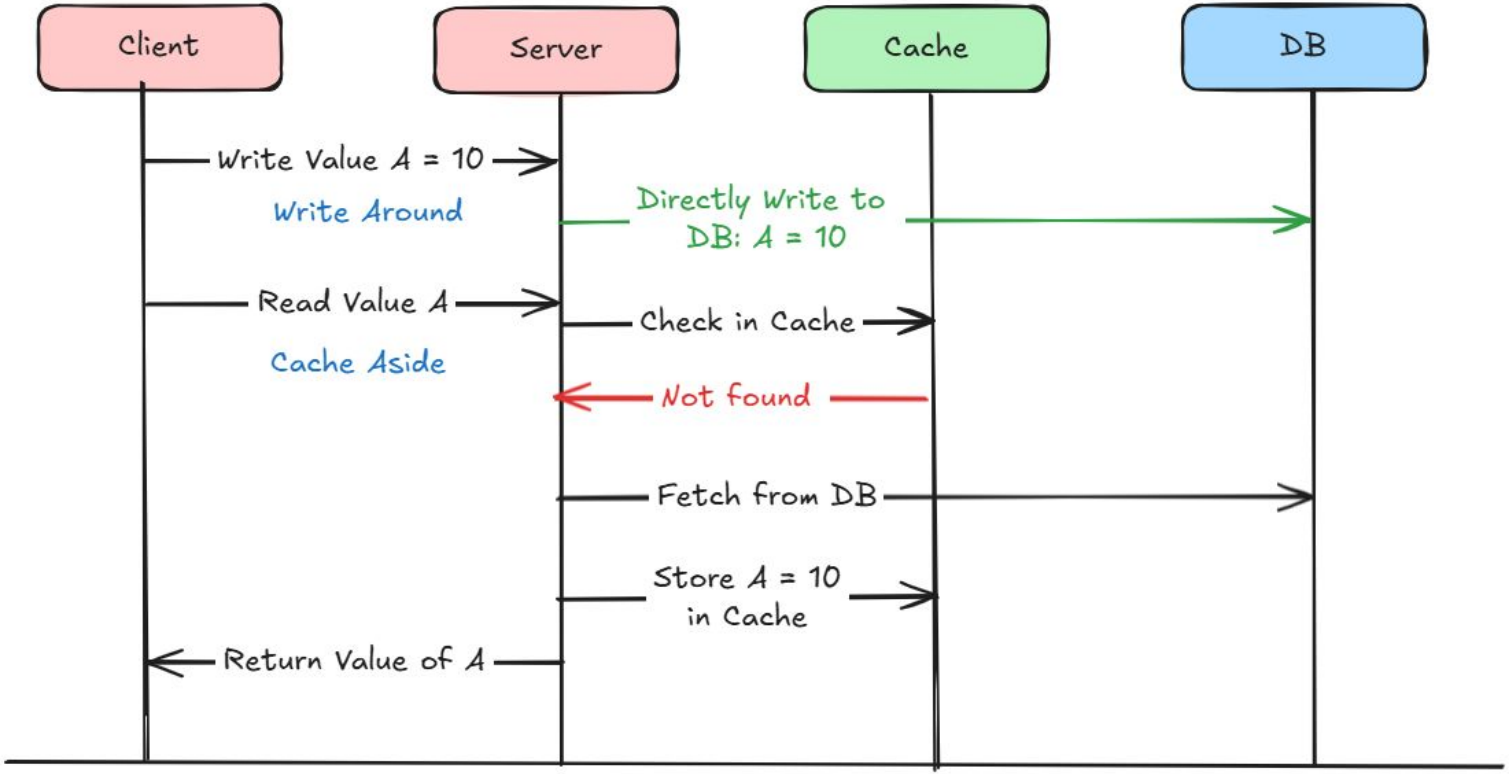




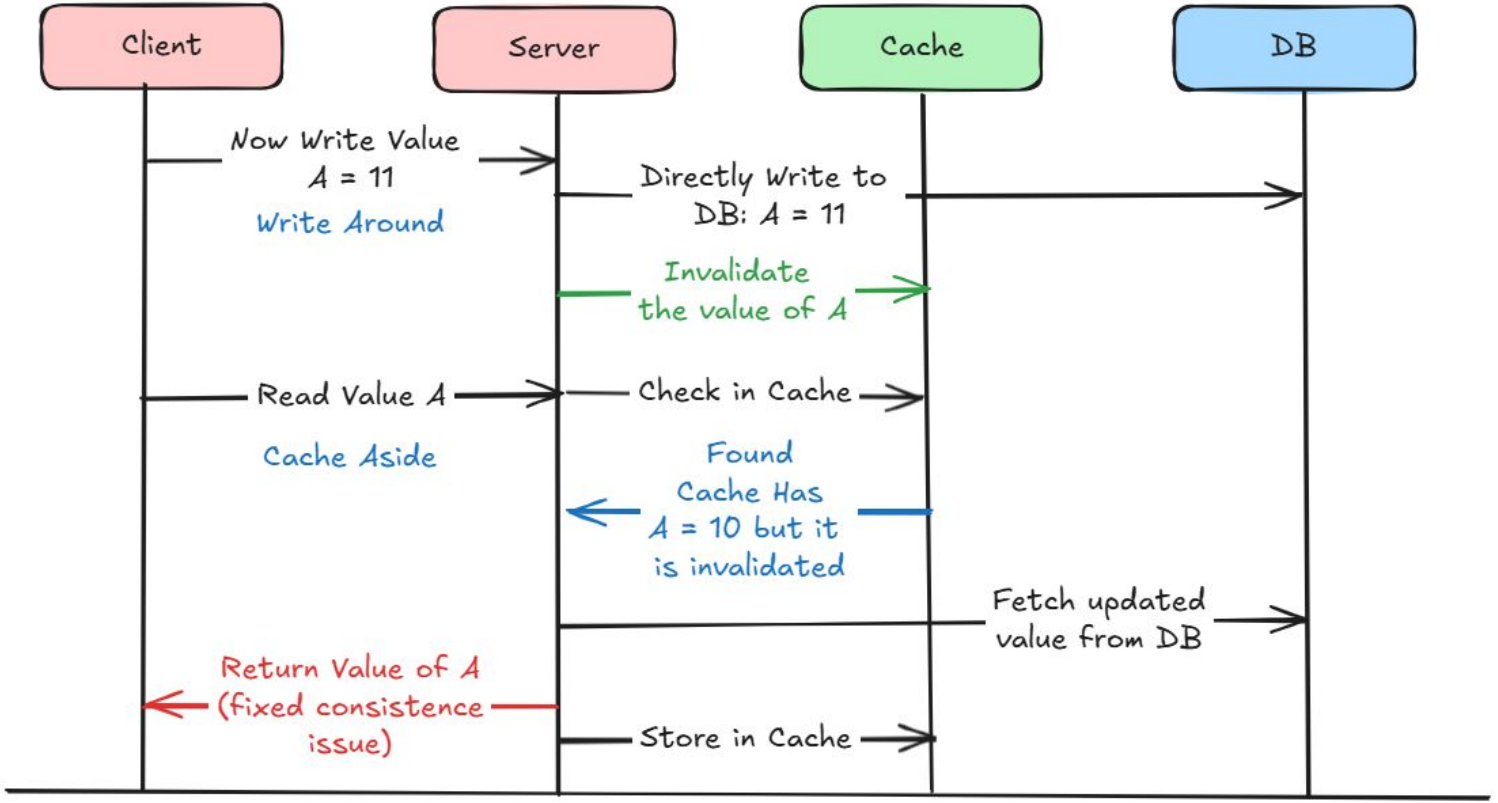
Have to choose a strategy between Cache Aside & Read Through

Combination of
Read : Cache Aside
Write : Write Around

Combination of Cache Aside & Write Around Strategy



Combination of Cache Aside & Write Around Strategy



Write Around : Advantages

- Good Approach for Heavy Read applications

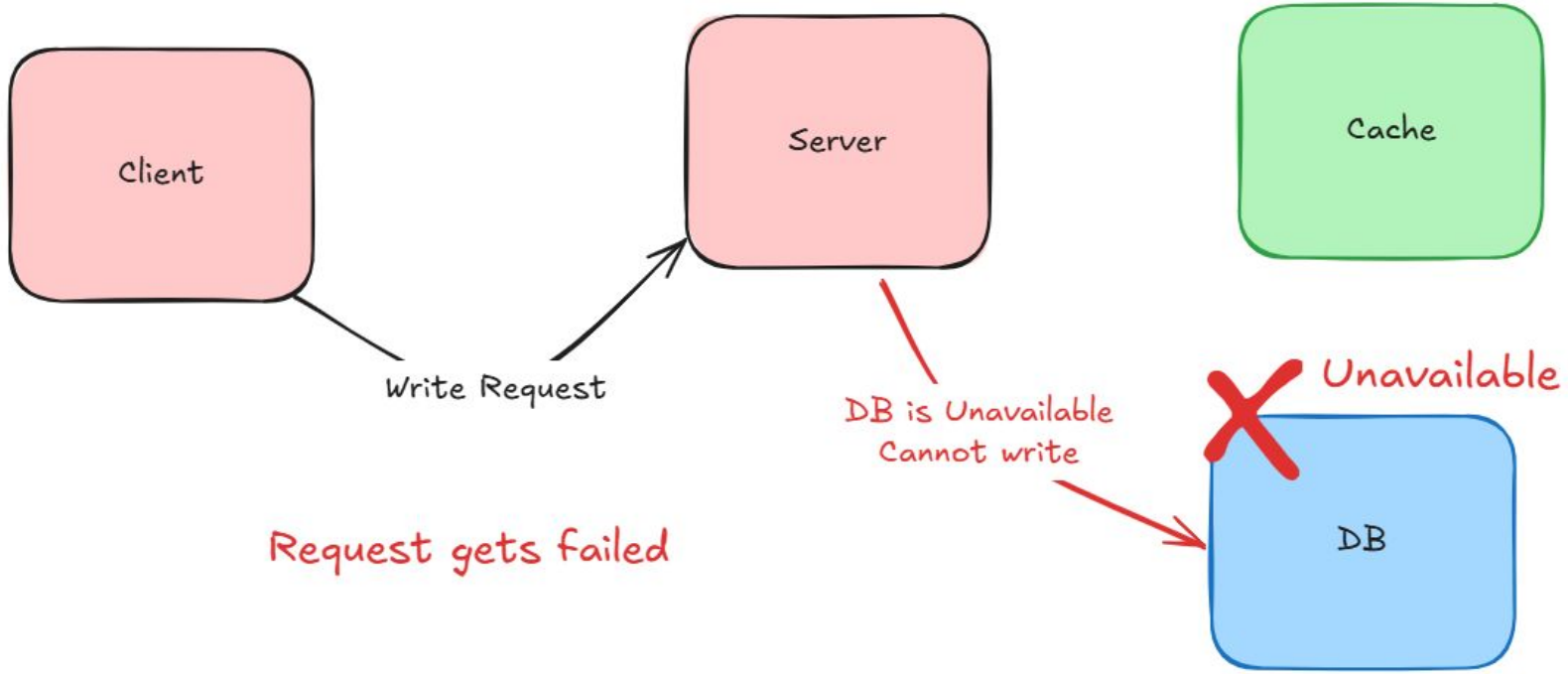
- Resolves the inconsistency problem between Cache and DB

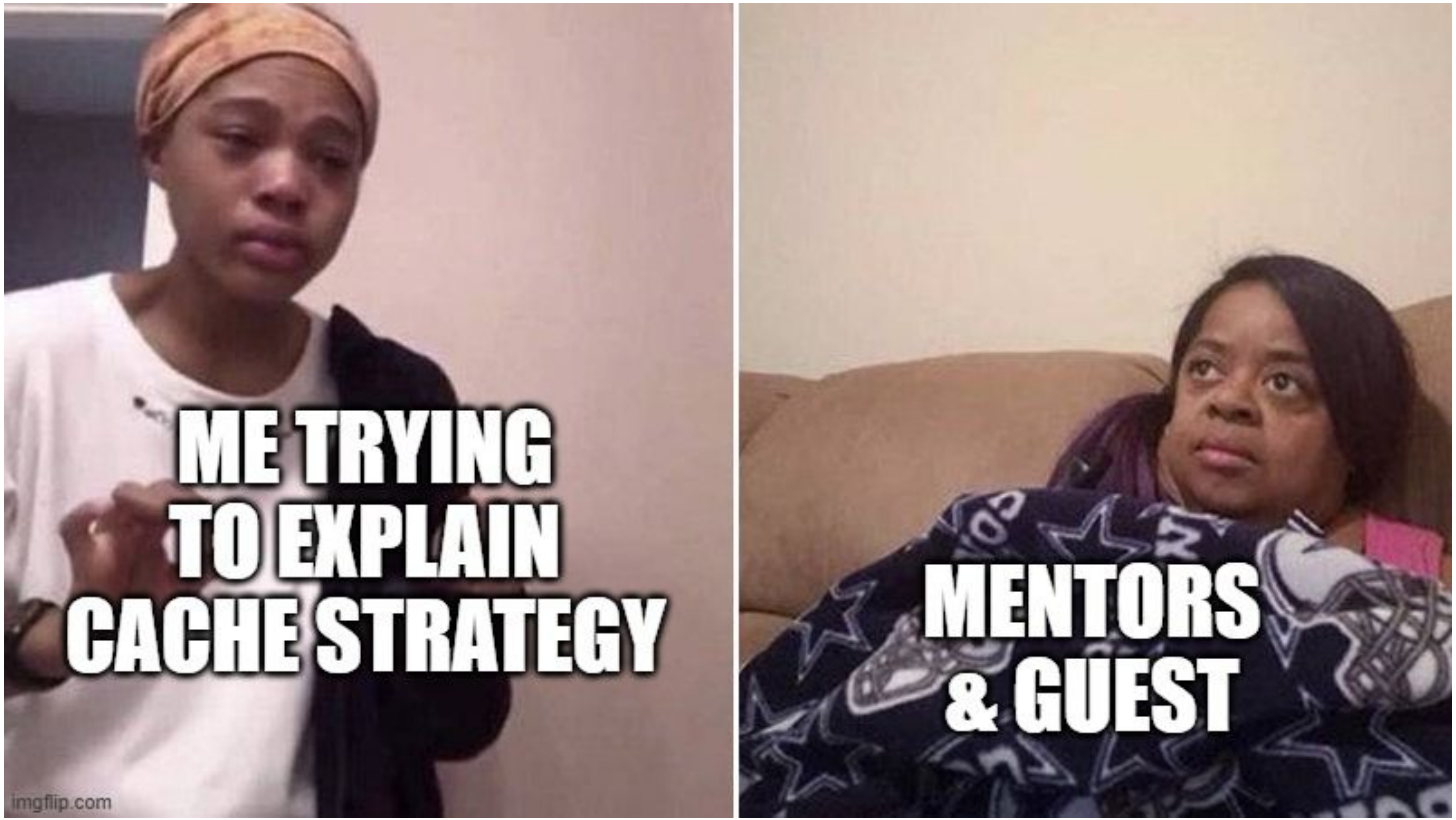
Write Around : Cons

- For new data read, there will always be Cache - Miss first.

- If DB is down, request for write operation will fail

If DB is down, request for write operation will fail

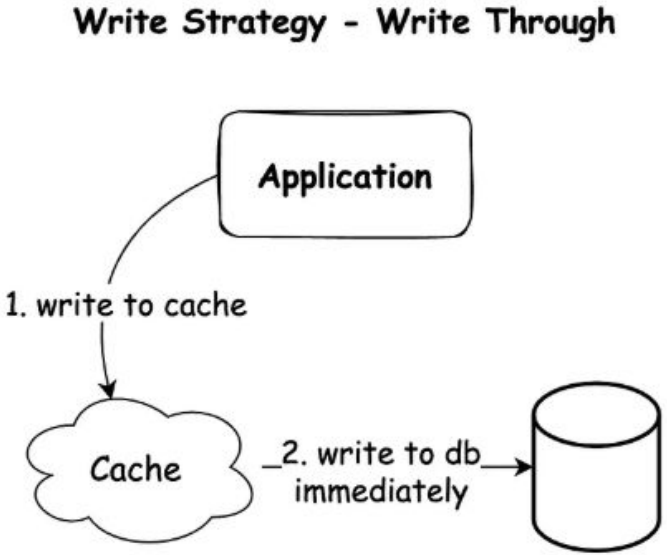



ME TRYING
TO EXPLAIN
CACHE STRATEGY
imgflip.com
MENTORS
& GUEST

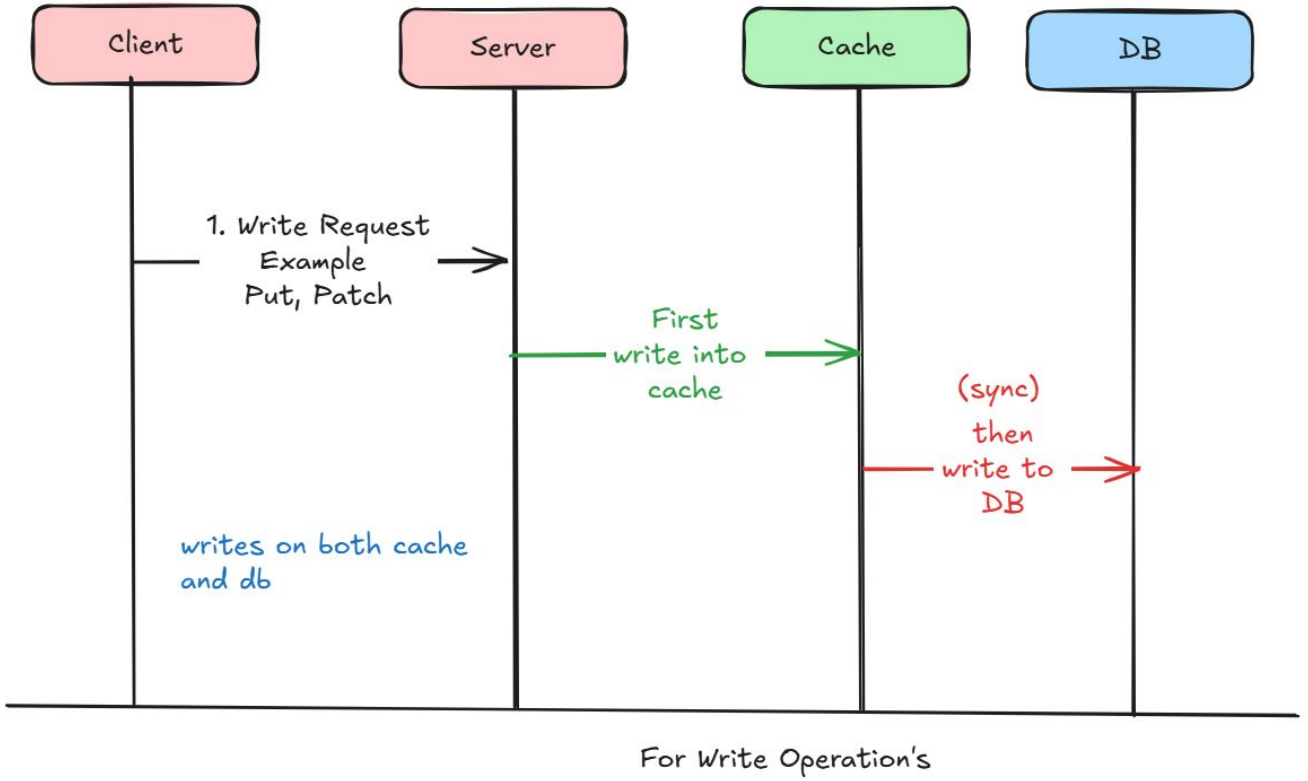
Write Through Caching Strategy

Write Through Strategy

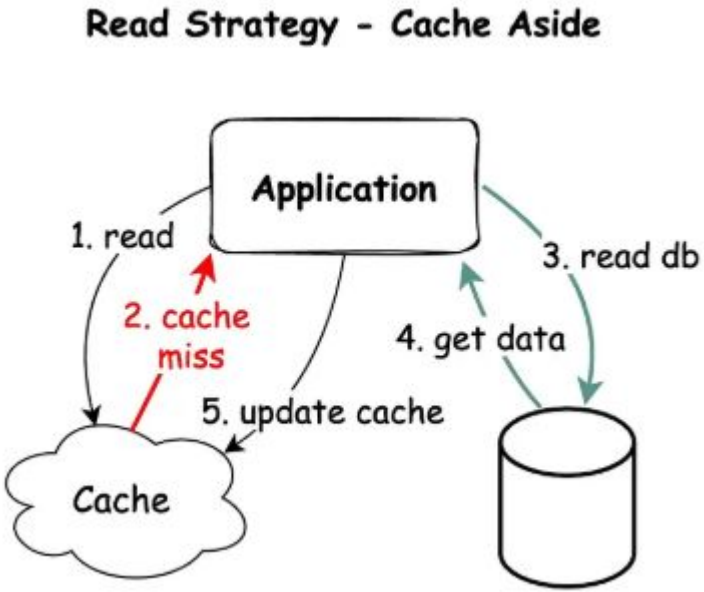
- First writes data into the cache
- Then in **synchronous** write data into the DB
- For Read : Choose between
  - Cache Aside
  - Read Through

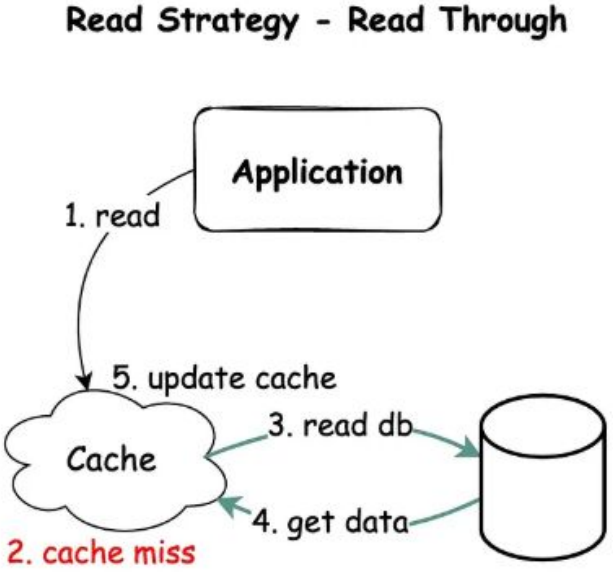


How write operation works?



What to do when read operations?





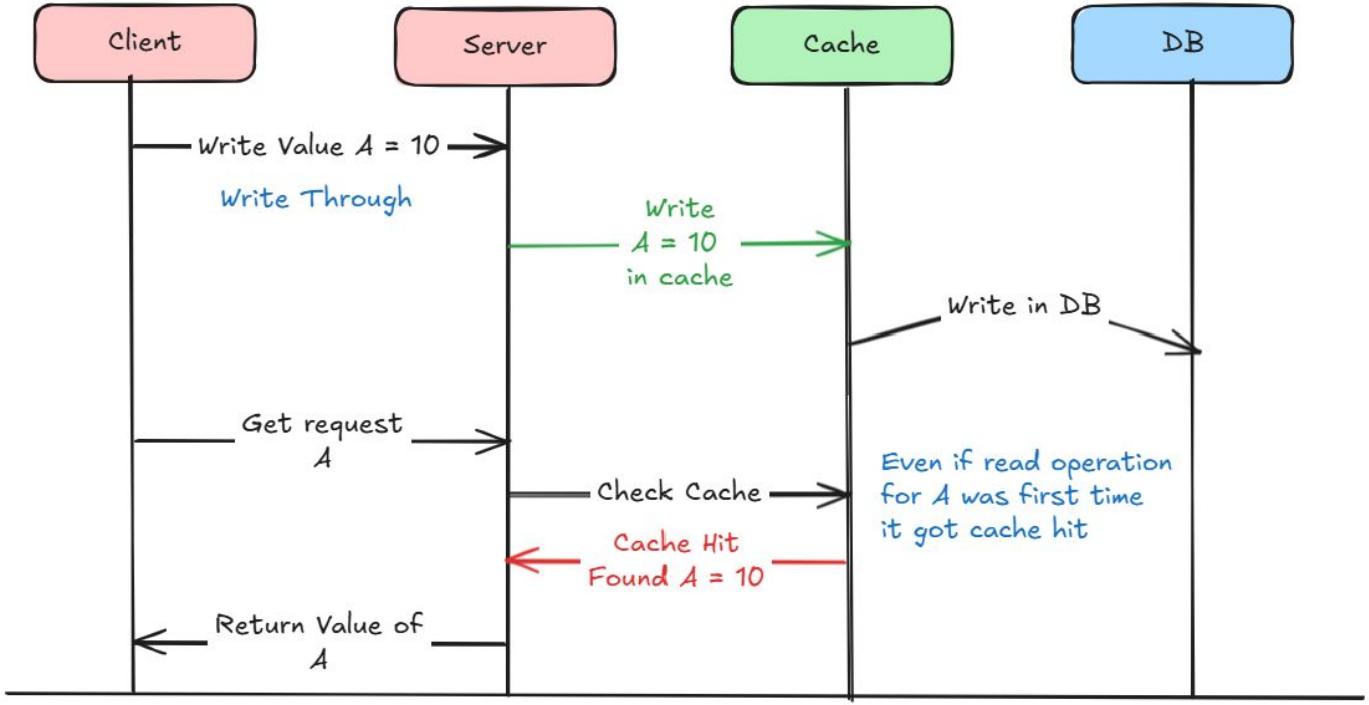
Have to choose a strategy between Cache Aside & Read Through

Write Through : Advantages

- Resolves the inconsistency problem between Cache and DB

- Increase Cache Hit chances : Even if for new data read there is a chance of cache hit

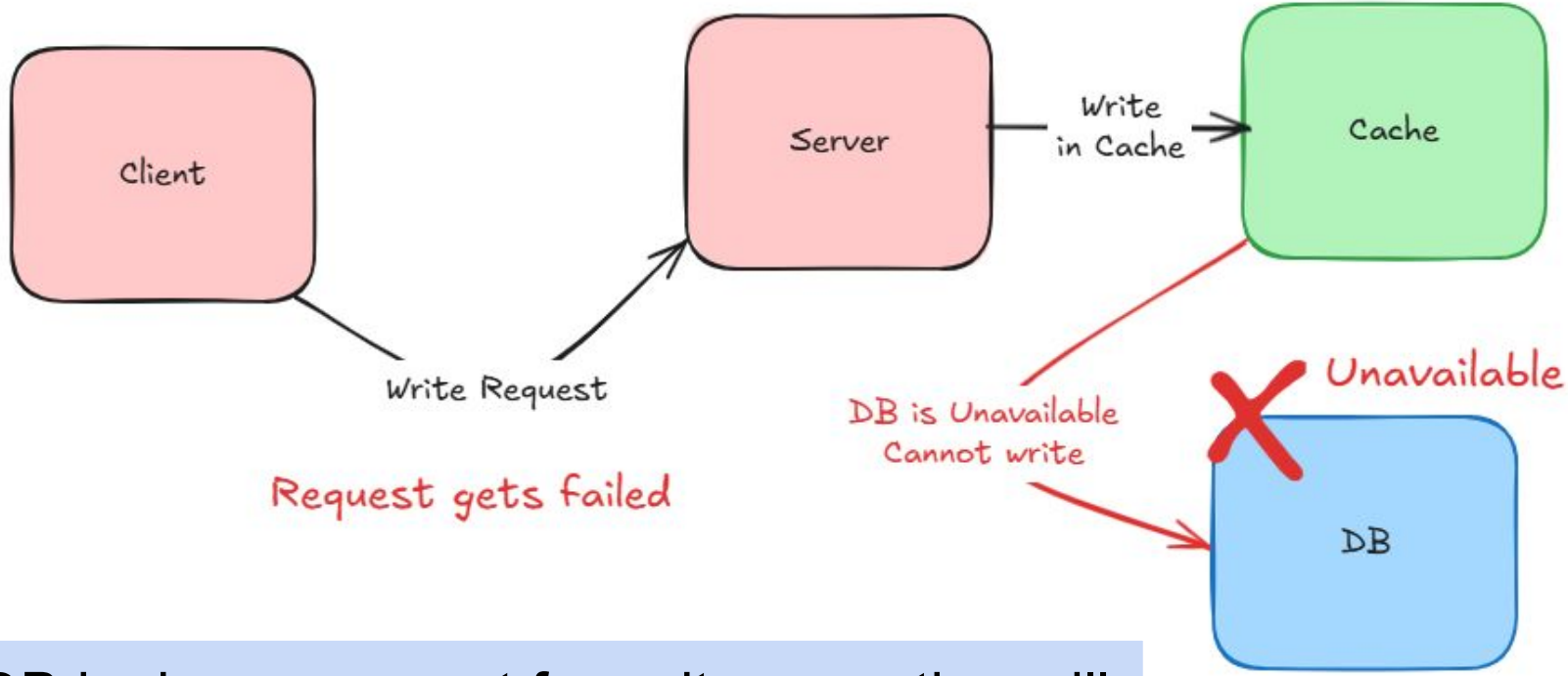
Increase Cache Hit chances : Even if for new data read there is a chance of cache hit



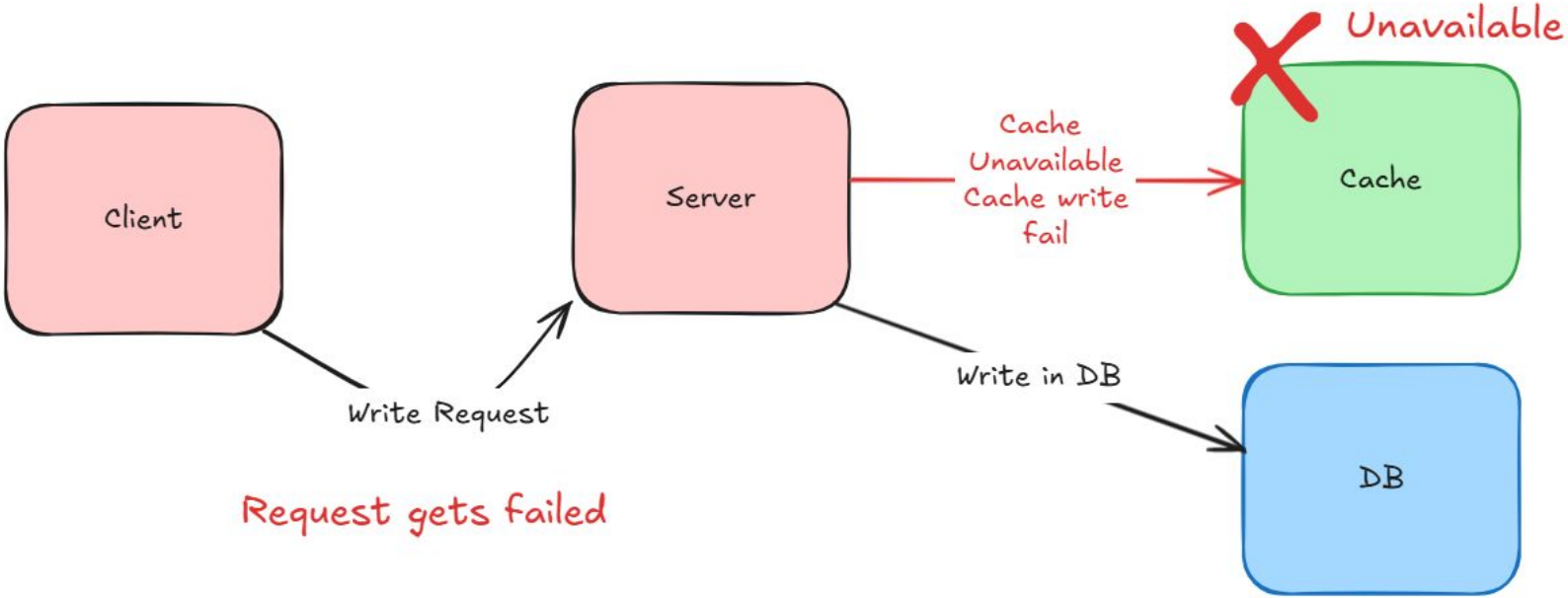
Write Through : Cons

- If any of DB or Cache is Unavailable, request will fail.

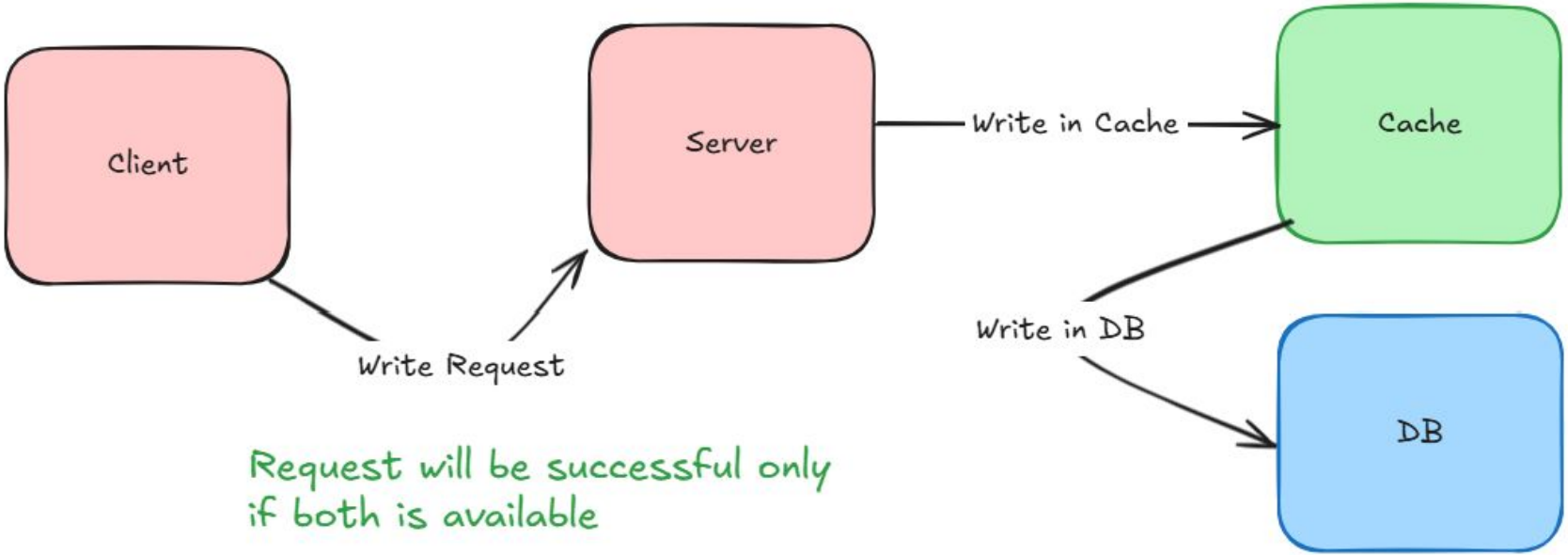
- 2 phase commit, need to be applied to maintain transaction property



If DB is down, request for write operation will fail Even if cache is available



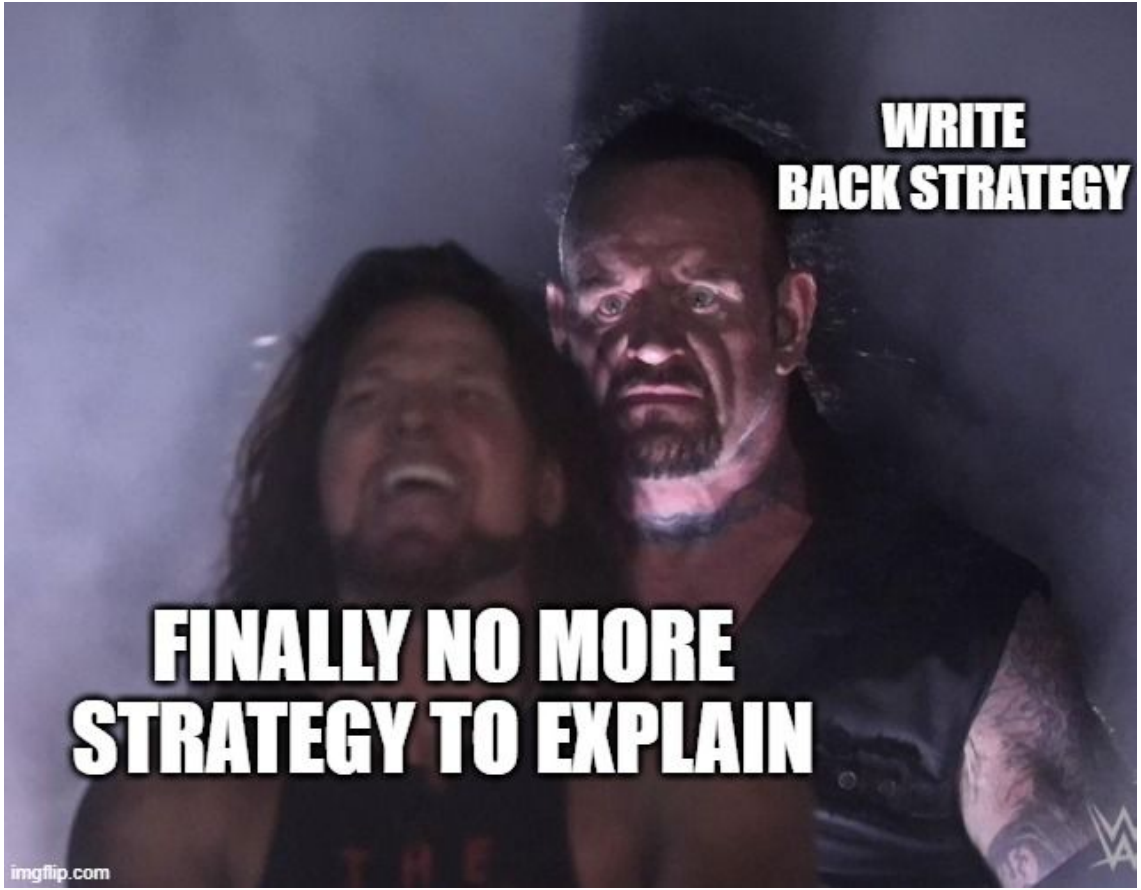
If Cache is down, request for write operation will fail Even if DB is available



Both Cache and DB must be available for successful operation

Write in Cache	**Write in DB**	Action Output
Successful	Failed (Unavailable)	**Roll back** write in cache
Failed (Unavailable)	Successful	**Roll back** write in DB
Successful	Successful	Okay !

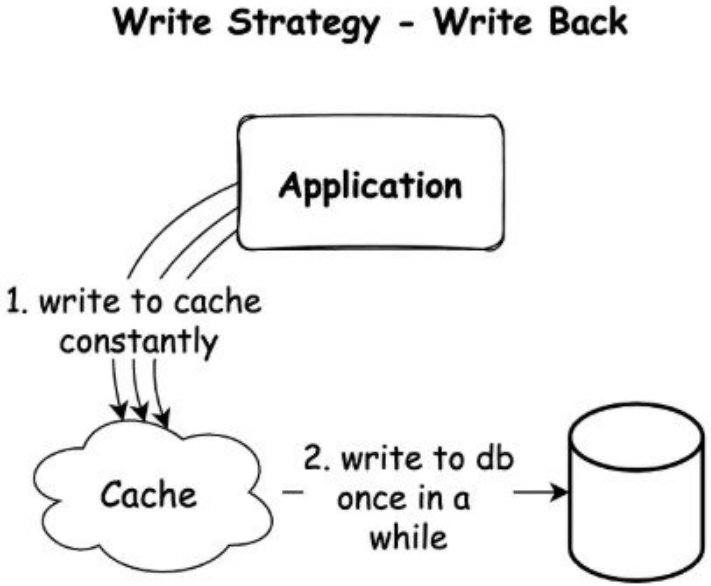
2 phase commit, need to be applied to maintain transaction property


WRITE
BACK STRATEGY
FINALLY NO MORE
STRATEGY TO EXPLAIN
imgflip.com

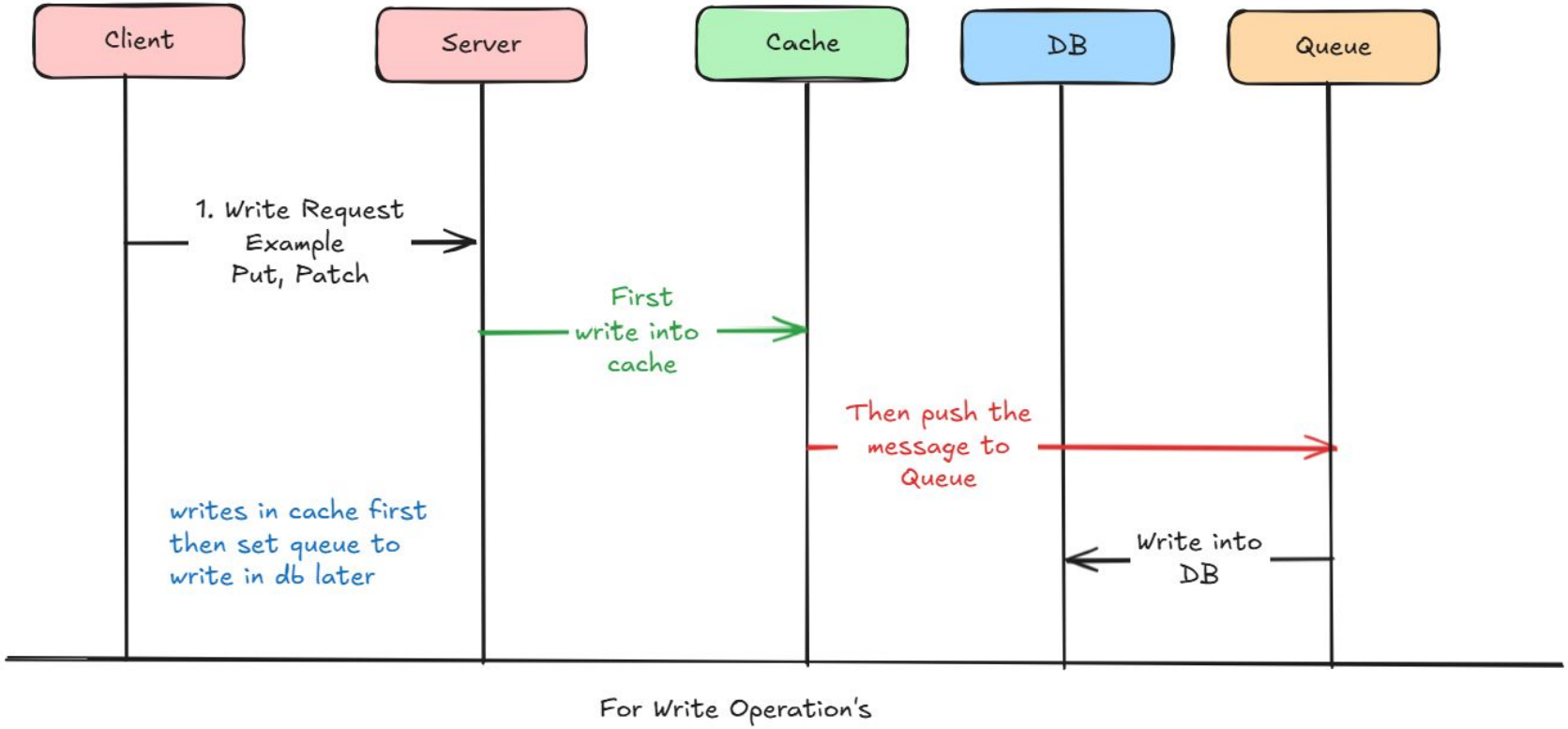
Write Back Caching Strategy

Write Back Strategy

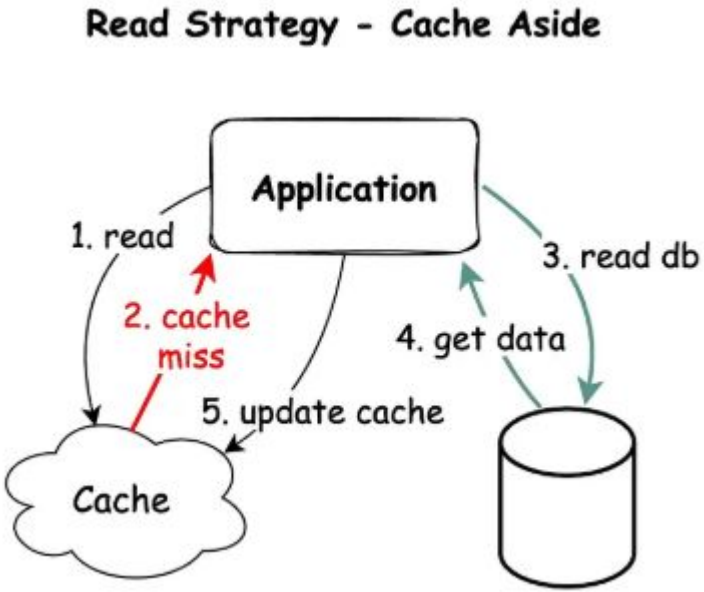
- First writes data into the cache
- Then in **asynchronous** write data into the DB
- For Read : Choose between
  - Cache Aside
  - Read Through

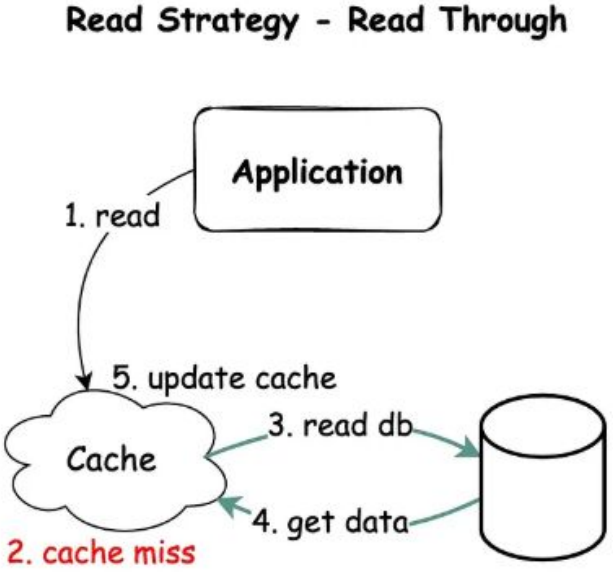


How write operation works?



What to do when read operations?





Have to choose a strategy between Cache Aside & Read Through

Write Back : Advantages

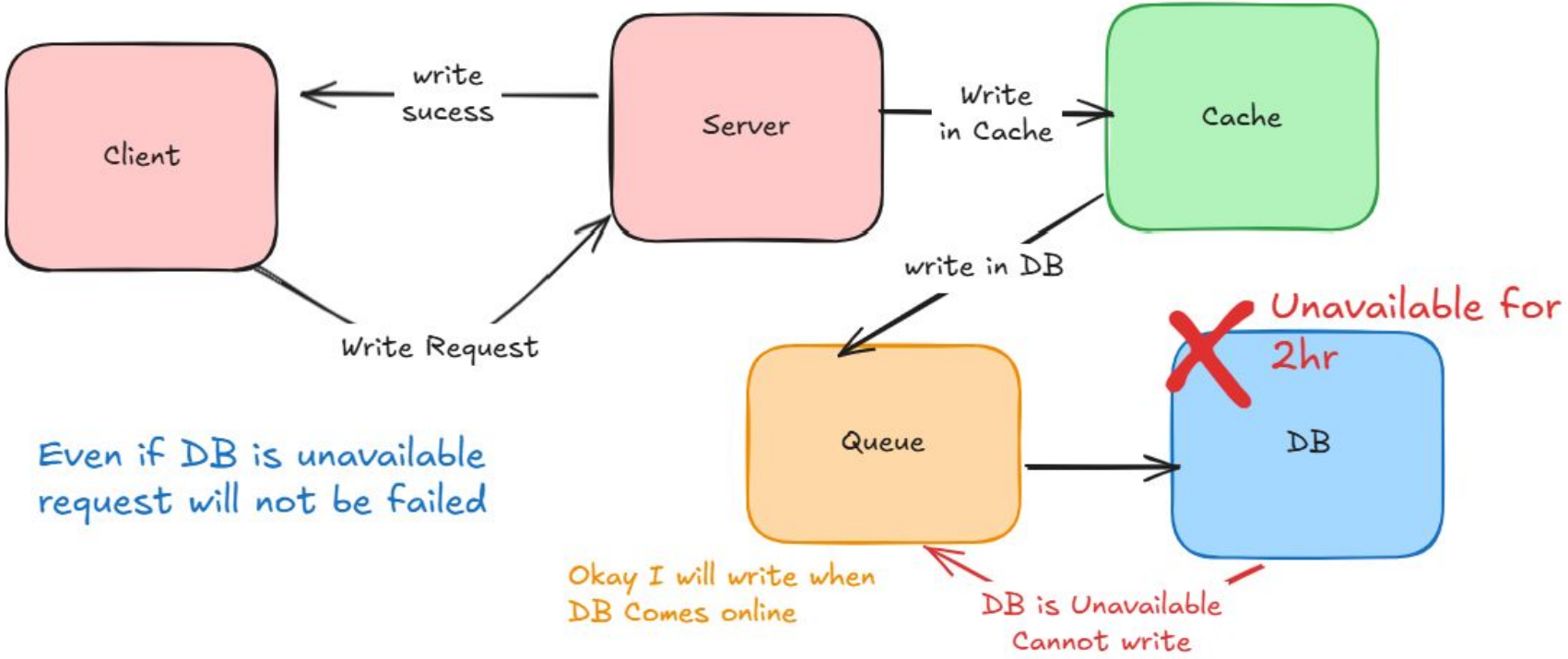
- Resolves the inconsistency problem between Cache and DB

- Increase Cache Hit chances : Even if for new data read there is a chance of cache hit

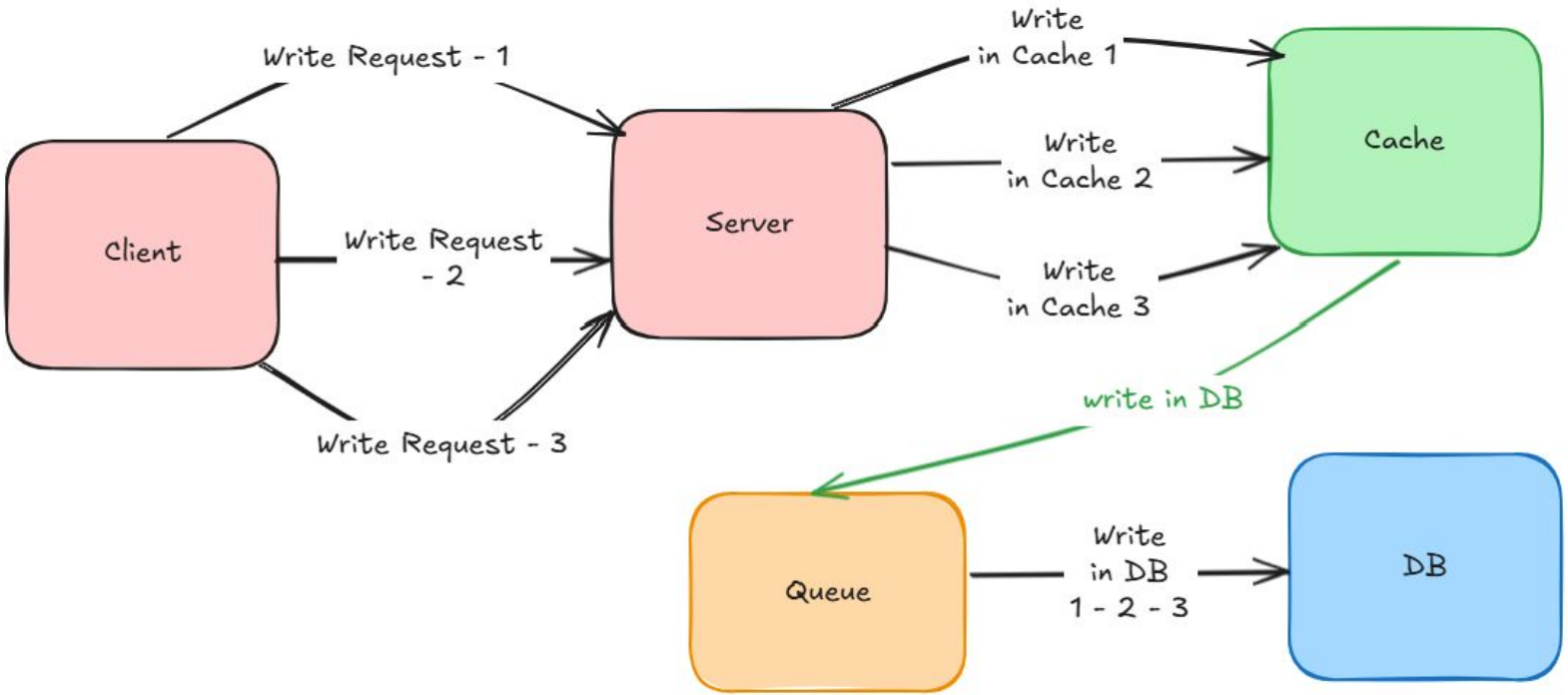
Write Back : Advantages

- Good for Write Heavy Applications
- Improves the write operation latency as writing into DB happens asynchronously
- Even if DB Fails, Write Operation will still works

Even if DB Fails, Write Operation will still works



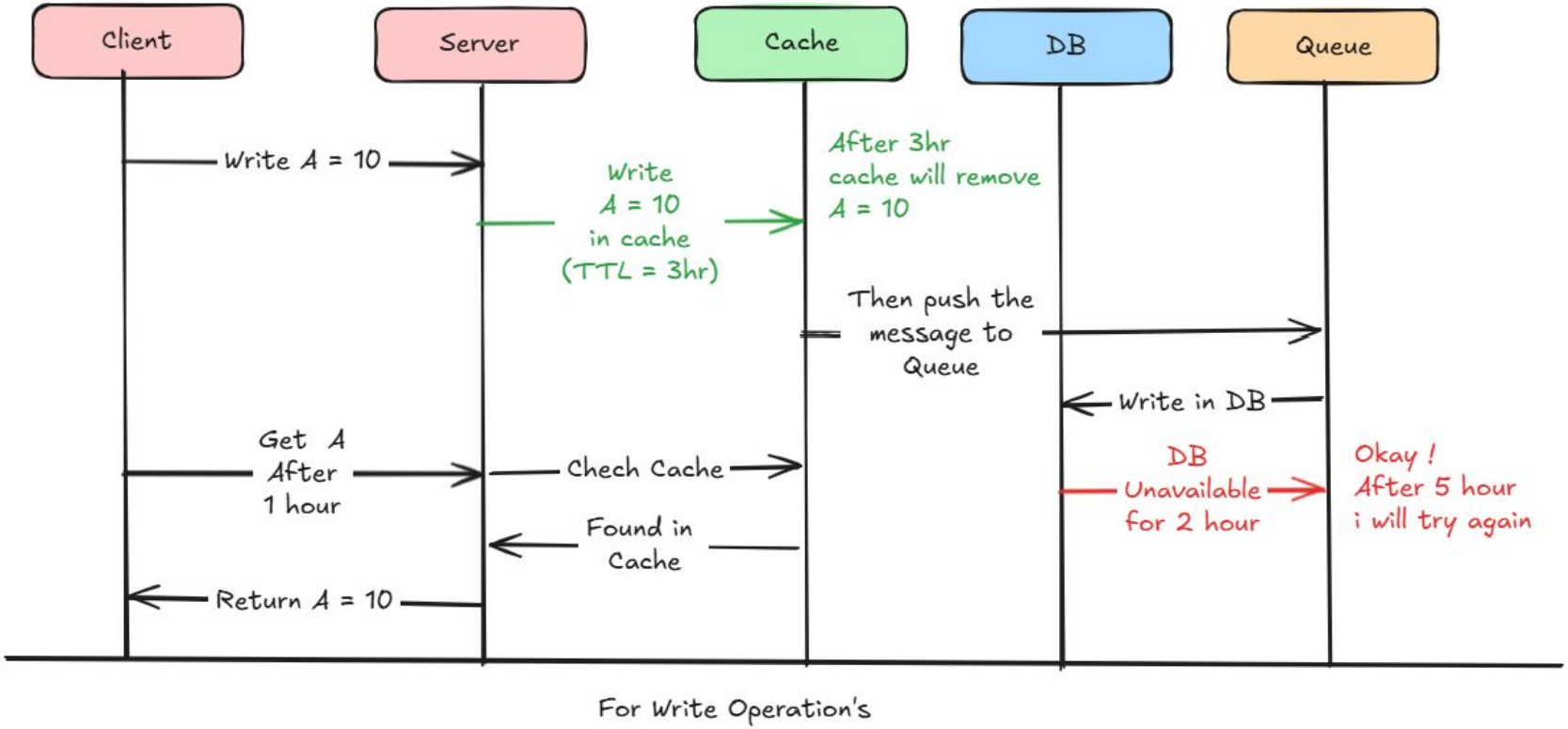
Improves the write operation latency as writing into DB happens asynchronously



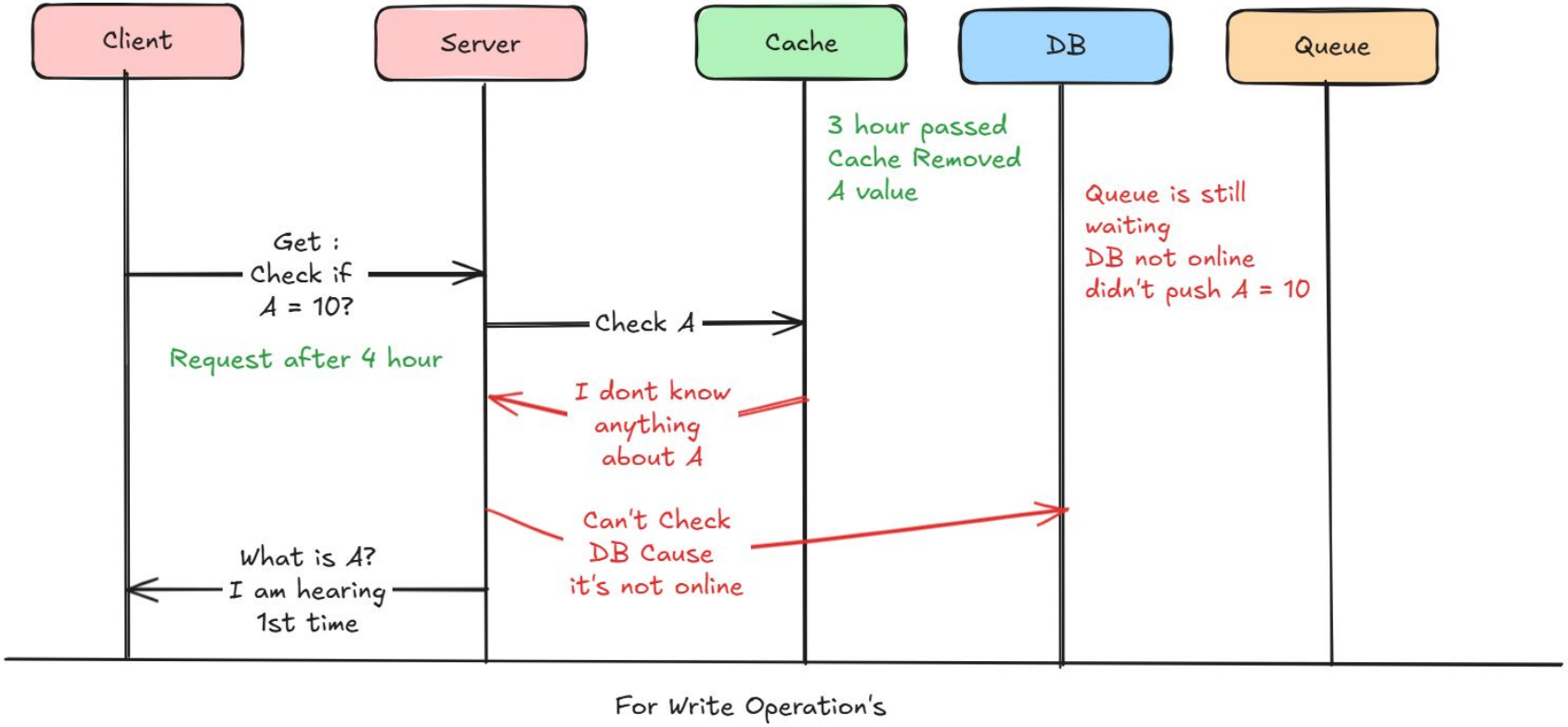
Write Back : Cons

- Chance of issues when data is removed from cache and DB write still not happen yet

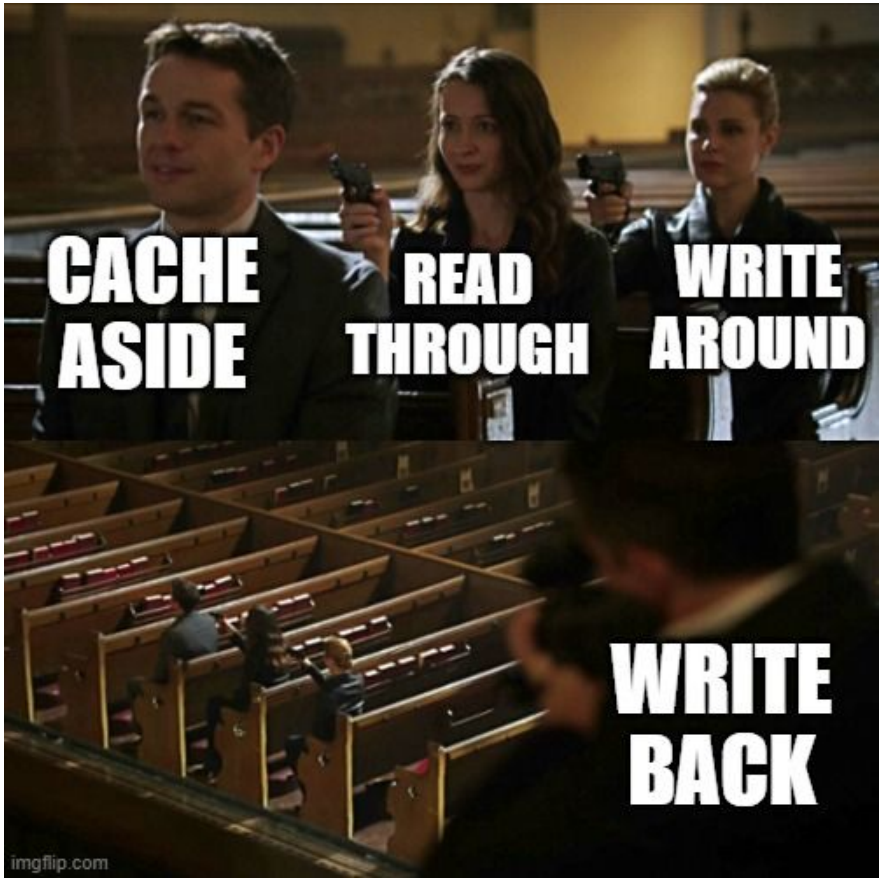
- Write-back caching improves performance but risks data loss, complexity, and consistency issues



For Write Operation's



For Write Operation's


CACHE ASIDE
READ THROUGH
WRITE AROUND
WRITE BACK
imgflip.com