Caching Basics, Strategies, and Modern Web Usages

K M Fatin Ishraq
Associate Software Engineer
Cefalo Bangladesh Ltd

Sumonta Saha Mridul
Trainee Software Engineer
Cefalo Bangladesh Ltd

Sanzida Afrin Promi
Trainee Software Engineer
Cefalo Bangladesh Ltd

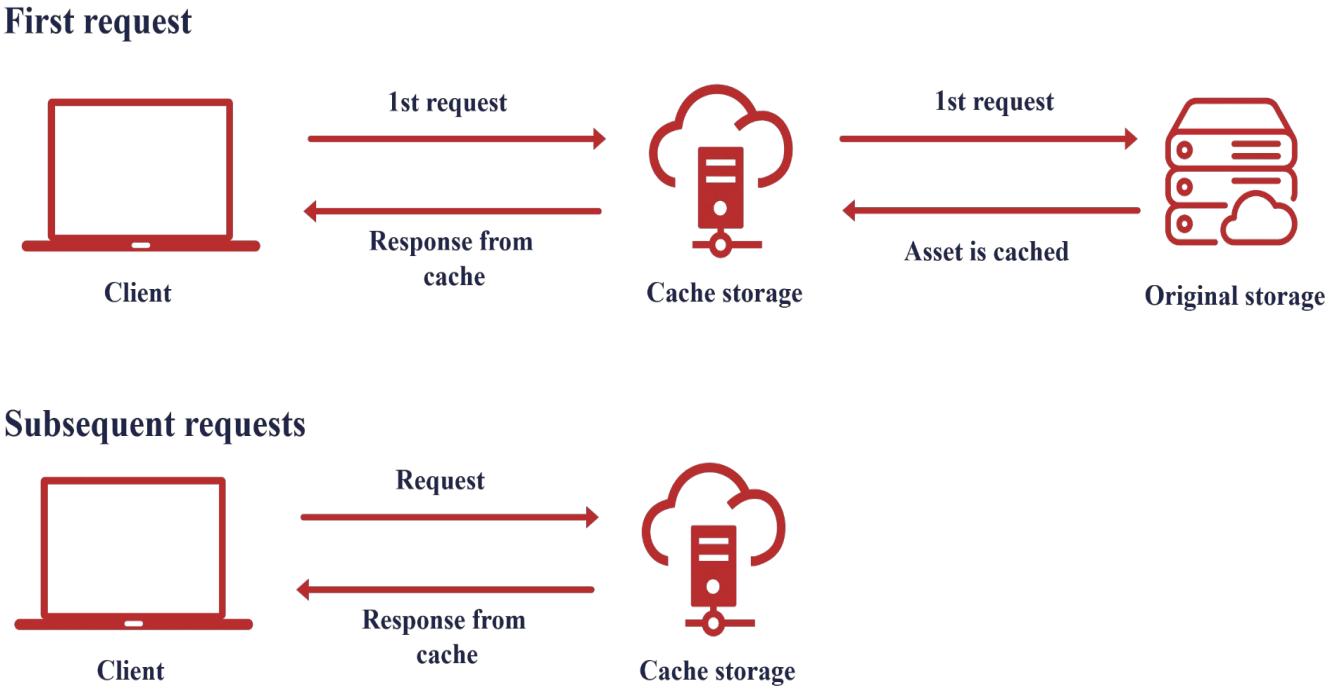
Caching

An Overview of HTTP Caches

Sanzida Afrin Promi
Trainee Software Engineer
Cefalo Bangladesh Ltd

What is Caching?

Caching is the process of storing copies of given resource temporarily in a cache .



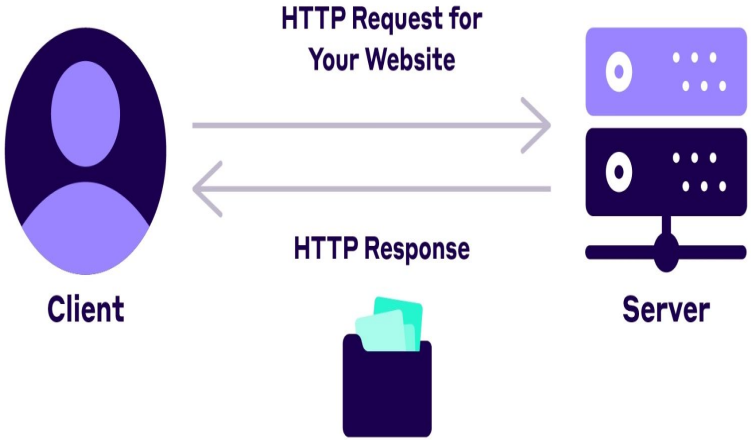
Web Caching

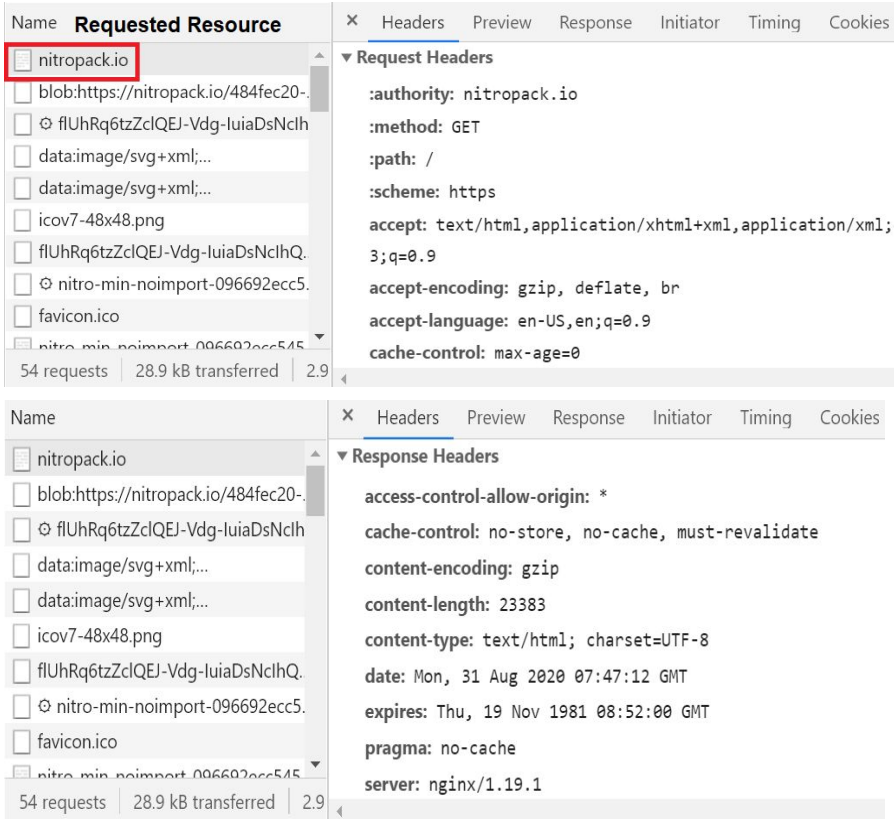
Web caching involves storing copies of web files, like HTML pages or images, on a user's device or intermediary servers.

Of all the ways to improve website performance, caching is the most important one.

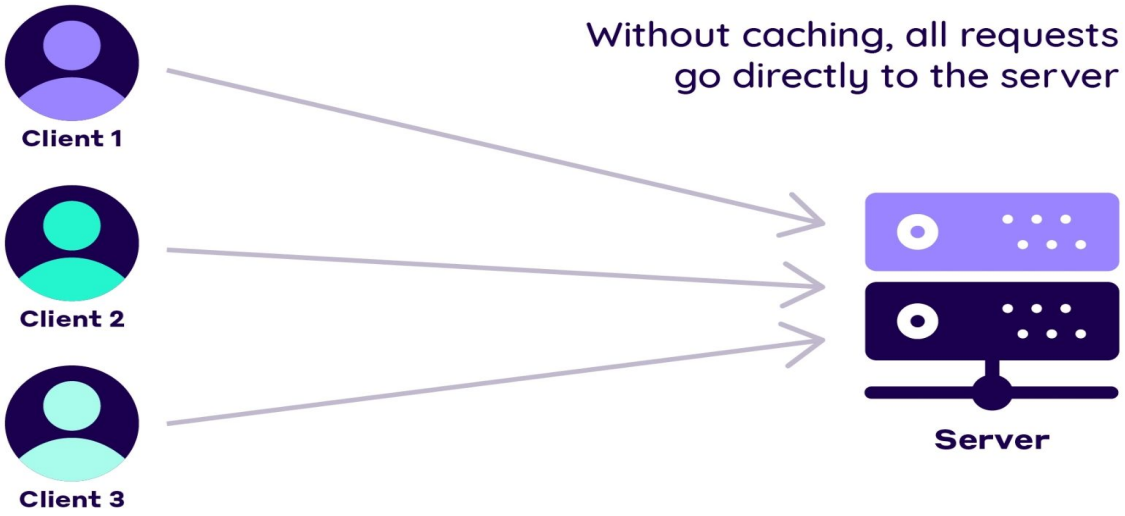
How the Web Works?

How the Web Works?





Why Every Web Service Needs a Caching Layer?

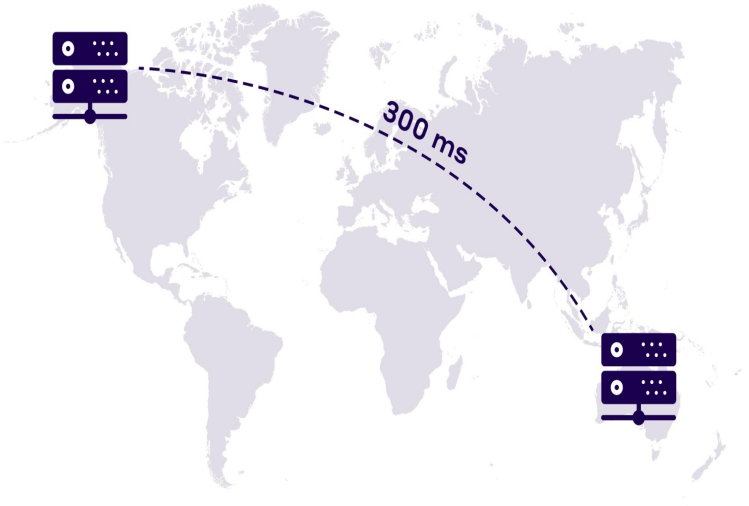


Origin server has to handle all incoming requests.

Why Every Web Service Needs a Caching Layer?

- Web server is probably pretty far away from most visitors.This increases the latency and makes your website slower than it should be.

- Second, repeat visitors have to re-download the same files every time. That's a massive waste of bandwidth.



How Web Caching Works?

Storing a copy of your website's resources in a different place called a web cache.

Their job is to sit between the origin server and the user and save HTTP responses - HTML documents, images, CSS files, etc.

Two crucial distinctions Of web caching

- Browser caching;
- Proxy server caching

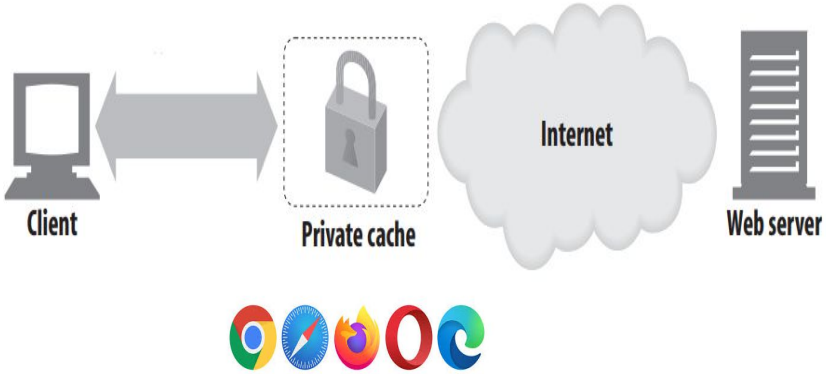
Benefits of caching

- Reduce redundant data transfers
- Reduce latency or distance delays
- Reduce demand on origin server
- Hide network failures


LEFT
EXIT 12
Server
Cache
Client
imgflip.com

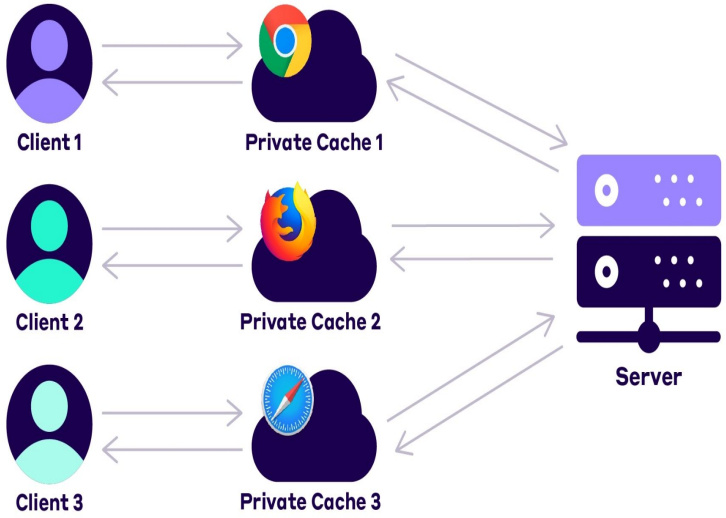
Private Cache

- Dedicated to a single user
- Web browsers have built in private caches
- Improves offline browsing of cached content
- E.g. cache of Firefox, Google Chrome



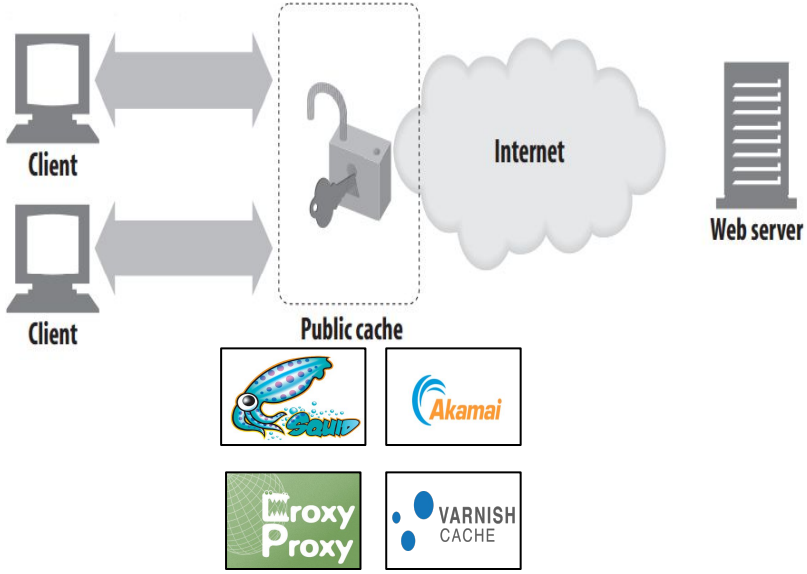
Browser Cache

- A temporary storage area in memory or on disk
- Holds the most recently downloaded Web pages
- Typically cache what are known as "static assets"



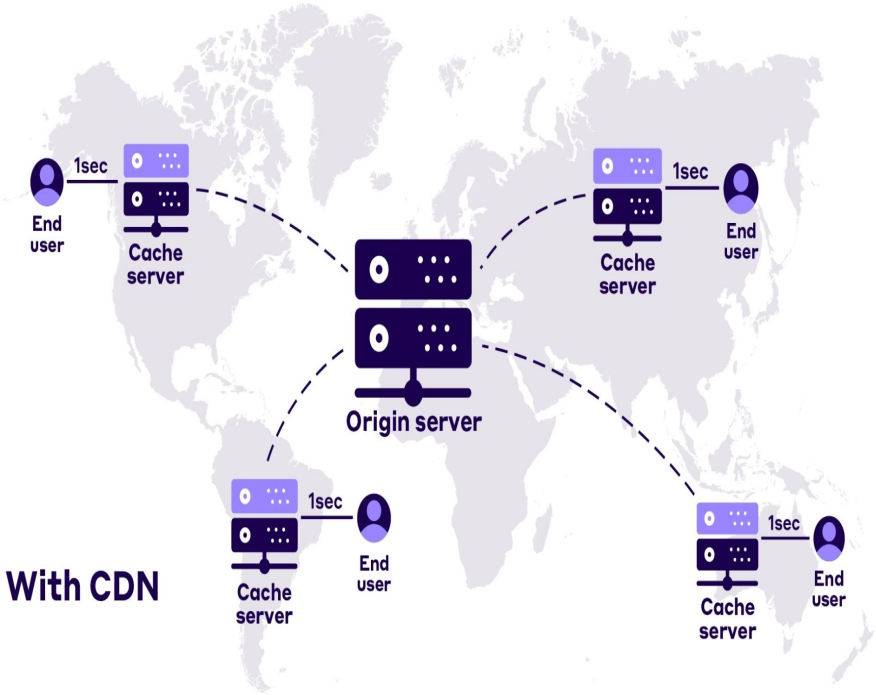
Public Cache

- Known as shared proxy caches
- Responses to be reused by more than one user
- Popular resources are reused a number of times
- Reducing network traffic and latency
- E.g. proxy and reverse proxy cache



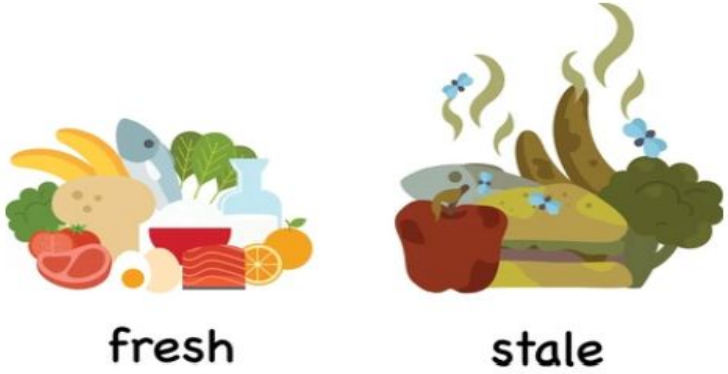
Proxy Cache

- Interpret and respond to requests on behalf of the original server
- Used for saving bandwidth, authentication, request filtering, response filtering etc.
- ISPs and large corporations often set them up on their firewalls as intermediaries



Life Cycle of Cached Contents

- Fresh : response is still valid and can be reused
- Stale : response has already expired.



How do we control who can cache the content?

When is cached content considered fresh?

What to do if the content gets stale?

Caching Rules

Defined by HTTP headers

HTTP Headers Related to Caching

- Cache-Control
- Pragma
- Range
- Expires
- Age
- If-Modified-Since
- If-None-Match
- If-Unmodified-Since
- If-Match
- ETag
- Last-Modified
- Vary
- Warning

Cache-Control Directives

- public
- private
- no-cache
- no-store
- Must-revalidate
- proxy-revalidate
- max-age=<seconds>
- s-maxage=<seconds>
- immutable
- stale-while-revalidate=<seconds>
- stale-if-error=<seconds>
- max-stale[=<seconds>]
- min-fresh=<seconds>
- only-if-cached


GET /styles/main.css HTTP/1.1
Accept-Encoding: gzip, deflate, br
Client
Server
HTTP/1.1 200 OK
Date: Wed, 27 Dec 2024 12:00:00 GMT
Content-Type: text/css
Content-Length: 4520
Connection: keep-alive
Cache-Control: max-age=3600, public
Expires: Wed, 27 Dec 2024 13:00:00 GMT
Last-Modified: Tue, 26 Dec 2024 10:00:00 GMT
ETag: "abc123"
Age: 300
Vary: Accept-Encoding
Server: Apache/2.4.41 (Ubuntu)
Client
Server

How do we control who can cache the content?

- Cache-Control: public  

- Cache-Control: private 

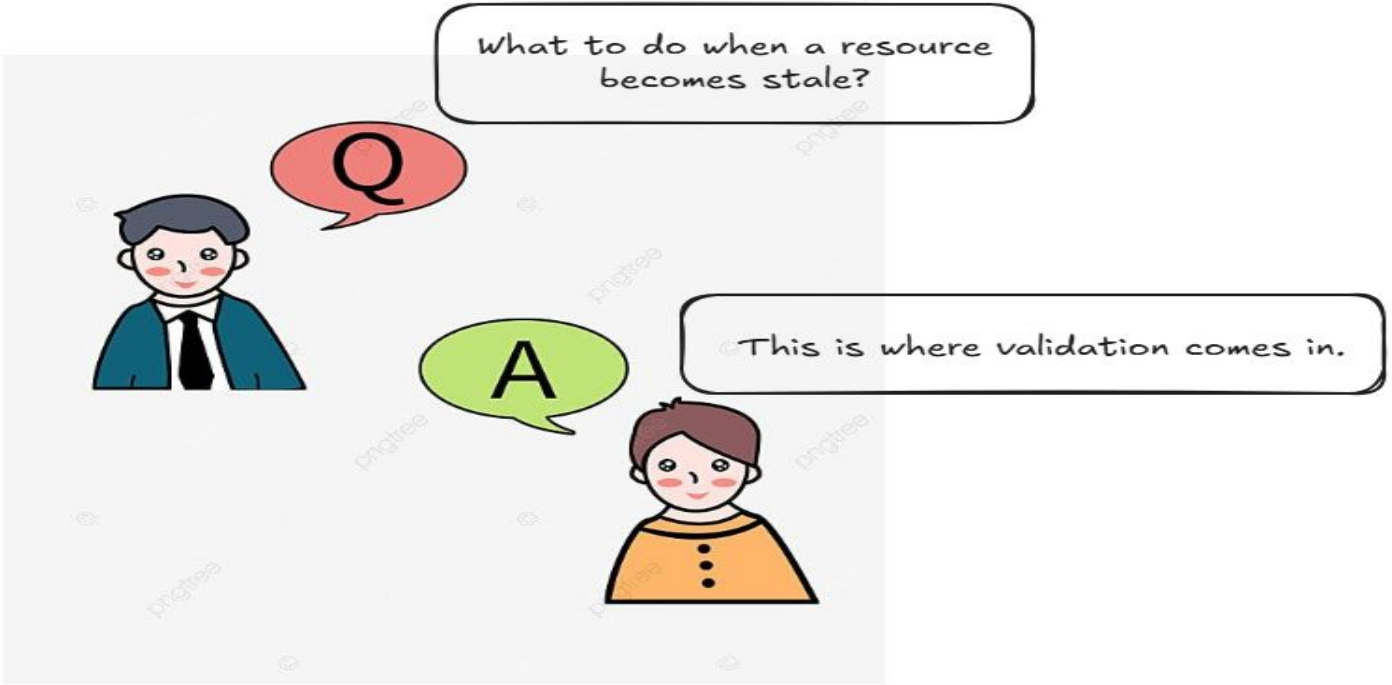
A Content : Fresh or Stale?

Fresh:

- Elapsed time is within the **max-age** duration.
- Current date is within the **Expires** value.

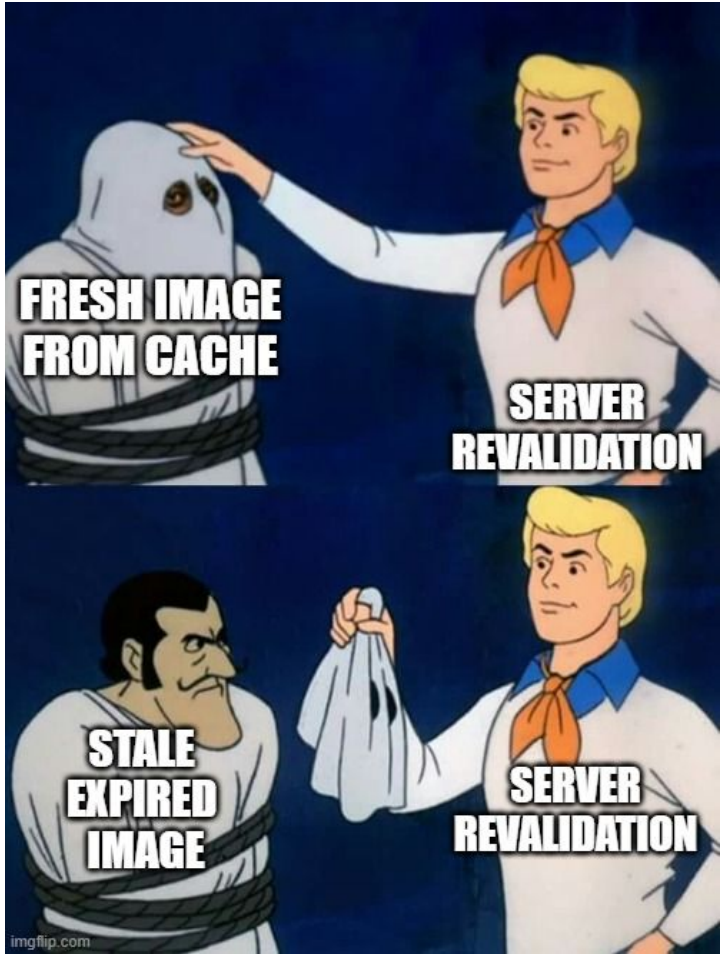
Stale:

- Elapsed time exceeds the **max-age** duration.
- Current date is past the **Expires** value.

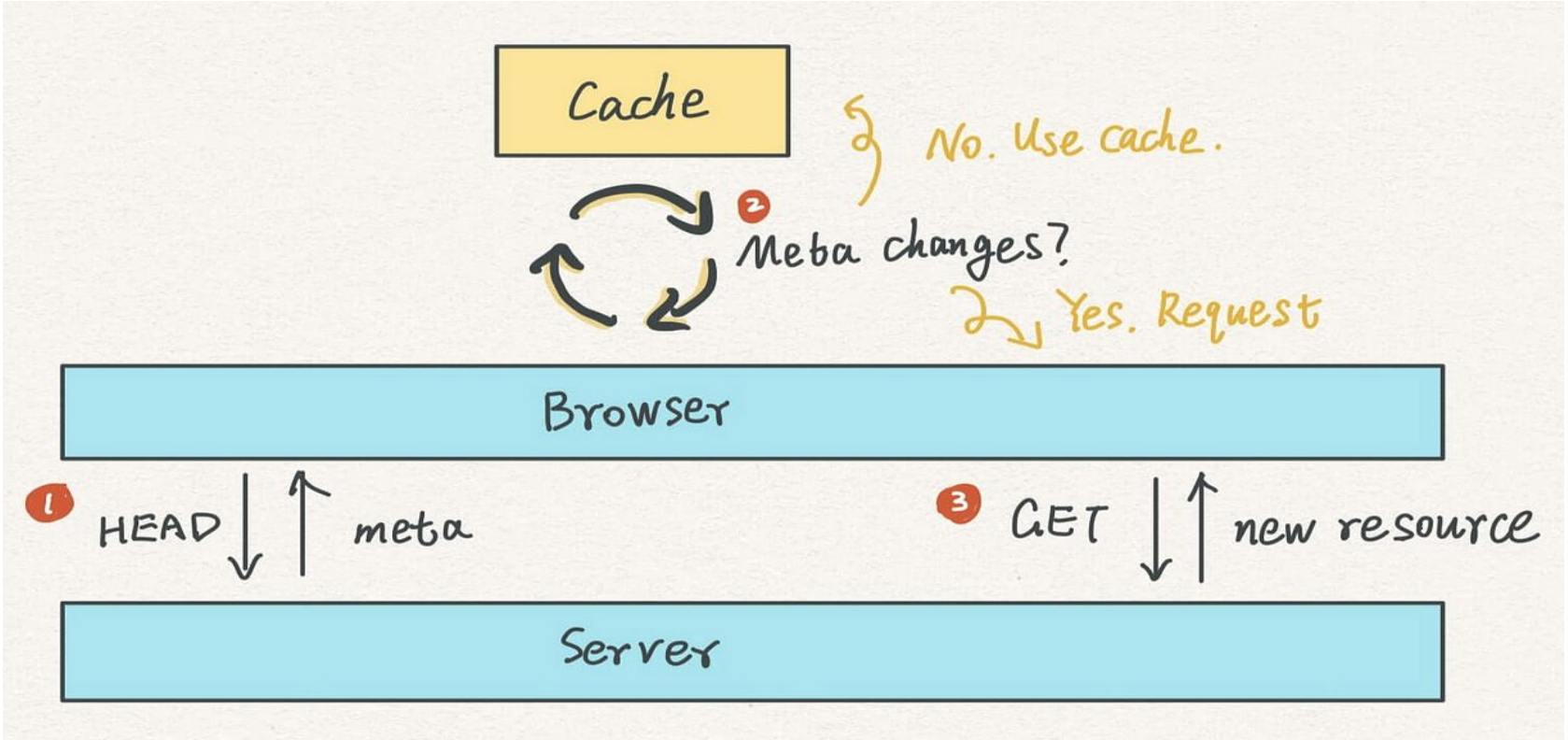

What to do when a resource becomes stale?
Q
A
This is where validation comes in.

Server Revalidation : Why needed?

- A cached document expiration doesn't mean it's different from the origin server
- Cache needs to ask the origin server whether the document has changed
- Unmodified Cached content can be reused.


FRESH IMAGE
FROM CACHE
SERVER
REVALIDATION
STALE
EXPIRED
IMAGE
SERVER
REVALIDATION
imgflip.com

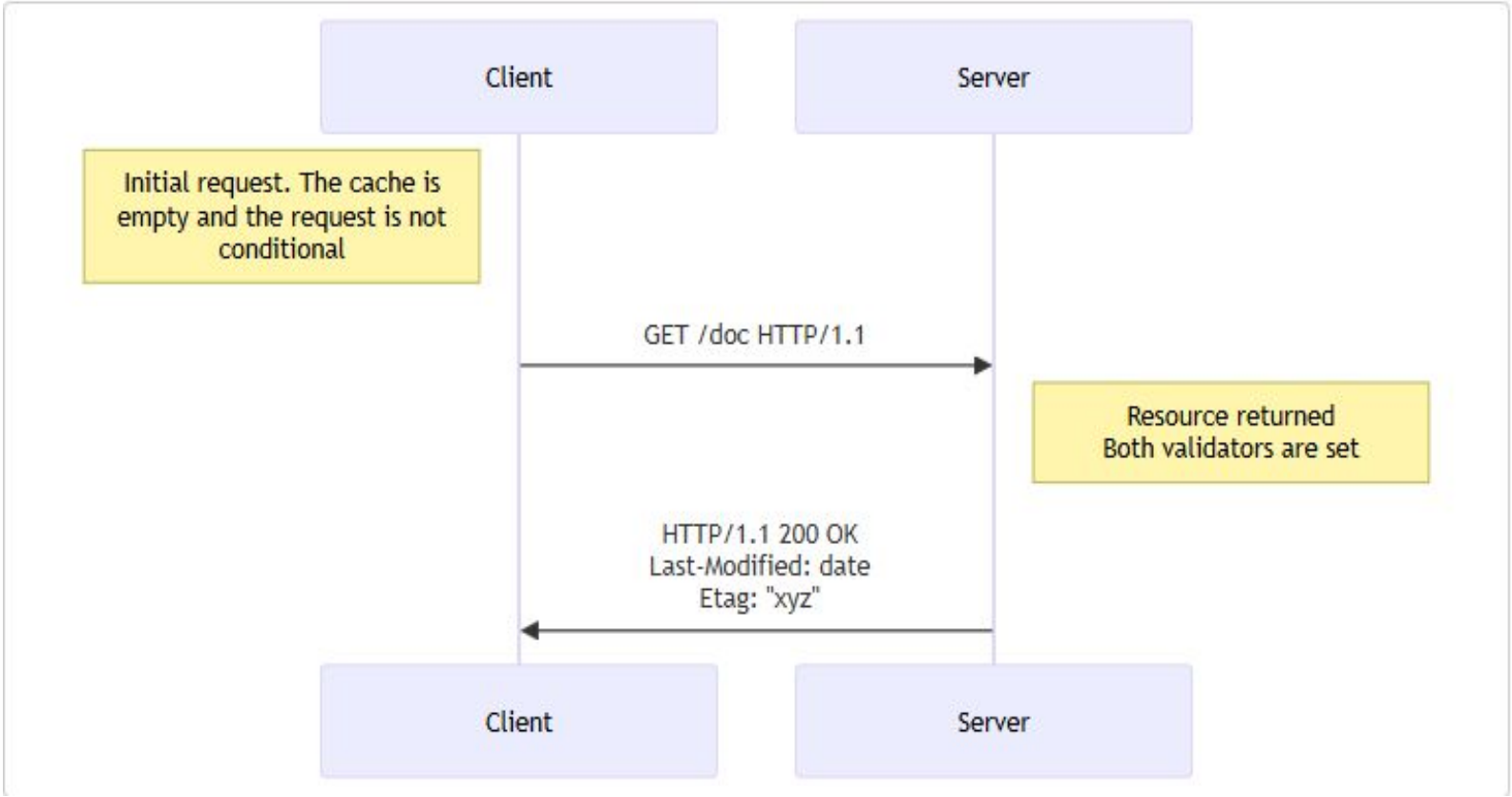
Server Revalidation



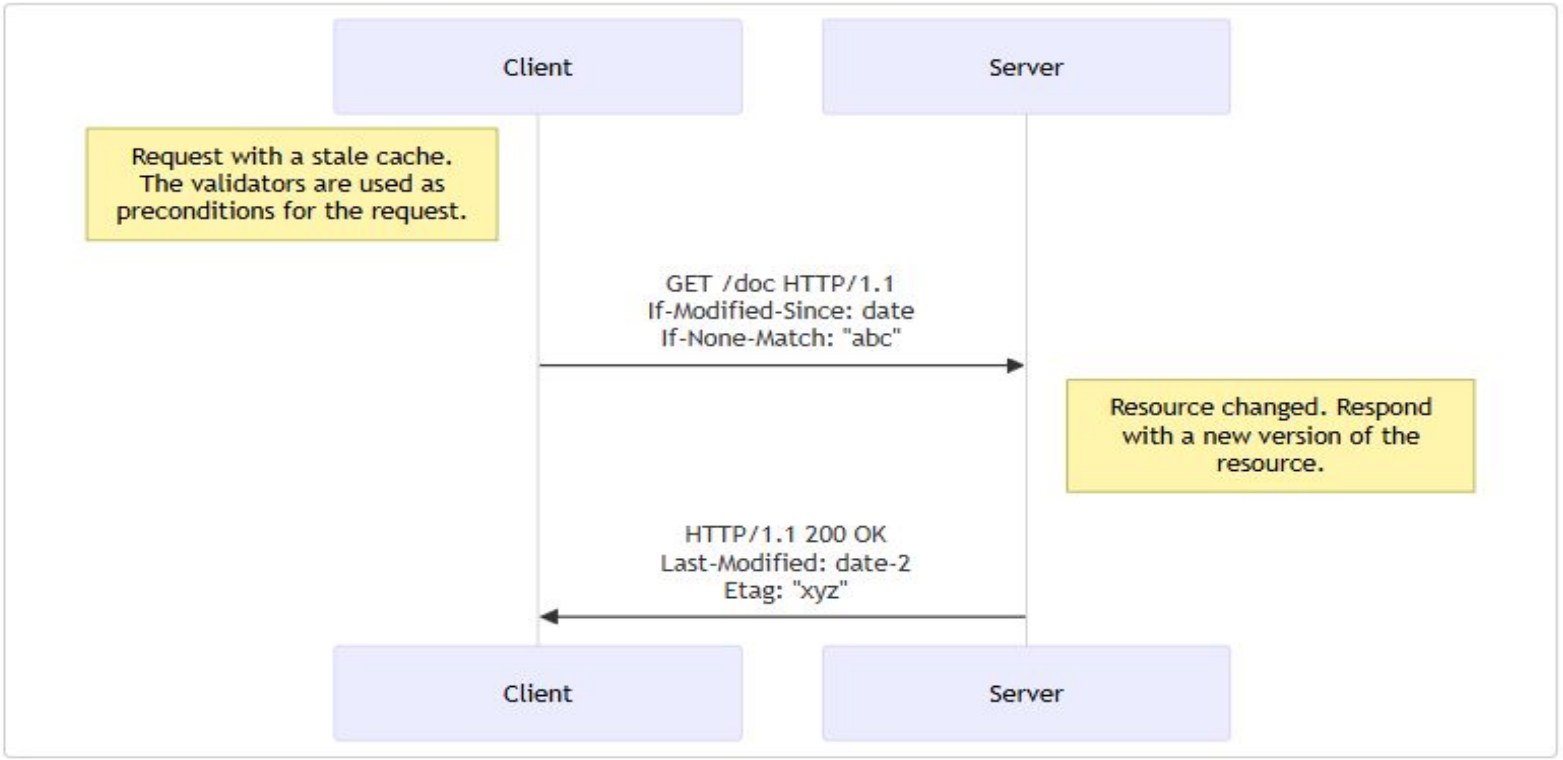
How is validation performed for cached resources?

- Using the Last-Modified and If-Modified-Since headers
- Using the Etag and If-None-Match header

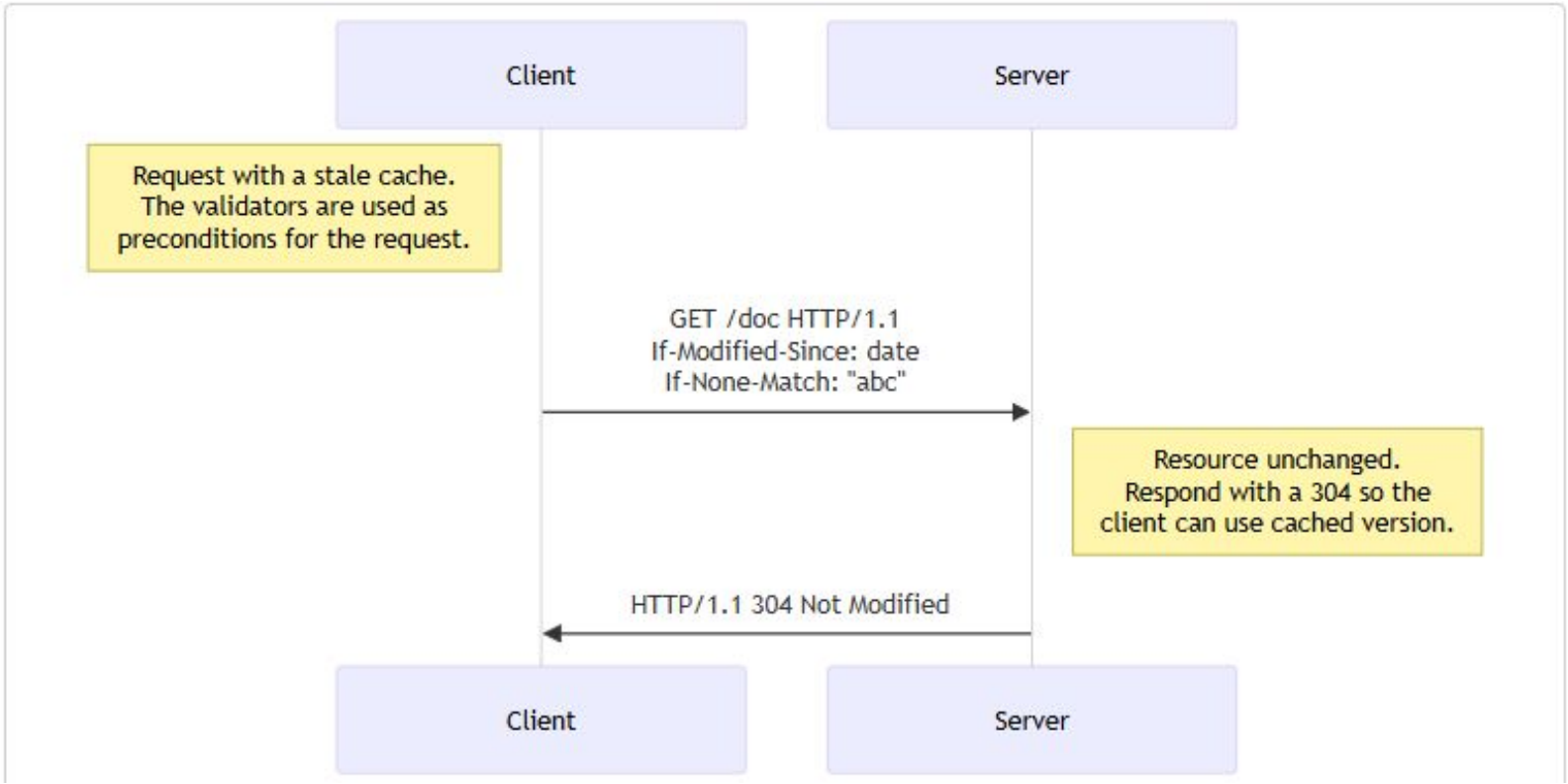
Revalidation



If-Modified-Since : true
If-None-Match : true



If-Modified-Since : False
If-None-Match : False



Force Revalidation

- To always fetch the latest content from the server.
- By adding **`Cache-Control: no-cache`** to the response.

Don’t Cache

When dealing with

- Sensitive data
- Dynamic or frequently-changing data
- Security concerns

By adding **`Cache-Control: no-store`** to the response.

Avoiding Revalidation

Cache-Control: max-age=31536000, immutable

The **immutable** directive explicitly indicates that revalidation is not required because the content never changes.

Deletion of Cached Contents

Manual Deletion: Users can clear browser cache via settings.

Cache Expiry: Content is replaced or revalidate when it exceeds max-age or Expires.

Cache Invalidations: Developers use techniques like versioned URLs to force deletion.

No-Store Directive: Prevents caching entirely, avoiding stored content

Cache Busting

When: Done when updated resources need to replace cached ones.

Why: Prevents users from accessing stale content.

How: Use versioned URLs (e.g., file-v2.js) or query strings (e.g., file.js?v=2).

Purpose: Ensures users get the latest updates seamlessly.

Some Best Practices While Caching

- If both **Expires** and **Cache-Control: max-age** are available, **max-age** is defined to be preferred.
- Using
  Cache-Control:max-age=3600,must-revalidate,proxy-revalidate
  together
- Use of kitchen-sink headers to deal with the outdated implementation.
  Cache-Control: no-cache,max-age=0, must-revalidate
  Cache-Control: no-store,no-cache,max-age=0, must-revalidate

Some Best Practices While Caching

- **Use ETag** for dynamic or frequently changing resources to ensure precise, content-based validation.
- **Use Last-Modified** for a simple solution when resource modification times are reliable and consistent.

What Happens When Cache-Control Information is Missing?

Browsers use heuristics (rules or algorithms) to decide how long to keep a cached resource, often based on the last-modified date or the age of the cached resource. This is called **Heuristic Caching .**

A Quick Overview of

Server Side Caching Strategy

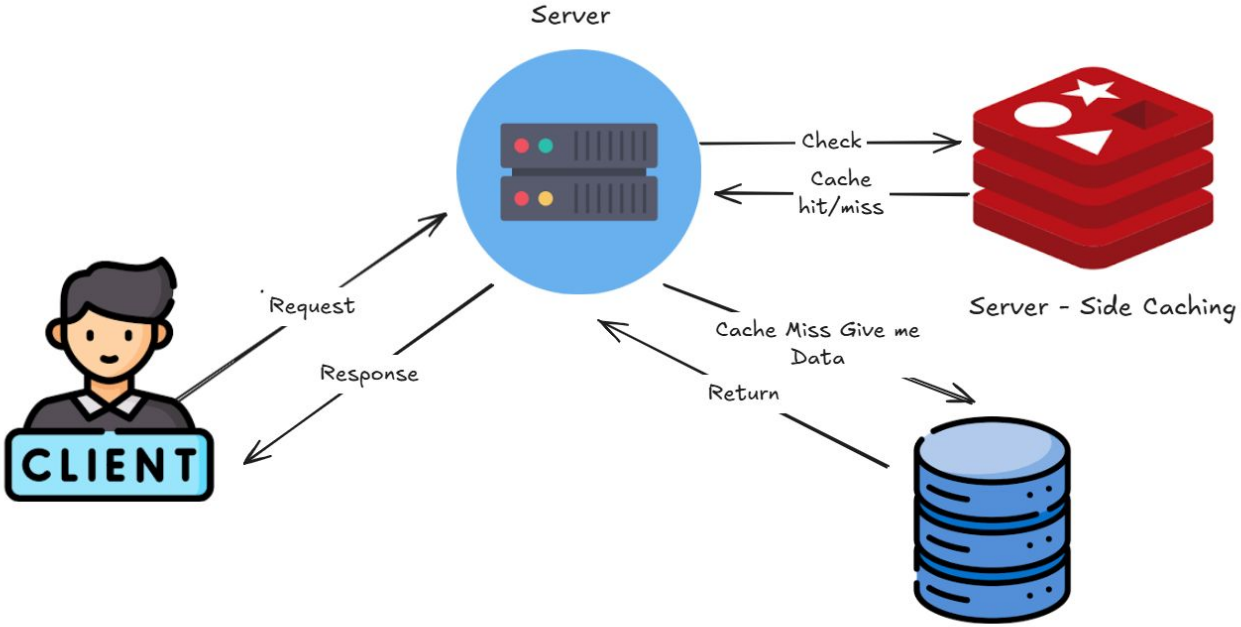
Sumonta Saha Mridul

Trainee Software Engineer

Cefalo Bangladesh LTD.


ME
SANZIDA
FATIN BHAI
imgflip.com

What is Server Side Caching?



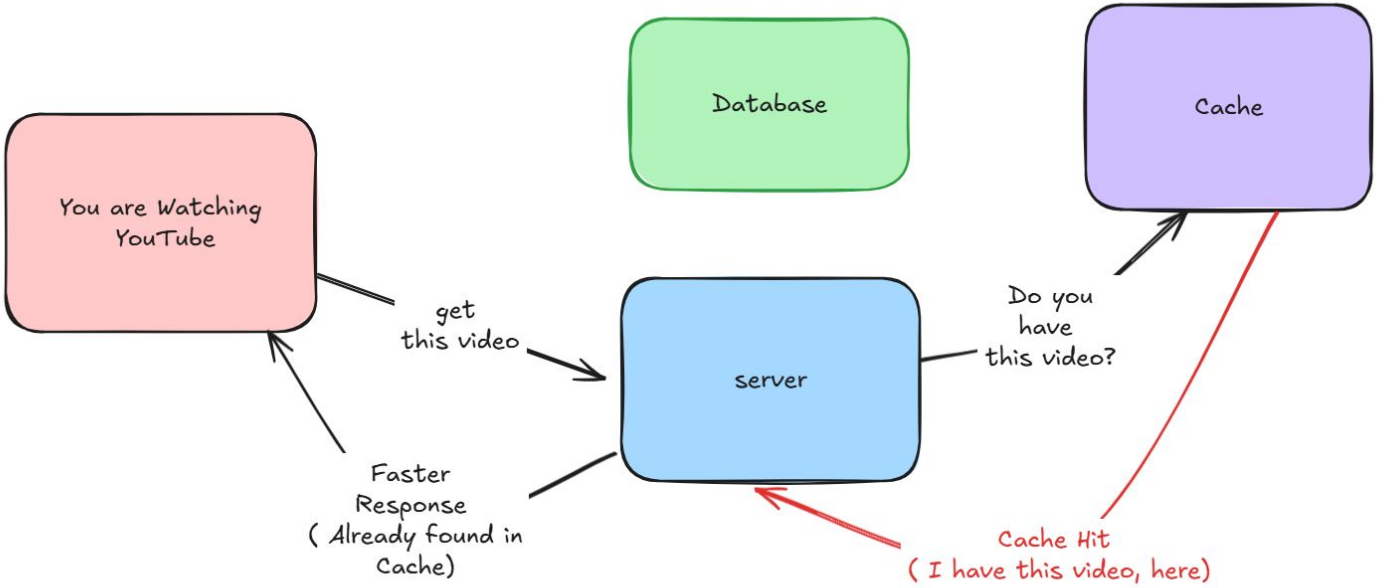
Server-side caching generally refers to the cache layer positioned between the server's application logic and the database.

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.

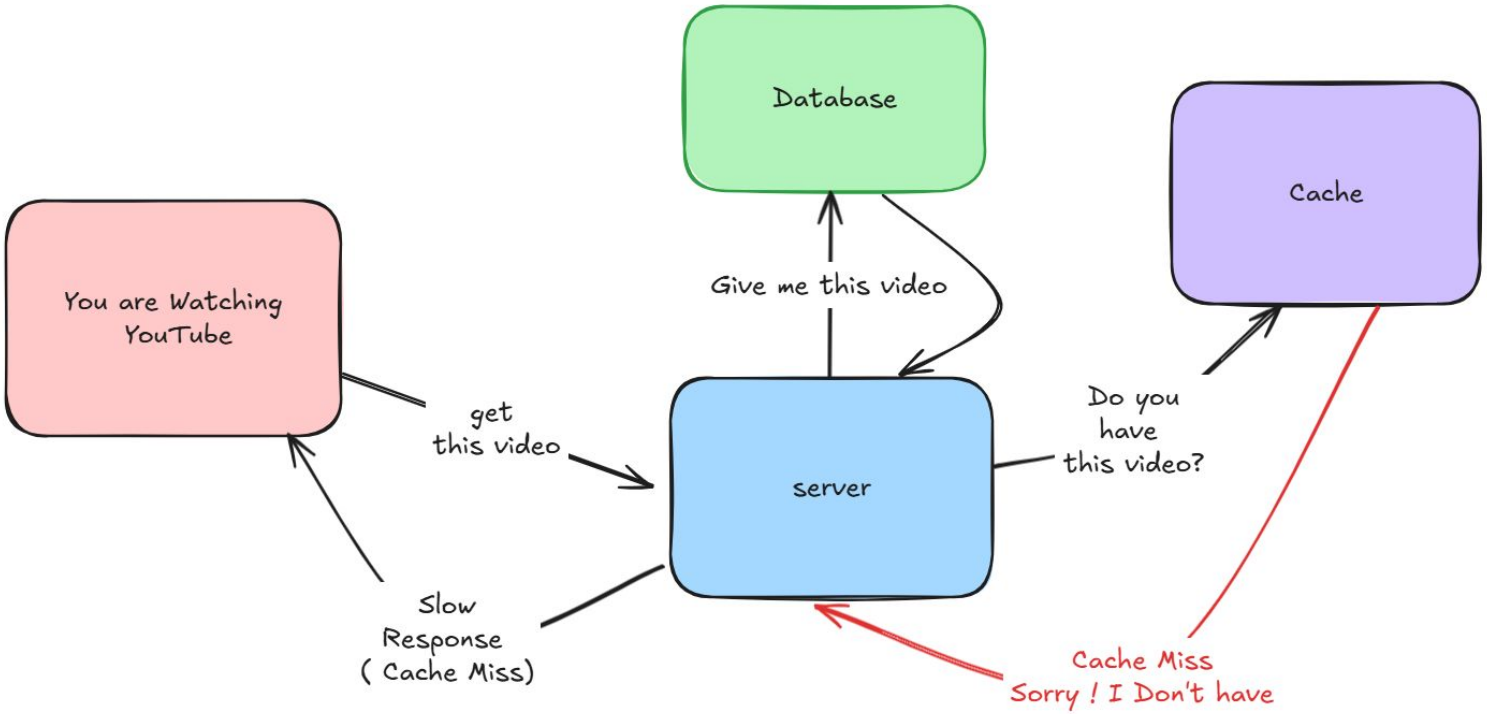
What is Cache Hit?



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?

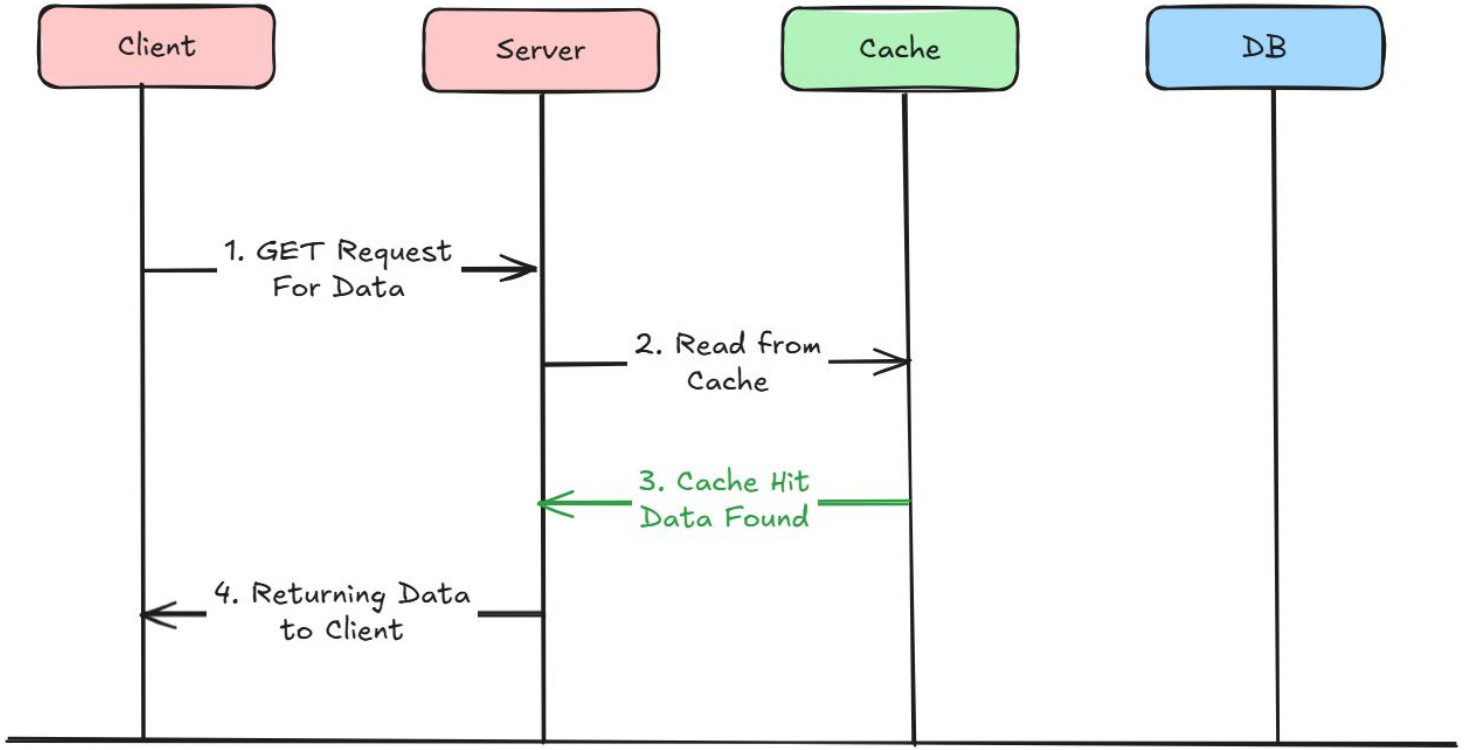


Cache Aside Caching Strategy

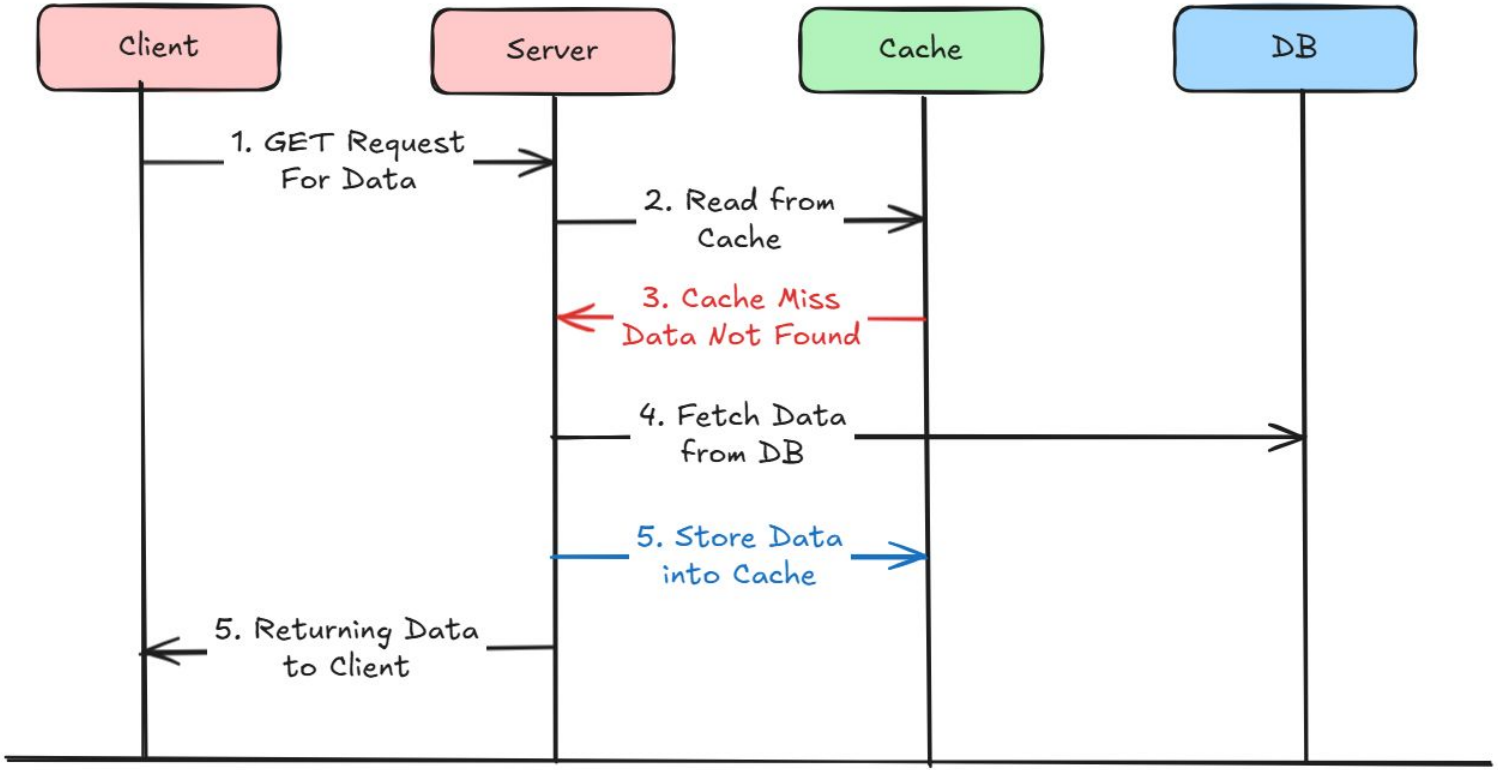
Cache Aside Strategy

- 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?



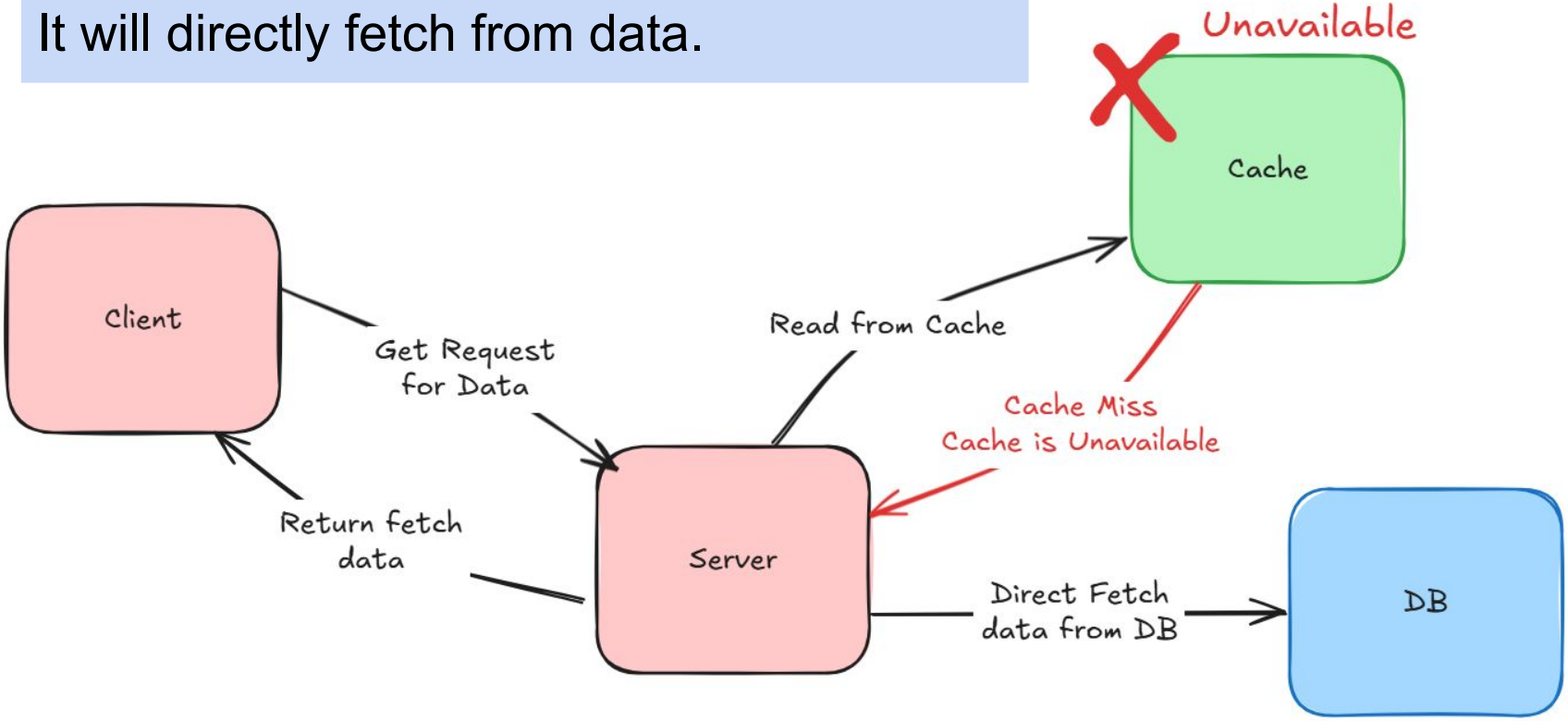
What happens in Cache Miss?



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)

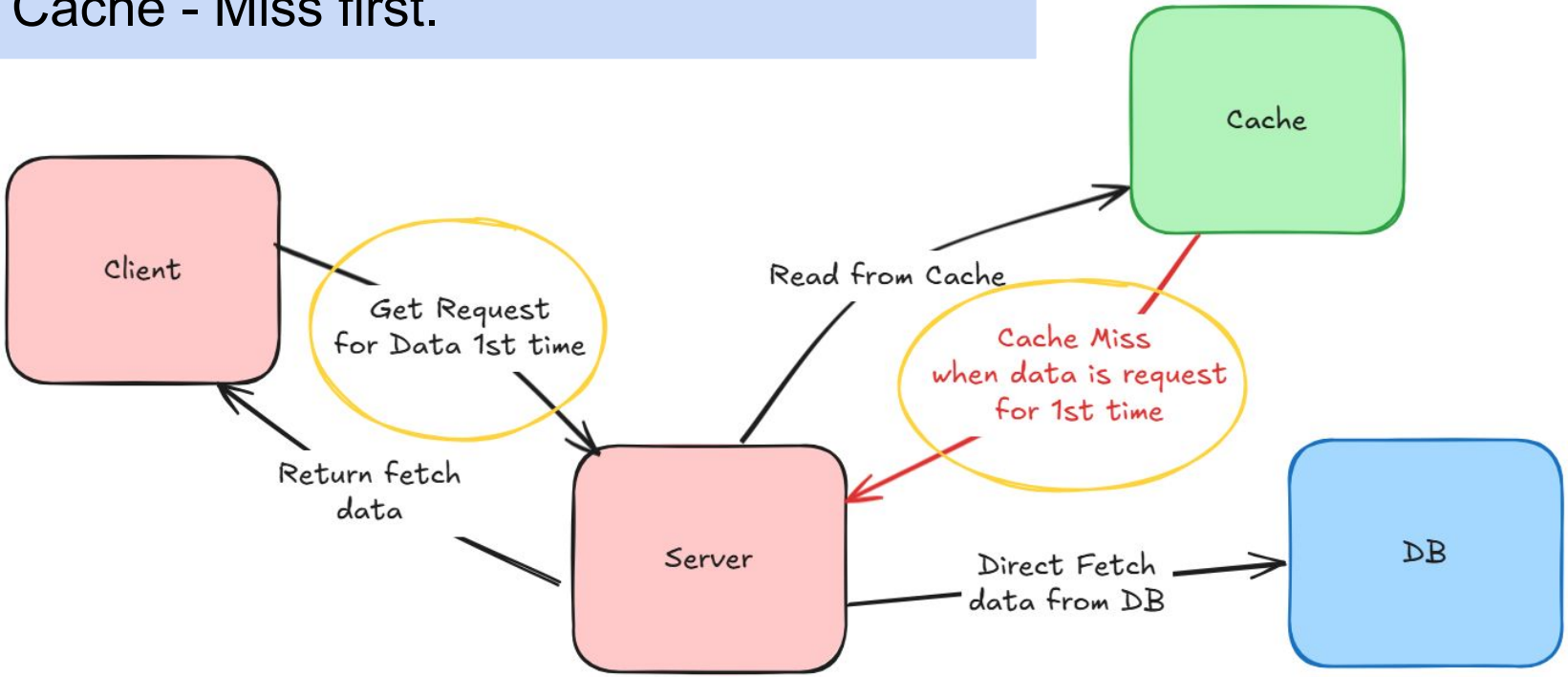
```json
{
  "UserID": 1,
  "Name": "John Doe",
  "Email": "john.doe@example.com",
  "RecentPosts": [
    {
      "PostID": 101,
      "Content": "Hello, world!"
    },
    {
      "PostID": 102,
      "Content": "Loving the sunny weather today!"
    }
  ]
}
```

Cache (Optimized for Access)

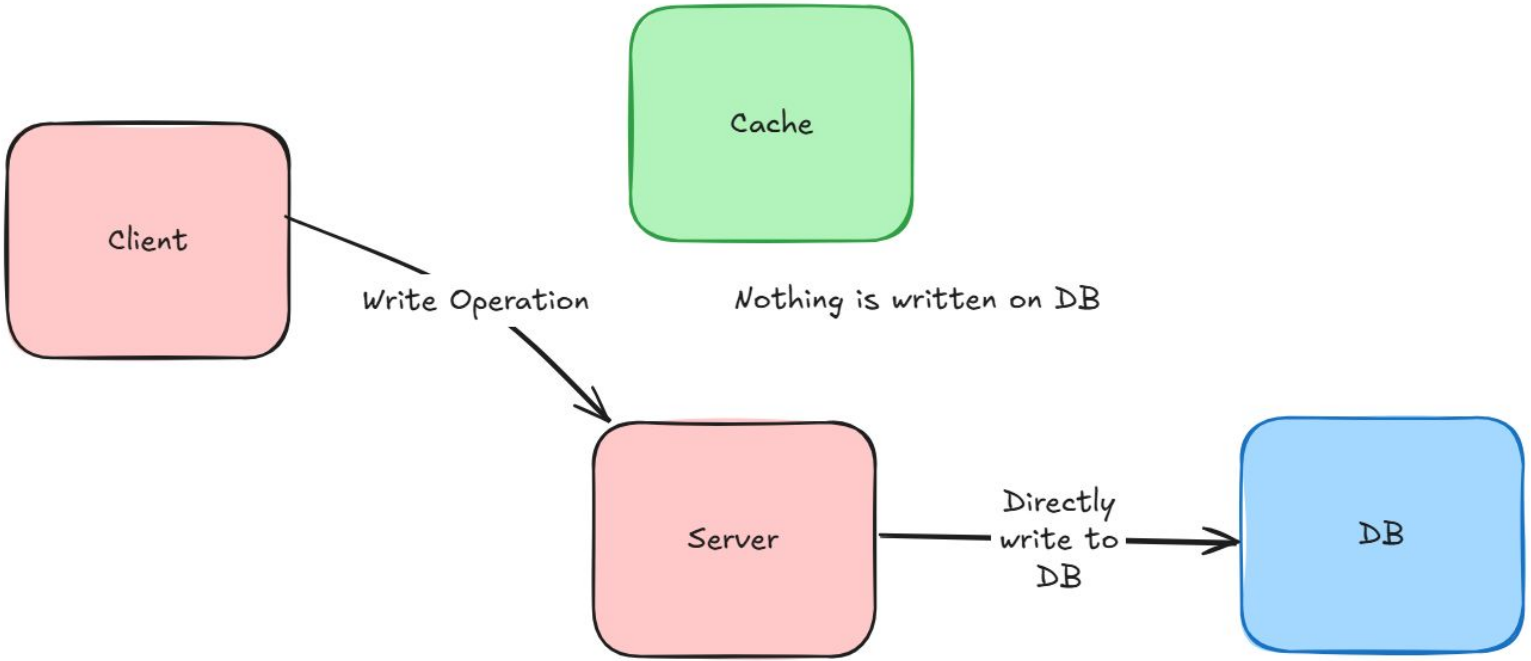
Cache Aside : Cons

- 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

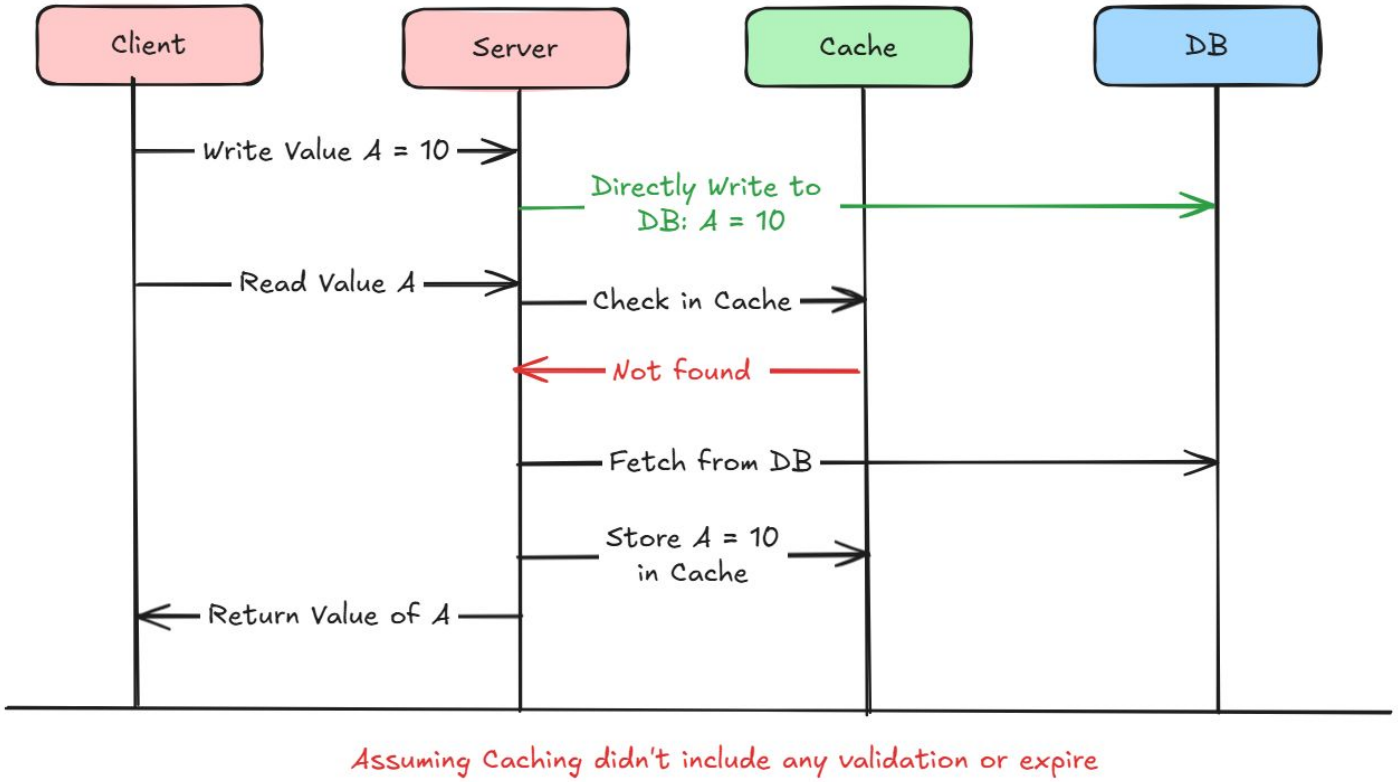
For new data read, there will always be Cache - Miss first.



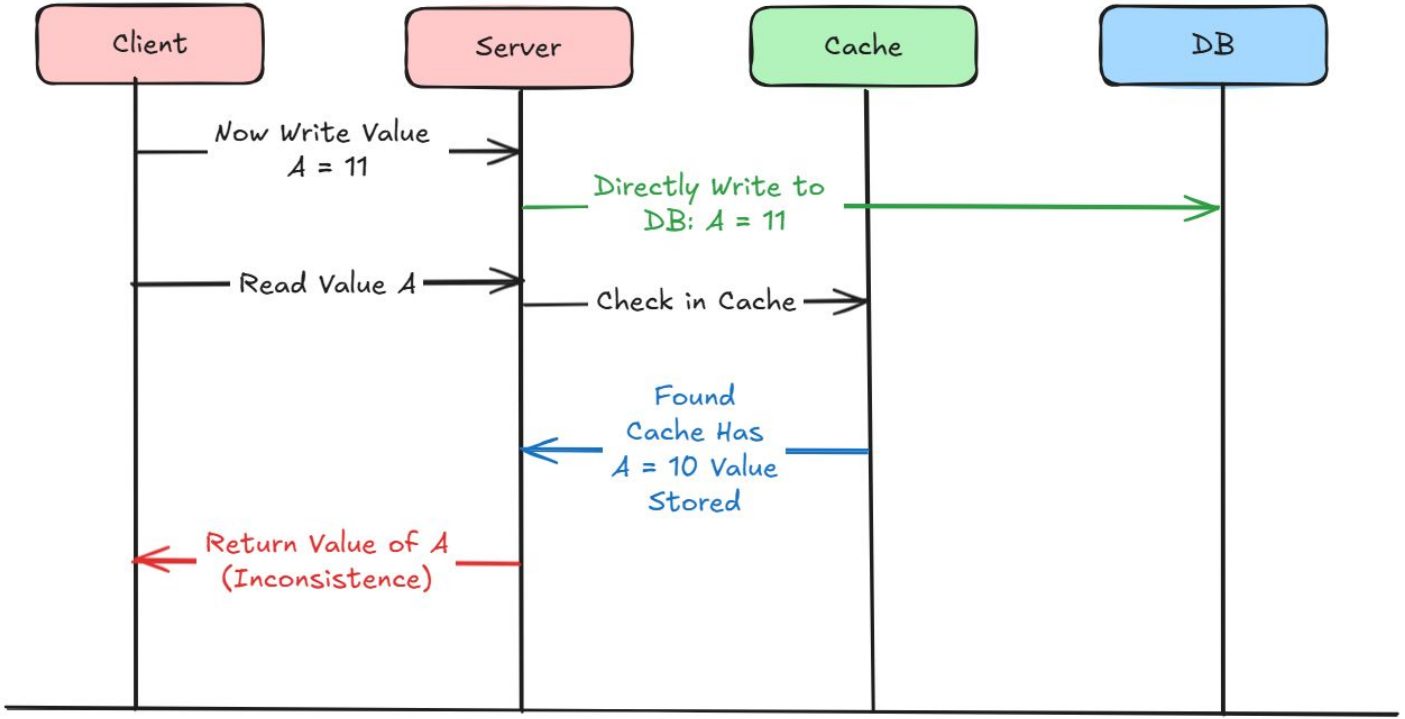
Write Only Method will always directly write to DB



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

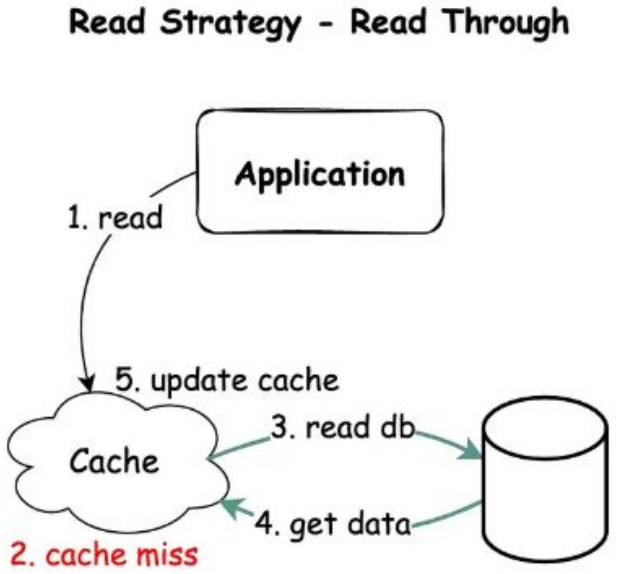


Assuming Caching didn't include any validation or expire

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

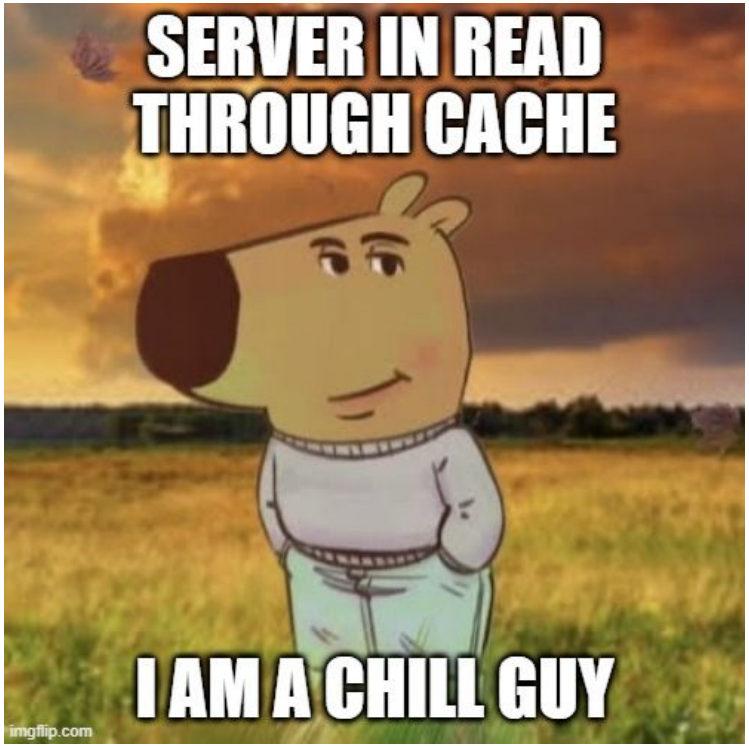
(Server has to do all of this task)

Cache Aside

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

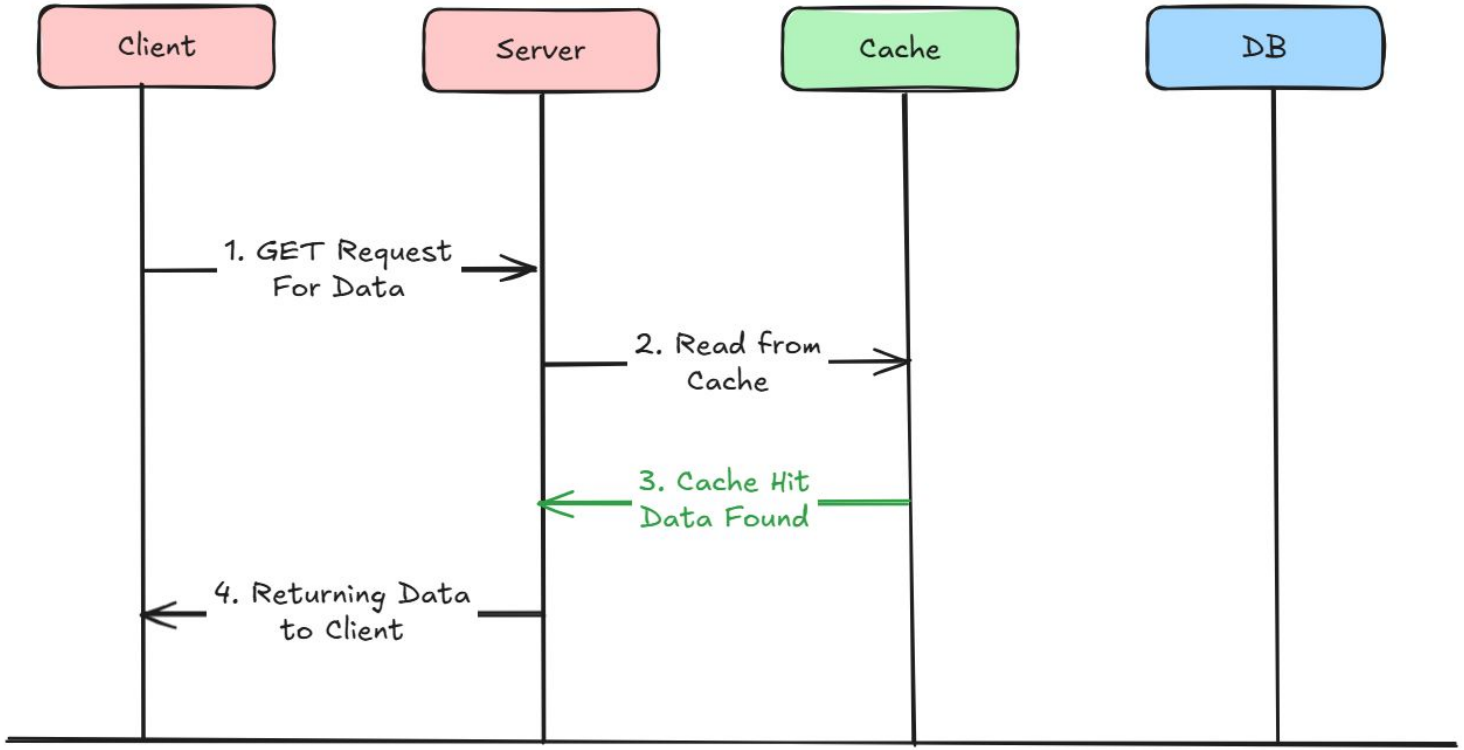
(Server has no tasks to do here)

Read Through

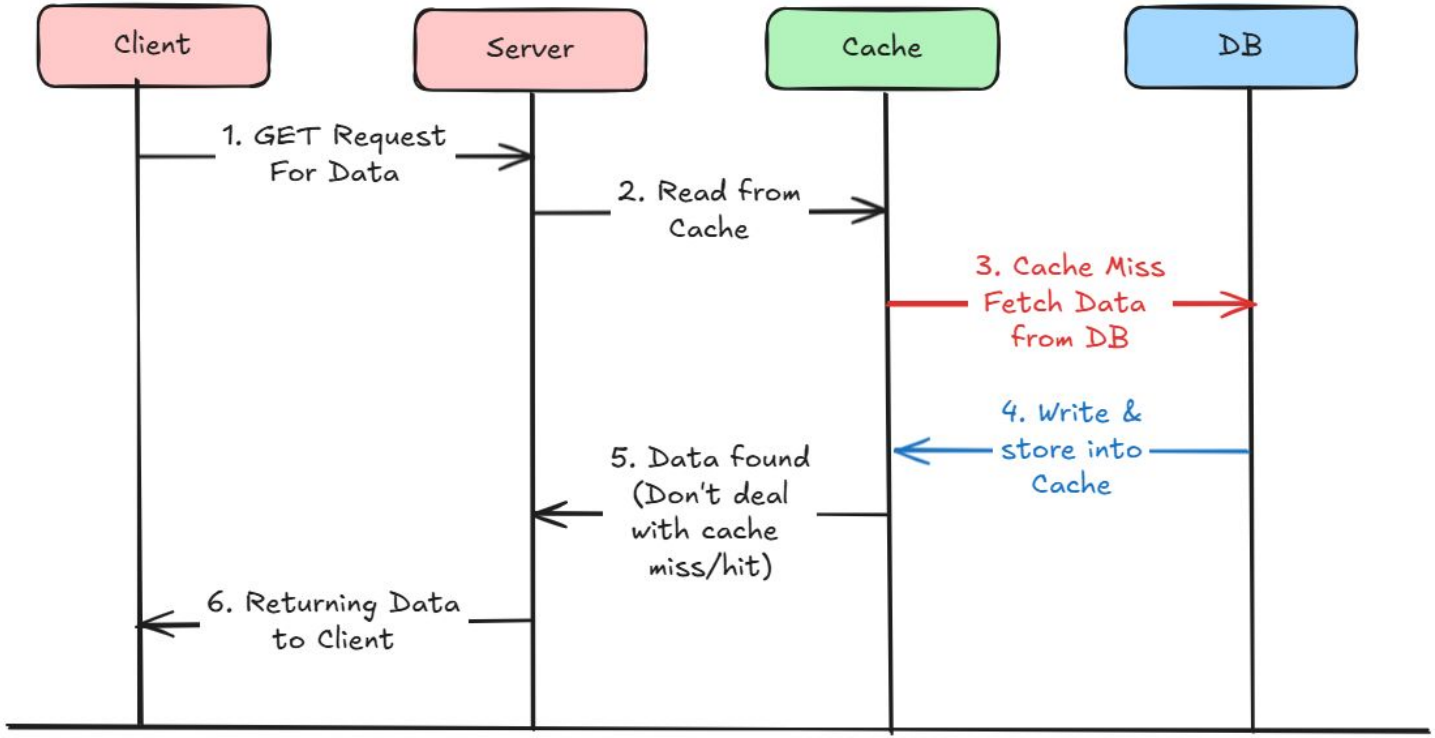

SERVER IN READ THROUGH CACHE
I AM A CHILL GUY
imgflip.com


CACHE MISS
SERVER IN CACHE ASIDE
imgflip.com

What happens in Cache Hit? (Same)



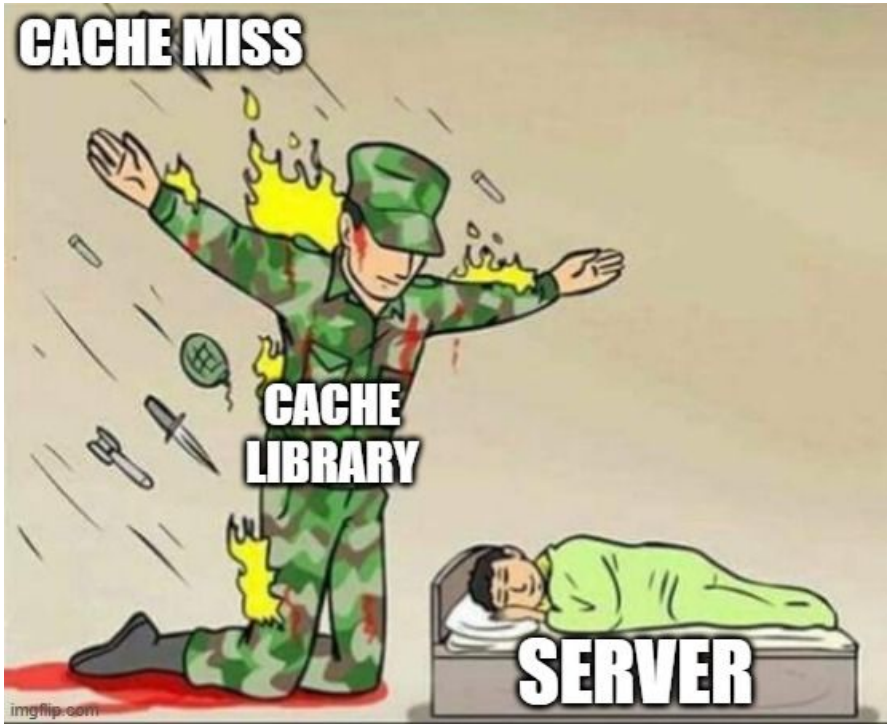
What happens in Cache Miss?



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

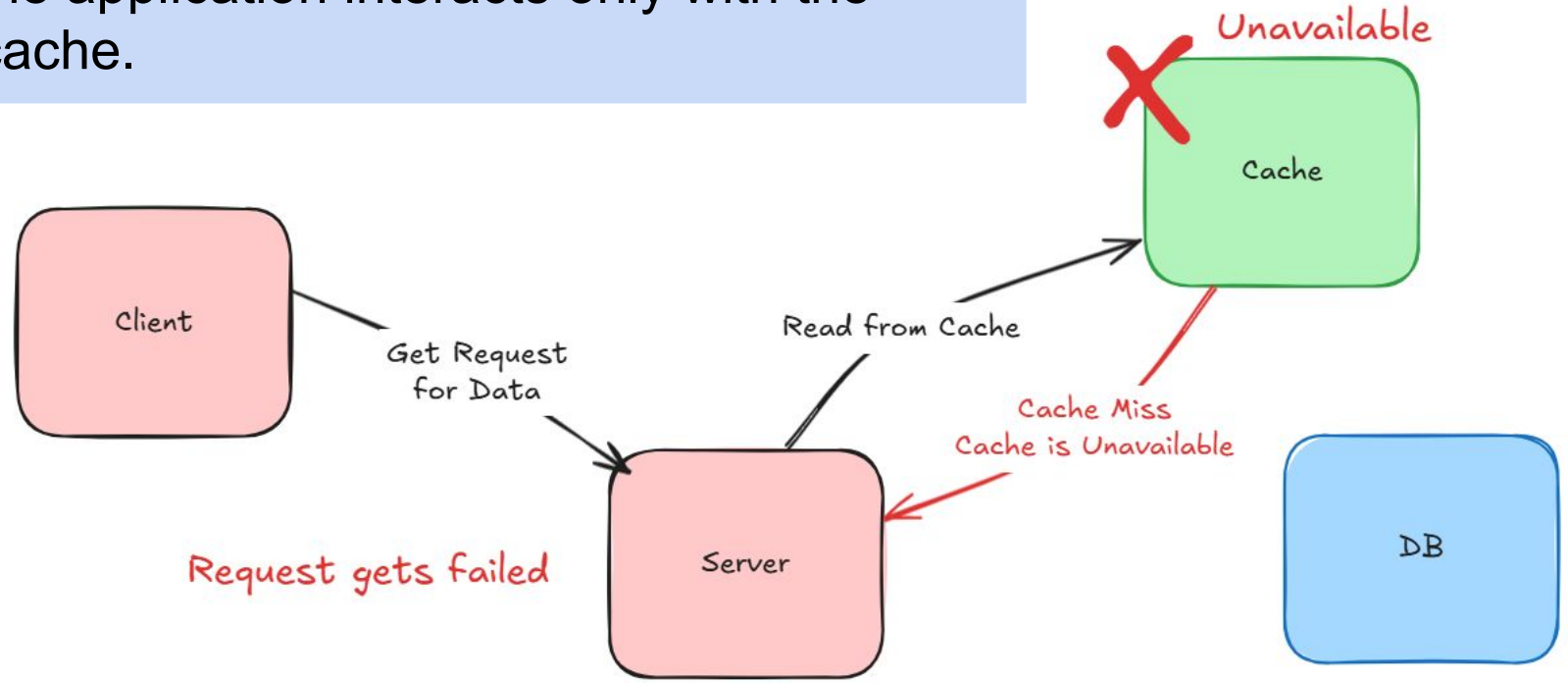
- Write Only Method will always directly write to DB
- For new data read, there will always be Cache - Miss first.
- If appropriate caching is not used during write operation, there is a chance of inconsistency between Cache and DB

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!"
    }
  ]
}
```

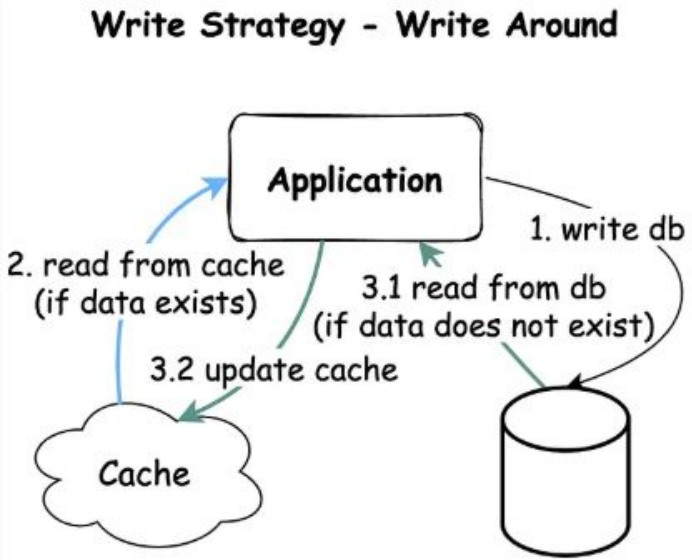
Database (Structured for Storage)

Cache (Optimized for Access)

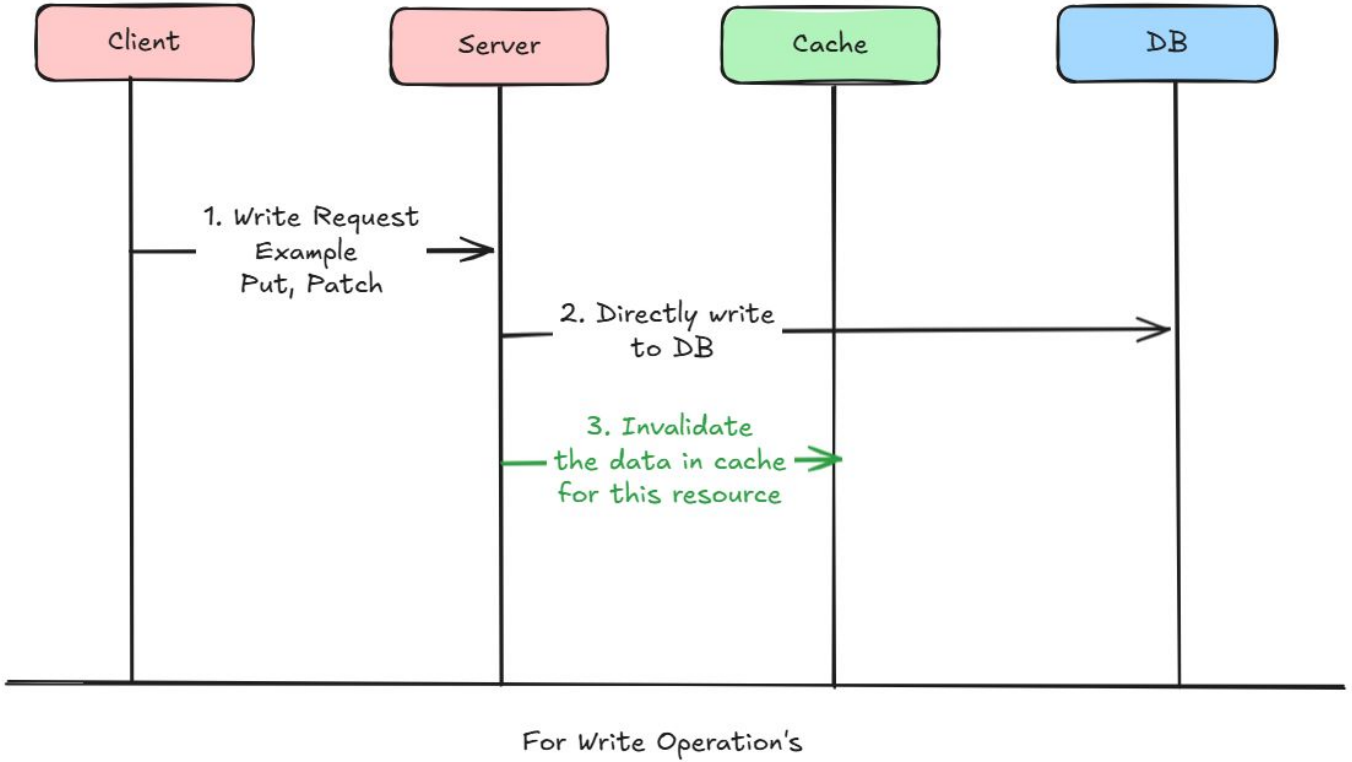
Write Around Caching Strategy

Write Around Strategy

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

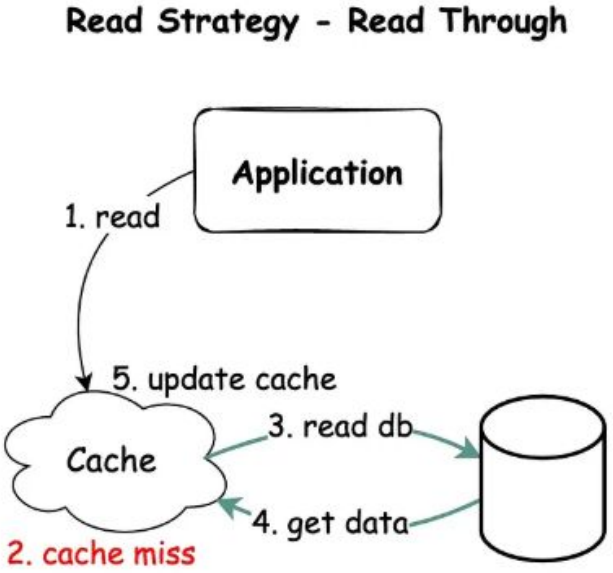


How write operation works?



For Write Operation's

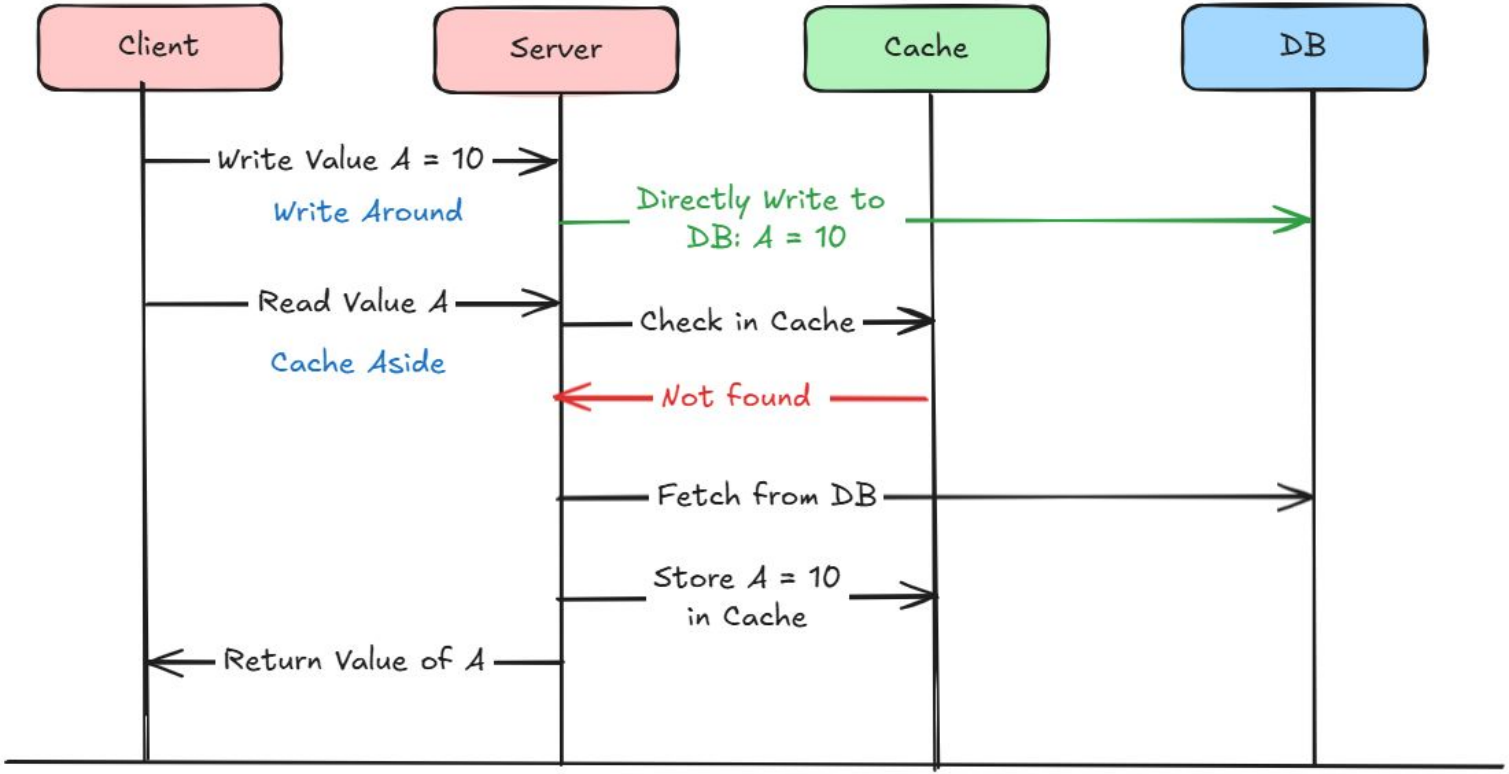
What to do when read operations?



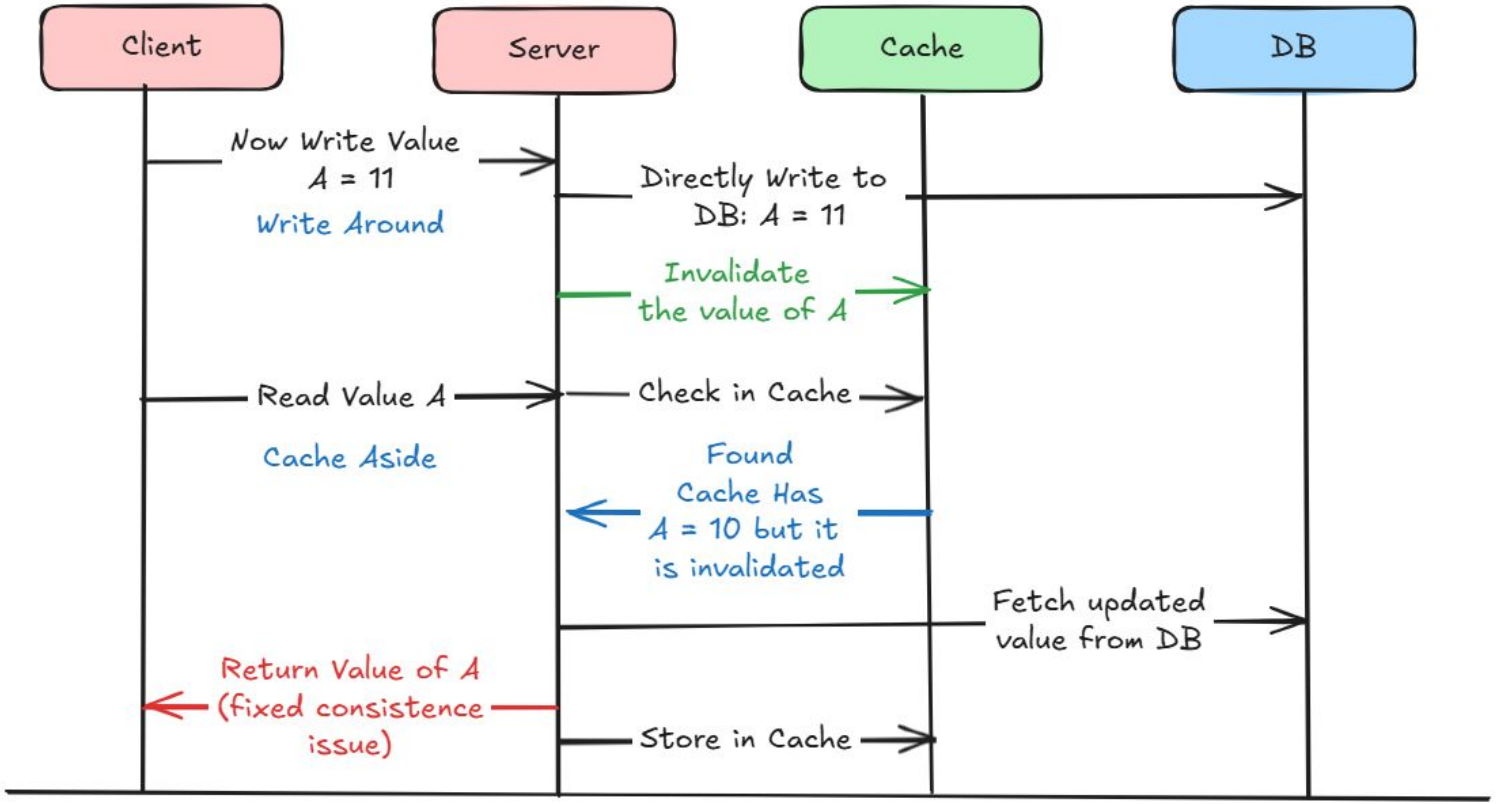
Have to choose a strategy between Cache Aside & Read Through

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

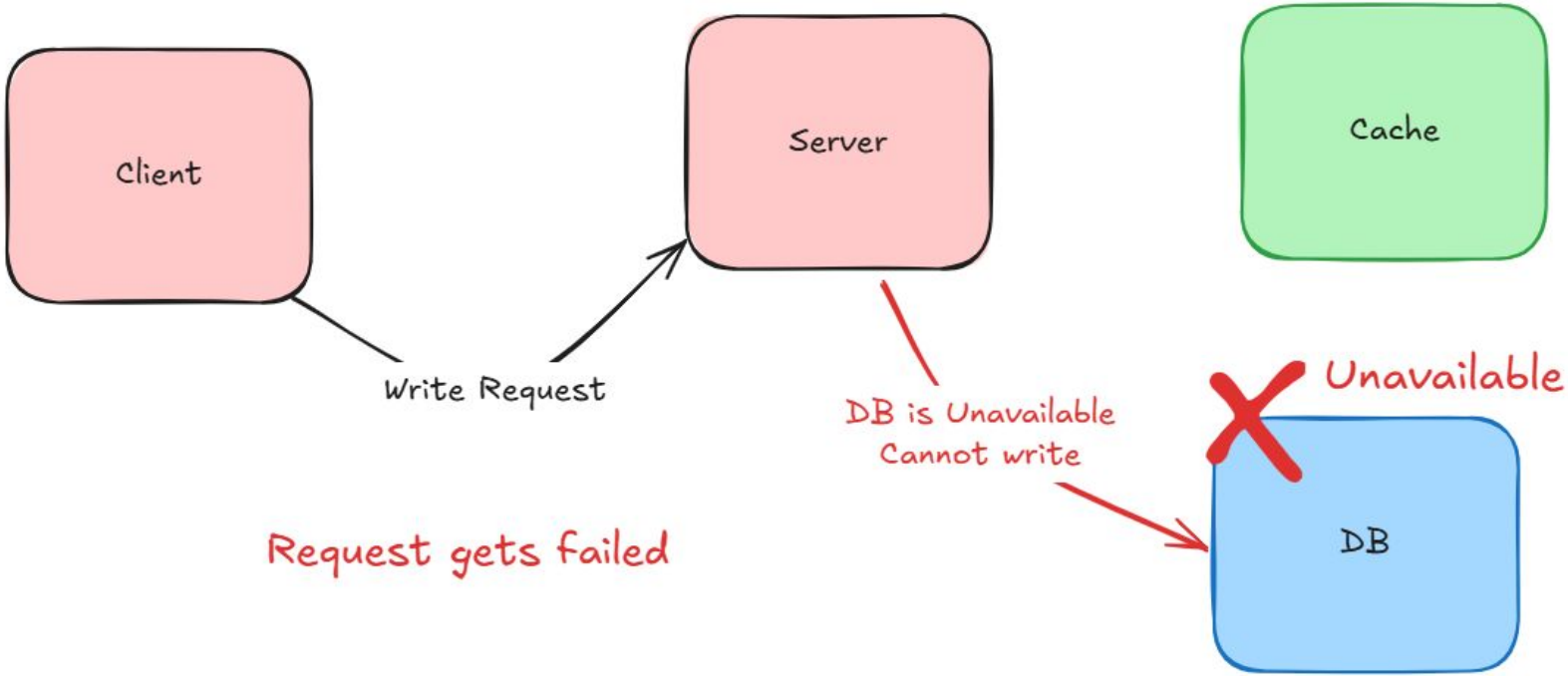
- Resolves the inconsistency problem between Cache and DB

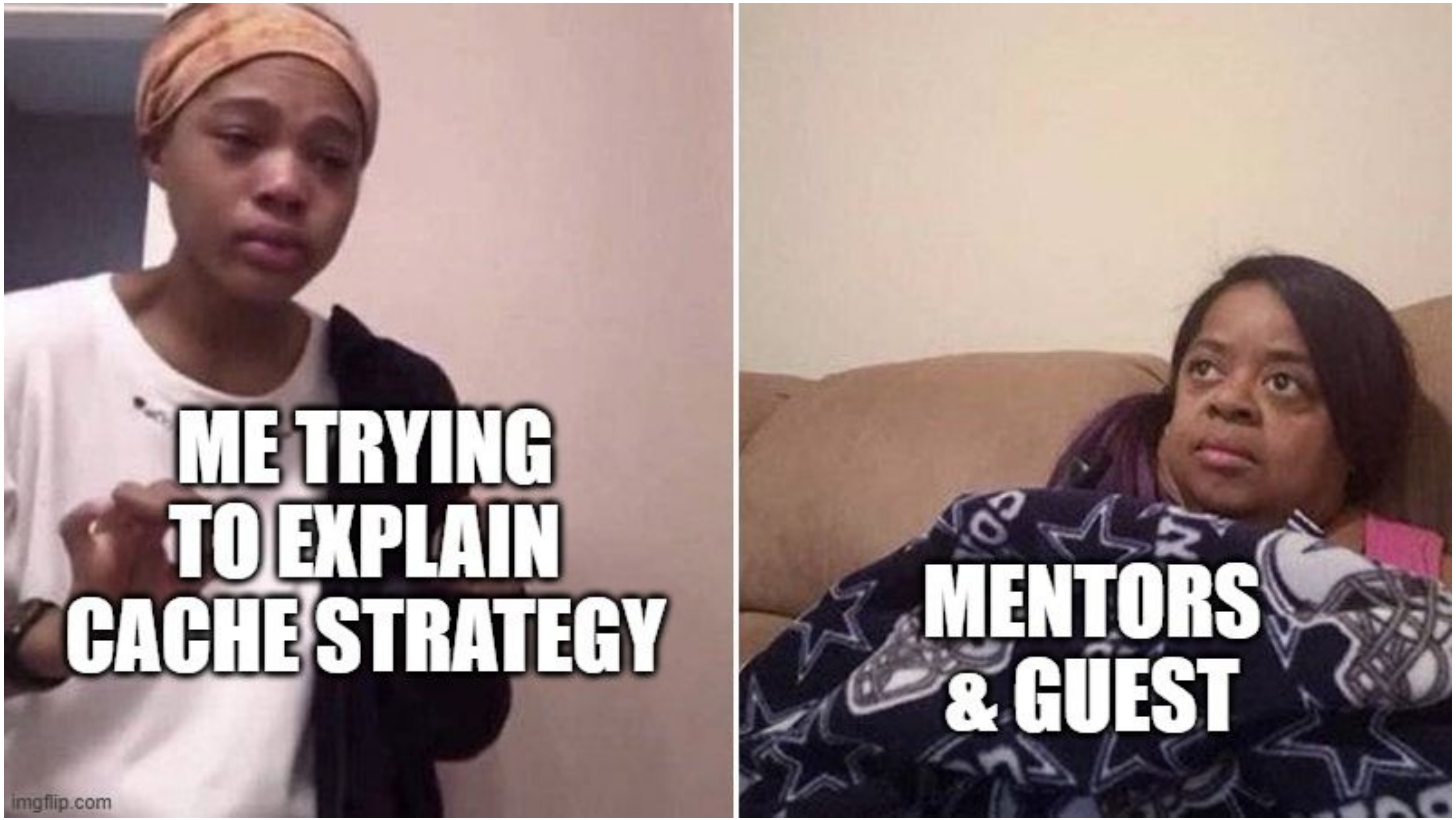
Write Around : Cons

- 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

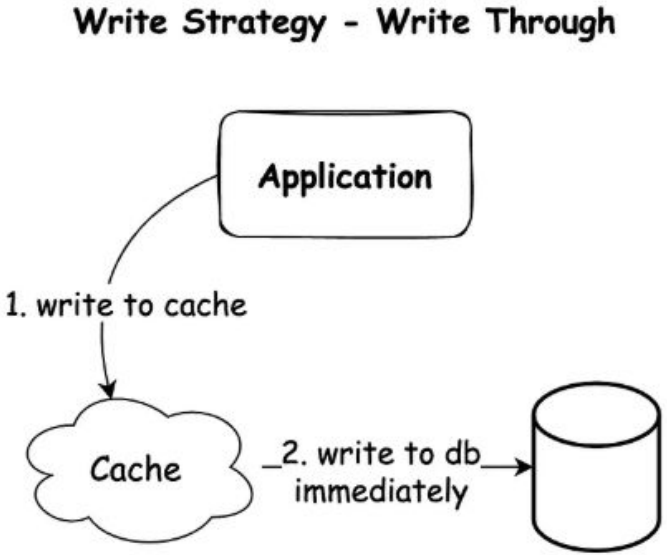



ME TRYING
TO EXPLAIN
CACHE STRATEGY
imgflip.com
MENTORS
& GUEST

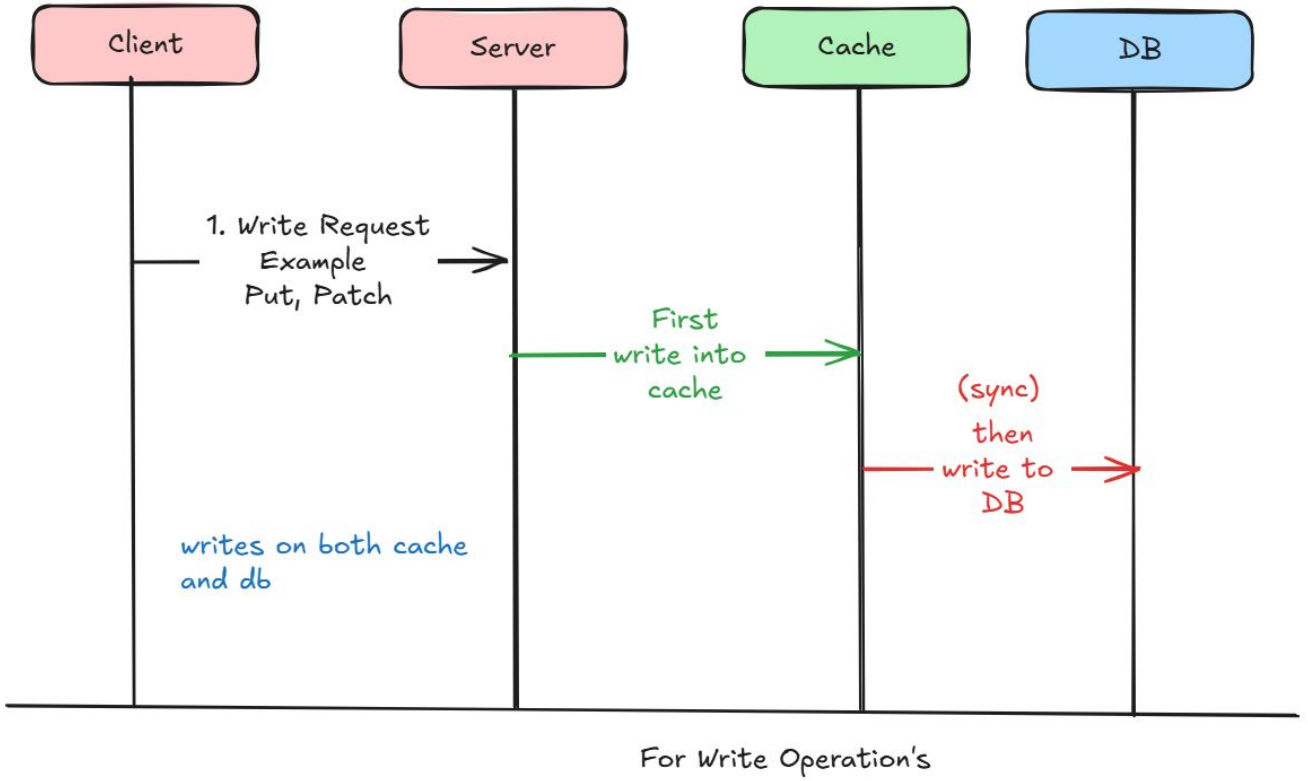
Write Through Caching Strategy

Write Through Strategy

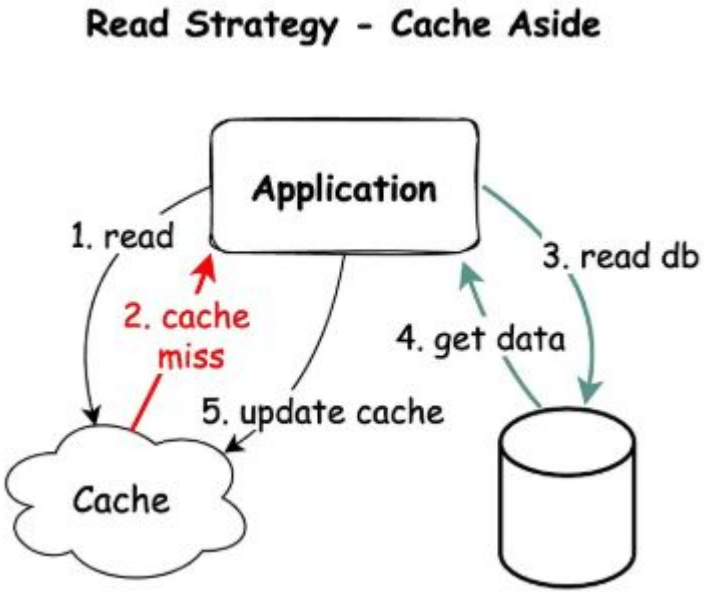
- First writes data into the cache
- Then in **synchronous** write data into the DB
- For Read : Choose between
  - Cache Aside
  - Read Through

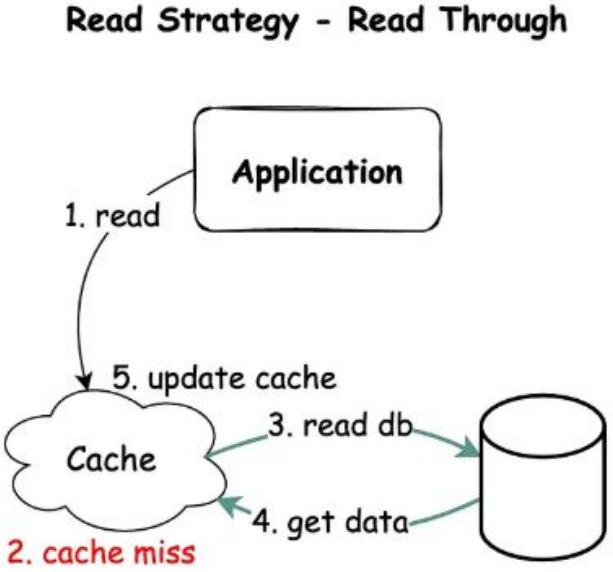


How write operation works?



What to do when read operations?





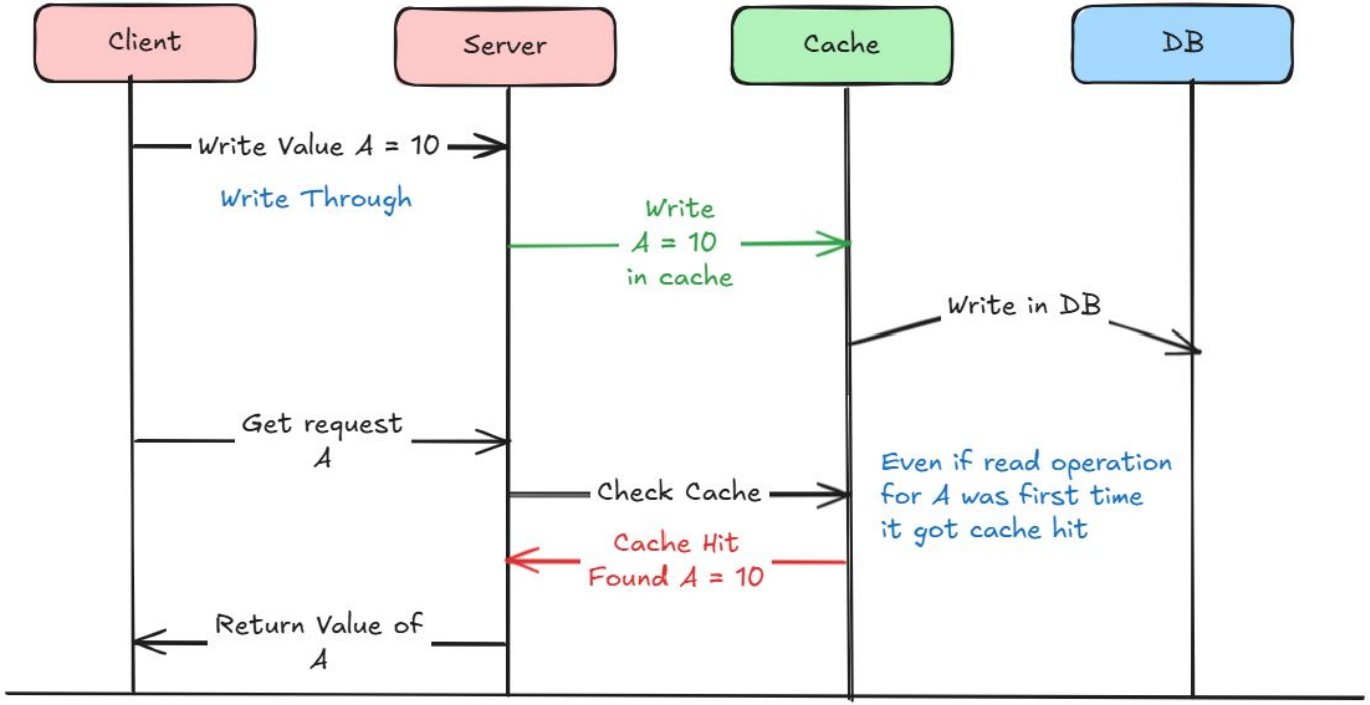
Have to choose a strategy between Cache Aside & Read Through

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

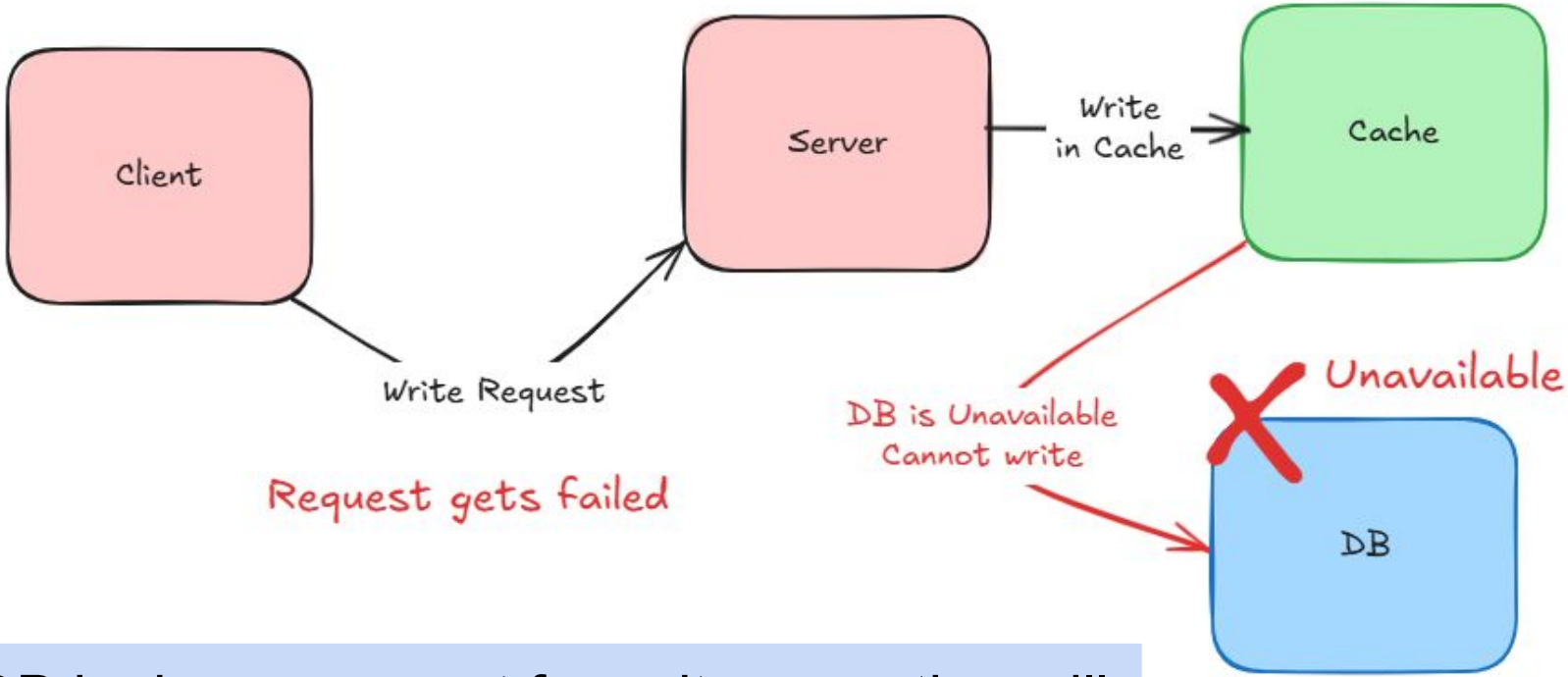
Increase Cache Hit chances : Even if for new data read there is a chance of cache hit



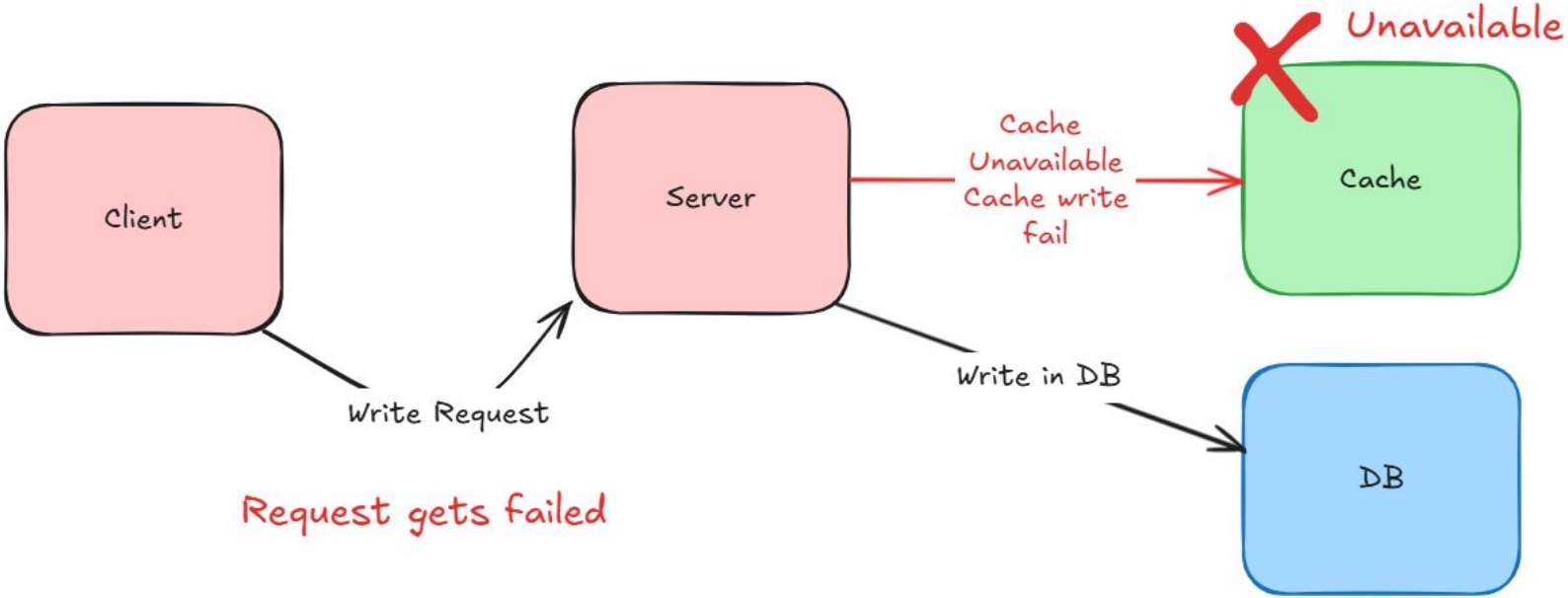
Write Through : Cons

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

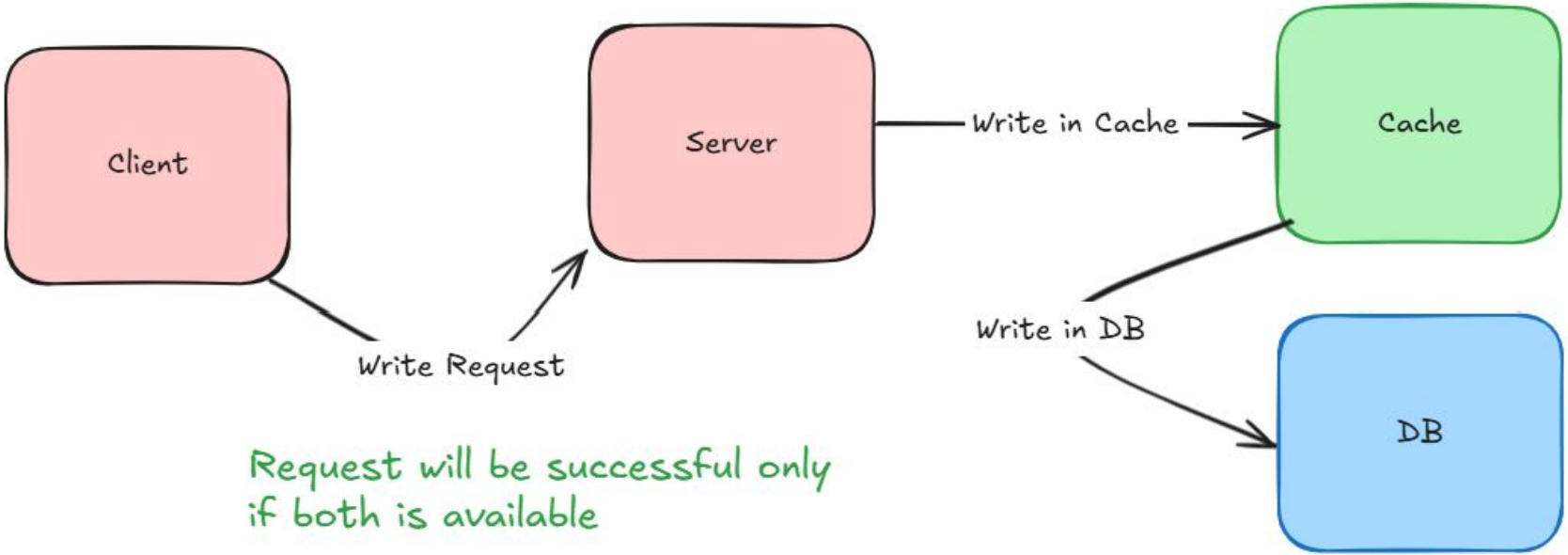
- 2 phase commit, need to be applied to maintain transaction property



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



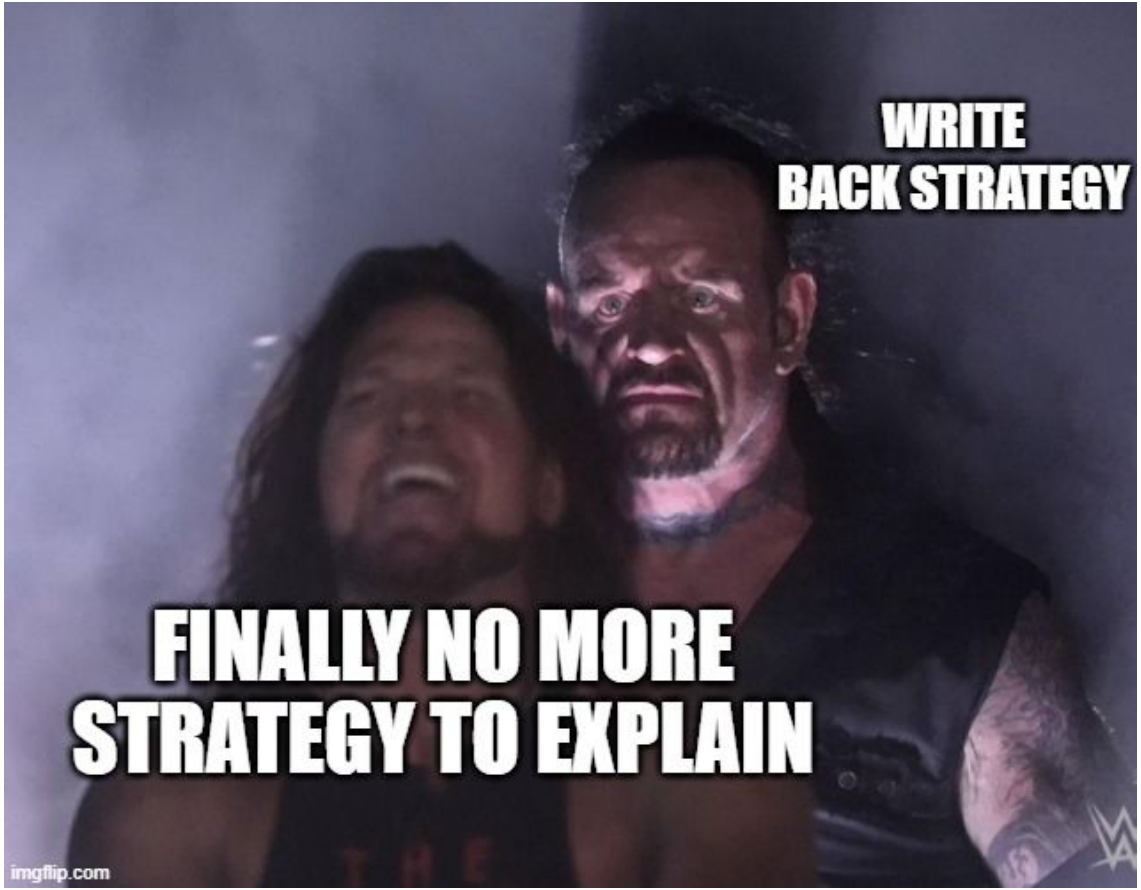
If Cache is down, request for write operation will fail Even if DB is available



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 !

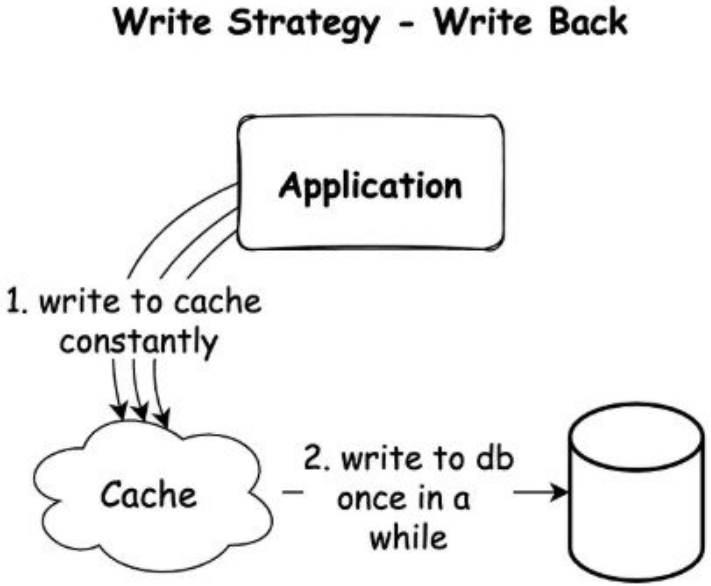
2 phase commit, need to be applied to maintain transaction property


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

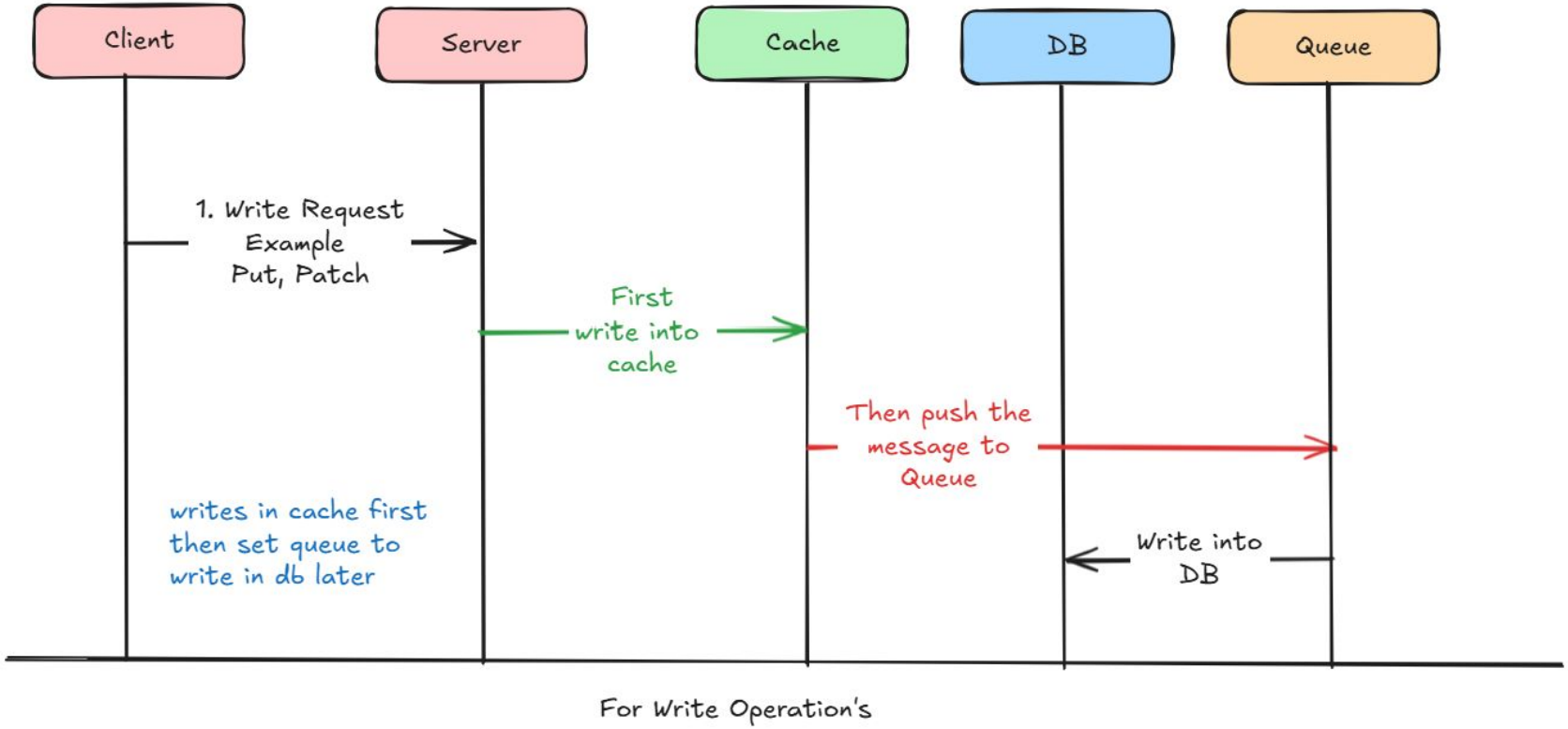
Write Back Caching Strategy

Write Back Strategy

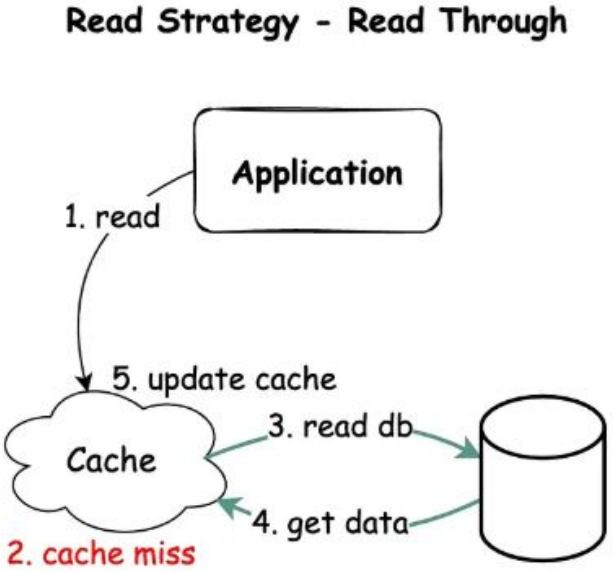
- First writes data into the cache
- Then in **asynchronous** write data into the DB
- For Read : Choose between
  - Cache Aside
  - Read Through



How write operation works?



What to do when read operations?



Have to choose a strategy between Cache Aside & Read Through

Write Back : Advantages

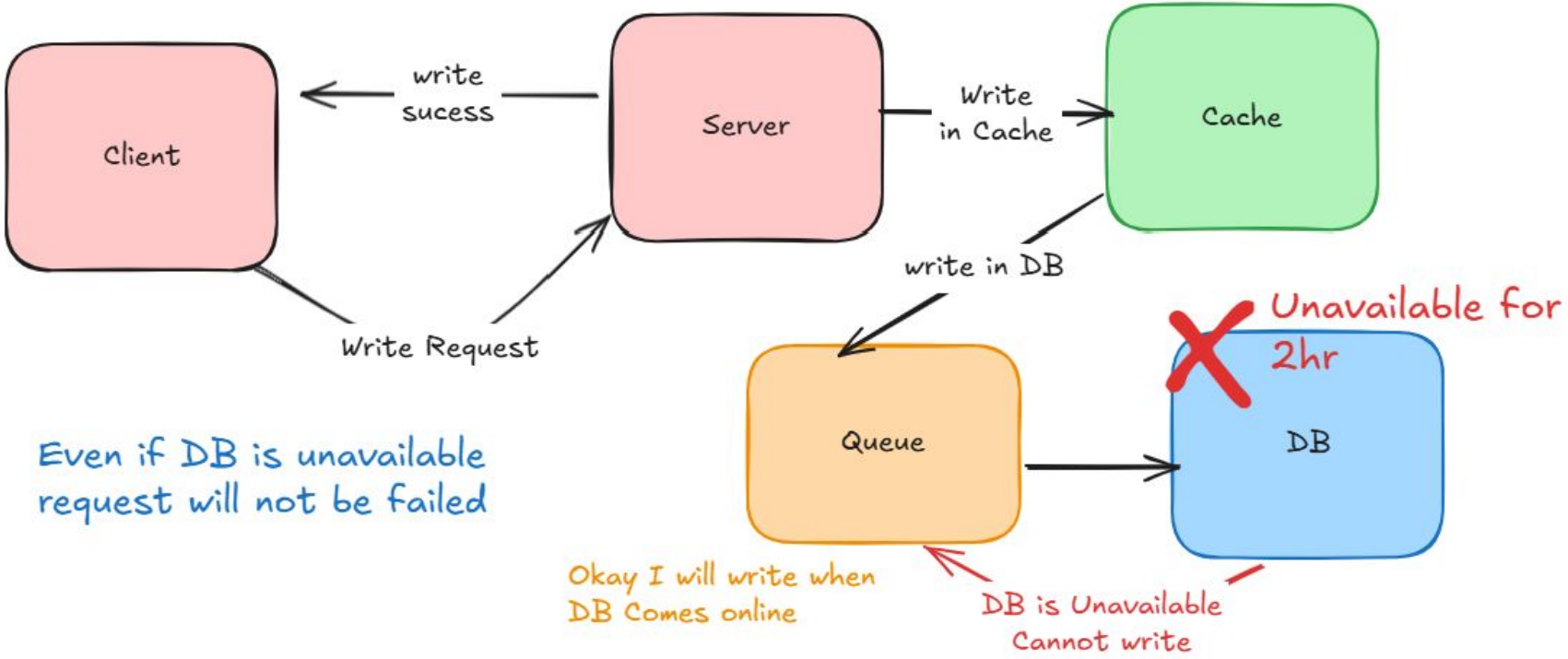
- Resolves the inconsistency problem between Cache and DB

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

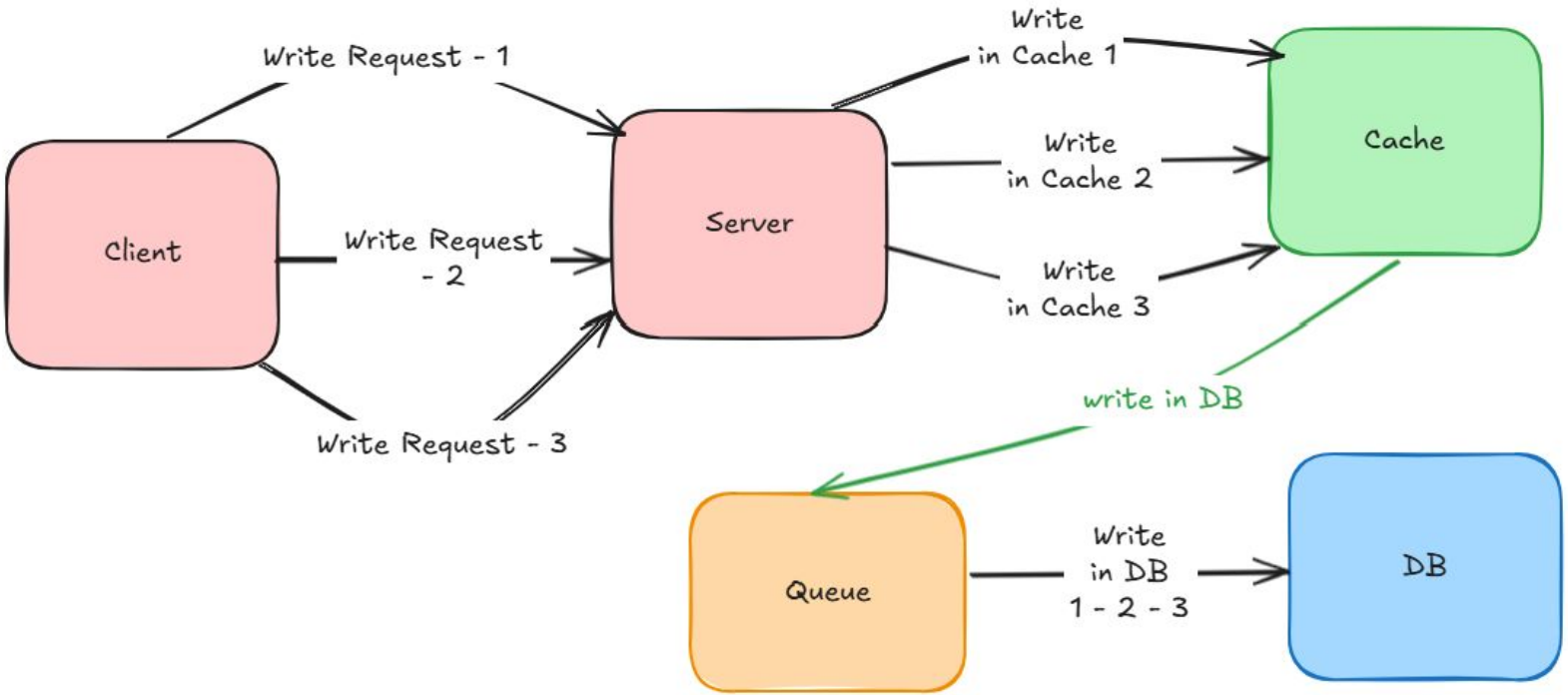
Write Back : Advantages

- 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



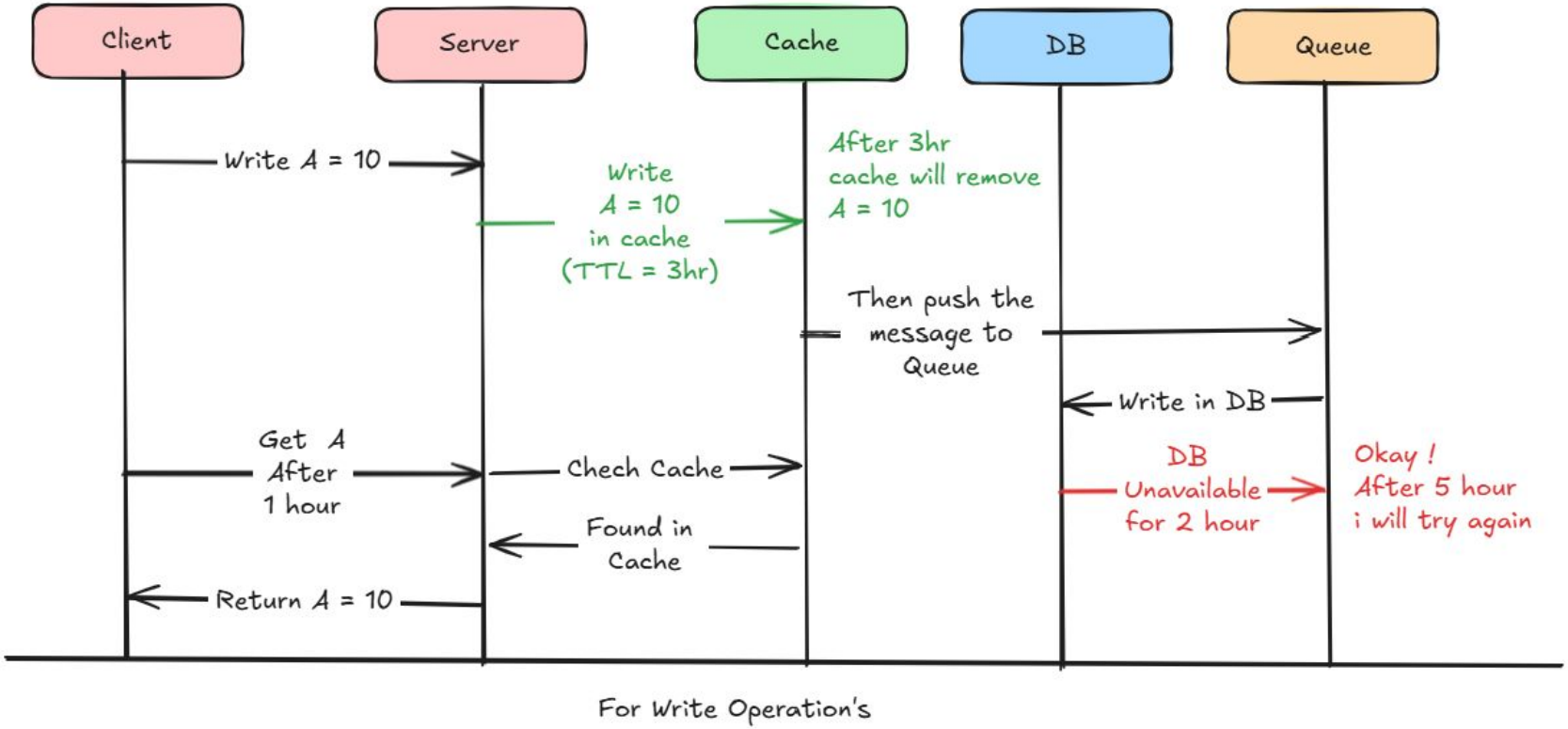
Improves the write operation latency as writing into DB happens asynchronously



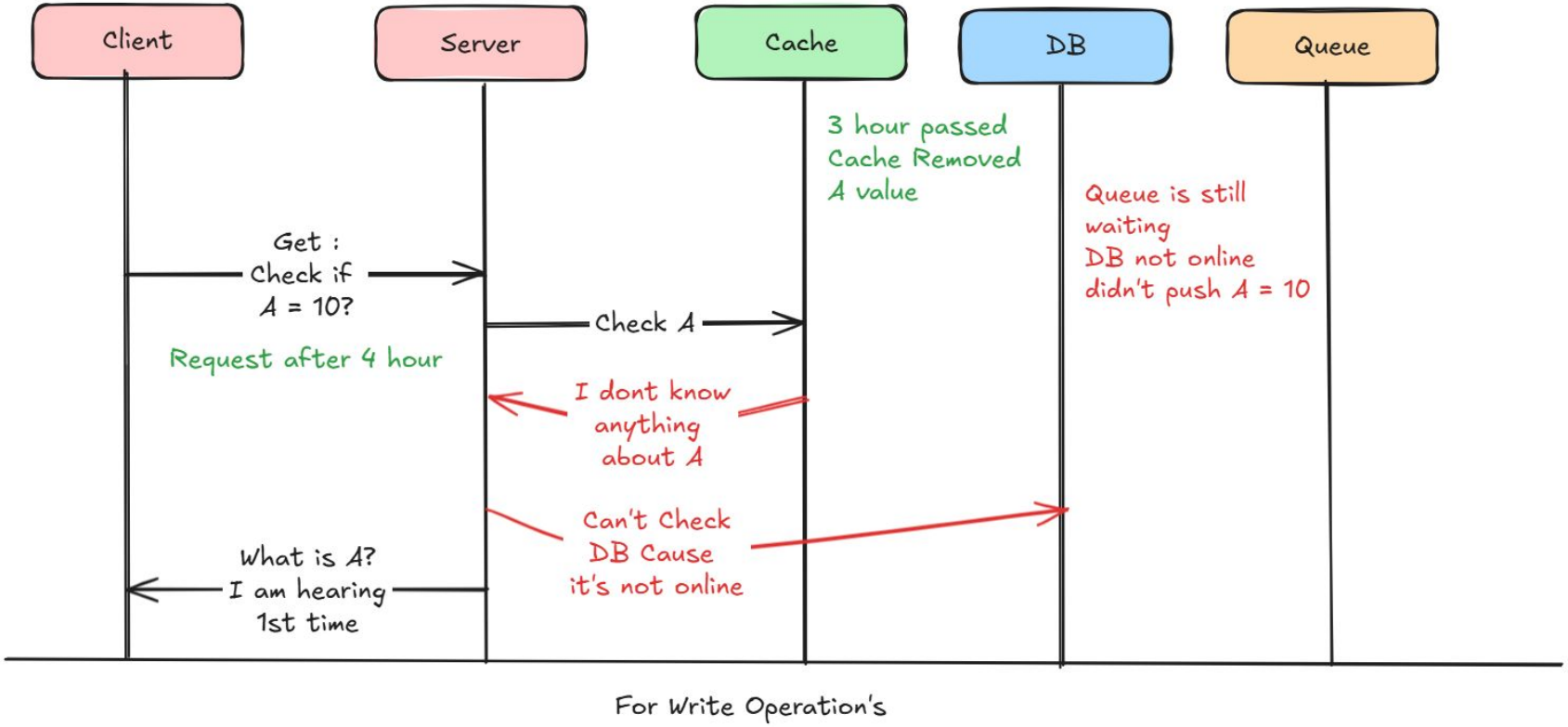
Write Back : Cons

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

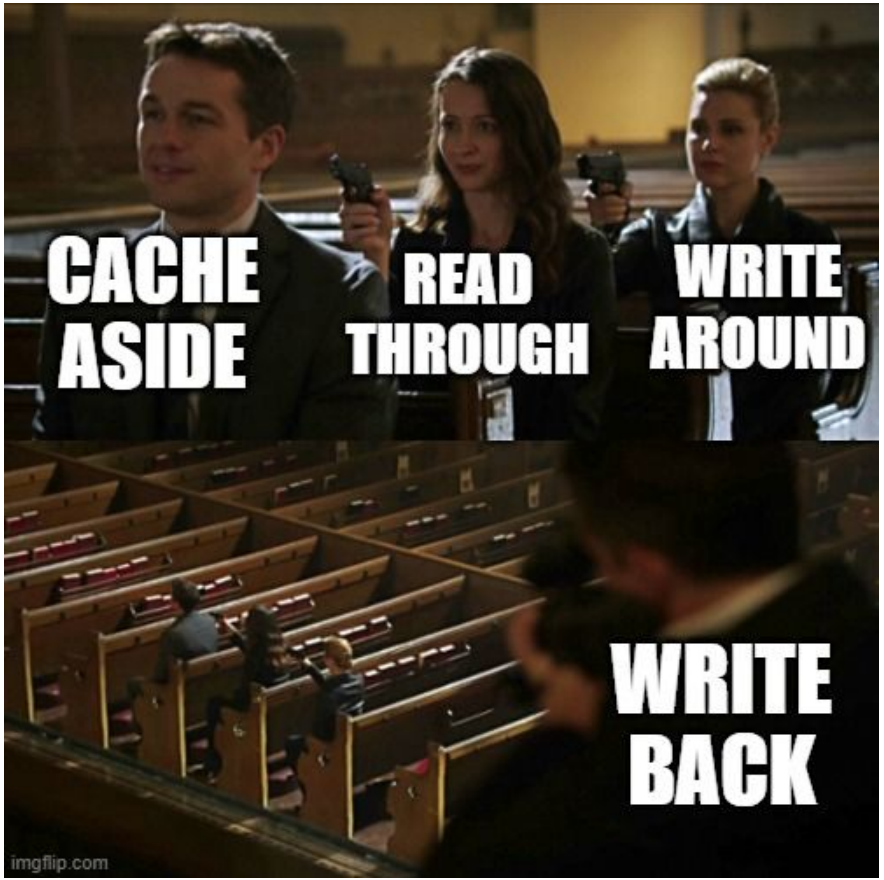
- Write-back caching improves performance but risks data loss, complexity, and consistency issues



For Write Operation's



For Write Operation's


CACHE ASIDE
READ THROUGH
WRITE AROUND
WRITE BACK
imgflip.com


GROUP 1
IS BEST
GROUP 2
IS BEST
SHOAIB BHAI
imgflip.com

Caching in Modern Web

Accelerating Modern Web Technologies

K M Fatin Ishraq

Associate Software Engineer

Cache, We meet again!

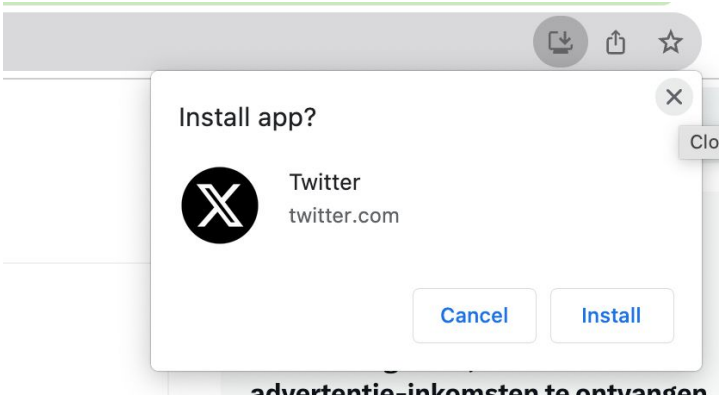
What is PWA?

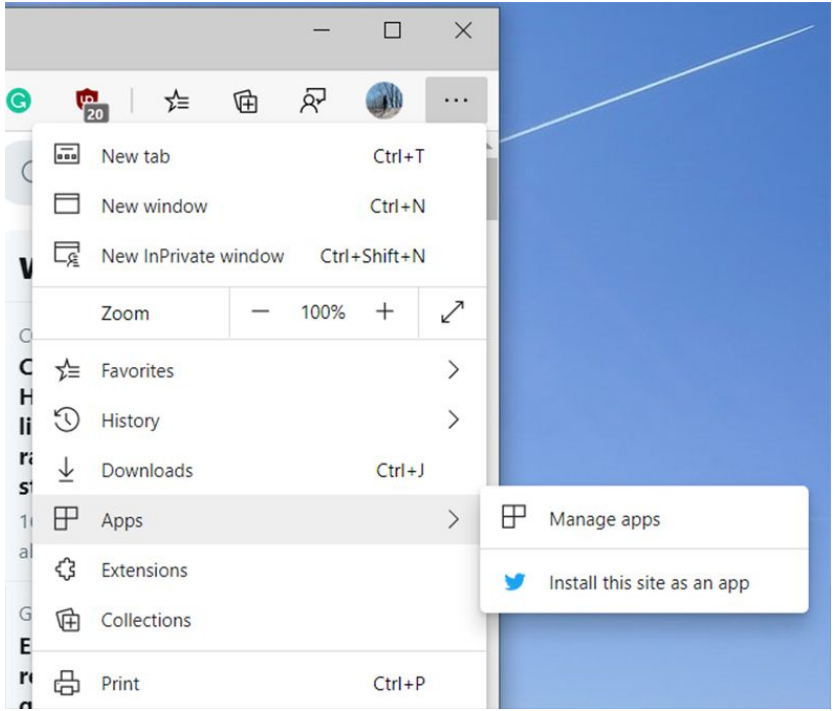
Progressive Web App.

Progressive? Fancy word!

But Why?

Does this looks familiar?

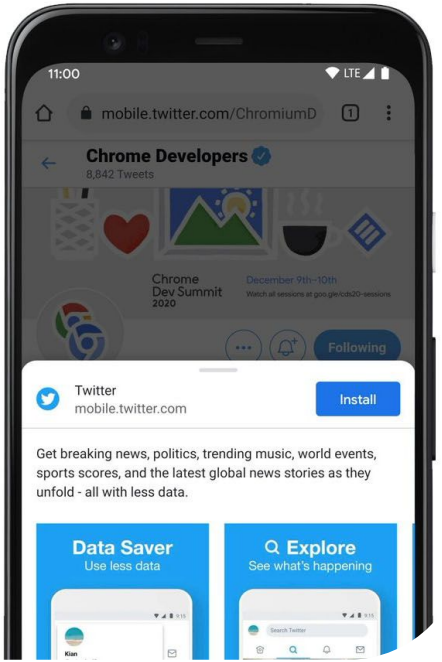




This One?

Can be installed simply by ***Add to home screen*** feature.

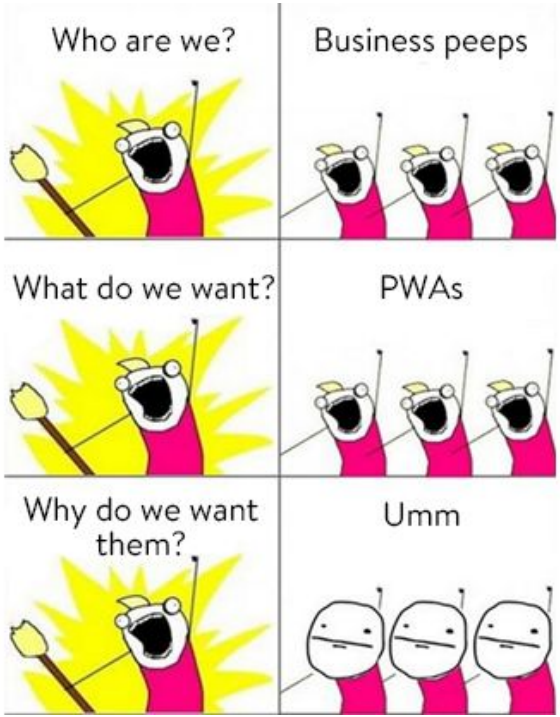
Rich Install Dialogue for Progressive Web Apps



So Why do we need PWAs?

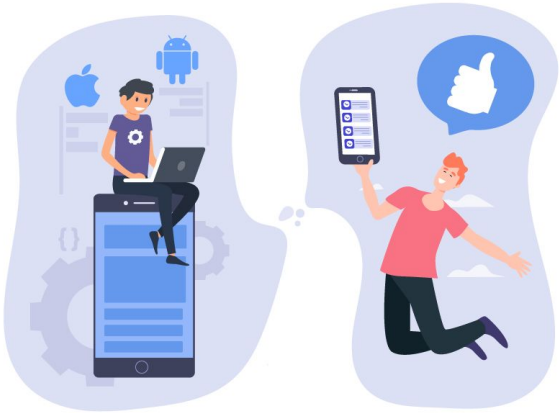
PWAs' actually good for businesses.

But why?

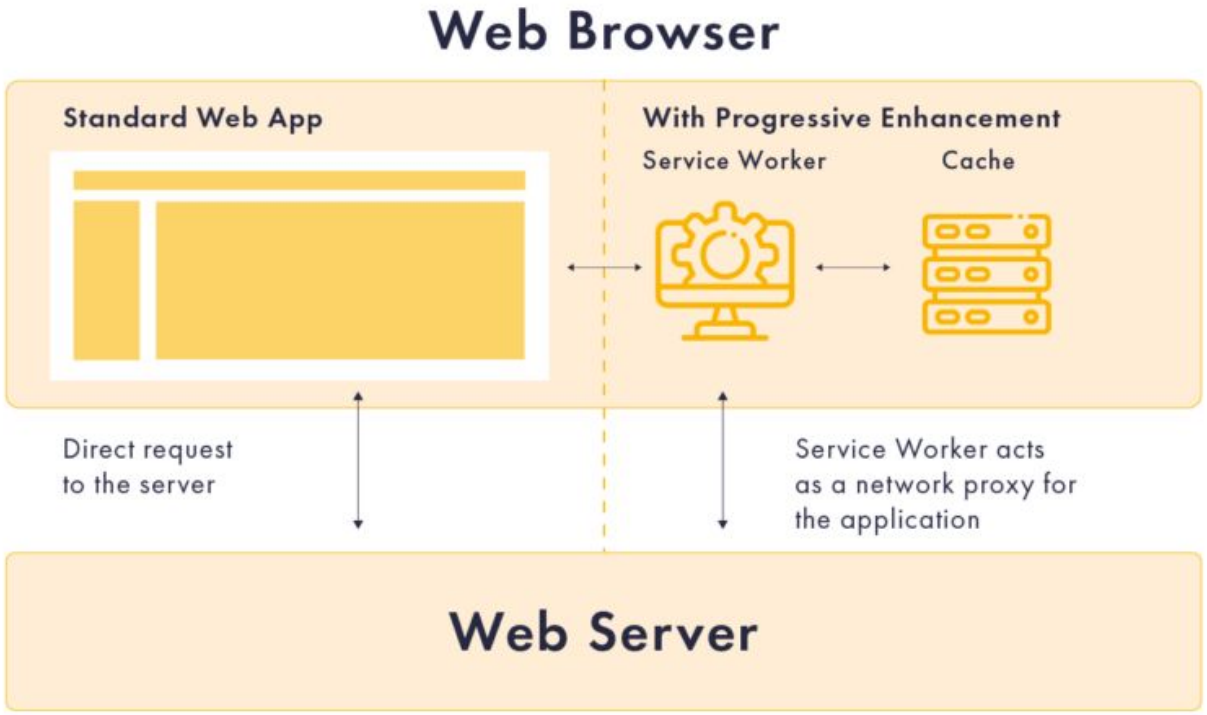


PWA combines the best features of web and mobile apps.

FAST	Reliable	Engaging
Loads super fast! Thanks to smart caching.	Works offline or even poor network connections.	Offer features like push notifications

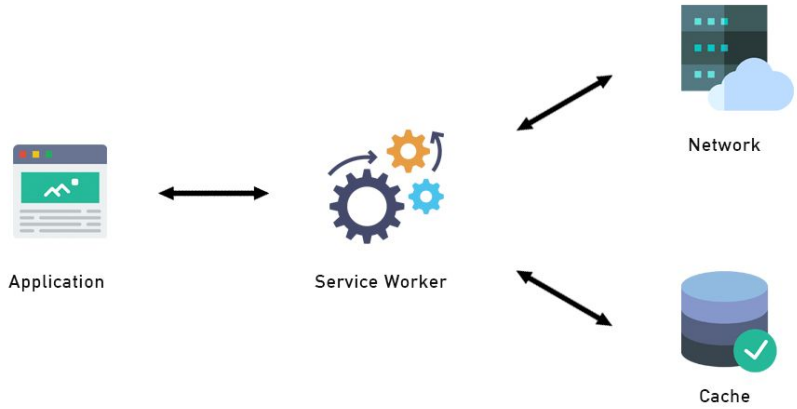


How is it different from traditional web app?



Let’s put it simply!

- The browser requests resources from the server.
- Service Worker Intercepts!
- First Load -> Resources are fetched from the server.
- Subsequent loads -> Use Cache!
- When offline -> Use content from cache



Key characteristics of PWAs

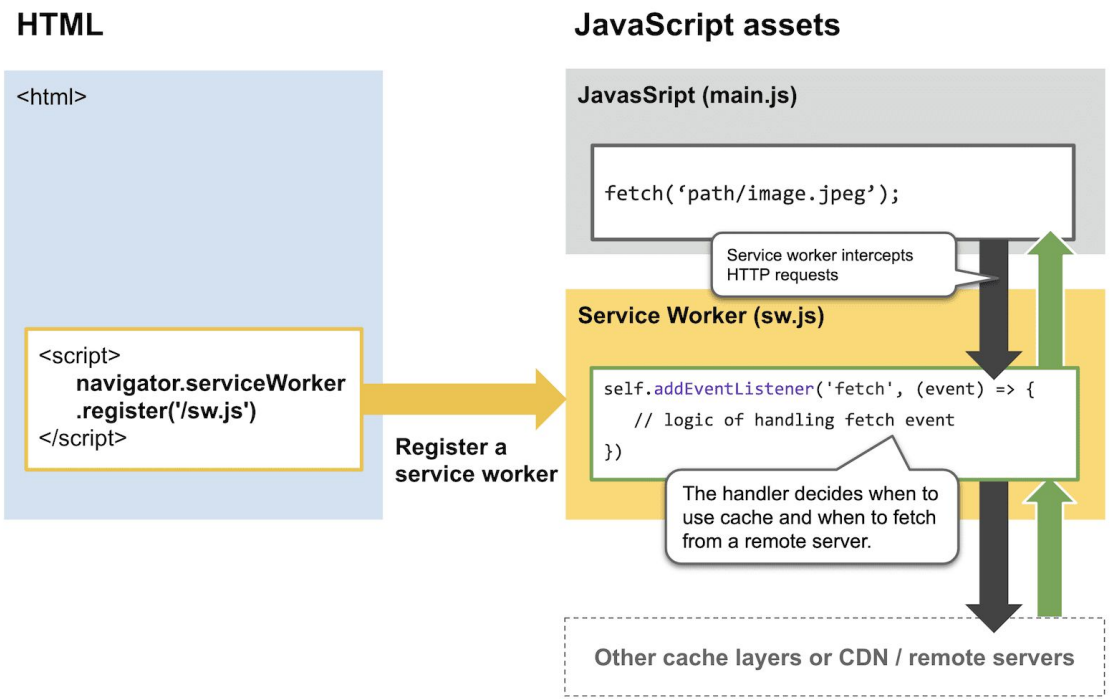
Service Workers: The Mastermind!

- Background scripts that enable caching, offline functionality and push notifications.
- A JSON file that allows users to install PWAs on their devices

Let’s get technical!

Service Worker Caching

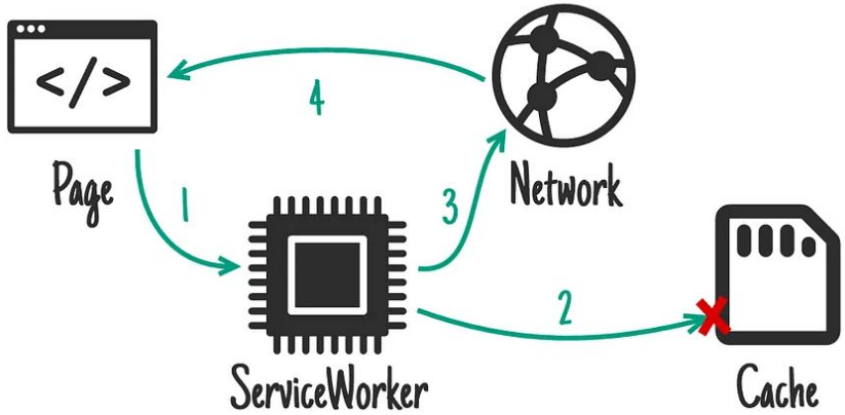
The service worker cache provides more fine-grained control over what is cached and how, compared to the HTTP cache.



Caching strategies of PWA

Cache First, Network Fallback

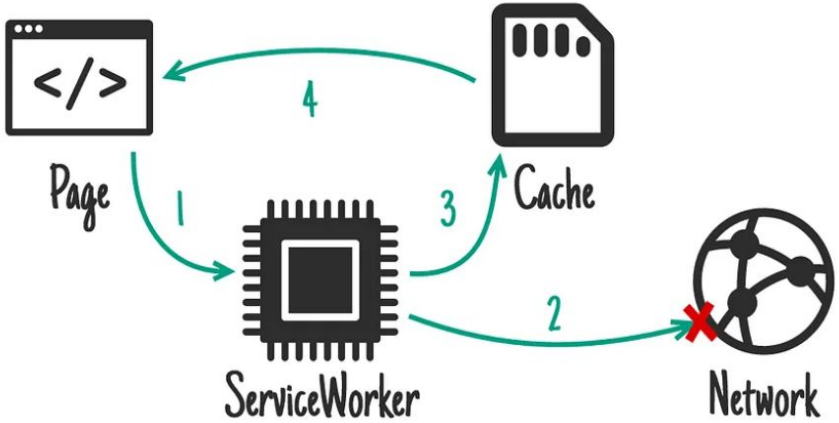
- Serves cached content first, falls back to network if unavailable.
- Best for static resources like images, CSS, or fonts.
- Offers fast load times but risks serving outdated data.



Caching strategies of PWA

Network First, Cache Fallback

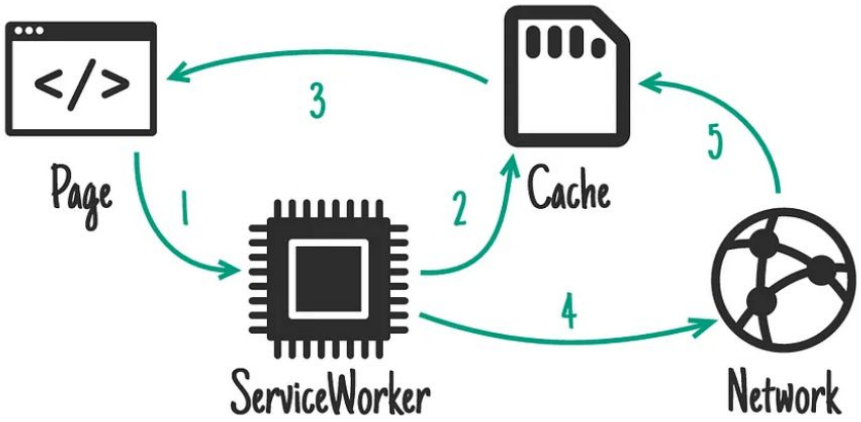
- Fetches fresh content from the network uses cache only if the network fails.
- Ideal for real-time APIs or frequently updated data.
- Ensures freshness but can be slower and dependent on network availability.



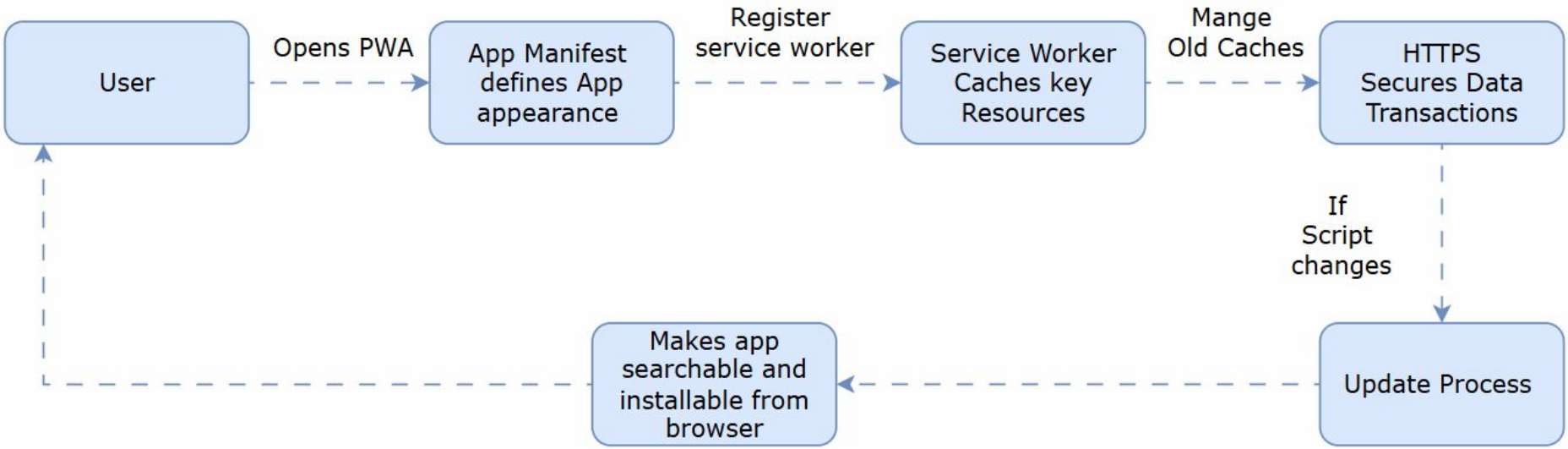
Caching strategies of PWA

Stale While Revalidate

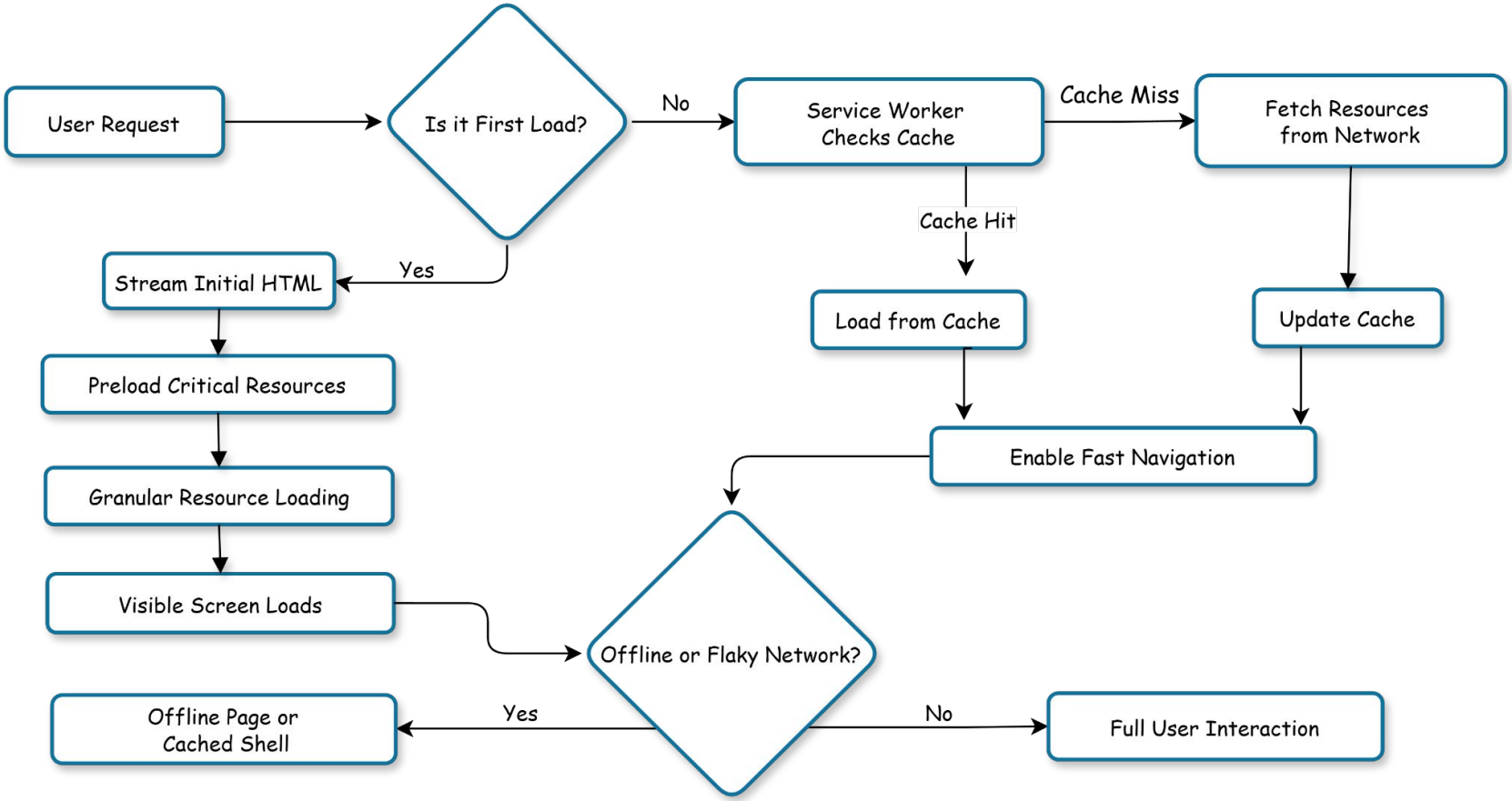
- Serves cached content immediately and updates cache in the background.
- Suitable for non-critical data
- Balances speed with relatively fresh data but may show outdated content briefly.



Overview of the process



Twitter PWA


User Request
Is it First Load?
No
Service Worker Checks Cache
Cache Miss
Fetch Resources from Network
Cache Hit
Yes
Stream Initial HTML
Load from Cache
Update Cache
Preload Critical Resources
Enable Fast Navigation
Granular Resource Loading
Visible Screen Loads
Offline or Flaky Network?
Offline Page or Cached Shell
Yes
No
Full User Interaction

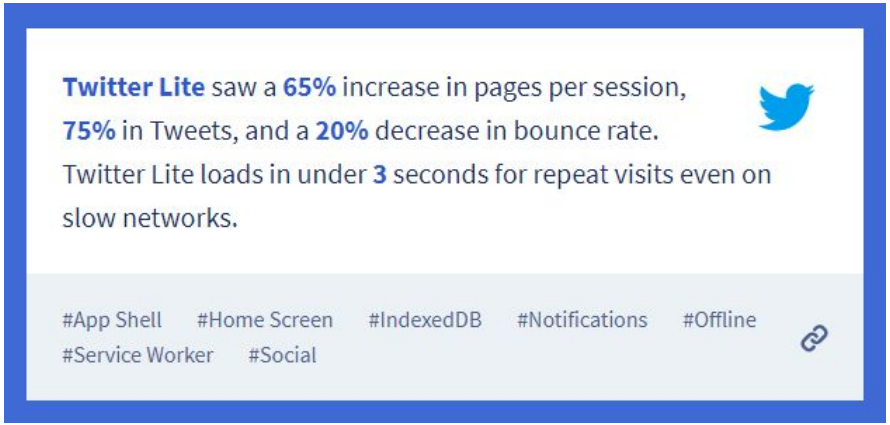
So what happened after this?

Problems

- Slow on mobile networks
- Used storage more
- Low user engagement on mobile devices

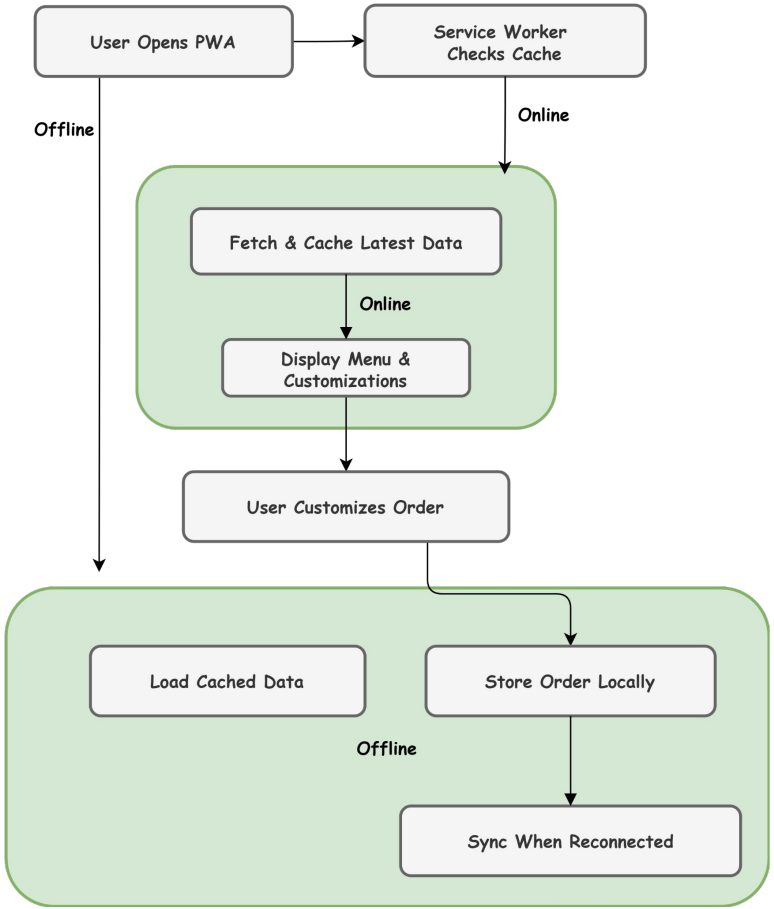
After PWA

- Instant Loading
- Lower data consumption
- Only 600 KB in app size



Starbucks PWA

- Storing Menu Items and Customizations
- Service Workers Updating in the background!



So that’s why they want it!

Let’s differentiate

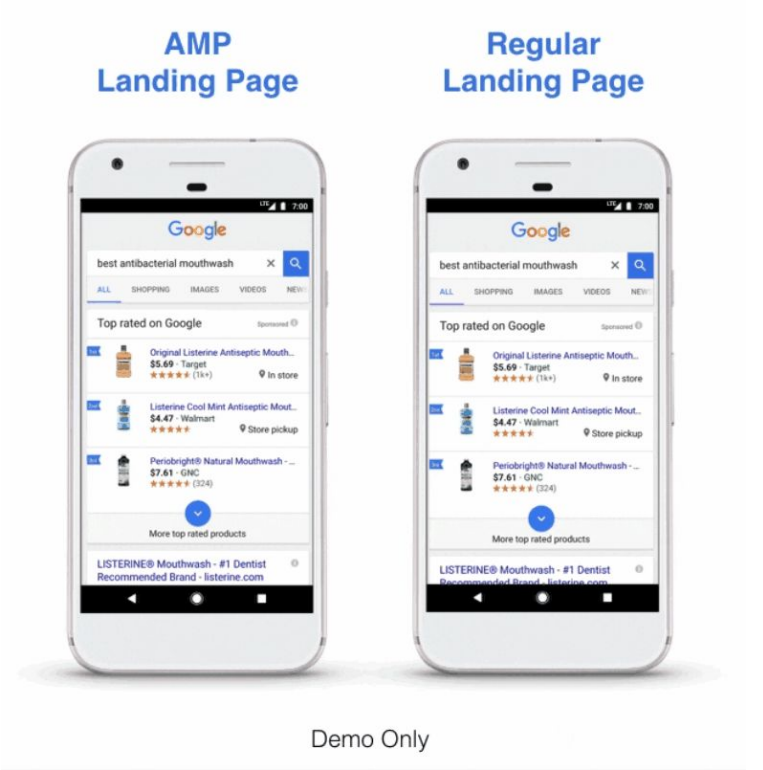
Checks All the boxes!

(Cause I have added only the checked boxes)

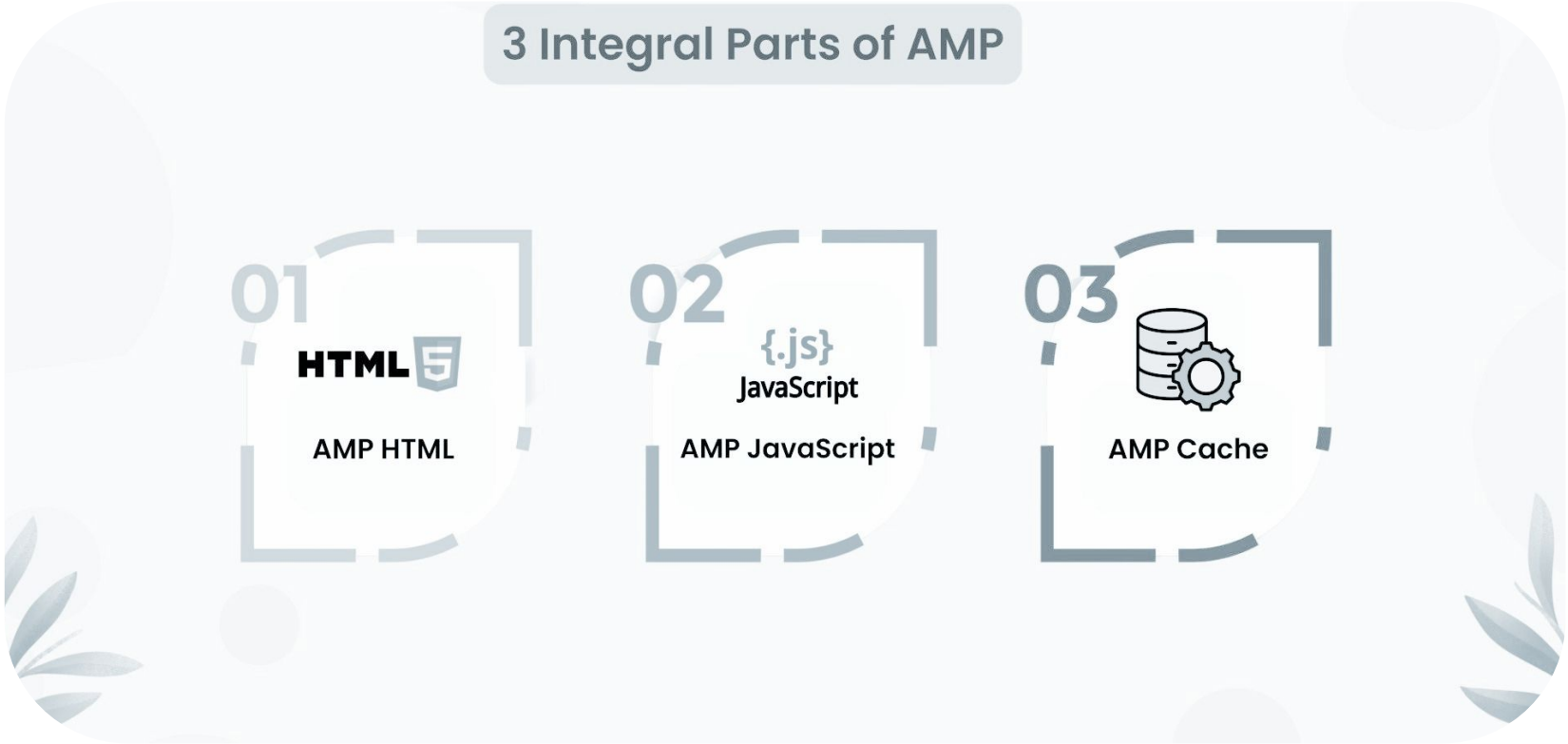

WEBSITE WHEN THEY
USE PWA
imgflip.com

AMP Pages

Accelerated Mobile Pages (AMP) is an open-source project by Google aimed at creating fast-loading mobile web pages.

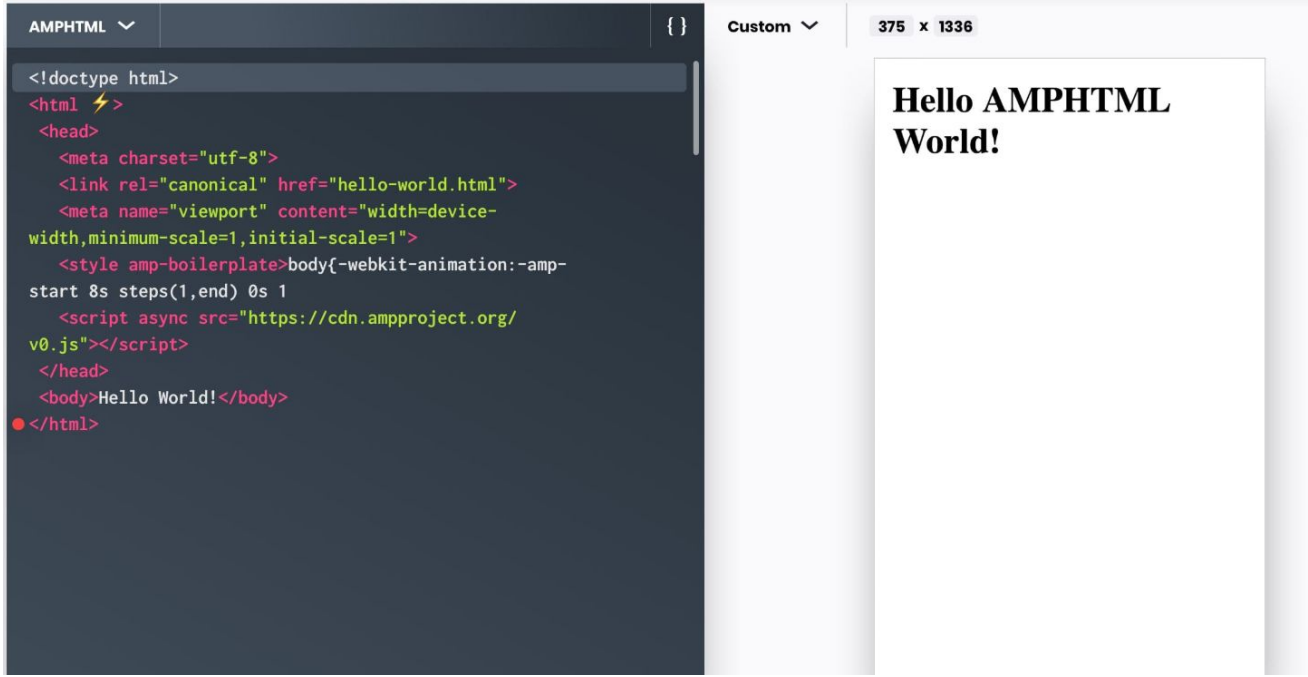


AMP Pages


3 Integral Parts of AMP
01
HTML
AMP HTML
02
{.js}
JavaScript
AMP JavaScript
03
AMP Cache

AMP HTML

Platforms like Google Search identify AMP pages using <html ⚡> or <html amp> tags and automatically add them to the AMP Cache after crawling.

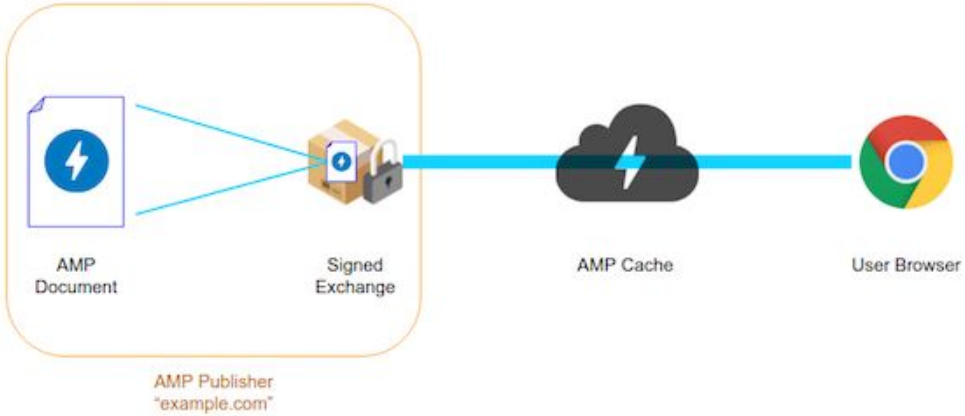


AMP JS

```
<amp-script src="http://example.com/my-script.js" width="300" height="100">
  <p>A single line of text</p>
</amp-script>
```

`amp-script` lets you write and run your own JavaScript in a way that maintains AMP's performance guarantees.

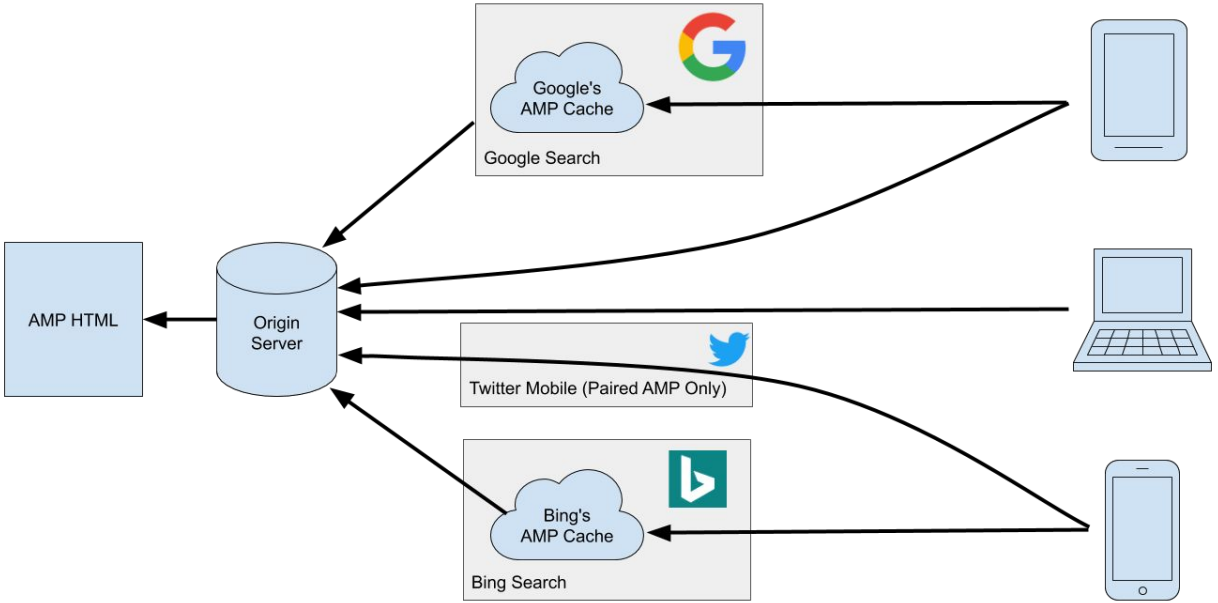
So What is an AMP Cache?



An AMP Cache is a proxy-based content delivery network (CDN) for delivering valid AMP documents. AMP Caches are designed to:

- Serve only valid AMP pages.
- Allow AMP pages to be preloaded efficiently and safely.
- Perform additional user-beneficial performance optimizations to content.

How does my AMP page get cached?



- Platform Discovery
- Cache URL Request
- Publisher Addition

AMP in Washington Post

23%

Increase in mobile search users who return within 7 days

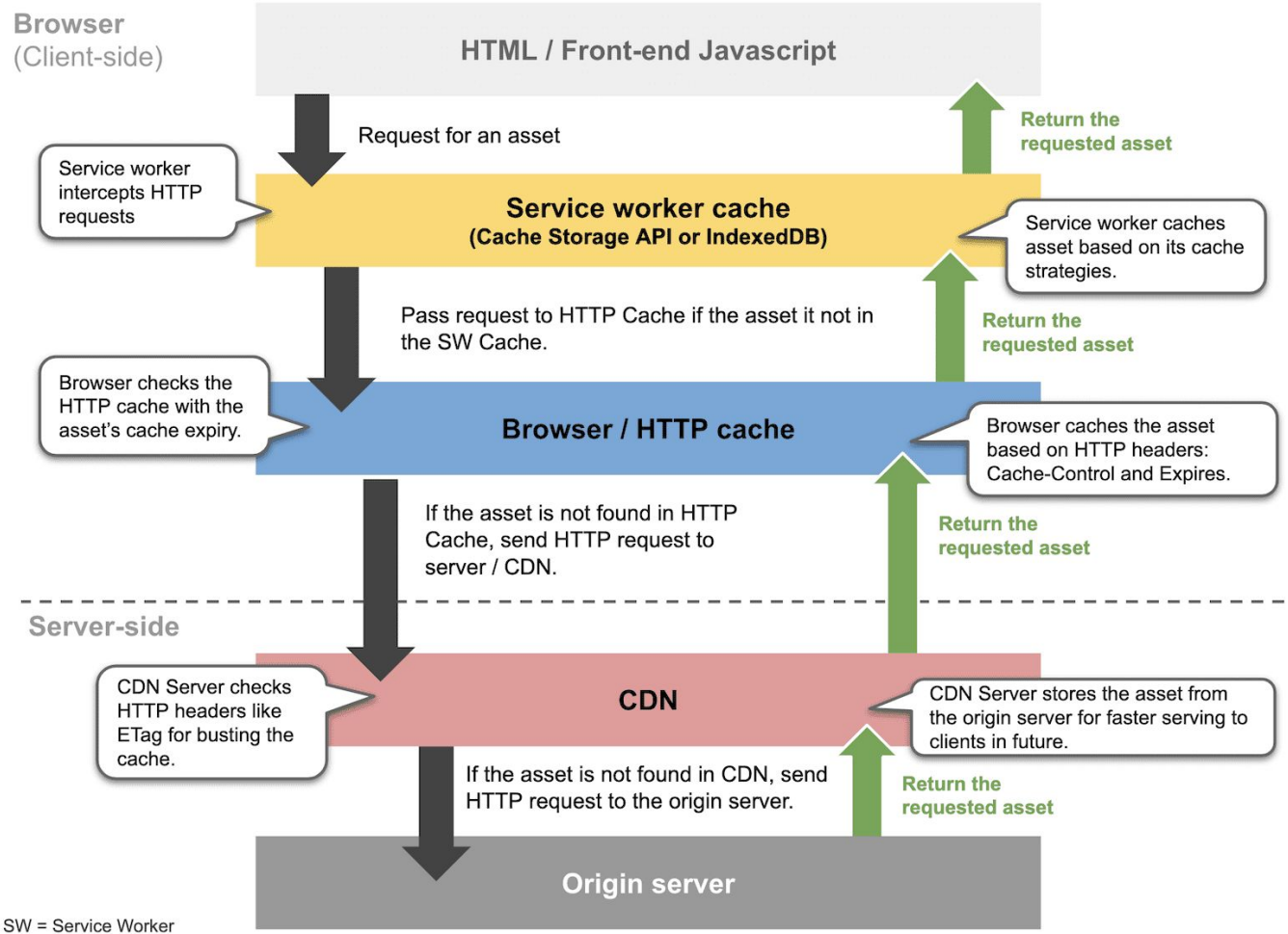
88%

Improvement in load time for AMP content versus traditional mobile web

1000+

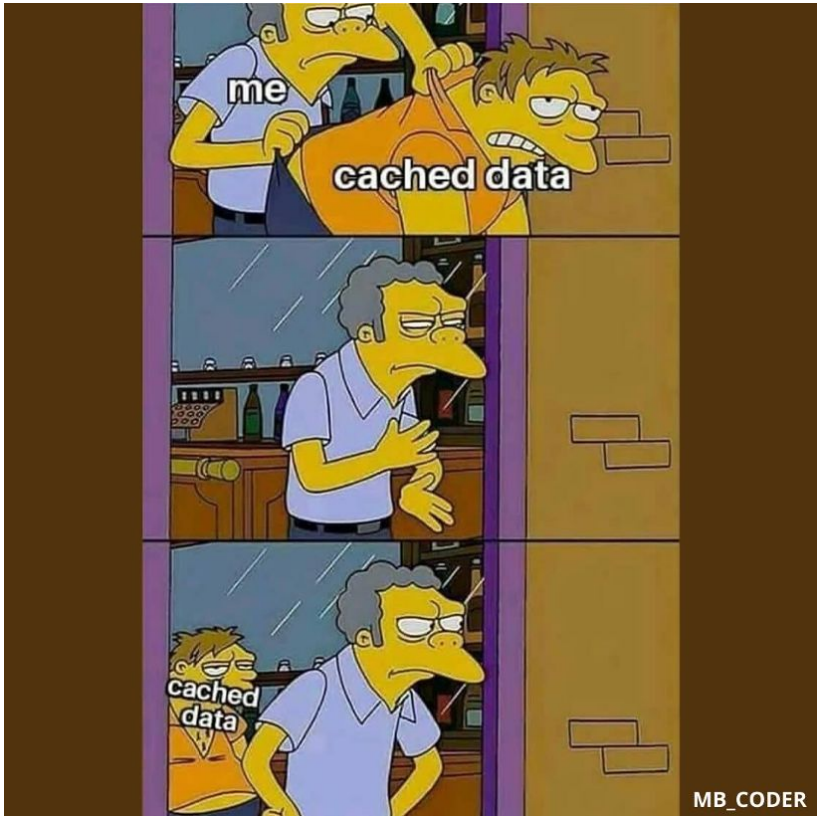
Articles the Washington Post publishes in AMP HTML daily

From amp.dev

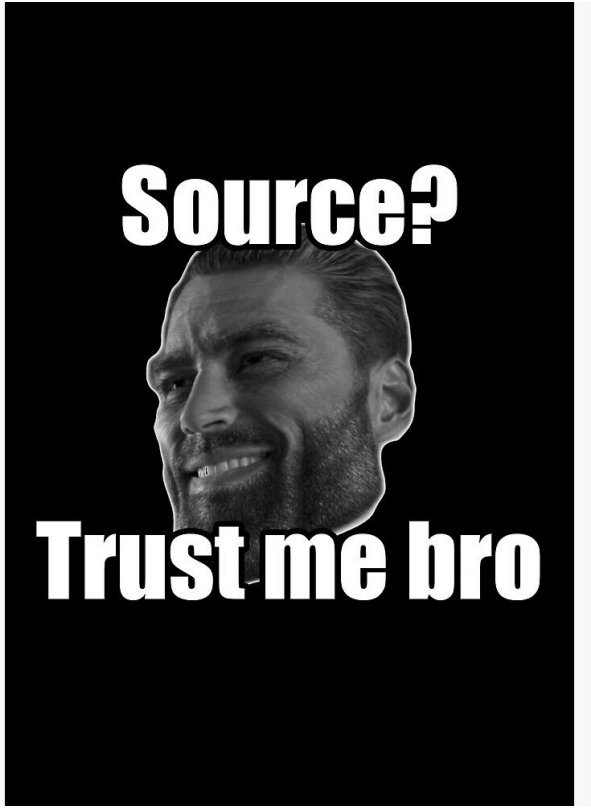


* SW = Service Worker

That’s why this happens!



Thank you for bearing with us for so long.


Source?
Trust me bro

References (Caching Basics)

1. https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching
2. https://nitropack.io/blog/post/web-caching-beginners-guide
3. https://www.mnot.net/cache_docs/
4. https://tomayko.com/blog/2008/things-caches-do

References (Caching Strategy)

1. https://youtu.be/RtOyBwBICRs?si=IrqpxAe4-qnbXMZB
2. https://blog.bytebytego.com/p/top-caching-strategies
3. https://blog.algomaster.io/p/top-5-caching-strategies-explained
4. https://codeahoy.com/2017/08/11/caching-strategies-and-how-to-choose-the-right-one/
5. https://youtu.be/2zIFUqTx_TU?si=jeT9xfqCpSKMQ2F4
6. https://wp-rocket.me/wordpress-cache/different-types-of-caching/
7. https://youtu.be/m7mPhY95uVg?si=PR8oM0TgRYUgA1O4

References (Caching in Modern Web- PWA)

1. https://developer.chrome.com/docs/workbox/caching-strategies-overview
2. https://pagepro.co/blog/what-is-pwa/
3. https://medium.com/animall-engineering/best-caching-strategies-progressive-web-app-pwa-c610d65b2009
4. https://alokai.com/blog/pwa
5. https://anuradha.hashnode.dev/role-of-service-worker-in-pwas
6. https://www.zignuts.com/blog/what-is-pwa
7. https://simicart.com/blog/pwa-service-worker/
8. https://clevertap.com/blog/progressive-web-apps/
9. https://web.dev/articles/service-worker-caching-and-http-caching

References (Caching in Modern Web- AMP)

1. https://amp.dev/documentation/guides-and-tutorials/develop/custom-javascript#enhance-amp-components
2. https://amp.dev/success-stories/washingtonpost
3. https://instapage.com/blog/amp/
4. https://www.tigren.com/blog/pwa-vs-amp/


ANY QUESTIONS?
imgflip.com