Day 6 Cache, Proxy, CDN

Lecture → Trending page
→ Tab
→ Video
→ Thumbnail
→ Text
→ Trending image
→ Tags

Browser → Site Redirects
→ Description, Name, Image, Price, Category

Post Ranking
Trending / Hot / Recent / Fresh

Search → Place → static contents

Zomato → Text search market report
→ Just match report

Suggestions → Menu
→ Highly ordered recommendations

Cache

Read not write from DB for all static contents, especially when we allow this using Caching

Caching → Locality of reference principle

→ Precalculated Results
→ Pre-generated Data
→ Strong copies

When to have Cache?

We should identify cache as much as possible.
We can't store that big, we cache ideas to the user as possible

Cache | DB
→ Commonly used contents
→ Stale
→ Hybrid Library
→ Restricting
→ Referability
→ Changes
→ Large DBs

→ Cache Eviction Policy
→ what all data to remove when cache is full

1) LRU - (Least Recently Used)
Least Recently Used
(MRU) - most cache for LRU
→ Tag → Type → Date

2) FIFO
3) LIFO
4) MRU
5) RR

Most LRU is most frequently read but LRU is most frequently read

Cache hit: If data is present in Cache

Cache Miss: If data is absent in Cache

we try to increase # of cache hits so our cache eviction policy signal

Maximum Cache hit
Minimum Cache miss

To prevent the stale data, it is called Cache Invalidation

Data is not consistent between DB & Cache

So we need to fetch and in which order we should R/W to DB or Cache

Approach 1

This is write through Cache

Approach 2

This is write back Cache

So data becomes Write Cache before hour approach

Write Through Cache | Write Back Cache

Pros
→ Data consistency
→ Reliable

Cons
→ Latency is high
→ Performance hit
→ Write Through on write

Pros
→ Low latency
→ Performance high
→ Write Back

Cons
→ Data Integrity & Reliability

Client's Request Cache

Cache coherency:
If we have different Cache in different servers, data is not consistent. So when this one updated Cache.

Rules is most commonly used related Time improve Cache coherency

Rules | Manual

→ Consistent storage
→ Distributed Cache

→ No consistent storage

Distributed Cache

Advantages/Disadvantages
→ Scale
→ Complexity
→ Cache coherency
→ Cache Eviction
→ Required extra infrastructure
→ Cache invalidation is wrong

Client

so redis is used as distributed cache. Redis stands for key, value. Redis is key value database.

redis free or redislabs.com

RC

1) we can cache mostly accessed objects content
2) Cache vs DB

Proxy

→ Intermediary Component that acts on behalf of another entity (entity can be client, server, or client)

Client A → Server → Proxy → Response → Client B

Proxy on behalf of client

Reverse Proxy

Forward Proxy

→ Client side proxy
→ Anonymity requests
→ Content filtering
→ Access Control
→ Caching
→ Server may not aware about client

→ Server side proxy
→ Load balancing
→ Security, Firewall, can act as web firewall
→ Caching

→ Client hide or Hide DB can internally talk using gRPC

Service Routing

DNS
Name → service IP mapping
Resolution, Proxy, Load Balancing
Why not? Stale Cache, Load balancing

Key concern
Can we place Proxy in front of LB?
Proxy in front of LB
Proxy behind LB

→ Usually LB + Service Discovery are centralized together
→ Advantages: Avoid Cross-AZ traffic
→ Service registration
→ Service registration: The new server after starting, with Service Discovery of Service Routing

Load balancer is distributed. Even if it fails, one server, is never but yourself has 4A of backup

Load Balancing Strategies

→ Round Robin
→ Simple & random
→ Will not aware how many load they received
→ Weighted Round Robin
→ Least Connections
→ Least req. to server with fewer connections
→ Best latency on Local
→ Weighted Round Robin
→ Latency weighted allocation
→ Least Response Time
→ Least Connections
→ Least Response Time
→ Least Connections
→ Least Response Time
→ Least Connections