Caching & Content Delivery Network (CDN)

I. The Role of Caching

Caching is the first line of defense in System Design. It acknowledges a simple truth: Fetching data from memory (RAM) is orders of magnitude faster than fetching it from a disk (Database).

1. Where do we Cache?

- **Client-side: Storing assets in the browser to avoid redundant network calls.**
- **CDN (The Edge): Storing static assets (images, JS, CSS) at servers physically close to the user.**
- **Application Cache: Using tools like Redis or Memcached to store database query results or session data.**

> **Interview Talk Track:**
>
> **"I implement caching to solve two main problems: High Latency and Database Bottlenecks. By moving frequently accessed, 'read-heavy' data from the disk into RAM, we can reduce response times from 100ms+ down to sub-10ms. This not only makes the app feel faster but also saves our database from crashing during traffic spikes."**

II. Caching Strategies (Write/Read Patterns)

How you move data into the cache is a critical architectural decision.

1. Cache-Aside (Lazy Loading)

The application looks at the cache first. If it's a miss, it fetches from the DB and updates the cache.

- **Best for: General-purpose, read-heavy workloads.**

2. Write-Through

Data is written to the cache and the DB at the same time.

- **Best for: Maintaining high consistency between cache and DB.**

3. Write-Behind (Write-Back)

The app writes only to the cache. The cache then updates the DB asynchronously (later).

- Best for: Write-heavy applications (like logging or real-time counters).
- Risk: If the cache server crashes before the DB is updated, data is lost.

Interview Talk Track:

"When deciding on a caching strategy, I look at the Read vs. Write ratio. For most web apps, I use the Cache-Aside pattern because it's simple and robust. However, if the system needs to handle massive bursts of writes—like a 'Like' button on a viral post—I'd consider a Write-Behind strategy to batch those updates and reduce the pressure on my database."

III. Cache Invalidation & Eviction

"There are only two hard things in Computer Science: cache invalidation and naming things." — *Phil Karlton*

1. Invalidation: How to handle stale data?

- TTL (Time to Live): Set an expiration timer.
- Write-through: Update cache whenever the DB is updated.

2. Eviction: What happens when the cache is full?

- LRU (Least Recently Used): Discards the data that hasn't been looked at for the longest time. (Most common).
- LFU (Least Frequently Used): Discards data that is accessed the least often.

IV. Content Delivery Network (CDN)

A CDN is essentially a distributed cache for the entire planet.

- Edge Servers: Located in "Points of Presence" (PoPs) globally.
- Origin Server: Your main backend server.

Interview Talk Track:

"A CDN is essential for any global application. Without it, a user in Tokyo would have to wait for a 200ms round-trip to a server in New York just to load a logo. By using a CDN, we serve that logo from a server in Tokyo itself. This

significantly reduces the 'Time to First Byte' (TTFB) and protects our origin server from being hammered by requests for static files."

Practice Interview Questions (Day 17–18)

1. "What is the 'Cache Stampede' problem, and how do you fix it?"

- **Answer: A Cache Stampede happens when a very popular item expires (TTL ends) and thousands of users hit the database at the same time to refresh it. We fix this by using 'X-Fetch' (re-calculating the value slightly before it expires) or using locking so only one request goes to the DB to refresh the cache.**

2. "Why shouldn't you cache everything?"

- **Answer: Caching adds complexity and costs. If data changes constantly (like a stock price ticker), the overhead of invalidating the cache might make it slower than just reading from the DB. Caching is for data that is read frequently but changed infrequently.**

3. "Explain the difference between a Pull CDN and a Push CDN."

- **Answer: * Pull CDN: The CDN asks your server for content only when a user requests it. (Lower maintenance).**
    - **Push CDN: You manually upload content to the CDN. (Better for very large files or infrequent updates).**

4. "How do you handle sensitive user data (like a bank balance) in a cache?"

- **Answer: We generally avoid caching highly sensitive, rapidly changing data. If we must, we ensure the cache is encrypted, has a very short TTL, and is never cached at the 'Edge' (CDN) where it could be exposed to other users.**

5. "What metric helps you decide if your cache is effective?"

- **Answer: The Cache Hit Ratio. If your ratio is 95%, your cache is doing great. If it's 10%, you are likely caching the wrong data or your TTL is too short, and you're still putting too much load on your database.**