10 Cache Concepts for Software Engineers

1. Key Points:

- Client-side caching includes browser cache for CSS, JavaScript, images, and service workers for offline functionality.
- Server-side caching reduces backend load through page, fragment, and object caching.
- Database caching involves row-level storage for hot data and query caching for frequent queries.
- Application-level caching stores specific data objects and results of expensive calculations.
- Distributed caching spreads data across multiple machines for scalability and fault tolerance.
- Content Delivery Networks (CDNs) store static content on edge servers to reduce global latency.
- Cache replacement policies include LFU (least frequently used), LRU (least recently used), and MRU (most recently used).
- Hierarchical caching involves multiple cache levels (L1, L2) balancing speed and capacity.
- Cache invalidation strategies include TTL expiration, manual invalidation, and event-based updates.
- Caching patterns include write-through, write-behind, and write-around, each managing data consistency and freshness.

2. Summary: The email presents ten essential caching concepts for software engineers. These cover various cache types —client-side, server-side, database, application-level, and distributed caching—each optimized for different performance and scalability needs. It emphasizes the importance of cache replacement policies (LFU, LRU, MRU), hierarchical caching for balancing speed and capacity, and strategies for cache invalidation to prevent stale data. Additionally, it discusses caching patterns such as write-through, write-behind, and write-around, which help maintain data consistency. Understanding these caching strategies is crucial to building high-performance, scalable systems while managing the risks of stale data.

3. Actionable Insights:

- Deep dive into cache replacement policies to understand their impact on cache hit ratios and system performance.
- Explore hierarchical caching architectures to optimize for speed and capacity.
- Study cache invalidation techniques, especially event-based invalidation, to maintain cache consistency.
- Analyze different caching patterns and select appropriate ones based on system requirements.
- Understand how to implement distributed caches for scalability and fault tolerance in large systems.
- Investigate CDN strategies for reducing latency through edge server caching.
- Review common caching pitfalls related to stale data and bugs to ensure robust cache management.
- Apply these concepts to improve existing system performance and scalability strategies.