Alex Xu - System Design Interview (2nd ed.)

1. Scale from zero to millions of users

Single server setup

Database

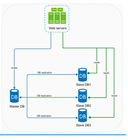
Which databases to use?

- Relational (SQL or RDBMS)
- Non-relational (NoSQL)
  - Key-value
  - Graph
  - Columns
  - Documents

Vertical scaling vs horizontal scaling

- Vertical scaling (scale up)
  - has a hard limit — It is impossible to add unlimited CPU and memory to a single server
  - does not have failover and redundancy — If one server goes down, the website/app goes down with it completely
- Horizontal scaling (scale-out)

Load balancer



Database replication

- Most applications require a much higher ratio of READS to WRITES — thus, the number of slave databases in a system is usually larger than the number of master databases
- Pros
  - Better performance — allows more queries to be processed in parallel — all writes and updates happen in master nodes; whereas, read operations are distributed across slave nodes.
  - Reliability — do not need to worry about data loss, because data is replicated
  - High availability — even if a database is offline, you can access data stored in another database server

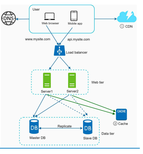
Cache

- Cache tier
  - "read-through cache" strategy
  - Q: If we have few similar requests at the same time - should they all go for data or only one should go while others should wait until data appears in the cache (it is much harder to implement)?
- Considerations for using cache
  - Decide when to use cache
  - Expiration policy
  - Consistency
  - Mitigating failures
    - A single cache server represents a potential single point of failure (SPOF)
    - multiple cache servers across different data centers are recommended to avoid SPOF
  - Eviction Policy
    - Least-recently-used (LRU) — the most popular policy
    - Least Frequently Used (LFU)
    - First in First Out (FIFO)

Content delivery network (CDN)

- A CDN is a network of geographically dispersed servers used to deliver STATIC content
  - DYNAMIC content caching is a relatively new concept and beyond the scope of this book

- Q: similar question here - will CDN prevent similar requests to our server at the same time?
- Considerations of using a CDN
  - Cost — infrequently used assets can be moved out of CDN
  - Setting an appropriate cache expiry
  - CDN fallback — clients should be able to detect the problem and request resources from the origin
  - Invalidating files
    - Use CDN's API
    - Use file versioning, e.g. image.png?v=2

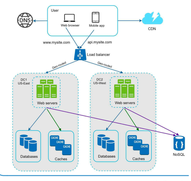


Stateless web tier

- STATEFUL architecture
  - Servers remember client data (state) from one request to the next
  - Needs sticky sessions in load balancer
- STATELESS architecture
  - State data is stored in a shared data store and kept out of web servers
    - NoSQL data store is usually chosen here as it is easy to scale

Data centers

- Users are geoDNS-routed



- Q: Why not keep the state in each DC? Why is it shared? — I believe that if it is just a user session and other NON-critical session data (no changes to databases then it can be stored in each DC)
- Should continue work when the whole DC goes down! — Several technical challenges here
  - Traffic redirection
  - Data synchronization
  - Test and deployment

Message queue

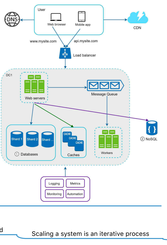
- Decouples the components
  - the producer can post a message to the queue when the consumer is unavailable to process it
- Q: Should it be inside DC or cross-DC?

Logging, metrics, automation

- Logging — Monitoring error logs is important
- Metrics
  - Host level metrics — CPU, Memory, disk I/O, etc
  - Aggregated level metrics — the performance of the entire database tier, cache tier, etc
  - Key business metrics — daily active users, retention, revenue, etc
- Automation — Continuous integration is a good practice

Database scaling

- Vertical scaling
- Horizontal scaling
  - also known as sharding
  - The sharding key - the most important factor to consider
    - Complexities to consider
      - Resharding data
      - Celebrity problem — aka a hotspot key problem
      - Join and de-normalization — A common workaround is to de-normalize the database so that queries can be performed in a single table



Millions of users and beyond

- Scaling a system is an iterative process — More fine-tuning and new strategies are needed to scale — e.g. you might need to optimize your system and decouple the system to even smaller services