System Design Interview Study Notes II

Yusuf Ali Koyuncu · Follow
10 min read · Dec 21, 2023

Listen Share

Hello there, in this article I am here with the 2nd part of this series. You can access the previous part of the series from the link below. So let's get started;

System Design Interview Study Notes I

Hello, the requirements of the applications we have developed in line with the increasing number of users in recent...

Performance vs Scalability

- When the resources of a system are increased, the service is called scalable if its performance increases in the same proportion.
- Applications should be designed with scalability in mind, they cannot be designed and implemented later.
- If there is a performance problem; the system is slow for a single user.
- If there is a scalability problem; the system is fast for a single user but slow under heavy load.

Latency vs Throughput

- Latency is the time required to perform an action or produce a result.
- Throughput is the number of actions or results executed per unit time.
- The goal is minimum latency, maximum throughput.
- Example;

— Latency → To produce a car in 8 hours.

— Throughput → 120 cars per day, 5 cars per hour.

- **Availability (A):** Each request responds with the latest state of the information, without being precise.
- **Partition Tolerance (P):** The system continues to function despite being partitioned due to network failure.
- Consistency-Partition Tolerance; Waiting for a response from a partitioned drop can cause a timeout. Consistency-Partition Tolerance is a good choice if atomic read and write meets your business needs.

- Availability-Partition Tolerance; Responses return the latest version of the most recent non-partitioned data available on any node. Write operations take time when the partition is resolved. Choose Availability-Partition Tolerance if the system needs to work despite external failures.

Consistency Patterns

- **Weak Consistency**; After typing, readers may or may not see it. It works well for real-time use, e.g. video chat, online games, etc.
- **Eventual Consistency**; After typing, readers will eventually see it. Data is replicated asynchronously. Example; It is used in systems such as DNS, e-mail etc. It is preferred in systems with high availability.
- **Strong Consistency**; Readers will see it after writing. It is replicated simultaneously. For example; used in systems such as file system, RDBMS etc.

Availability Patterns

Fail-over

- Active-Passive; Ping is sent between active and passive servers. If the ping is interrupted, the passive takes over the IP of the active server. This structure is also known as master-slave.
- Active-Active; Both servers are active, and the load is shared between them. If the servers are public, DNS must know both IPs. This structure is also known as master-master.

Replication

- There are two methods; master-master and master-slave.

Availability in Numbers

- Measured by uptime or downtime.
- In sequence = Availability (Total) = Availability (Foo) * Availability (Bar)
- In parallel = Availability (Total) = 1 — (1 — Availability (Foo)) * (1 — Availability (Bar))

Disadvantages

- Fail-over adds more hardware and complexity.
- There is a potential for data loss (there may be breaks or errors when copying data from the active system to the passive system).

Domain Name System — DNS

It is the structure that translates the “www.example.com” structure into an IP address. It works in a hierarchy. Likewise, it provides information about which DNS server to communicate with when searching without a router or ISP. It can be browser or OS cache (with TTL).

1. NS record (name server) — determines the DNS server for the domain/subdomain.
2. MX record (mail exchange) — specifies the mail server for the message.
3. A record (address) — redirects a name to an IP address.
4. CNAME (canonical) — converts a name to NAME or CNAME (example.com to <www.example.com>).

- DNS Method; Weighted round-robin, Latency-based, Geolocation-based

Disadvantages

- DNS server access is delayed even with cache.
- DNS server methodology is complex and usually managed by large companies or governments.

Content Delivery Network — CDN

A globally distributed network of proxy servers that deliver content closer to the user. It usually hosts static content, but sometimes dynamic content is supported by cloud providers. Users get content from data centers close to them. Your server does not have to serve the requests that the CDN fulfills.

- **Push CDNs:** Works well for sites with little traffic or sites whose content doesn't change much (low traffic, high storage)
- **Pull CDNs:** Works well for high traffic sites. The first request may take a long time due to cache processing. Works with TTL (high traffic, low storage)

Disadvantages:

- Cost may increase depending on traffic.
- Content may become stale if updated before the TTL expires.
- Requires changing URLs for static content to point to the CDN.

Load Balancer

Distributes client requests to request processing places such as servers or databases. It prevents requests from going to unhealthy servers, prevents overloading of resources, and helps eliminate single points of failure. It also provides SSL termination and Session persistence. To protect against failures, it is a common practice to set up multiple instances in active-passive or active-active structure. It can route traffic according to different methods; Random, Least loaded, Session/cookies, round-robin or weighted round-robin, Layer 4 and Layer 7.

- **Layer 4**: Distributes load by looking at the transport layer. There is IP based routing without packet content. Provided by NAT.
- **Layer 7**: Distributes load by looking at the application layer. It can contain headers, messages and cookies.
- **Horizontal Scaling**: Can improve performance and availability.

Disadvantages:

- If the load balancer does not have enough resources or is not configured properly, bottleneck may occur.
- Complexity increases.

Reverse Proxy Server

It centralizes the internal services and provides a unified interface to the outside. It receives the incoming request, has the internal servers respond to it and returns the response to the client. Some servers provide both load balancer and reverse proxy server.

Benefits;

- Provides increased security
- Increased scalability and flexibility
- SSL termination
- Compression for response
- Caching
- Static content

Disadvantages:

- Complexity increases.
- Having one tannin becomes a single point of failure. Defining more leads to complexity.

Application Layer

Separating the web layer from the application/platform layer allows you to scale and configure both layers independently. SRP supports small, autonomous services that work together. A Service Discovery structure is established for these services to find each other.

Disadvantages:

- Different approaches are required for a loosely coupled system.
- It is complex in terms of deployment and operation.

Databases

Relational database management system (RDBMS)

A relational database, such as SQL, is a collection of data items organized in tables. Its most important structure is “ACID”.

- **Atomicity:** Transactions are all or nothing.
- **Consistency:** Any transaction will bring the database from one current state to another.
- **Isolation:** The results of transactions serially and concurrently are the same.
- **Durability:** Once a transaction is done, it will remain so.

Ways to scale Relation databases;

Replication:

- **Master-Slave:** The master provides read and write service, replicates writes to one or more slaves that only provide read service. Slaves can also replicate to other slaves in a tree-like fashion. If the master goes offline, the system can continue to operate in read-only mode until a slave is promoted to master or a new master is provisioned. Additional logic is required to promote a slave to a master.
- **Master-Master Replication:** Both masters provide read and write services and coordinate with each other on write operations. If one of the masters goes down, the system can continue to work with both reads and writes. You will need a load balancer or you will need to make changes to your application logic to determine where to write. Violates ACID.

Disadvantages:

- There is a possibility of data loss if the master fails before any newly written data is replicated to other nodes.
- Writes are replayed to the read replicas. If there are too many writes, the read replicas can get bogged down from replaying writes and cannot read as much.
- The more read slaves there are, the more replicas you need to do, leading to greater replication latency.
- On some systems, writing to the master can create multiple threads to write in parallel, while read replicas only support writing sequentially with a single thread.
- Replication adds more hardware and additional complexity.

Federation (Functional Partitioning):

- Partitions databases according to their function. For example, instead of one monolithic database you can have three databases: forums, users and products, resulting in less read and write traffic to each database and therefore less replication latency. Smaller databases result in more data that can fit in memory, resulting in more cache hits due to improved cache locality. Since there is no single central master serializing writes, you can write in parallel and increase throughput.

Disadvantages:

- Federation is not effective if your schema requires large functions or tables.
- You need to update your application logic to determine which database to read and write to.
- Combining data from two databases is more complicated with a server connection.

Sharding:

- Distributes data across different databases so that each database can only manage a subset of the data. Taking a user database as an example, as the number of users increases, more shards are added to the cluster.

Disadvantages:

- You need to update your application logic to work with shards, which can result in complex SQL queries.
- Data distribution in a shard can be unbalanced. For example, a number of power users in one shard may cause more load to be placed on that shard compared to others.
- Rebalancing introduces additional complexity. A sharding function based on consistent hashing can reduce the amount of data transferred.
- Merging data from multiple shards is more complex.

Denormalization:

- Attempts to improve read performance at the expense of some write performance. Backup copies of data are written to multiple tables to avoid expensive joins. When data is distributed through techniques such as federation and sharding, managing joins between data centers further increases complexity. Denormalization can eliminate the need for such complex joins.

Disadvantages:

- Data is duplicated.
- Constraints can help keep redundant copies of information in sync, which increases the complexity of database design.
- Under heavy write load, a non-normalized database may perform worse than its normalized counterpart.

SQL tuning:

- It is important to benchmark and profile to simulate and uncover bottlenecks. SQL tuning is a broad topic and complex. Use good indexes, tighten the schema, avoid expensive joins, create partition table, create query cache.

NoSQL

It is a collection of data items represented in a key-value store, document store, wide column store or a graph database. The data is denormalized and joins are usually done in the application code. Most NoSQL repositories lack true ACID operations, preferring BASE instead.

BASE is often used to describe the properties of NoSQL databases. Compared to the CAP Theorem, BASE favors availability over consistency.

- **Basically available:** the system guarantees availability.
- **Soft state:** the state of the system can change over time, even without input.
- **Eventual consistency:** if the system does not receive input for a certain period of time, the system will become consistent for a certain period of time.

Key-Value Store:

- Usually allows O(1) read and write operations and is usually supported by memory or SSD. It allows metadata to be stored with a value. It is high performance and is usually used for simple data models or in-memory cache layer. Additional operations are handled in the application layer.

Document Store:

- A document stores all the information for a given object (XML, JSON). The document structure provides APIs or a query structure. Documents are organized by collection, tags, metadata or directories. Documents can be organized together but have different fields. It provides high flexibility and is used to work with data structures that change from time to time.

Wide Column Store:

- The basic unit of data is a column. A column can be grouped into column families. You can access each column independently with a row key and columns with the same row key form a row. Each value has a timestamp. Offers high availability and scalability. Usually used with very large data.

Graph Database:

- Each node is a record and each arc is a relationship between two nodes. Many-to-many or many-to-many relationships represent complex relationships. It provides high performance for data models with complex relationships such as social networks.

Time-series Database:

- A database optimized for time-stamped or time-series data. Time-series data are simple measurements or events that are tracked, monitored, sub-sampled and collected over time. This can be server metrics, application performance monitoring, network data, sensor data, events, clicks, transactions in a marketplace, and many other types of analytical data. A time series database is built specifically to handle time-stamped measurements and events or metrics. A TSDB is optimized for measuring change over time. The features that make time series data very different from other data workloads are data lifecycle management, summarization, and wide range scanning of many records.

SQL or NoSQL

Causes of SQL:

- Structured data
- Solid scheme
- Relational data
- The need for complex joining
- Operations
- Clear patterns for scaling
- More built-in: developers, community, code, tools, etc.
- Searches by index are very fast

Reasons for NoSQL:

- Semi-structured data
- Dynamic or flexible schema
- Non-relational data
- No need for complex merges
- Store large numbers of TB (or PB) of data
- Very data intensive workload
- Very high throughput for IOPS

> The greatest obstacle to discovery is not ignorance — it is the illusion of knowledge. — Daniel J. Boorstin

Reference

- https://github.com/madd86/awesome-system-design
- https://economictimes.indiatimes.com/definition/systems-design
- https://github.com/binhnguyennus/awesome-scalability
- https://github.com/donnemartin/system-design-primer

System Design Interview | Software Development | Software Engineering | Software Architecture | Distributed Systems

Follow

Written by Yusuf Ali Koyuncu

137 Followers

Back-end Software Engineer - Part-Time Open-Sourcerer. Focuses on Go.

More from Yusuf Ali Koyuncu

Yusuf Ali Koyuncu

System Design Interview Study Notes I

Hello, the requirements of the applications we have developed in line with the increasing number of users in recent years and different use...

10 min read · Dec 12, 2023

200

Yusuf Ali Koyuncu

Software Architecture Study Notes III

Hello there, in this article I am here with the 3rd part of this series. This is the continuation of Distributed Systems Patterns. You can...

14 min read · Jan 23, 2024

1

Yusuf Ali Koyuncu

Software Architecture Study Notes II

Hello there, in this article I am here with the 2nd part of this series. You can access the previous part of the series from the link...

10 min read · Jan 17, 2024

53

Yusuf Ali Koyuncu

Software Architecture Study Notes I

Hello there, in recent years, new software architectures have been developed due to increasing internet usage and new needs. For this...

10 min read · Jan 8, 2024

3

See all from Yusuf Ali Koyuncu

Recommended from Medium

Yusuf Ali Koyuncu

System Design Interview Study Notes I

Hello, the requirements of the applications we have developed in line with the increasing number of users in recent years and different use...

10 min read · Dec 12, 2023

200

Chaudhary Vivek Kadiyan

20 Tricky Java Architect Interview Question 2024

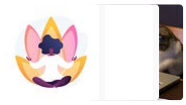
I am posting some of the tricky and important interview questions that have been asked to me , related to Java/Spring...

4 min read · Feb 4, 2024

187 5

Lists

General Coding Knowledge
20 stories · 917 saves



Stories to Help You Grow as a Software Developer
19 stories · 810 saves

Leadership
46 stories · 235 saves

Coding & Development
11 stories · 445 saves

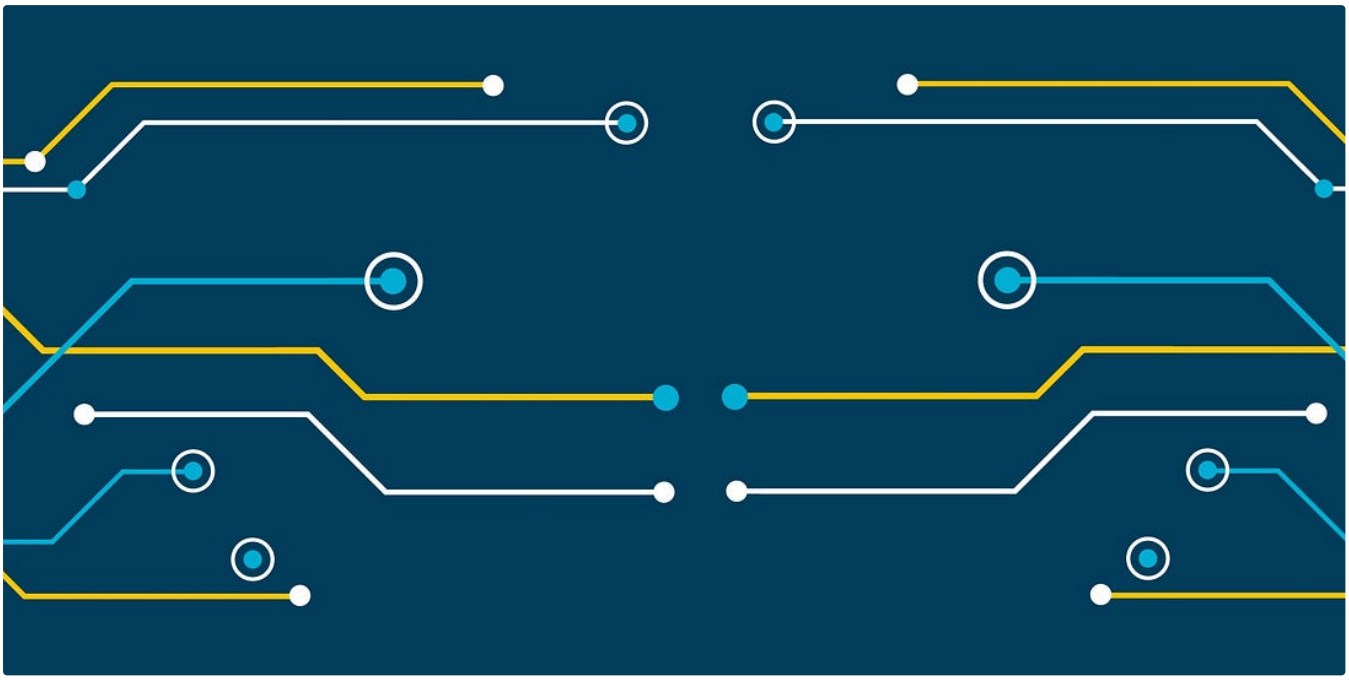
Vishal Barvaliya

Golden Rules for System Design Interview.

System Design is explained in detail with examples.

· 17 min read · Jan 6, 2024

595 2



Capital One Tech in Capital One Tech

10 microservices design patterns for better architecture

Consider using these popular design patterns in your next microservices app and make organization more manageable.

11 min read · Jan 10, 2024

1K 4

Kajol Kumari

Must-Know System Design Concepts: A Comprehensive Guide

System design is a critical aspect of software engineering, and understanding key concepts is essential for building scalable, reliable...

5 min read · Dec 3, 2023

154 1

Talha Şahin

High-Level System Architecture of Booking.com

Take an in-depth look at the possible high-level architecture of Booking.com.

✨ · 8 min read · Jan 10, 2024

 2.5K  18