Basic Concepts for System Design Interview Preparation

Why is system design important?

For large-scale software applications, System design provides a high-level understanding of the components and their relationships. In other words, it helps us to define architecture by breaking down the system into small components and making it easier to understand.

So it is critical for software engineers or tech professionals to understand various system design concepts to make informed decisions about scalability, performance and various tradeoffs. On another side, system design is also one of the important concepts that tech companies ask during the interview process.

What is System Design?

System design deals with designing a system that meets functional and non-functional business requirements. While designing such systems, one needs to think about various components, complexity, performance, etc. So it would be best to have an excellent understanding of essential system design terms.

We need to know about scalability(horizontal and vertical), caching, load balancing, data partitioning, various types of databases, network protocols, database sharding, and many more. Similarly, we need to consider various tradeoffs like: latency vs throughput, performance vs scalability, consistency vs availability, etc. Overall, system design is an open-ended discussion topic. That's why most top tech firms prefer to have one or two system design interview rounds.

So let’s dive in to get familiar with the essential concepts used in system design.

Availability

Availability means: System should always remain up and return the response of a client request. In other words, whenever any user want to use the service, system must be available and satisfy user request.

Availability can be quantified by measuring the percentage of time system remain operational in a given time window. We usually define availability percentages of a system in terms of number of 9s (as shown in the table below).

enjoyalgorithms.com

Availability %	Downtime/Year	Downtime/Month	Downtime/Week
90% (one nine)	36.53 days	72 hours	16.8 hours
99% (two nines)	3.65 days	7.20 hours	1.68 hours
99.9% (three nines)	8.77 hours	43.8 minutes	10.1 minutes
99.99% (four nines)	52.6 minutes	4.32 minutes	1.01 minutes
99.999% (five nines)	5.25 minutes	25.9 seconds	6.05 seconds
99.9999% (six nines)	31.56 seconds	2.59 seconds	604.8 milliseconds
99.99999% (seven nines)	3.15 seconds	263 milliseconds	60.5 milliseconds
99.999999% (eight nines)	315.6 milliseconds	26.3 milliseconds	6 milliseconds
99.9999999% (nine nines)	31.6 milliseconds	2.6 milliseconds	.6 milliseconds

For designing a highly available system, we need to consider several factors like redundancy, load balancing, failover mechanisms, and disaster recovery strategies. Our goal should be to minimize downtime and ensure that system is always accessible and functioning as intended.

Throughput

Throughput is one of the key metrics to measure the performance of a system. It is the amount of data or the number of requests that can be processed within a given time period. In other words, we use throughput to measure the system capacity to handle multiple requests or data

streams concurrently.

High throughput is important when we are handling a large volume of data or a high number of concurrent requests. One excellent way to achieve high throughput is: Splitting up the requests and distributing them to various machines.

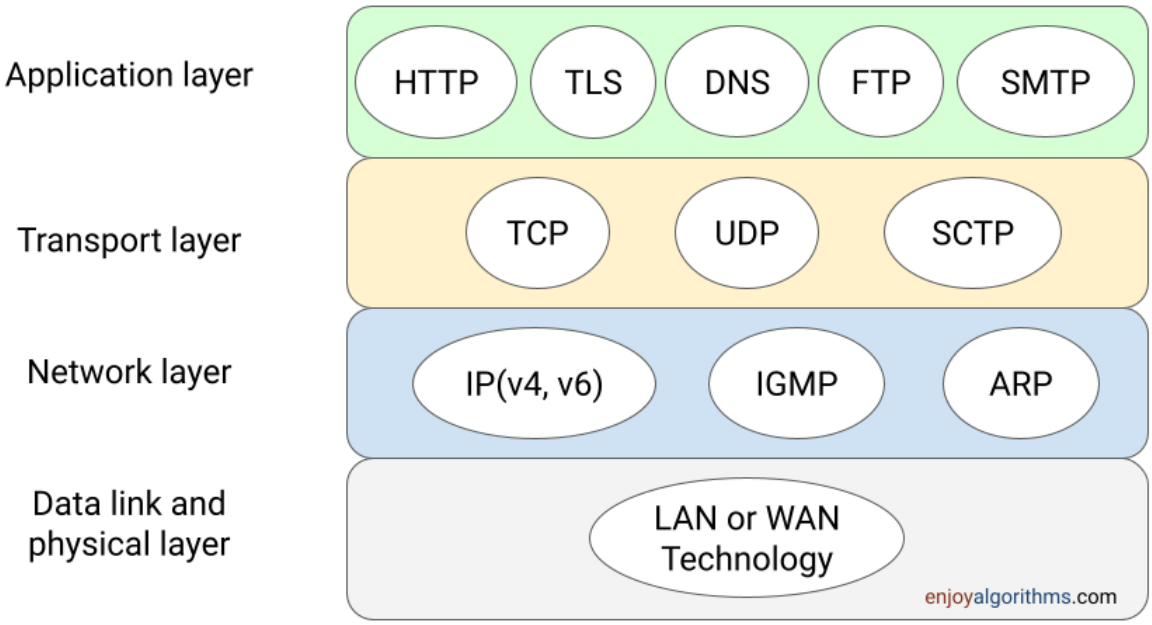
Latency

Latency is another important metric related to the performance of a system. It is a measure of the time duration to produce the result. In other words, it is the time taken by a request to receive a response from a server. This includes the time taken for a request to travel through the network, processing the request, database queries, and other operations before generating the response.

Lower latency means: Quick response times, higher system performance and a smooth user experience!

Network Protocols

Almost every system has dependency on networks. It acts as a platform for communication between users and servers or different servers. On another side, protocols are the rules governing how servers or machines communicate over the network. Some of the common network protocols are HTTP, TCP/IP, etc.



Load Balancing

Load balancers are machines that balance the load among various servers. To scale the system, we can add more and more servers. So there must be a way to direct requests to these servers so that there is no heavy load on one server. In other words, load balancers are used to to prevent a single point of failure.

- Load balancers distribute traffic, prevent service from breakdown, and contribute to the system reliability.
- Load balancers act as traffic managers and help us to maintain system throughput and availability.

Proxies

Proxy is present between the client and server. When a client sends a request, it passes through the proxy and then reaches the server. Proxies are of two types: Forward Proxy and Reverse Proxy.

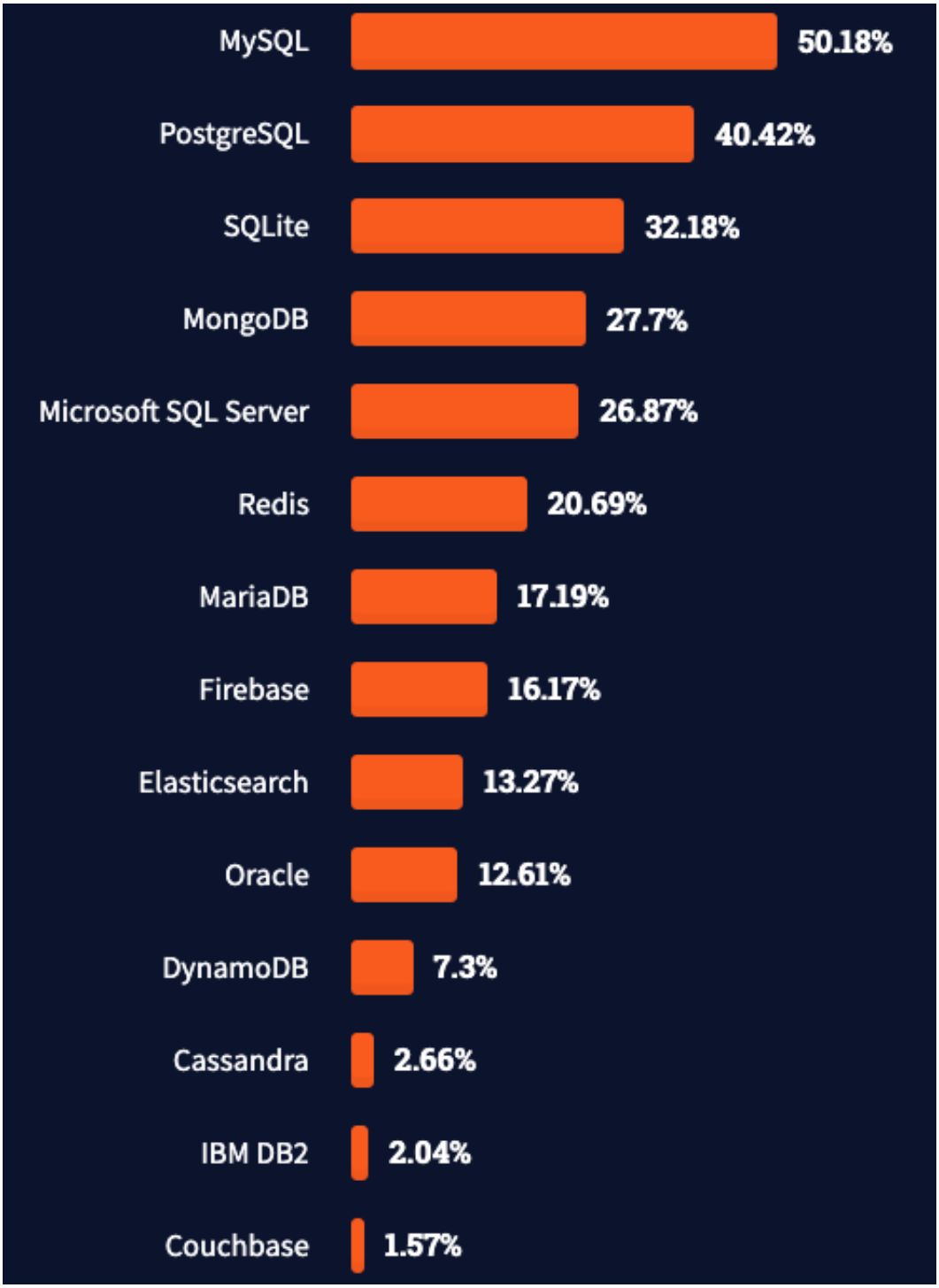
The forward proxy acts as a mask for clients and hides the client's identity from the server. Similarly, Reverse proxy acts as a mask for servers and hides server's identity from the response.

Databases

Systems are associated with data that needs to be stored for later retrieval. One way to classify databases is based on their structure, which can be either relational or non-relational.

Relational databases, such as MySQL and PostgreSQL, enforce strict relationships among data and have a highly structured organization. Non-relational databases, such as Cassandra and Redis, have flexible structures and can store both structured and unstructured data. They are often used in systems that are highly distributed and require high speed.

Database Partition is the way of dividing database into smaller chunks to increase performance of the system. It is another critical concept used to improve latency and throughput so that more and more requests can be served. Here is the list of some popular databases based on Stack Overflow Developer Survey 2021.



ACID vs BASE

Relational and Non-Relational Databases ensure different types of compliance. Relational Databases are associated with ACID compliance, while Non-Relational associated with BASE compliance.

ACID: Atomicity, Consistency, Isolation, Durability

ACID compliance ensures the relational nature of databases and ensures that transactions are executed in a controlled manner. A transaction is an

interaction with the database.

- **Atomicity** ensures that a group of operations are treated as a single unit of work. If any of these operations fail, entire transaction will fail. This is important in transactions as it ensures that all or nothing is achieved.
- **Consistency** ensures that each transaction must comply with a set of rules. It ensures that the database state changes do not corrupt the data and that transactions can move from one valid state to another.
- **Isolation** means that transactions are executed independently of each other, without affecting other transactions. This ensures concurrency in database operations.
- **Durability** ensures that any data written to the database is persisted and will remain there, even in the event of a system failure.

BASE: Basically Available Soft State Eventual Consistency

BASE compliance maintains integrity of NoSQL databases and ensures their proper functioning. It is a key factor in building the scalability of NoSQL databases.

- **Basically Available** ensures that the system is always available.
- **Soft state** provides flexibility to the system and allows it to change over time for faster access.
- **Eventual Consistency** ensures that the data in the system takes some time to reach a consistent state and eventually becomes consistent.

SQL vs NoSQL

While designing any application, one needs to be clear about the type of storage according to the system requirements. If system is distributed in nature and scalability is essential, then NoSQL databases are the best choice to go with. No-SQL databases are also preferred when amount of data is huge.

Simultaneously, SQL databases are favourable when data structure is more important. It is generally preferred when queries are complex and databases require fewer updates. However, there is always a trade-off when choosing between NoSQL vs SQL database. Sometimes, according to business requirements, a Polyglot architecture comprising both SQL and NoSQL databases is used to ensure application performance.

Scalability in distributed systems

As services grow and number of requests increases, system can become slow and performance may be affected. Scaling is the best way to address this issue, which involves increasing the capacity of system to handle more requests. There are two main ways to scale: horizontal scaling and vertical scaling.

Horizontal scaling involves adding more servers to distribute requests. This allows system to handle more traffic by spreading it across multiple servers. On the other hand, vertical scaling increases the capacity of a single machine by upgrading it with more resources such as CPU, memory, and storage. This allows the system to handle more traffic by increasing the capacity of a single machine.

Caching

Caching is an important technique that helps to improve the performance of a system by reducing its latency. It involves storing frequently used data in a cache so that it can be accessed quickly, instead of querying the database. However, implementing a cache also adds complexity to the system because it is important to maintain consistency between data stored in the cache and data stored in the main database.

Cache memory is relatively expensive, so the size of cache must be limited. To ensure the best performance, various cache eviction algorithms, such as LIFO, FIFO, LRU, and LFU, are used to manage the cache. These algorithms determine which data should be removed from cache when space is needed.

Consistent Hashing

Consistent hashing is a widely used concept in distributed systems because it offers flexibility in scaling the application. It is an improvement over traditional hashing methods, which are ineffective for handling requests over a network.

In consistent Hashing, users and servers are located in a virtual circular ring structure called a hash ring. The ring is considered infinite and can accommodate any number of servers, regardless of fixed allocation. Servers are assigned random locations based on a hash function. This concept allows for efficient distribution of requests or data among servers. It helps us achieve horizontal scaling, which increases throughput and reduces latency of the application.

CAP Theorem

CAP Theorem is an essential concept for designing networked shared data systems. It states that a distributed database system can only provide two of these three properties: consistency, availability, and partition tolerance. We can make trade-offs between three properties based on the use cases of our dbms system.

- **Consistency:** Consistency means that everything should go on in a very well-coordinated manner and with proper synchronisation. It ensures that system remains consistent and returns the results such that any read operation should give the most recent write operation.
- **Availability:** Availability means that the system is always available whenever any request is made to it. In other words, whenever any client requests the server, system should remain available and give the response irrespective of the failure of one or more nodes. Replication is used to ensure redundancy, which directly contributes to the availability of the system.
- **Partition Tolerance:** Partition Tolerance is necessary for any distributed system, so we always need to choose between availability

and consistency. Partition tolerance corresponds to the condition that the system should work irrespective of any failure or breakdown of nodes. Due to massive dependency on network calls, it is prevalent for a distributed system to fall into the trap of network failures. So partition tolerance is essential.

Some other important concepts to explore

- Server-Sent Events
- Long Polling
- Client Server Architecture
- Web Sockets
- Load Balancing Algorithms
- Peer to Peer Networks
- Master Slave Replication
- Throttling and Rate Limiting
- How to Choose the Right Database?
- Leader Election in Distributed Systems

Conclusion

System design is an essential skill to have and is equally important from the interview point of view. One needs to be well aware of all the trade-offs while designing any system. In this blog, we tried to cover all the basic concepts necessary for getting started. We hope that after reading this blog, you'll be now familiar with the basics of systems.

We are looking forward to your feedback in the message below. Enjoy learning, Enjoy system design!