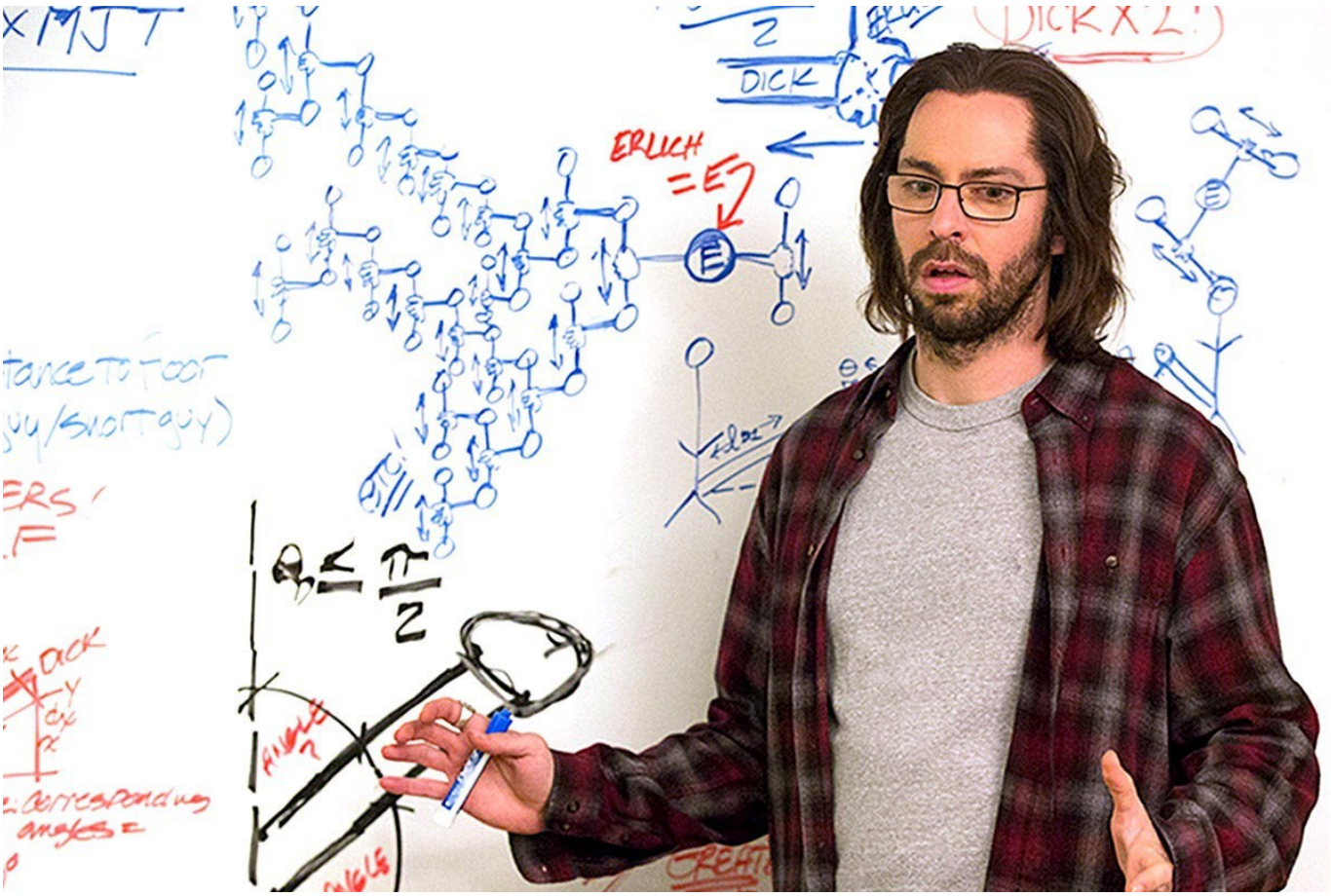
System Design Interviews 101

Tips to crack the system design interview at a Tech Company.

Animesh Gaitonde Follow

Mar 19 · 7 min read ★



What whiteboard interviews look like?

Most of the Tech Companies have a system design round as a part of their interview process. Candidates are generally asked to design a scalable system such as Facebook NewsFeed, Instagram stories, WhatsApp chat, CI/CD system, etc.

There are a lot of resources on the Internet to prepare for System Design interviews. The plethora of resources can become overwhelming for any developer to digest. However, there are common patterns observed across the questions asked by Tech Companies.

Having crystal clear idea of System Design Concepts & fundamentals helps any candidate in coming out with flying colours.

In this article, I am going to elaborate on a systematic and organized strategy to help anyone clear the interviews. Along the way, I'll also illustrate various questions asked and approaches to tackle them.

Why System Design Interviews?

Today, most of the Tech Companies invest in building scalable high performant systems. Hence, it's essential for an Engineer to have amazing design skills.

System Design Interviews are intentionally open-ended and ambiguous. It gives an Interviewer a chance to evaluate a candidate along different dimensions. In addition to design skills, it also helps to interviewer judge a candidate's overall thought process, thinking, algorithms knowledge and communication skills.

High-Level Strategy

I follow the following five-step approach in a System Design Interview:-

- Requirements & Clarifying questions
- Capacity Estimation
- Design Objectives
- API Design & Algorithms
- Database, Cache Design & Sharding

We will walk through each of the above steps in detail.

Requirements & Clarifying questions

When a problem such as Design a backend system for Uber is thrown at us, thousands of features cross our mind. In fact systems like Uber, Facebook etc evolved over years of Engineering effort & rigour. Designing a large-scale system in 45–60 mins isn't straight forward.

Since we can't design all features of a large scale system as time is limited, it's essential to narrow down the scope of the problem. Requirements can be classified into functional &

non-functional requirements. You can think of all the simple features that can be designed on a whiteboard or a piece of paper in the interview duration.

Uber

For eg: In the case of the Designing backend for Uber, you can list down the following requirements in scope:-

1. Riders must be able to book & cancel a ride
2. The system must match a Rider with a driver
3. Drivers can cancel an assigned ride
4. Drivers & Riders can both rate each other after the ride
5. Riders can pay the driver using multiple payment modes

Furthermore, asking clarifying questions is a must. You need to understand who are the users of the system, how they are going to interact with it, what are the usage patterns going to be like. These questions will help you in identifying the bottlenecks, removing redundancies & scaling the overall system.

Capacity Estimation

On clarifying the requirements of the system, the next question that must be asked is how many users are going to leverage our system? This is a significant question as it will help answer many design bottlenecks later.

A rough idea of active users helps you estimate how many requests your system will receive and you can compute the data storage required as well. Furthermore, you can get an idea of whether a system is a read-heavy or a write-heavy one. This will influence your choice of data store (SQL Vs NoSQL) and the need for a caching layer in your application.

Twitter

Let's say you are designing a system like Twitter. In this case, you can make the following estimates:-

- 200 Mn active users with 600 Mn daily tweets
- Every user having on an average of 100 followers
- A tweet retweeted at least 10 times on an average
- A person sharing 1 media file in every 10 tweets

On computing the estimates, you will get clarity on the extent of horizontal scaling or number of servers needed in the system. Moreover, using the rough estimate for the

amount of data that needs to be stored, will help you plan your database servers accordingly.

Design Objectives

One of the primary requirement for any system is Latency & Throughput. Latency is the amount of time it takes to process a given request from the client. Whereas Throughput refers to the number of requests that can be processed within a given time interval.

For a high-performant system, it's essential to have minimum latency with maximum throughput. If you are designing a Shopping website which takes a long time to load, then it results in poor user experience. Further, on a black Friday sale day, your system's performance mustn't degrade due to increased load.

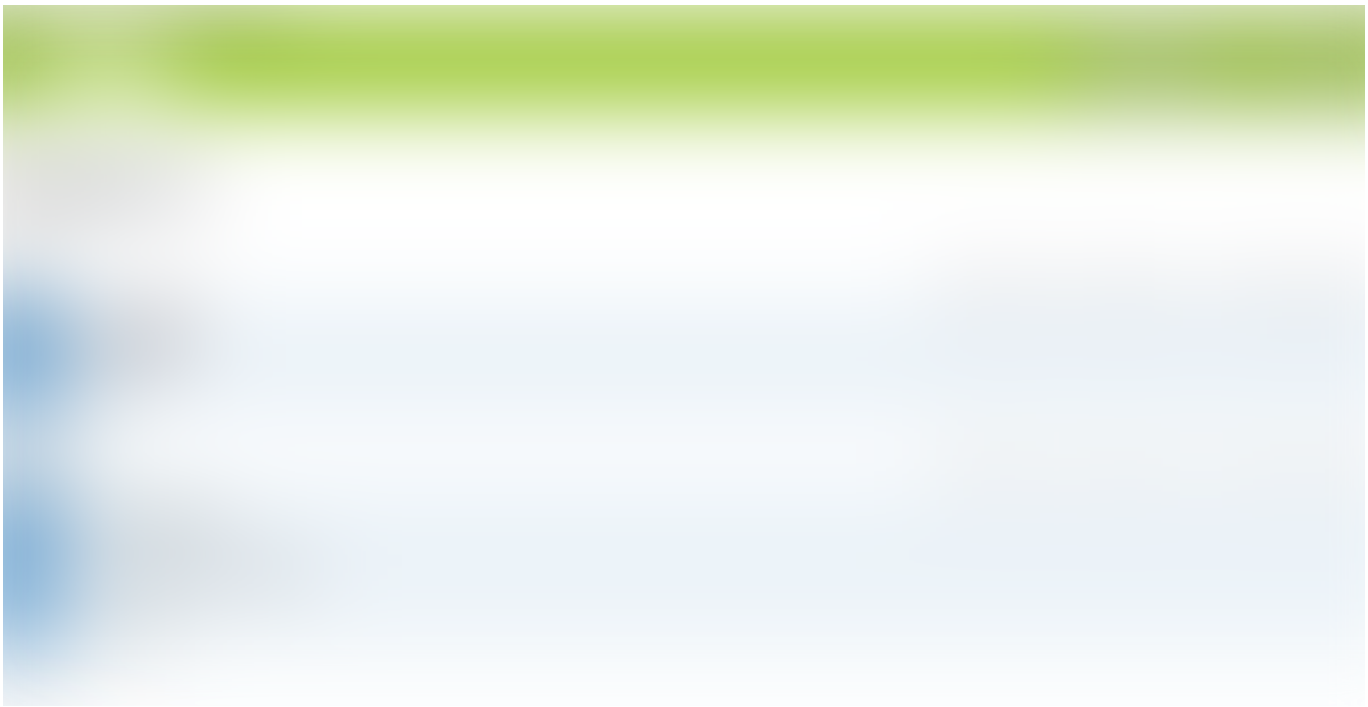
The second design goal is to choose between Availability & Consistency. As per the CAP theorem, you can either have a CP or an AP system to tolerate failures in networks. The choice between Availability & Consistency is influenced by what type of system is being designed.

CAP theorem

If you have a Banking system which directly deals with user's money, Consistency becomes of prime importance. Whereas, if you are designing a Facebook feed, it's okay to miss out a status update from a page or a friend.

API Design & Algorithms

For each of the requirements that are in the scope of the problem, it's necessary to sketch the basic APIs that the backend system would be exposing. A candidate can mention the requests, responses, status codes, HTTP Method to use, etc. Depending upon the problem, once can discuss whether to use *REST*, *GraphQL* or *GRPC* for API design.



API Design

You can further move to draw the high-level components that will be used in the system. It's good to start with client-facing components i.e API gateway, Load Balancers. You can talk about what kind of strategies can be used in Load Balancers to distribute the load evenly among the Application servers.

API Gateway & Load Balancer

If you are asked to design a system like Youtube, Netflix, or Instagram, you can also talk about CDN (Content Delivery Network) which facilitate the serving of static image or video files.

For every API, you'll need to see if an algorithm needs to be implemented on the Server side or whether it's a plain CRUD API. For instance:- While designing twitter, to get news-feed you'll need to come up with a news feed algorithm. In case a user follows another user, it will be a plain CRUD API.

Database, Cache Design & Sharding

The capacity estimation will give you an idea of how much data your system needs to handle. Moreover, you should have a clear idea of the entities involved in data modelling & relationship between them. You need to figure out whether your data is structured or unstructured and whether it needs schema flexibility or not. The answers to the above question will influence your choice of using either a SQL or a NoSQL database.

Cassandra / MySQL

Most of the systems are read-heavy systems. Hence, you can speed up all your database read queries by creating an index on a column or a set of columns depending upon the access patterns. You can have a master-slave architecture where writes are performed on the master while reads are done from the slave. This will isolate reads from the writes and thereby improve the performance of the system.

Reads from a database result in IO operation which is computationally expensive to perform. You can add a caching layer between the application and the database to minimize the disk IO. This will further boost the performance of the system.

While adding a Cache, you will need to choose between the Cache eviction strategies and also on the Caching patterns such as Write through, Write back, Cache aside, etc

Redis

If you are designing a system for millions of users, data will not fit on a single database server. Hence, it's important to distribute the data among multiple database servers. This is known as data partitioning. The Interviewer might question you about different data partitioning strategies that you can use to distribute the data and discuss the trade-offs of using each.

. . .

Building expertise in system design takes a lot of time, experience and effort. You can refer to interesting Tech Blogs where Companies have nicely articulated the manner in which they solved challenging problems.

Additionally, you can try solving as many design problems as you can. Forums like LeetCode, & Carrer cup will help you connect with like-minded Engineers & you might even come across a different thought process.

To summarize, with a structured approach and clarity of fundamentals of distributed systems & enough practice, one can easily ace the System Design Interview at any Tech Company.

References

1. System Design Primer
2. Educative.io — Grokking the System Design Interview
3. Interview bit — System Design Course
4. Swagger Image
5. API gateway & Load Balancers