8 Things You Need to Know Before a System Design Interview

What is the most frequently asked question regarding interview preparation to us? I can tell immediately: how to prepare system design interview?

Many people are afraid of system design interview as there's no certain pattern to prepare and the question is quite flexible and unpredictable. What's more, system design questions are usually open-ended so that there's no standard or correct answer, which makes the preparation process even harder.

We've spent the **past whole month** for this guide to tell you things you'd better know before your system design interview, and at the same time let you be more carefree as system design interview is not as hard as many people thought, certain ways can definitely help you be good at it.

1. What are evaluated in a system design interview?

We all know that a coding interview is focused on those basic knowledge of a candidate, so his general technical skills, analysis ability is tested.

However, few people can tell clearly the purpose of conducting system design interviews. So before jumping into tips, it's better to understand system design interview from interviewer's perspective.

In a system design interview, the candidate is often asked to design a new system in order to solve an open-ended problem like designing the URL shortening service. Sometimes the problem can be quite general like how do you design the recommended system for Youtube.

During this process, discussion is the core. The candidate is more likely to lead the conversation and discussion high-level components, details, pros and cons, and everything with the interviewer.

Compared to coding interview, system design interview is much more similar to software engineer's daily work.

During the interview session, your communication and problem-solving ability are mainly evaluated. Given an open-ended problem, how do you analyze the issue, how do you solve it step by step, how do you explain your idea and discuss with others, how to you evaluate your system and optimize it are what interviewers mostly care.

So let's see what you can do to prepare for it.

2. Get your hands dirty

The best way to prepare system design interview is always thru real projects and practices. Many people start their preparation process quite early like 6 months or 1 year in advance, then this is definitely the best practice for you.

A common pattern we saw is that the more practical experiences you have, the better you are at system design interview. It's quite easy to understand because those system design questions are all from real life product and people who have worked on many projects before tend to have a better sense on these problems or it's just one of the problem they have solved before.

When asked to design Youtube recommendation system, it's similar to many other recommendation systems say Amazon's system since a lot of concepts are common here.

If you have some experience with recommendation, or you've read some articles/books or have thought about it, you must be able to come up with some initial ideas at least.

If you don't know what to work on, here're some suggestions for you:

- Build a small service/product to solve a real problem you have
- Contribute to open source projects at Github
- Find a topic that interests you like machine learning, network etc. and search for some projects you can work on

What really matters is getting your hands dirty to work on some real life projects. It could take a long while before you can see your improvement, but at that point, you will notice how straightforward those interview questions are. Also, you will benefit a lot from this in the long run.

3. Be familiar with basic knowledge

I can’t recall how many times I’ve emphasized this point, but it’s really important for system design interview. As system design questions are open-ended and may cover many technical fields, the basic knowledge here is much more than data structure and algorithm.

First of all, there’s no doubt you should be very good at data structure and algorithm. Take the URL shortening service as an example, you won’t be able to come up with a good solution if you are not clear about hash, time/space complexity analysis.

Quite often, there’s a trade-off between time and memory efficiency and you must be very proficient in the big-O analysis in order to figure everything out.

There are also several other things you'd better be familiar although it's possible that they may not be covered in your interview.

- Abstraction. It's a very important topic for system design interview. You should be clear about how to abstract a system, what is visible and invisible from other components, and what is the logic behind it. Object oriented programming is also important to know.
- Database. You should be clear about those basic concepts like relational database. Knowing about No-SQL might be a plus depends on your level (new grads or experienced engineers).
- Network. You should be able to explain clearly what happened when you type "gainlo.co" in your browser, things like DNS lookup, HTTP request should be clear.
- Concurrency. It will be great if you can recognize concurrency issue in a system and tell the interviewer how to solve it. Sometimes this topic can be very hard, but knowing about basic concepts like race condition, dead lock is the bottom line.
- Operating system. Sometimes your discussion with the interviewer can go very deeply and at this point it's better to know how OS works in the low level.
- Machine learning (optional). You don't need to be an expert, but again some basic concepts like feature selection, how ML algorithm works in general are better to be familiar with.

Remember, the point is here asking you to learn all these stuff from scratch, which may take you more than a year. What really matters is the basic concepts behind each topic. For instance, it's totally okay if you can't implement neural network in the interview, but you should be able to explain it within a sentence.

4. Top-down + modularization

This is the general strategy for solving a system design problem and ways to explain to the interviewer. The worst case is always jumping into details immediately, which can only make things in a mess.

Instead, it's always good to start with high-level ideas and then figure out details step by step, so this should be a top-down approach. Why? Because many system design questions are very general and there's no way to solve it without a big picture.

Let's use Youtube recommendation system as an example. I might first divide this into front-end and backend (the interviewer may only ask for backend or a specific part, but I'll cover the whole system to give you an idea). For backend, the flow can be 3 steps: collect user data (like videos he watched, location, preferences etc.), offline

pipeline that generating the recommendation, and store and serve the data to front-end. And then, we can jump into each detailed components.

For user data, we can list features that we think are relevant to videos a user may like. For pipeline, we can discuss how to train the dataset etc.. We can go even deeper. Since Youtube has a huge dataset, the offline pipeline must run over a huge number of data, then MapReduce or Hadoop might be used.

We can continue this analysis infinitely by going deeper and deeper, but the idea I want to explain here is that you should always have a big picture.

Another tip here is modularization. Like I illustrated above, it's better to divide a system into different components, which is also a good practice in real life projects. Modularization not only can make your design much clearer to both yourself and the interviewer, but also make testing much easier.

5. Pros & cons

System design questions are often given without much restriction. As a result, there's no clear cut between good solutions and bad solutions. In this case, you are responsible to understand what is the best approach in different scenarios.

The most common trade off is between time and memory. You can directly tell the interviewer about the pros and cons for each solution and ask him to clarify the constraints like how much memory you have.

Also when asked to optimize the system, you can also put several common constraints there, for example, if you are designing something for driver's license, you can tell the interviewer that it's reasonable to assume the max

length of a license is maybe 7, and in this way you might be able to store all license in memory, based on which you can further optimize your system.

6. Estimation

You'd better have a good sense of numbers when doing estimation, which is even more important in real projects. More specifically, you should have a clear estimation of how much memory your system or program would cause, how fast it runs, and based on your estimation, how would you adjust your design.

If we use the driver license example, of course you can say let's assume the memory is enough to store all license in US. But it'll be more impressive that you first estimate how much memory you need to store them.

To estimate the memory cost, you should count how many licenses are there if the max length is 7, and what data structure you're gonna use to store and then figure out how much memory you need, which will give you clear idea whether this approach is feasible.

Also when deciding storage, memory of course is not the only solution. Beside storing everything in memory, you can store in disk, or store in multiple computers as well. Selecting the best approach is really a matter of estimating time and storage cost. Figuring out the bottle neck of the execution time and memory limit will give you a much clearer picture of the whole system.

7. Mock interview

Quite honestly, it's not very easy to practice system design interview by yourself since there's no standard answer for it. So the suggestion is always doing this in front of some experienced engineers.

Also thru this process, you'll spend majority of your time communicating and discussing with the interviewer, which is what system design interview mostly about. So in short, we strongly encourage you to practice system design interview with others instead of by yourself.

Websites like Gainlo allow you to have mock interviews with employees from Google, Amazon etc., which can be really helpful.

8. Learn from existing system

This approach is what I usually suggest people to do. Whenever you are curious about some system, try to figure out how this system was designed. You can try to design by yourself first and then compare with how it is actually designed. Most importantly, try to understand why it's designed in this way. It's very likely that certain constraints that forced the system to be like this, like data size, speed requirement etc..

http://highscalability.com has a lot of good articles about how real world systems are designed. It might be a little overkilled for system design interview, but it's always good to know about them.

Also you will notice that even for the same kind of system, different company may have totally different ways of designs. You'll definitely learn a lot from exploring this.

Conclusion

Having system design interview is a lot of fun since it's much closer to real world products. The key to preparation is being clear about what's expected in the interview and spend enough time and effort on things that really matter.

If it's scary to you that there's no standard answer to system design question, you may also take it as your answer is always correct.

I hope this post will make you less anxious about system design interview and let me know what you think about it.