Technical Interviews Technical Interview Questions +3

How do I prepare to answer design questions in a technical interview?

Answer Interesting · 1.3k Request

Question merged

You were redirected because the question How should I prepare system design questions for Google/Facebook Interview? was merged with this question.

Answer Wiki

Here are a few useful links referred in the answers:

1. Grokking the System Design Interview
2. HiredInTech's Training Camp for Coding Interviews
3. Refdash - Real time interview & feedback from senior engineers
4. http://highscalability.com/
5. http://www.gainlo.co
6. Problem Solving in Data Structures & Algorithms Using Java: The Ultimate Guide to Programming Interview
7. Video Session: System design questions and talk about how to approach solving them
8. Structured, Rigorous bootcamp: Large Scale Systems Design Interview Preparation Bootcamp
9. Introduction to Distributed Systems Bootcamp
10. Mock System Design Interview
11. GeekyPrep.com. Join us, Prepare for Interviews, Get Hired!!
12. Interviewbit.com

33 Answers

Anton Dimitrov, Founder & Interviewer at HiredInTech.com
Updated Jan 29, 2015

Originally Answered: How should I prepare system design questions for Google/Facebook Interview?

In my view there are two components to solving system design questions at interviews:

1) Having enough knowledge and skills to come up with good solutions
2) Using the right process to approach the questions

Probably 1) doesn't need any justification and below I will list a few resources that could be a good start

Related Questions

What is the best interview question you have ever used or heard?

What are the best interview questions that are tricky and require presence of mind to answer them?

What interview questions does Google ask their user experience designer candidates?

What is a good approach for product design interview questions?

What is the toughest question ever asked in any interview?

How did you prepare for your software engineering technical interviews? For all past interns and current software engineers out there, when yo...

What are some typical design interview questions for software engineers?

What are some of the best answers to the question "How would you design Twitter" in a system design interview?

What are some good blogs about algorithms and technical interviews?

How can one be well prepared to answer data structure/algorithm questions in interviews?

+ Ask New Question

More Related Questions

Question Stats

1,315 Publicly Interested
420,592 Views
Last Asked Mar 22
9 Merged Questions
Edits

engine", or "Create this hierarchy of classes, which does X".

Obviously, to solve such a question in full, companies need weeks to years and big teams of engineers. So, how on Earth should one person answer the question in 30 minutes.

Usually, such a discussion needs to start at the higher level where the interviewee presents their idea of how the architecture would be structured. Only after that, if the interviewer is ok with the higher level design one could go into more details about how they would tackle specific bottlenecks and other limitations. Most often, the interviewer would lead the discussion in one or another direction based on their goals.

In short, it's important to practice doing that for enough system design problems. Just take a few from the internet and give it a try. Here are a few examples:

1) Design a URL shortening service.
2) How would you design the feature in LinkedIn where it computes how many hops there are between you and another person?
3) If you were to design a web platform for online chess games, how would you do that?

As for some good resources to start with, I would recommend this video lecture from prof. David Malan from Harvard, which covers many useful aspects of designing scalable systems:

I also find this 4-part article very interesting: Scalability for Dummies - Part 1: Clones .

Finally, the HighScalability blog has some nice examples of real-life architectures: http://highscalability.com/

There is a lot to be said about this topic and it's a very exciting matter. We've tried to assemble a chapter on system design questions at tech interviews in

I'll be happy to hear your comments on whether it's useful.

Good luck!

316.9k Views · View Upvoters

Frank Kelly, Software Engineer who has endured many inane interview experiences

Updated Feb 16, 2014

Originally Answered: How should I prepare system design questions for Google/Facebook Interview?

Nothing beats experience but . . . .

here are some key principles to remember

1) Asynchronous is good (use Queues, Topics etc.)

2) Parallel is good (Multi-threading, load balancing etc.)

3) Avoid points of contention e.g. synchronization

4) Avoid writing to disk until you must - cache like crazy

5) Scale out not up

6) At web scale the speed of light is a problem

7) At web scale everything fails - networks, load balancers etc.

And here are some good articles to read and presentations to watch

http://s3.amazonaws.com/AllThing...

High Scalability - High Scalability - Amazon Architecture

http://www.addsimplicity.com/dow...

Scalability Best Practices: Lessons from eBay

http://www.hpts.ws/papers/2009/session9/shoup.pdf

Shopzilla - Performance By Design

http://static.googleusercontent....

facebook architecture for 600M users

:ructure at Facebook

Scaling Memcache at Facebook

Scuba: Diving into Data at Facebook

Scalability, Availability & Stability Patterns
LinkedIn - A Professional Network built with Java Technologies and ...
LinkedIn Data Infrastructure (QCon London 2012)
Global Netflix Platform
Timelines at Scale
Big Data in Real-Time at Twitter

Hope that helps!

137.8k Views · View Upvoters

Your response is private.

Is this answer still relevant and up to date? Yes No

Add a comment... Recommended All

Nishant Rai, Software Engineer at Rubrik
Updated Sep 8, 2017

System design questions have been gaining lots of traction recently.

Many companies have rounds dedicated to it in their interview process (My **Microsoft** and **Uber** interviews involved such questions). I personally love such questions as I think they help make a much better judgement of the candidate.

Upvote · 401 Downvote

To answer your question, let us focus on the **Why**, **What** and **How** of system design interviews.

compared to algorithmic rounds. Note that a design round is a very open ended discussion and companies generally focus on the following,

- **Systems:** Check the candidate's knowledge about systems
- **Design:** Check the quality of the design proposed by the candidate and also their abilities to identify potential issues in their solution.
- **Communication:** Checking how well the candidate can convey their ideas

An important thing is that the ultimate goal of such interviews is not the answer. The main focus lies in the journey.

What should one expect in a system design interview?

Such an interview generally kicks off with a very open ended problem such as design a Quora-like system, design your own social network (like Facebook). As the discussion proceeds, the interviewer will add extra constraints or assumptions to narrow the scope of the problem.

How should one prepare for a system design interview?

Based on my interview experiences, I can conclude the following facts regarding the process of tackling such questions.

- **Practice helps:** I know, I know.. I can make this statement anywhere and at anytime in the world and it will still be true. But this fits perfectly in the case of system design interviews. The knowledge you gain from practicing even five design problems exponentially improves your chances of acing your interviews.
- **Ace it using a bag of tricks:** It's unfair to reduce the design interviews to just a bag of tricks, but I must. There are a popular set of ideas you can use to construct the solution to almost any design problem asked in a (realistic) interview.
- **Have a structured approach:** A round in a system design interview consists of only one problem, which means it's a really intensive one (mostly since it's so open-ended). Starting to approach the problems in a chaotic way is one of the best ways to fail. Multiple resources point to a systematic way to tackle such problems (And it actually works!).

Let's look at the above three points in detail.

Practice Helps:

The best way to practice would be directly jumping to the problem solving stage and learning on the fly. There are many resources you can use as reference. A few which I used during my interview preparation are given below,

- Free Resources: *HiredIn* , *Gainlo*

Bag of Tricks:

These are a few popular tricks I've seen being used in multiple design problems (and personally use them in interviews as well). You might be able to spot a few more if you go through the resources provided above.

- **Usage of a cluster:** It's safe to assume that we have a cluster of nodes to use (Since it's consistent with most real world systems). It also opens possibilities which makes things faster and more efficient.
- **Avoid Sync Issues:** Avoid introducing synchronization issues in your system. For example,
  - We need to maintain the same value (which can be changed in between) on multiple nodes of the cluster
- **Multi-Threading:** Try to use it whenever possible, makes things much quicker
- **Shard data:** One of my favorite tricks! I've used it in almost every system design interview I've faced. Possibilities include sharding based on: Order of appearance, Some hash value, Arbitrarily, etc.
- **Load Balancing**
- **Account for system failures:** Discussing the pros-cons of your system in such cases is extremely important
- **Cache:** Use caching for faster query response time

Structured Approach:

Here's the plan I used to follow during my interviews. There are many resources which provide detailed steps for the same.

- **Understand the system requirements:** As mentioned earlier, the original problem is extremely open ended. Getting a clear idea of the requirements is critical to developing a good design. Do not make any assumptions; Ask questions till you're satisfied; Understand the constraints involved.
- **Visualize an abstract design:** This should be based on the constraints clarified in the previous step. This gives you a chance to consolidate all the information and reaffirm your design choice. Be sure to focus on the basic components of the system and the relationships between them.
- **Bottlenecks and Challenges:** Think about the bottlenecks these components can face with increased load.
- **Use the bag of tricks:** Once you've settled on the components and challenges involved, start carving the final design using the tricks discussed earlier.

I hope this was helpful! Feel free to express your views in the comments.

Helpful Links:

- CareerCup: System Design Questions
- InterviewBit: System Design

8k Views · View Upvoters

Upvote · 42 Downvote

Add a comment... Recommended All

Mohan Gupta, Amazon, Microsoft, Flipkart, Cisco
Answered Nov 8

I am assuming you are talking about design rounds at big tech companies like Facebook, Google etc.

Unlike machine-codding or problem-solving, design rounds are more abstract and very difficult to learn just by reading. Design is more of an art than science just like painting or music.

A bit about system design:

A great system design is built by years of experience designing high scale systems and when you set out to solve a design problem you just know it in your gut why something feels wrong or right, so that you can double down on it and see the 'why' of it.

One needs to understand what works in real life use cases and what problems generally occur and to design to mitigate those. From experience, one knows that software systems change very often and your design should be flexible enough to easily accommodate new use cases.

Another aspect of the modern design is to scale horizontally where you can add new servers and be able to server increasing number of users. Also, fault tolerance, in real-world servers go down, disks crash, the network is unreliable, a good system is designed ground-up to handle such scenarios.

Preparing for a design round:

There is no substitute for actual experience but when it comes to interviews, there is a pattern to it and it's predictable. If you know what/how you are being evaluated, there are ways to master it. **Interviews are more about meeting someone's expectation and the way you answer a question rather than getting the right answer.**

In fact for design I guess there is no such thing as a 'perfect answer'.

Assuming you do not have experience building web-scale systems. The next best alternatives are to learn about the most common elements used currently and to understand 'why' and 'how' are they used. Almost all large-scale systems have same things used over and over again and there is a genuine reason behind that. Just to give you an idea, here are a few design components and a brief on 'why' and 'how' are they used:

1. **Caching**: I cannot think of any large service which doesn't use many layers of

it from the hard disk and in most cases, it is 'OK' to serve a response which is a few minutes stale.

Looking at search results, they don't change every minute, Google will cache the results for a query for few minutes and server that over and over again. Actually searching from its billions of pages of index might take 10s of seconds and a lot of CPU power but getting from cache takes just 2-5 milliseconds and almost no CPU.

A system that can support N users can scale to 10-100 times just by putting a cache in front of it.

From the chips on the CPU of the server to your browser's cache, there might be 10s of layers of cache between the service and what you see on your browser client.

2. **Load balancers:** How do you handle a million users at a time? How about a billion? One server (a computer) can handle only so many requests at a time.

How about we use 100s of servers to do provide the same service? How do you make sure that each server is getting equally utilised and no server is being bombarded with requests or what to do if one of these 100s of server is down? All these things are handled by special software/hardware called Load-balancers. They are the point of contact where the initial request comes and they then intelligently route requests to these 100s of servers, making sure to equally distribute load and taking care of removing dead servers automatically.

3. **Micro-services:** How do you design Netflix? A service which shows you millions of videos, predicts your tastes, automatically detects your device, optimizes the image shown for a video, debits your subscription fee etc. If you write one big monsterware to do all these things, every new feature release will take years and any minor change will cause riots between teams at Netflix. **As services get bigger, with every new feature the software gets complicated and new changes slower and slower**. Here come micro-services. A micro-service is a fully operational service which does just a small piece of work. 100s of micro-services talk to each other over well-defined APIs to bring alive the Netflix experience. If you need a new feature, just create a new micro-service and plug it into Netflix stack.

Any change can be isolated to a small code base so that any bug/issue is just limited to that service. If a micro-service fails, rest of the system can still function and a majority of the users won't even notice.

There are 10s of more such component/ideas used for a good design. Here are some of the resources you can refer to:

1. High Scalability - : Here you read about the design and architecture of all the major companies in the world. Though you may not find a detailed explanation of 'why' things are done the way they are but will get a sneak peek into what components/ideas are used.

3. Course on cracking the design interview: In-depth course to understand what the interviewer is looking for and how to structure your design interview answers. Can see live how to perfectly answer a design question and also provides you real interview-setting mock design interviews to get you ready for the D-day.

Hope this helps.

Samyak Datta, Interviewed with Google, Directi and Facebook
Answered Oct 23, 2014

Originally Answered: How do I answer system design questions in a technical interview?

I faced system design questions while interviewing for both Google and Directi, co-incidentally during my final round of interview for both companies. Personally, I feel that these questions make up for an excellent measure to gauge a candidate's knowledge and software engineering skills. You should treat these problems as discussions and not as your conventional interview questions. In contrast to algorithmic problems which have a consistent and clear approach to a solution (or solutions), system design questions tend to be very open-ended. The interviewer may often consciously make room for ambiguities as a test of your design skills. From what I've learnt during my interview process, there are some points which you should keep in mind, while tackling system design problems.

Keep it simple, stupid (KISS)!

At the very outset, you should ensure that all the system requirements, design goals, optimization objectives are crystal clear before you proceed to brainstorm on possible solutions. As an example, suppose you are asked to design a minimalist text editor using appropriate data structures. A bad approach would be to straight-away start with advanced concepts such as tries. Agreed, tries are optimized for fast insertion and search, but you need to ask yourself, "Is that really required for a minimalist editor". It is always a good idea to note down all the data structures you know and write the pros and cons of using each one of them in the problem domain at hand. This not only clears your thought process, but also ensures that you do not over-complicate things. You will be surprised to know how many seemingly complicated systems can be implemented using data structures which are as simple and elegant as linked lists and stacks!

Divide and Conquer

The problem statements might seem overwhelming at first, but there is

that they can be easily decomposed into such fragments and solving the individual sub-problems should be a matter of reducing them to a known algorithm. For example, if I am supposed to design a technique to compress log records (lossless compression), my approach would be to first decide on what kind of fields a log record might contain and then think about efficient ways of compressing each one of them. Date and time can be stored as seconds elapsed since the beginning of the year (assuming we are interested in logs that are not more than a year old, this should fit in a 32 bit integer), verbose error descriptions can be replaced by short error codes which are mapped to their respective descriptions on a separate hash map, and so on!

Think out loud

Another very important strategy is to always keep communicating with your interviewer. As stated earlier, system design questions do not have a 0-1 answer. Whatever solution you proffer, there is always scope for optimizations. In such a scenario, it is imperative to let your interviewer have a glimpse of your thought process. Though this advice is true in general for the entire spectrum of question formats, it holds special significance here. Following this has the added benefit of the fact that the interviewer would be able to correct you if you digress from the intended path and he may have some ideas that you can further build upon in your solution.

Knowledge is power

Since, the systems that you would be asked to design are most likely going to be a simplified prototype of a commercial system, you might end up having to use some Machine Learning or Data Science in your solution. So, it's a good idea to brush up on some common M.L. and N.L.P. concepts before your interview! If you are asked to implement the "Did you mean" feature in Google search, in addition to everything else, it wouldn't hurt you to know about how spelling correctors work (N.L.P.)!

Sachi Bafna, B. Des Graphic Design, MAEER'S MIT Institute of Design, Pune, Maharastra (2020)

Answered Oct 7, 2017

The first reaction of most people to design based questions in a technical interview is fear. I would attribute this to the fact there is no clear-cut way to prepare and the question is quite flexible and unpredictable. The questions are open-ended, making the preparation even harder. A coding interview is focused on your basic knowledge and analytic skills.

What the interviewer wants from you:

interviewer. During the interview session, your communication and problem-solving ability are mainly evaluated.

2. **Clarity in basic concepts** like abstraction, database, network, concurrency, operating systems, and machine learning.
3. **How you communicate your design idea.** They want to know how you analyze the issue, how you solve it step by step, and how you explain your idea and discuss with others.

How to prepare

1. First of all, there's no doubt you should be very good at data structure and algorithm. Make sure you're up to snuff on your skills. Practice and do mocks.
2. Have mock sessions. Don't think of it as an interview—just try to come up with the best solution you can. Design interviews are similar to actual design sessions, so getting better at one will make you better at the other.
3. Learn how databases and operating systems work under the hood. These technologies are not only tools in your belt, but also a great source of design inspiration. If you can think like a DB or an OS and understand how each solves the problems it was designed to solve, you'll be able to apply that mindset to other systems.
4. Practice white board coding. You'll be expected to work out your solution on a board. So don't let lack of practice throw you off your game.
5. Come up with your own standardized framework on how to answer these questions. Once you have your process in sections, you'll find it much easier to tackle this answer.
6. Work on your speed. These problems are wide and bottomless, and the point of the interview is to see how much volume you can cover in a given period of time.

Finally - just remember to remain calm and have your ideas and processes organized. Don't be a scatter-brain! Make sure you understand what they're asking off you and then take a structured approach to communicating your ideas.

You can also help other candidates crack their interview and land a job, by submitting your Interview experience on this page: Share interview questions & advice | AmbitionBox

If you are an experienced professional, please feel free to submit your company review on this page: Write a company review | AmbitionBox

To access free interview preparation resources, visit Ambitionbox Blog - A one stop solution for all your needs related to job interview preparation.

Fahim ul Haq, former Software Engineer at Facebook
Answered Oct 26, 2017

It's a multi-step process and your path to preparation depends on the following two factors

1. Your knowledge of basic distributed system theory e.g. Consistency, CAP theorem, Storage and Redundancy, Map Reduce, Caching etc.
2. Your experience in working on a distributed system.

You don't have to read complete papers but you should be able to have a reasonable idea about what are the major components in your system and how all of the individual subsystems (e.g. your load balancer, web front-ends, task queues, storage servers) work together to efficiently serve hundreds of millions of users.

If you are looking for a resource to prepare for System Design Interviews, take a look at Grokking the System Design Interview .

As an example I've noticed that during interviews, candidates struggle to describe when to use SQL vs. NoSQL vs. HDFS etc. Most likely, a large system will include all three types of storage (and even more).

Here are some sample questions and solutions that will help you get an idea about the type of questions that are typically asked during these interviews

1. Designing a URL Shortening service like TinyURL
2. Designing Instagram
3. Designing Facebook's Newsfeed
4. Designing Youtube or Netflix
5. Designing Uber backend

Top Stories from Your Feed