master 1 branch 0 tags Go to file Code

Jeevan-kumar-...	dcafc27 on Aug 8, 2020	25 commits
basics	Add files via upload	15 months ago
bin	Add files via upload	15 months ago
designs	Add files via upload	15 months ago
img	Add files via upload	15 months ago
LICENSE	Initial commit	15 months ago
README.md	Update README.md	15 months ago
SUMMARY....	Add files via upload	15 months ago
book.json	Add files via upload	15 months ago

Grokking System Design Interview

Source: educative

Interview Process

- Scope the problem
  - Don't make assumptions.
  - Ask clarifying questions to understand the

About

Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

Readme

GPL-3.0 License

Releases

No releases published

Packages

No packages published

Languages

constraints and use cases.
    - Steps
        - Requirements clarifications
        - System interface definition
- Sketch up an abstract design
    - Building blocks of the system
    - Relationships between them
    - Steps
        - Back-of-the-envelope estimation
        - Defining data model
        - High-level design
- Identify and address the bottlenecks
    - Use the fundamental principles of scalable system design
    - Steps
        - Detailed design
        - Identifying and resolving bottlenecks

Distributed System Design Basics

- Key Characterics
- Load balancing
- Caching
- Sharding
- Indexes
- Proxies
- Queues
- Redundancy
- SQL vs. NoSQL
- CAP Theorem
- Consistent Hashing
- Client Server Communication

System Designs

- Short URL Service
- Pastebin
- Instagram
- Dropbox
- Twitter
- Youtube
- Twitter Search
- Web Crawler
- Facebook Newsfeed
- Yelp
- Uber Backend
- Ticketmaster

System Design Interviews: A step by step guide

SYSTEM DESIGN PREPARATION

- How to prepare for and answer system design questions

Objective

Learning about and implementing large-scale distributed system is not easy. I do not want to give the impression that it's something that can be learnt in a month. What this repository aims to achieve, is for software engineers and students to get a rough idea of how the thought process of designing a large scale works and how big companies have managed to solve really hard problems. Along with that, there is a recent trend for companies to

README.md

levels if they haven't gotten the opportunity to work on such systems themselves.

This is a collection of links/documents for the following use cases: a) Prepare for a system design or open-ended rounds. b) Learn more about how large-scale systems work and thought process of designing a new system.

Index

- [ ] Starting point
- [ ] basics
- [ ] How to answer in interviews
- [ ] Steps how I approach the system design questions in interviews
- [ ] Common Design questions
- [ ] architecture
- [ ] company engineering blog links
- [ ] Low on time ?

Starting point

For a very broad overview please go through these lectures, really useful:

- Gaurav Sen's system design series Starts from simple stuff like load balancing and message queues, then moves to building full systems like Whatsapp and Tinder.
- david malans cs75 scalability talk Feel free to go through other lectures if needed.
- david huffman's talk , scaling up talk (Youtube link)
- scalability for dummies
- Designing data intensive appliations This is by far one of the best books about large-scale systems and the practical challenges encountered during building them. It's focussed more on data-oriented applications though.

These talks should give you a starting point on how to think about such problems.

Basics

But before you begin, here are some topics(in no particular order) which in my opinion you should have a decent idea of before proceeding.

1. Operating system basics: how a file system, virtual memory, paging, instruction execution cycle etc work (For starters silbershatz should be enough. If you already have decent knowledge try stallings book on OS)
2. Networking basics: Should know the TCP/IP stack, basics of how Internet, HTTP, TCP/IP work at the minimum. cs75 on youtube (1st lecture) should give a broad overview. I personally love networking-a top down approach.
3. Concurrency basics: threads, processes, threading in the language you know. Locks , mutex etc.

4. DB basics: types of DB's (SQL vs noSQL etc), hashing and indexing, EAV based databases, Sharding, caching for databases, master-slave etc
5. A basic idea of how a basic web architecture is: say load balancers, proxy, servers, Database servers, caching servers, precompute, logging big data etc. Just know broadly what is each layer for.
6. very basic summary of what the CAP theorem is (Have never been asked about the theorem itself, but knowing it will help you in designing large-scale systems.

How to answer in interviews

- I found hiredintech videos an excellent place to start with. The way how to approach a design question as given in the link is really useful. It goes into how we start with clearing the use-cases of the system, then thinking in the abstract manner of the various component and the interactions. Think about the bottlenecks of the system and what is more critical for your system (eg latency vs reliability vs uptime etc) Address those giving the tradeoff of your approach.

- system design in crack the coding interview: good approach on how to begin attacking a problem by first solving for a small usecase then expanding the system.

- The best way to prepare for such questions is do mock interviews, pick any topic (given below) try to come up with a design and then go and see how and why it is designed in that manner. There is absolutely no alternative to practice!! Whiteboarding a system design question is similar to actually writing code and testing it! Just reading will only take you so far.

Steps how I approach the system design questions in interviews

These are the steps I go through mentally in the interviews, followed by actual interview experiences:

- a) **Be absolutely sure you understand the problem being asked**, clarify on the onset rather than assuming anything
- b) **Use-cases**. This is critical, you MUST know what is the system going to be used for, what is the scale it is going to be used for. Also, constraints like requests per second, requests types, data written per second, data read per second.
- c) Solve the problem for a **very small set**, say, 100 users. This will broadly help you figure out the data structures, components, abstract design of the overall model.
- d) Write down the various components figured out so far and how will they interact with each other.
- e) As a rule of thumb remember at least these :
- i. processing and servers
- ii. storage
- iii. caching
- iv. concurrency and communication
- v. security
- vi. load balancing and proxy
- vii. CDN
- viii. Monetization: if relevant, how will you monetize? eg. What kind of DB (Is Postgres enough, if not why?), do you need caching and how much, is security a prime concern?
- f) **Special cases** for the question asked. Say designing a system for storing thumbnails, will a file system be enough? What if you have to scale for

facebook or google? Will a nosql based database work?

- g) After I have my components in place, what I generally try to do is look for minor optimization in various places according to the use-cases, various tradeoffs that will help in better scaling in 99% cases.
- h) [Scaling out or up] (http://highscalability.com/blog/2014/5/12/4-architecture-issues-when-scaling-web-applications-bottlene.html)
- i) Check with the interviewer is there any other special case he is looking to solve? Also, it really helps if you know about the company you are interviewing with, what its architecture is, what will the interviewer have more interest in based on the company and what he works on?

Common Design questions

It generally depends what you are and you will be working on. Also what your level is but these are some of the more frequent interview questions.

- Design amazon's frequently viewed product page (eg. which shows the last 5 items you saw)
- Design an online poker game for multiplayer. Solve for persistence, concurrency, scale. Draw the ER diagram for this
- Design a [url compression system] (http://www.hiredintech.com/system-design/the-system-design-process/)
- Search engine (generally asked with people who have some domain knowledge): basic crawling, collection, hashing etc. Depends on your expertise on this topic
- Design dropbox's architecture. good talk on this

- Design a picture sharing website. How will you store thumbnails, photos? Usage of CDNS? caching at various layers etc.
- - Design a news feed (eg. Facebook , Twitter): news feed
- Design a product based on maps, eg hotel / ATM finder given a location.
- Design malloc, free and garbage collection system. What data structures to use? decorator pattern over malloc etc.
- Design a site like junglee.com i.e price comparision, availability on e-commerce websites. When and will you cache, how much to query, how to crawl efficiently over e-commerce sites, sharding of databases, basic database design
- A web application for instant messaging, eg whatsapp, facebook chat. Issues of each, scaling problems, status and availability notification etc.
- Design a system for collaborating over a document simultaneously (eg google docs)
- (very common:) top 'n' or most frequent items of a running stream of data
- Design election commission architecture : Let's say we work with the Election Commission. On Counting day, we want to collate the votes received at the lakhs of voting booths all over the country. Each booth has a voting machine, which, when connected to the network, returns an array of the form {[party_id, num_votes],[party_id_2, num_votes_2],...}. We want to collect these and get the current scores in real time. The report we need continuously is how many seats is each party leading in. Please design a system for this.
- Design a logging system (For web applications, it is common to have a large number of servers running the same application, with a load balancer in front to

distribute the incoming requests. In this scenario, we want to check and alarm in case an exception is thrown in any of the servers. We want a system that checks for the appearance of specific words, "Exception", "Disk Full" etc. in the logs of any of the servers. How would you design this system?)

Architectures :

Personally I looked into the following architectures:

- Basics of google search
- Basics of messaging frameworks like Kafka , queuing architectures like rabbitmq.
- Broad overview and advantages of Redis , mongodb , cassandra.
- Google file system
- [Google architecture] (http://highscalability.com/google-architecture)
- Instagram and other image based social networks
- Memcache scaling by facebook
- Twitter scaling and facebook feeds
- facebook graph api
- facebook haystack needle architecture
- youtube architecture and optimizations for video

Company engineering blog links

courtesy checkcheckzz

Depending on where you are interviewing, go through the company blog. VERY USEFUL IN INTERVIEWS! It really helps if you have an idea of the architecture, as the questions asked will generally be of that domain and your prior knowledge will help out here.

- Airbnb Engineering

- Amazon
- Amazon AWS
- Bandcamp Tech
- BankSimple Simple Blog
- Bitly Engineering Blog
- Cloudera Developer Blog
- Dropbox Tech Blog
- Engineering at Quora
- Etsy Code as Craft
- Facebook Engineering
- Flickr Code
- Foursquare Engineering Blog
- Google Research Blog
- Groupn Engineering Blog
- High Scalability
- Instagram Engineering
- LinkedIn Engineering
- Oyster Tech Blog
- Pinterest Engineering Blog
- Songkick Technology Blog
- SoundCloud Backstage Blog
- Square The Corner
- THE REDDIT BLOG
- The GitHub Blog
- The Netflix Tech Blog
- Twilio Engineering Blog
- Twitter Engineering
- Uber Engineering
- Walmart Labs Tech Blog
- WebEngage Engineering Blog
- Yammer Engineering
- Yelp Engineering Blog

- Smarkets Blog

Low on time ?

I would HIGHLY recommend you do not take a shortcut unless you have a week or so for an interview. System design is best learnt by practising, shortcuts might help you in the short term, but would recommend coming back to this link for an in-depth understanding after the interview

- a) Go through cs76 and Udacity's links given above for scaling systems.
- b) Go through the engineering blog of the company you are interviewing in (or if its a startup go through the link of the company closest to yours)
- c) See this talk: http://www.hiredintech.com/system-design/the-system-design-process/ and develop a process for how to answer such questions.
- d) Remember these terms, just roll over them in your interview in your mind, and if relevant mention it in the interview

1. processing and servers
2. storage
3. caching
4. concurrency and communication
5. security
6. load balancing and proxy
7. CDN
8. Monetization

Best of luck 👍, feel free to send pull requests to add more content to this git!