Problem Solving Workshop #37 Tech Interviews and Competitive Programming Meetup

August 11, 2018 https://www.meetup.com/tech-interviews-and-competitive-programming/

Instructor: Eugene Yarovoi (can be contacted through the group Meetup page above under Organizers)

Design Question Edition

More practice questions: glassdoor.com, careercup.com, geeksforgeeks.org
Have questions you want answered? Contact the instructor, or ask people on Quora. You can post questions and follow the instructor and other people who write about algorithms and design.

Generally speaking, in design questions, there's always lots of ambiguity, so you would start by asking clarifying questions to the interviewer. Since you don't have an interviewer here, assume what you think are reasonable answers to the questions you would ask, or ask the people you're working with to fill in for the role of the interviewer.

"URL Shortener" (Overall Design & Data Flow)

How would you design a URL shortener service? A URL shortener service is a site where I can enter a URL (for example a link to this document on Github) and the site will generate a URL for me like http://myurlshort.com/gh30fnqfd that I can then post somewhere as a shortened URL. When a user goes to that URL, they should be redirected to the URL I originally registered (e.g. this document on Github).

How would you build this service? Consider what data you would need to store and how you would represent and store it, and what enhancements to the service you can make.

"Executor Service" (Feature Brainstorming & Data Flow)

A large company wants to implement an internal "executor service". This will be a service that runs jobs on demand. At the most basic level, the user will upload a "package" that contains an executable plus any data files that need to come with it, and run that executable on a worker machine, reporting back any output. Think about what features this system should provide to provide utility to users and prioritize them, and then discuss how you would build the system. Discuss how the APIs you create will look like, what data will be stored under the hood, and what algorithms your service will use to do its job. Here are some sample potential use cases:

Jake is a data scientist who is running some very intensive computations. He has 1000 1MB data files he wants to process. Processing just one of the files on his local machine using his algorithm takes hours, so he will use this system to speed things up by parallelizing the work.

Lauren is doing machine learning, and her local machine doesn't have a GPU, something that would massively speed up her algorithm. Some worker machines will come with GPUs, so she'll want to use this service to execute her program on one of those machines.

Dave works on a product that deals with maps. Periodically, they recalculate certain metrics about the maps, which is a series of computationally intensive batch-jobs, and so an automated service that detects when the map metrics need to be recalculated will need to kick off some jobs periodically through your service.

"Twitter Trends" (Feature Brainstorming & Algorithmic Design)

Imagine you are Twitter. How would you design a way for people to see what content is currently trending on Twitter? In this hypothetical, assume such a thing does not already exist, but assume that the Twitter service itself already exists and is already being used by milions of people.

The goal of this feature is to inform Twitter users about what's going on in the world and what content has been "hot" lately. You should interpret this mission statement in a fairly broad sense.

As a narrower starting point, consider how you would design a "top 1000 trending hashtags" (hashtags that are used a lot acorss posts). Some points you should be sure to address:

- How do you propose to define “top trending hashtags” so that it makes sense relative to the stated use case?
- How will you know which hashtags are the top trending 100 at any given time?
- When you design this feature, you should assume Twitter is already a large-scale distributed system. How will you add the logic necessary for your “trending hashtags” service to the existing Twitter infrastructure? To answer this, you will need to state some reasonable assumptions for what the Twitter infrastructure looks like, and how you would add your service on top of that.
- How will users access the top trending hashtags?

Then consider how else you can identify and show to users trending content on Twitter.

"Download ALL the things!" (Algorithmic Design)

The interviewer states the problem like this: Consider an existing large website where there are lots of images submitted by users. For example, Wikipedia, Pinterest, DeviantArt. How would you design a web crawler to crawl all pages belonging to that domain, and download all the images?

At this point, you would need to ***ask clarifying questions*** *before you can attempt the problem. Imagine the conversation goes as follows:*

You: How large is the website and how many images do you estimate there are?

Interviewer: The site may have on the order of 1 billion images.
You: Each image may be 100KB-1MB, so we're talking hundreds of terabytes of data. I assume we want to download this data into some kind ***distributed*** database, since 1 machine can't hold this.
Interviewer: Yes. And no requirement to organize the images in any way, just download them all.
You: I will set up some distributed data storage system. Can I assume I have that set up and an API call that adds 1 image to the storage once I have that image downloaded locally on a worker machine?
Interviewer: Yes. Just design the crawler. You should mention what data schema you will use to organize any data your crawler directly stores in a database, though. You don't have to worry about the database's internal implementation details, as this is a system built by someone else.
You: How do we know what pages are available in the domain? Is the website organized so that we will find all the pages by starting from the homepage for the domain and following links?
Interviewer: Yes. Also assume for the time being you don't have to deal with robots.txt or the like.
You: Can I assume a static page structure too, the link structure of the site isn't changing much?
Interviewer: Start with that assumption, you can think how to weaken this assumption later.
You: Can I assume images are referenced by img tags, and they're not embedded in some other medium like Flash?
Interviewer: Yes. The images will be easily accessible like that.
You: Because there's so much data, this process may take a long time.
Interviewer: Yes. Think about your design in the context of this huge scale.
You: And probably the websites will detect you're crawling them and block you.
Interviewer: True. Assume that doesn't happen, or that you can circumvent it. Don't focus your answer too much on this aspect. Let's get to the core logic first.

At this point, there's still plenty of ambiguity, but assume reasonable answers of your choice for your remaining questions. Design the system.

Guidance: For these kinds of problems involving distributed systems, first figure out how you would solve the problem at small scale. What if you have the entire domain downloaded locally on one machine and there's only a few pages? Then think how you will make it efficient enough for millions of pages. Then, try to make more and more subsystems into distributed systems.