Handling Interview Questions

Background

While it’s important to have a bunch of knowledge surrounding systems, any successful interview is going to be predicated not just on possessing the knowledge, but also conveying it in an effective and concise manner. In this final video, let’s discuss how to tackle an actual systems design interview.

Requirements Clarifications

The most important part of any interview is figuring out which part of a service you are actually going to build. No one wants you to build all of Twitter or Facebook, but rather do a high level design of a small chunk.

Try to figure out what the interviewer wants you to focus on early on by asking:

- The scale of the application in number of users, as well as the performance goals
  - If Twitter: are we optimizing for low latency tweets, or instead home page reads?
- Functional requirements
  - If Twitter: can we post tweets, do we have DMs, do we have images?

Capacity Estimations

It is important to get a sense of the scale of our application early on so that we can start to think about whether something like caching or partitioning is necessary.

Ask:

- Ratio of reads to writes?
- Number of users? Average size of each upload image?

From there try to figure out the necessary amount of storage needed, and what the common use cases of most users is - because this is what we want to optimize for.

API Definitions

At this point, start coming up with some rest endpoint definitions that you will have in your service for the client to interact with - you can just list these out as function calls.

```
create_user(email, password_hash)

send_tweet(user_id, text)
```

They do not have to be super detailed, obviously in a real service they will be much more complicated.

Data Tables

Start thinking about schemas of data that you may want to store. This is something that may have to be modified a bit later once the design starts coming together, as it may turn out that the data store that you choose may be insufficient for a problem, and you have to rethink your schema.

Tweets Table:

- User_id: int
- Text: string
- Timestamp: date
- Metadata: json

Get Designing!

For every single piece of the design, make sure to carefully inspect it to see that it is not a lone point of failure - and if so, make modifications to change this. In pretty much any systems design problem, expect to talk about the following:

- Load balancing
- Replication
- Sharding
- Database choice (sometimes multiple is the best answer!)
- Using a microservices design

You Will Get Grilled

Remember, it is an interviewer’s job not just to figure out what choices you would use, but rather why you used them. If you can’t list the tradeoffs of any design choice that you made versus possible alternatives, you seem like you just memorized a solution. You don’t have to have a perfect design, but you need to be able to justify what you came up with as a feasible solution.

Often times, an interviewer comes into a problem wanting to focus on one aspect of it. Try to figure out what that aspect is early on, and then prove to them that you know about it.

Conclusion

Good luck everyone, I’ll start doing some interview problems and we’ll get through it together!