System Design Round Prep

Handling the Questions

- Communicate - Stay engaged with interviewer, ask them questions, be open about issues in system
- Go broad first - Don't dive straight into the algorithm or get excessively focused
- Use the whiteboard - Helps interviewer follow proposed design. Draw a picture.
- Acknowledge interviewer concerns - Don't brush off concerns. Validate and make changes
- Be careful about assumptions - can dramatically change the problem
- State assumptions explicitly - allows interviewer to correct you if you're mistaken
- Estimate when necessary
- Stay in the driver's seat.

Design Step-By-Step

1. Scope the Problem
2. Make reasonable assumptions
3. Draw the major components
4. Identify the key issues
5. Redesign for the key issues.

Algorithms that Scale: Step-By-Step

Step 1: Ask Questions

- Understand problem

Step 2: Make Believe

- How would you solve all on one machine with no memory limitations.

Step 3: Get Real

- How much data can you fit on one machine?
- What problems will occur when you split up the data?

Step 4: Solve Problems:

- Mitigate or remove the issue.
- May require fundamentally altering the approach
- Do new problems emerge?
- Poke holes in your own solution

Key Concepts

1. Horizontal vs Vertical Scaling

- Horizontal - increasing the number of nodes (i.e. adding more servers)
- Vertical - increase the resources of a specific node (i.e. adding additional memory)

2. Load balancing

- Allows system to distribute load easily so that one server doesn't crash and take down the whole system.
- Requires a network of cloned servers

3. Database Denormalization and NoSQL

- Denormalization - adding redundant information into a db to speed up reads
- NoSQL - does not support joins, and structures data in a different way. Designed to scale better

4. Database Partitioning and Sharding

- Vertical Partitioning - partitioning by feature
- Key/Hash Based Partitioning - Uses some part of the data to allocate N servers and put data on a mod(key, n)
- Directory-based partitioning - you maintain a lookmup table for where data can be found

5. Caching

- In-memory cache can deliver rapid results.
- If it cant find cached memory, it looks to the data store to find it.

6. Async Processing and Queues

- Pre-Process Jobs
- Notifying user when the process is done

7. Networking Metrics

- Bandwidth - max amount of data that can be transferred in a unit of time
- Throughput - actual amount of data that is transfered
- Latency - how long it takes data to go from one end to the other

8. MapReduce

- Typically used to process large amount of data
- Map takes some data and emits a key, value pair
- Reduce takes key and associated values, and reduces them in some way, emitting a new key and value

Considerations

1. Handle Failures
2. Availability and Reliability
3. Read-heavy vs Write-Heavy
4. Security