What is a System Design Interview?

The purpose of a system design interview is to assess a candidate's ability to design and understand complex systems. It's a crucial part of the hiring process for roles that involve system architecture and engineering, such as software engineers, system architects, and DevOps engineers. Here's a breakdown of what these interviews aim to evaluate:

1. Evaluating Technical Proficiency

- **Design Skills**: Assessing the ability to design a scalable, efficient, and robust system architecture.
- **Problem-solving Skills**: Gauging how you approach and solve complex, open-ended problems.
- **Technical Knowledge**: Understanding your familiarity with various technologies, databases, frameworks, and protocols.

2. Understanding Approach and Methodology

- **Requirements Gathering**: Your ability to understand and outline the requirements of a system before diving into the solution.
- **Balancing Trade-offs**: How you balance various trade-offs in system design, such as between scalability and cost, or performance and reliability.
- **Decision-making Process**: Evaluating your reasoning behind choosing certain technologies or architectures over others.

3. Testing Soft Skills

- **Communication**: Your ability to clearly articulate your thought process and design choices.
- **Collaboration**: Assessing how you interact with interviewers, which can mimic real-life collaboration with team members.
- **Adaptability**: Your response to new information or feedback during the interview, reflecting your ability to adapt in real-world projects.

4. Real-world Application

- **Practicality**: Ensuring that your designs can be practically implemented and are not just theoretical.
- **Scalability and Performance**: Your understanding of how the system would perform under real-world constraints and loads.

5. Innovation and Creativity

- **Creative Thinking**: Looking for innovative solutions and ideas that showcase your ability to think outside the box.
- **Future-Proofing**: How you design systems with future growth, maintenance, and potential challenges in mind.

What System Design Interview is Not About

- It’s not about getting the perfect answer. In fact, there often isn’t one “correct” solution in system design.
- It’s not a test of memorization. While familiarity with certain tools and technologies is helpful, understanding the concepts is more important.

Conclusion

The system design interview is a critical tool for employers to assess not just your technical abilities, but also your problem-solving approach, communication skills, and overall fit for roles that require designing and working with large-scale systems. It's a comprehensive evaluation of how you would handle real-life challenges in system architecture and engineering.

Functional vs. Non-functional Requirements

In the context of system design interviews, understanding functional and non-functional requirements is key to showcasing your ability to design a system that meets both the specific actions it should perform and how it should perform them.

Functional Requirements

- **Definition**: These are the requirements that define what a system is supposed to do. They describe the various functions that the system must perform.
- **Examples**:
  - A user authentication system must validate user credentials and provide access levels.
  - An e-commerce website should allow users to browse products, add them to a cart, and complete purchases.
  - A report generation system must collect data, process it, and generate timely reports.

Importance in Interviews

- **Demonstrates Understanding of Core Features**: Shows you know what the system needs to do to satisfy its primary objectives.
- **Basis for System Design**: Functional requirements often form the backbone of your system design.

Non-Functional Requirements

- **Definition**: These requirements describe how the system performs a task, rather than what tasks it performs. They are related to the quality attributes of the system.
- **Examples**:
  - Scalability: The system should handle growth in users or data.
  - Performance: The system should process transactions within a specified time.
  - Availability: The system should be up and running a defined percentage of time.
  - Security: The system must protect sensitive data and resist unauthorized access.

Importance in Interviews

- **Showcases Depth of Design Knowledge**: Demonstrates your understanding of the broader implications of system design.
- **Highlights System Robustness and Quality**: Reflects how well your system design can meet real-world constraints and user expectations.

Integrating Both in Interviews

- **Scenario-Based Discussions**: When presented with a scenario, identify both the functional (what the system should do) and non-functional (how the system should do it) requirements.
- **Balancing Act**: Exhibit your ability to balance both types of requirements, showing that you can design a system that not only meets its functional goals but also performs effectively, securely, and reliably.

In System Design Interviews

When you're in a system design interview, here's how you can handle these requirements:

1. **Clarify Requirements**: Start by asking questions to understand both functional and non-functional requirements. Interviewers often leave these vague to see if you'll probe for more details.

2. **Prioritize**: Not all requirements are equally important. Identify which ones are critical for the system's success.

3. **Trade-offs**: Discuss trade-offs related to different architectural decisions, especially concerning non-functional requirements. For example, a system highly optimized for read operations might have slower write operations.

4. **Use Real-World Examples**: If you can, relate your points to real-world systems or your past experiences. This shows practical understanding.

5. **Balance**: Ensure you're not focusing too much on one type of requirement over the other. A well-rounded approach is often necessary.

Remember, in system design interviews, interviewers are often interested in seeing how you think and approach problems, not just your final solution. Demonstrating a clear understanding of both functional and non-functional requirements is key to showing your comprehensive knowledge in system design.

What are Back-of-the-Envelope Estimations?

Back of the envelope estimations in system design interviews are like quick, rough calculations you might do on a napkin during lunch - they're not detailed or exact, but give you a good ballpark figure. These rough calculations help you quickly assess the feasibility of a proposed solution, estimate its performance, and identify potential bottlenecks.

Purpose

Back-of-the-envelope estimation is a technique used to quickly approximate values and make rough calculations using simple arithmetic and basic assumptions. This method is particularly useful in system design interviews, where interviewers expect candidates to make informed decisions and trade-offs based on rough estimates.

Why is Estimation Important in System Design Interviews?

During a system design interview, you'll be asked to design a scalable and reliable system based on a set of requirements. Your ability to make quick estimations is essential for several reasons:

1. **Indicates System Scalability**: Highlights your understanding of how the system can grow or adapt.
2. **Validate proposed solutions:** Estimation helps you ensure that your proposed architecture meets the requirements and can handle the expected load.
3. **Identify bottlenecks:** Quick calculations help you identify potential performance bottlenecks and make necessary adjustments to your design.

4. **Demonstrate your thought process:** Estimation showcases your ability to make informed decisions and trade-offs based on a set of assumptions and constraints.
5. **Communicate effectively:** Providing estimates helps you effectively communicate your design choices and their implications to the interviewer.
6. **Quick Decision Making**: Reflects your ability to make swift estimations to guide your design decisions.

Estimation Techniques

1. Rule of thumb

Rules of thumb are general guidelines or principles that can be applied to make quick and reasonably accurate estimations. They are based on experience and observation, and while not always precise, they can provide valuable insights in the absence of detailed information. For example, estimating that a user will generate 1 MB of data per day on a social media platform can serve as a starting point for capacity planning.

2. Approximation

Approximation involves simplifying complex calculations by rounding numbers or using easier-to-compute values. This technique can help derive rough estimates quickly and with minimal effort. For instance, assuming 1,000 users instead of 1,024 when estimating storage requirements can simplify calculations and still provide a reasonable approximation.

3. Breakdown and aggregation

Breaking down a problem into smaller components and estimating each separately can make it easier to derive an overall estimate. This technique involves identifying the key components of a system, estimating their individual requirements, and then aggregating these estimates to determine the total system requirements. For example, estimating the storage needs for user data, multimedia content, and metadata separately can help in determining the overall storage requirements of a social media platform.

4. Sanity check

A sanity check is a quick evaluation of an estimate to ensure its plausibility and reasonableness. This step helps identify potential errors or oversights in the estimation process and can lead to more accurate and reliable results. For example, comparing the estimated storage requirements for a messaging service with the actual storage used by a similar existing service can help validate the estimate.

Types of Estimations in System Design Interviews

In system design interviews, there are several types of estimations you may need to make:

1. **Load estimation:** Predict the expected number of requests per second, data volume, or user traffic for the system.
2. **Storage estimation:** Estimate the amount of storage required to handle the data generated by the system.
3. **Bandwidth estimation:** Determine the network bandwidth needed to support the expected traffic and data transfer.
4. **Latency estimation:** Predict the response time and latency of the system based on its architecture and components.

5. **Resource estimation:** Estimate the number of servers, CPUs, or memory required to handle the load and maintain desired performance levels.

Process

1. **Understand the Scope:** Clarify the scale of the problem - how many users, how much data, etc.
2. **Use Simple Math:** Utilize basic arithmetic to estimate the scale of data and resources.
3. **Round Numbers for Simplicity:** Use round numbers to make calculations easier and faster.
4. **Be Logical and Reasonable:** Ensure your estimations make sense given the context of the problem.

Practical Examples

1. Load Estimation

Suppose you're asked to design a social media platform with 100 million daily active users (DAU) and an average of 10 posts per user per day. To estimate the load, you'd calculate the total number of posts generated daily:

```
100 million DAU * 10 posts/user = 1 billion posts/day
```

Then, you can estimate the request rate per second:

```
1 billion posts/day / 86,400 seconds/day ≈ 11,574
requests/second
```

2. Storage Estimation

Consider a photo-sharing app with 500 million users and an average of 2 photos uploaded per user per day. Each photo has an average size of 2 MB. To estimate the storage required for one day's worth of photos, you'd calculate:

```
500 million users * 2 photos/user * 2 MB/photo = 2,000,000,000 MB/day
```

3. Bandwidth Estimation

For a video streaming service with 10 million users streaming 1080p videos at 4 Mbps, you can estimate the required bandwidth:

```
10 million users * 4 Mbps = 40,000,000 Mbps
```

4. Latency Estimation

Suppose you're designing an API that fetches data from multiple sources, and you know that the average latency for each source is 50 ms, 100 ms, and 200 ms, respectively. If the data fetching process is sequential, you can estimate the total latency as follows:

```
50 ms + 100 ms + 200 ms = 350 ms
```

If the data fetching process is parallel, the total latency would be the maximum latency among the sources:

```
max(50 ms, 100 ms, 200 ms) = 200 ms
```

5. Resource Estimation

Imagine you're designing a web application that receives 10,000 requests per second, with each request requiring 10 ms of CPU time. To estimate the number of CPU cores needed, you can calculate the total CPU time per second:

```
10,000 requests/second * 10 ms/request = 100,000 ms/second
```

Assuming each CPU core can handle 1,000 ms of processing per second, the number of cores required would be:

```
100,000 ms/second / 1,000 ms/core = 100 cores
```

System Design Examples

1. Designing a messaging service

Imagine you are tasked with designing a messaging service similar to WhatsApp. To estimate the system's requirements, you can start by considering the following aspects:

- **Number of users:** Estimate the total number of users for the platform. This can be based on market research, competitor analysis, or historical data.
- **Messages per user per day:** Estimate the average number of messages sent by each user per day. This can be based on user behavior patterns or industry benchmarks.
- **Message size:** Estimate the average size of a message, considering text, images, videos, and other media content.
- **Storage requirements:** Calculate the total storage needed to store messages for a specified retention period, taking into account the number of users, messages

per user, message size, and data redundancy.

- **Bandwidth requirements:** Estimate the bandwidth needed to handle the message traffic between users, considering the number of users, messages per user, and message size.

By breaking down the problem into smaller components and applying estimation techniques, you can derive a rough idea of the messaging service's requirements, which can guide your design choices and resource allocation.

2. Designing a video streaming platform

Suppose you are designing a video streaming platform similar to Netflix. To estimate the system's requirements, consider the following aspects:

- **Number of users:** Estimate the total number of users for the platform based on market research, competitor analysis, or historical data.
- **Concurrent users:** Estimate the number of users who will be streaming videos simultaneously during peak hours.
- **Video size and bitrate:** Estimate the average size and bitrate of videos on the platform, considering various resolutions and encoding formats.
- **Storage requirements:** Calculate the total storage needed to store the video content, taking into account the number of videos, their sizes, and data redundancy.
- **Bandwidth requirements:** Estimate the bandwidth needed to handle the video streaming traffic, considering the number of concurrent users, video bitrates, and user locations.

By applying estimation techniques and aggregating the individual estimates, you can get a ballpark figure of the video streaming platform's requirements, which can inform your design decisions and resource allocation.

Tips for Successful Estimation in Interviews

Estimation plays a crucial role in system design interviews, as it helps you make informed decisions about your design and demonstrates your understanding of the various factors that impact the performance and scalability of a system. Here are some tips to help you ace the estimation part of your interviews:

1. Break down the problem

When faced with a complex system design problem, break it down into smaller, more manageable components. This will make it easier to estimate each component's requirements and help you understand how they interact with each other. By identifying the key components and estimating their requirements separately, you can then aggregate your estimates to get a comprehensive view of the system's needs.

2. Use reasonable assumptions

During an interview, you may not have all the necessary information to make precise estimations. In such cases, make reasonable assumptions based on your knowledge of similar systems, industry standards, or user behavior patterns. Clearly state your assumptions to the interviewer, as this demonstrates your thought process and enables them to provide feedback or correct your assumptions if necessary.

Operation name	Time
L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	100 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	10,000 ns = 10 µs
Send 2K bytes over 1 Gbps network	20,000 ns = 20 µs
Read 1 MB sequentially from memory	250,000 ns = 250 µs
Round trip within the same datacenter	500,000 ns = 500 µs
Disk seek	10,000,000 ns = 10 ms
Read 1 MB sequentially from network	10,000,000 ns = 10 ms
Read 1 MB sequentially from disk	30,000,000 ns = 30 ms
Send packet CA->Netherlands->CA	150,000,000 ns = 150 ms

3. Leverage your experience

Drawing from your past experiences can be beneficial when estimating system requirements. If you have worked on similar systems or have experience with certain technologies, use that knowledge to inform your estimations. This will not only help you make more accurate estimations but also showcase your expertise to the interviewer.

4. Be prepared to adjust your estimations

As you progress through the interview, the interviewer may provide additional information or challenge your assumptions, requiring you to adjust your estimations. Be prepared to adapt and revise your estimations accordingly. This demonstrates your ability to think critically and shows that you can handle changing requirements in a real-world scenario.

5. Don’t Forget to Ask Clarifying Questions

Don’t hesitate to ask the interviewer clarifying questions if you’re unsure about a requirement or assumption. This will help you avoid making incorrect estimations and showcase your problem-solving abilities.

6. Communicate your thought process

Throughout the estimation process, communicate your thought process clearly to the interviewer. Explain how you arrived at your estimations and the assumptions you made along the way. This allows the interviewer to understand your reasoning, provide feedback, and assess your problem-solving skills.

Conclusion

Back of the envelope estimations are crucial in system design interviews as they showcase your ability to grasp the scale of a system quickly and assess the feasibility and resource needs of your design. It's a skill that demonstrates both technical knowledge and practical problem-solving ability.

Things to Avoid During System Design Interview

In a system design interview, while it's important to showcase your skills and knowledge, it's equally crucial to be aware of common pitfalls. Avoiding these mistakes can greatly improve your chances of success. Here are some key "don'ts" for a system design interview:

1. Don’t Ignore the Requirements

- **Neglecting to Clarify**: Failing to ask questions or clarify requirements can lead to a design that misses the mark.
- **Oversimplifying the Problem**: Don’t oversimplify the problem or ignore the complexities involved.

2. Don’t Dive into Details Too Soon

- **Rushing into Low-Level Details**: Starting with low-level details before establishing the high-level design can make your solution seem disjointed.
- **Losing Sight of the Big Picture**: Focus first on the overall architecture and how different components interact.

3. Don’t Stick Rigidly to One Idea

- **Being Inflexible**: Being too rigid with your initial idea can prevent you from considering better alternatives.
- **Ignoring Interviewer’s Hints**: The interviewer might provide hints or feedback; not adapting your design accordingly can be seen as a lack of collaboration or adaptability.

4. Don’t Overlook Trade-offs

- **Ignoring Trade-offs**: Every design decision has trade-offs. Not discussing these can show a lack of depth in your understanding.
- **Failing to Justify Decisions**: Be prepared to explain why you chose one approach over another.

5. Don’t Neglect Non-Functional Requirements

- **Overlooking Scalability, Reliability, etc.**: Focusing solely on functional aspects and neglecting non-functional requirements like scalability and reliability is a common mistake.
- **Not Considering System Constraints**: Real-world constraints such as cost, time, and existing technology stack should be considered.

6. Don’t Under-Communicate

- **Poor Explanation**: Failing to clearly articulate your thoughts can leave interviewers unsure about your understanding and approach.
- **Not Engaging the Interviewer**: This is a dialogue, not a monologue. Engage with the interviewer, ask questions, and be receptive to feedback.

7. Don’t Be Overconfident or Arrogant

- **Overconfidence**: Being overly confident can lead to dismissing valuable feedback or overlooking key aspects of the problem.
- **Not Acknowledging What You Don’t Know**: It’s okay not to know everything. Being open about areas you are unsure of is better than providing incorrect information.

Conclusion

A system design interview is not just about getting the right answer. It's about demonstrating your problem-solving approach, your ability to adapt, and how you communicate and collaborate. Avoiding these pitfalls can help you present yourself as a well-rounded candidate capable of handling the complexities of real-world system design.