Valid if interviewing for a backend engineer or generic software engineer

Make sure you are driving the discussion when answering these questions

This is obsolete now. Refer to https://www.interviewbit.com/courses/system-design/ going forward.

Pre-requisites :

Systems are complex, and when you're designing a system you're grappling with its full complexity. Given this, there are many topics you should be familiar with, such as:

* **Concurrency.** Do you understand threads, deadlock, and starvation? Do you know how to parallelize algorithms? Do you understand consistency and coherence?
* **Networking.** Do you roughly understand IPC and TCP/IP? Do you know the difference between throughput and latency, and when each is the relevant factor?
* **Abstraction.** You should understand the systems you're building upon. Do you know roughly how an OS, file system, and database work? Do you know about the various levels of caching in a modern OS?
* **Real-World Performance.** You should be familiar with the speed of everything your computer can do, including the relative performance of RAM, disk, SSD and your network (http://highscalability.com/numbers-everyone-should-know - Just the ball park numbers).
* **Estimation.** Estimation, especially in the form of a back-of-the-envelope calculation, is important because it helps you narrow down the list of possible solutions to only the ones that are feasible. Then you have only a few prototypes or micro-benchmarks to write.
* **Availability and Reliability.** Are you thinking about how things can fail, especially in a distributed environment? Do know how to design a system to cope with network failures? Do you understand durability ?
* Understand CAP theorem.
* Know about consistent hashing

First Step :

Cover everything here : http://www.hiredintech.com/system-design/
Watch the video at : https://www.youtube.com/watch?v=-W9F__D3oY4 (The video has few audience questions which is not properly audible, and it moves at a fairly slow pace. Maybe set the play speed to 1.25 / 1.5x).

If you have time left over, go through : http://book.mixu.net/distsys/ebook.html

Basic checklist

With every question you attempt, make sure you ask yourself the following questions where applicable :

- Have I understood the requirement correctly ? Do I completely understand the expectations in terms of performance, features and reliability. What are critical requirements for the system :
    - Does it need to be high throughput ?
    - Does it need to be highly available ?
    - Does it have high write traffic and hence concurrency issues, or is it read heavy.

  Check https://player.vimeo.com/video/86413525 and https://player.vimeo.com/video/86413528 for example.
- Is my design fault tolerant ? What happens when a machine goes down ? (application / database / loadbalancer machine).
- What can we compromise on from CAP (Consistency / Availability / Partitioning) ?
- For write heavy questions, how do you handle concurrent reads / writes ? Can writes be batched / sampled ?
- Have you taken care of data not fitting on a single machine ? How do you shard data across machines ?
- Once data is sharded, how do you correctly route the requests ?
- Is the current system latency sensitive ? Search typeahead for example is extremely latency sensitive. If yes, then how do you guarantee extremely low latencies ?
- Relational DB / NoSQL ? Why ?

We will cover around **20 case studies** here.
Make sure you attempt these cases yourself before looking into the final solution.

CASE STUDY 1 :

Design a URL shortening service.

Already covered at the hiredintech link shared.
Check https://player.vimeo.com/video/86413525 ,
https://player.vimeo.com/video/86413528 and https://player.vimeo.com/video/86413593
incase you missed it.

CASE STUDY 2 :

Design a simplified version of Twitter where people can post tweets, follow other people and favorite tweets.*

http://www.hiredintech.com/data/uploads/hiredintech_system_design_the_twitter_problem_beta.pdf

CASE STUDY 3 :

Design a search typeahead.

Clarifying questions :

- How many typeahead suggestions are to be provided ?
- Do we need to account for spelling mistakes ? Example : Should typing “mik” give michael as a suggestion because michael is really popular as a query ?
- What can be the length of a search query ?

Lets assume for this question, we focus on only providing 5 suggestions at max. We need not account for spelling mistakes, and assume that the suggestions will have the typed phrase as the strict prefix.

So, in effect we have a system which does 2 major things :

- Given a query, it gives back upto 5 typeahead suggestions.
- Given a search query (which is actually search for by the user), it updates the suggestions if needed.

One approach would be to store the data as a trie. But, do you construct a complete trie ? Is there a limit to the number of characters in a query ? Users can type any random string as a query and that can cause the trie size to blow up.
Alright, so we only construct the nodes that are needed.

How do we calculate the top 5 suggestions then ? Top 5 frequency query terms with the user typed query as prefix seems to be a good approach. How do we find top 5 frequency query results for a query ?
Do we go and traverse the whole subtree in the trie to find the top frequency terms ? If so, what can be the size of the subtree ? Do you think its going to grow too inefficient (especially because typeahead is latency sensitive) ?
Can we store some additional information on the node itself ? How about we store the top 5 terms along with the frequency on the top 5 terms ? Query becomes really fast then. Update in the trie would mean percolating up the new term with its frequency, and see if its eligible to be in top 5 at every node.
What about the updates ? Do you update the trie with every update ? Would that cause things to be really slow ? Would sampling work here ?
Now, what do you think about the trie size ? Do you think it fits on a single machine ?
There are 2 options here :

1) You shard the trie. How do you shard the trie ? Do you only shard it on the first level ?
2) Maintain the trie as a refined set of queries which are more frequent than a certain threshold. All query terms along with the actual frequencies are stored in another hashmap. How do you do the update ? Batch update ? Would you compromise on real time updates for recent trending search terms ? What if you trigger the entry / update on search terms when it crosses certain threshold post sampling ?

What about fault tolerance ? Replication ?
How about optimizations on the client side ? Do you trigger off a request to the backend on every keystroke ? Or do you wait for 100ms and trigger off request if there have been no other keystroke ?

CASE STUDY 4 :

Design Facebook Timeline

https://www.facebook.com/note.php?note_id=10150468255628920

CASE STUDY 5 :

Design an online multiplayer game

- How to Create an Asynchronous Multiplayer Game
- How to Create an Asynchronous Multiplayer Game Part 2: Saving the Game State to Online Database

- How to Create an Asynchronous Multiplayer Game Part 3: Loading Games from the Database
- How to Create an Asynchronous Multiplayer Game Part 4: Matchmaking
- Real Time Multiplayer in HTML5

CASE STUDY 6 :

Design notification system

http://stackoverflow.com/questions/9735578/building-a-notification-system

CASE STUDY 7 :

Design a trending topic system

http://www.michael-noll.com/blog/2013/01/18/implementing-real-time-trending-topics-in-storm/
A small gist at https://www.quora.com/How-does-Twitter-select-trending-topics

CASE STUDY 8 :

Design a Facebook like status system

http://stackoverflow.com/questions/7072924/what-is-the-design-architecture-behind-facebooks-status-update-mechanism

CASE STUDY 9 :

Design Facebook messages

https://www.facebook.com/notes/facebook-engineering/inside-facebook-messages-application-server/10150162742108920

CASE STUDY 10 :

Design Facebook places with check-in. Focus on implementing places suggestions as well
Hint : K-D tree

CASE STUDY 11 :

How would you design the feature in LinkedIn where it computes how many hops there are between you and another person? (Or degree of connection at Facebook)

Hint: Meet in the middle

CASE STUDY 12 :

Design a Google document system

https://neil.fraser.name/writing/sync/

CASE STUDY 13 :

Design a key value store (distributed)

http://www.slideshare.net/dvirsky/introduction-to-redis

CASE STUDY 14 :

Design gmail backend (think about how requirements should be different from Facebook messages)
http://blog.sajithmr.me/gmail-architecture/

CASE STUDY 15 :

How would you optimize an elevator system for a building with 50 floors and 4 elevators ? Optimize in terms of lowest wait times for the users.

http://www.quora.com/What-are-ways-to-optimize-the-service-algorithm-for-an-elevator
http://dan-nolan.com/how-i-would-optimize-the-elevators-in-our-office-building/

CASE STUDY 16 :

A random ID generator which generates unique IDs. Think about such a generator for a company like Google.
http://www.slideshare.net/davegardnerisme/unique-id-generation-in-distributed-systems

CASE STUDY 17 :

Design a caching system (Think about Write back vs write through and cache invalidation)

https://msdn.microsoft.com/en-us/library/dd129907.aspx

CASE STUDY 18 :

Design a scalable web crawling system
http://cis.poly.edu/tr/tr-cis-2001-03.pdf

CASE STUDY 19 :

Design a recommendation system (this is fairly wide scope. Points for narrowing down the requirements)

http://ijcai13.org/files/tutorial_slides/td3.pdf
http://tech.hulu.com/blog/2011/09/19/recommendation-system/

CASE STUDY 20 :

Design image sharing website (Flickr / Instagram)

http://highscalability.com/blog/2011/12/6/instagram-architecture-14-million-users-terabytes-of-photos.html

Interesting and Useful reads :

- http://highscalability.com/blog/2013/4/15/scaling-pinterest-from-0-to-10s-of-billions-of-page-views-a.html
- http://highscalability.com/youtube-architecture