Create a Photo Sharing App

After I published our first system design interview question post – how to design twitter (http://blog.gainlo.co/index.php/2016/02/17/system-design-interview-question-how-to-design-twitter-part-1/), we got so many requests for more tutorials like this.

Gainlo (http://www.gainlo.co/?utm_source=blog&utm_medium=System%20Design%20Interview%20Question%20-%20Design%20a%20Picture%20Sharing%20System&utm_campaign=blog) team has hand-picked a list of system design interview questions that are both classic and easy to extend. As a result, I believe these questions can be great examples to help you learn more about how to ace system design interviews.

Again, it's worth to note that the analysis provided by Gainlo (http://www.gainlo.co/?utm_source=blog&utm_medium=System%20Design%20Interview%20Question%20-%20Design%20a%20Picture%20Sharing%20System&utm_campaign=blog) team is only used for system design interview illustration. The real solution in production can be quite different as we significantly simplified the problem here.

Create a photo sharing app

How to create a photo sharing app like Instagram (https://www.instagram.com/?hl=en)?

More specifically, the system allows people to follow each other, share/comment/like pictures, and maybe some other features like explore (http://blog.instagram.com/post/122260662827/150623-search-and-explore), advertisement (https://business.instagram.com/advertising/) and so on so forth.

There are several reasons we'd like to analyze this problem here. First of all, picture sharing systems are quite popular. I'm not picking a weird issue that has almost no applications in real world. Instead, there are many similar products like Pinterest (http://www.pinterest.com), Flickr (https://www.flickr.com/) etc..

Secondly, the problem is general, which is extremely common in system design interviews. Usually, interviewers won't ask you to solve a well-defined problem, which is exactly what makes many people uncomfortable.

Lastly, the analysis covers a wide range of topics like scalability, database, data analysis etc. so that it can be reused in other system design interview question as well.

High-level solution

As we have emphasized multiple times in previous posts (http://8 Things You Need to Know Before a System Design Interview), it's recommended to start with a high-level solution and then you can dig into all sorts of details later.

The advantage of this approach is that you're gonna have a clear idea of what you are trying to solve and interviewers are less likely to be confused.

To design a picture sharing system, it's quite straightforward to identify two major objects – user object and picture object.

Personally, I'd like to use relational database (https://en.wikipedia.org/wiki/Relational_database) to explain as it's usually easier to understand. In this case, we will have a user table for sure, which contains information like name, email, registration date and so on. The same goes for picture table.

In addition, we also need to store two relations – user follow relation and user-picture relation. This comes very naturally and it's worth to note that user follow relation is not bi-directional.

Therefore, having such data model allows users to follow each other. To check a user's feed, we can fetch all pictures from people he follows.

Potential scale issues

The above solution should definitely work well. As an interviewer, I always like to ask what can go wrong when we have millions of users and how to solve it.

This question is a great way to test if a candidate can foresee potential scale issues (https://en.wikipedia.org/wiki/Scalability) and it's better than just asking how can you solve problem XYZ.

Of course, there're no standard answers and I would like to list few ideas as inspirations.

1. Response time

When users get to a certain number, it's quite common to see slow response time becomes the bottleneck.

For instance, one costly operation is to render users feed. The server has to go over everyone the user follows, fetch all the pictures from them, and rank them based on particular algorithms. When a user has followed many people with a large number of pictures, the operation can be slow.

Various approaches can be applied here. We can upgrade the ranking algorithm if it's the bottleneck, e.g. if we are ranking by date, we can just read the top N most recent pictures from each person with infinite scroll feature (https://www.google.com/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=twitter+infinite+scroll). Or we can use offline pipelines to precompute some signals that can speed up the ranking.

The point is that it's unlikely to have someone following hundreds of users, but it's likely to have someone with thousands of pictures. Therefore, accelerating the picture fetching and ranking is the core.

2. Scale architecture

When there are only tens of users and pictures, we may store and serve everything from a single server.

However, with millions of users, a single server is far from enough due to storage, memory, CPU bound issues etc.. That's why it's pretty common to see server crashes when there are a large number of requests.

To scale architecture, the rule of thumb is that **service-oriented architecture beats monolithic application.**

Instead of having everything together, it's better to divide the whole system into small components by service and separate each component. For example, we can have database separate from web apps (in different servers) with load balancers (https://en.wikipedia.org/wiki/Load_balancing_(computing)).

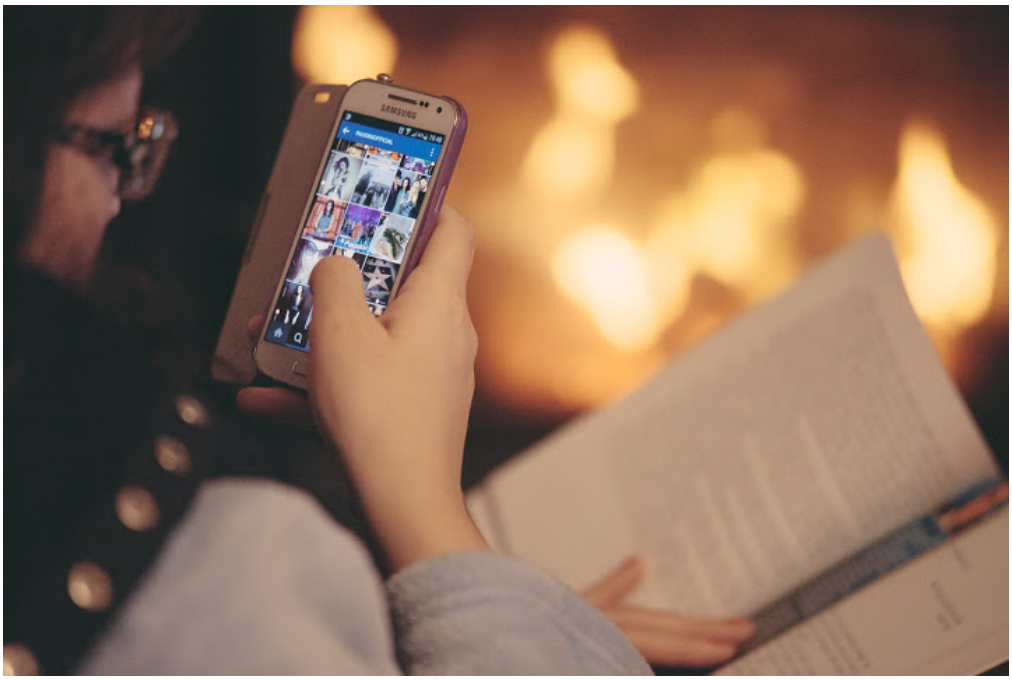
3. Scale database

Even if we put the database in a separate server, it will not be able to store an infinite number of data.

At a certain point, we need to scale the database. For this specific problem, we can either do the vertical splitting (partitioning) (https://en.wikipedia.org/wiki/Partition_(database)) by splitting the database into sub-databases like user database, comment database etc. or horizontal splitting (sharding) (https://en.wikipedia.org/wiki/Shard_(database_architecture)) by splitting based on attributes like US users, European users.

You can check this post (http://highscalability.com/blog/2014/5/12/4-architecture-issues-when-scaling-web-applications-bottlene.html) for deeper analysis of scalability issues.

Feed ranking



It's also interesting to discuss how to rank feeds (pictures) in users timelines.

Although it's quite straightforward to rank everything in chronological order, is it the best approach? Such open-ended question is very common in system design interviews.

Actually, there can be quite a few alternatives. For example, an algorithm that combines time and how likely the user will like this picture is definitely promising.

To design such algorithm, a common strategy is to come up with a scoring mechanism that takes various features as signals and computes a final score for each picture.

Intuitively, features that matter a lot include like/comment numbers, whether the user has liked many photos of the owner and so on. A linear combination (https://en.wikipedia.org/wiki/Linear_regression) can be used as a starting point due to simplicity.

Later on, more advanced machine learning algorithms like collaborative filtering (https://en.wikipedia.org/wiki/Collaborative_filtering) is worth to try.

Image optimization

Since a picture sharing system is full of images, I would like to ask what can be optimized related to images?

First of all, it's usually recommended to store all pictures separately in production. Amazon S3 (https://aws.amazon.com/s3/) is one of the most popular storage systems. However, you don't need to be able to come up with this.

The point is that images are usually of large size and seldom get updated. So a separate system for image storage has a lot of advantages. For instance, cache and replication can be much simpler when files are static.

Secondly, to save space, images should be compressed. One common approach is to only store/serve the compressed version of images. Google photos (http://www.ghacks.net/2015/05/29/a-close-look-at-google-photos-unlimited-storage-offer/) actually is using this approach with unlimited free storage.

Summary

There are still some topics that I haven't covered in this post like how to build the explore feature (http://blog.instagram.com/post/122260662827/150623-search-and-explore) in Instagram. I hope you can take some time to think about it.

Also for reference, you can check Instagram infrastructure (http://highscalability.com/blog/2011/12/6/instagram-architecture-14-million-users-terabytes-of-photos.html) and Flickr architecture (http://highscalability.com/flickr-architecture). However, I don't think they are very helpful to system design interviews as they are too focused on techniques rather than design principles.

If you find this post helpful, I would really appreciate if you can share it with your friends. Also you can check more system design interview questions (http://blog.gainlo.co/index.php/category/system-design-interview-questions/) and analysis here.

The post is written by **Gainlo (http://www.gainlo.co/?utm_source=blog&utm_medium=footer-link http%3A//blog.gainlo.co/index.php/2016/03/01/system-design-interview-question-create-a-photo-sharing-app/&utm_campaign=blog)** - a platform that allows you to have mock interviews with employees from Google, Amazon etc..

I'd like to learn more (http://www.gainlo.co/?utm_source=blog&utm_medium=footer http%3A//blog.gainlo.co/index.php/2016

Related Posts:

1. **Create a TinyURL System (http://blog.gainlo.co/index.php/2016/03/08/system-design-interview-question-create-tinyurl-system/)**
2. **Random ID Generator (http://blog.gainlo.co/index.php/2016/06/07/random-id-generator/)**
3. **Design News Feed System (Part 1) (http://blog.gainlo.co/index.php/2016/03/29/design-news-feed-system-part-1-system-design-interview-questions/)**
4. **Design Facebook Chat Function (http://blog.gainlo.co/index.php/2016/04/19/design-facebook-chat-function/)**

3 thoughts on “Create a Photo Sharing App”

NITIN
April 22, 2016 at 10:02 pm (http://blog.gainlo.co/index.php/2016/03/01/system-design-interview-question-create-a-photo-sharing-app/#comment-653)

i like reading your posts… they are simple so easy to understand for me as a beginner..
Suggestions:
=> The high level design is too abstract. I would like to see at least one end to end complete design solution (with all pieces combined)
=> I would like to see the “Takeaway” section in the design posts as well.

g.gainlo.co/index.php/2016/03/01/system-design-interview-question-create-a-photo-sharing-app/?replytocom=653#respond)

JAKE
April 26, 2016 at 9:54 pm (http://blog.gainlo.co/index.php/2016/03/01/system-design-interview-question-create-a-photo-sharing-app/#comment-679)

Hi Nitin,

This is really great feedback. I'd love to improve all these areas in our future posts!

gainlo.co/index.php/2016/03/01/system-design-interview-question-create-a-photo-sharing-app/?replytocom=679#respond)

KUNAL VERMA
July 9, 2016 at 11:54 am (http://blog.gainlo.co/index.php/2016/03/01/system-design-interview-question-create-a-photo-sharing-app/#comment-1396)

I liked your post, but can you please provide a tutorial about the drawing the designs like, for this question

In an online teaching system,there are n number of teachers and each one teaches only one subject to any number of students. And a student can join to any number of teachers to learn those subjects. And each student can give one preference through which he can get updates about the subject or class timings etc.
Those preferences can be through SMS or twitter/facebook or email..etc.
Design above system and draw the diagram for above.

what kind of diagrams? and what need to be their in diagram.

P.S. : I am not asking to solve this problem, just an approach for diagrams / class diagram / or whatever you feel like to add.

gainlo.co/index.php/2016/03/01/system-design-interview-question-create-a-photo-sharing-app/?replytocom=1396#respond)

LEAVE A REPLY

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment

‹ Warning! Are You a Slow Programmer in Interviews? (http://blog.gainlo.co/index.php/2016/02/27/warning-are-you-a-slow-programmer-in-interviews/)

Create a TinyURL System › (http://blog.gainlo.co/index.php/2016/03/08/system-design-interview-question-create-tinyurl-system/)

CATEGORIES

Coding Interview Questions (http://blog.gainlo.co/index.php/category/coding-interview-questions/)

Common Pitfalls (http://blog.gainlo.co/index.php/category/common-pitfalls/)

Common Questions (http://blog.gainlo.co/index.php/category/common-questions/)

Facebook Interview Questions (http://blog.gainlo.co/index.php/category/facebook-interview-questions/)

Mock Interview Thoughts (http://blog.gainlo.co/index.php/category/mock-interview-thoughts/)

Others (http://blog.gainlo.co/index.php/category/others/)

System Design Interview Questions (http://blog.gainlo.co/index.php/category/system-design-interview-questions/)

The Complete Guide to Google Interview Preparation (http://blog.gainlo.co/index.php/category/google-interview-preparation/)

Uber Interview Questions (http://blog.gainlo.co/index.php/category/uber-interview-questions/)