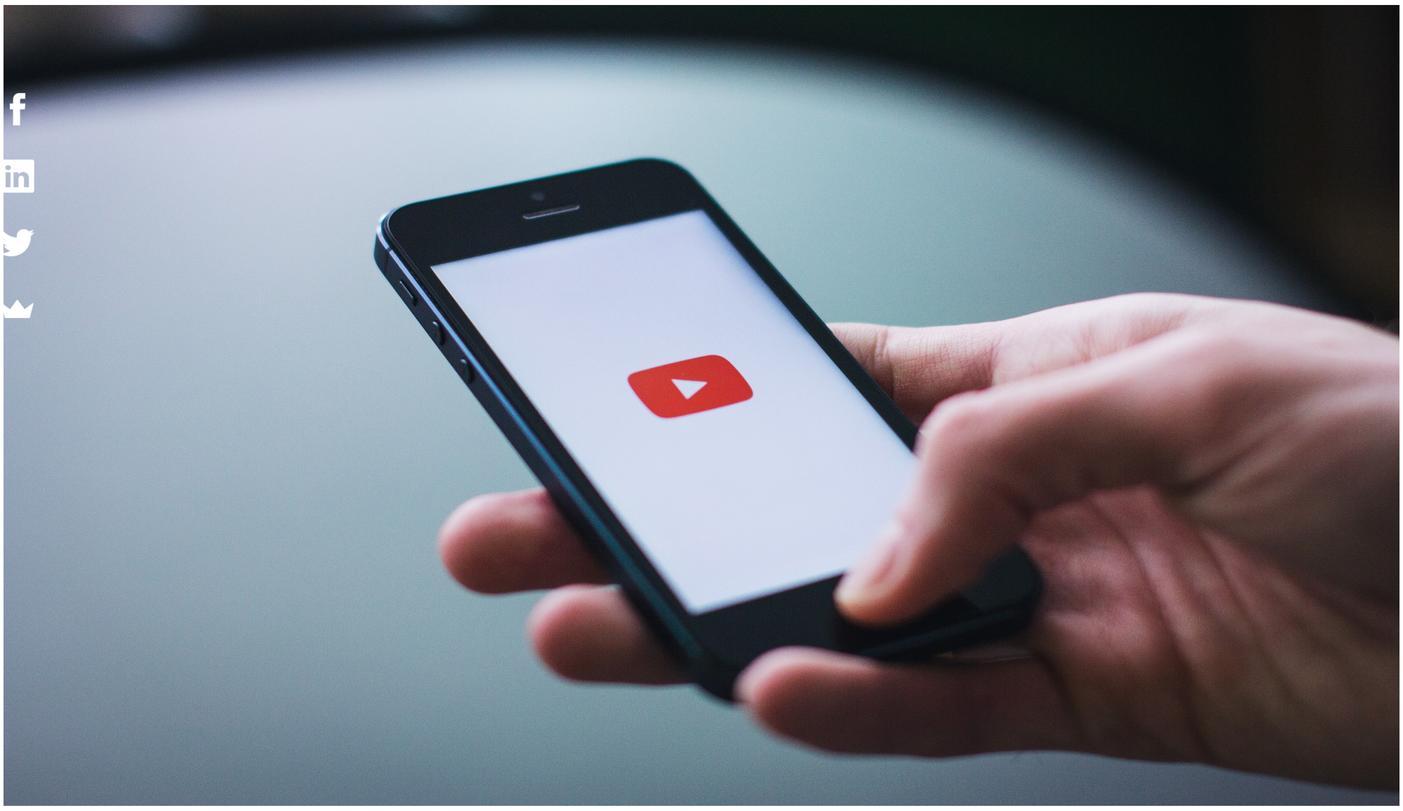

How to Design Youtube (Part II)

This is the second post about how to design Youtube. We'll continue our discussion from the first one (http://blog.gainlo.co/index.php/2016/10/22/design-youtube-part/) and please check it if you haven't read.

In the last post (http://blog.gainlo.co/index.php/2016/10/22/design-youtube-part/), we mainly talked about database and storage. This week, we'll cover much more topics, including scalability, web server, cache and security.

Scale the database

There are tons of problems to fix once the product has millions or even billions of users. Scalability is one of the most important issues to solve. Basically, storing all the data into a single database is not only inefficient but infeasible. So how would you scale the database for Youtube?

We can follow a lot of general rules when scaling the database. The most common approach is to scale only when you need it. In other words, it's not recommended to do all the work like partition your database at day one, because it's almost for sure that at the point you really need to scale, the whole infrastructure and product have been changed dramatically.

So the idea is to start from a single server. Later on, you may go to a single master with multiple read slaves (master/slave model (https://en.wikipedia.org/wiki/Master/slave_(technology))). And at some point, you'll have to partition the database and settle on a sharding approach. For instance, you can split the database by users' location and when a request comes, you'll route the request to the corresponding database.

For Youtube, we can further optimize it. The most important feature of Youtube is the video. Therefore, we can prioritize traffic by splitting the data into two clusters: a video cluster and a general cluster. We can give a lot of resources to the video cluster and other social network features will be routed to the less capable cluster. A more general idea here is that when solving scalability issue, you should first identify the bottleneck and then optimize it. In this case, the bottleneck is watching videos.

Cache

I will not talk much about cache for this topic as we've covered this in details in our previous post – Design a Cache System (http://blog.gainlo.co/index.php/2016/05/17/design-a-cache-system/). But several points are worth to mention here.

First of all, when talking about cache, most people's reaction is about server cache. In fact, front end cache is equally important. If you want to make your website fast and has low latency, you can't avoid setting cache for the front end. This is a very common technique when building a website interface.

Secondly, as we briefly discussed in the previous post, caching won't do a lot of good in terms of serving videos. This is mainly because majority usage of Youtube comes from those long tail videos and it'll be extremely expensive to set cache for all videos. So the general idea here is that if you are building a long tail product like this, don't place too much bet on the cache.

Security

There are a lot of things that can be discussed security in Youtube. I'd like to cover one interesting topic here – view hacking. Under each Youtube video, it shows the view count, which indicates how popular the video is. People can programmatically send requests to hack the view count, so how should we protect it?

The most straightforward approach is to if a particular IP issues too many requests, just block it. Or we can even restrict the number of view count per IP. The system can also check information like browser agent and user's past history, which potentially can block a lot of hacks.

People may use services like Tor (https://en.wikipedia.org/wiki/Tor_(anonymity_network)) to hide IP, and sites like Mechanical Turk (https://www.mturk.com/mturk/welcome) allows you to pay people to click the video with very low cost. However, hacking the system is harder than most people think.

For instance, a video with high view count but low engagement is very suspicious. With a large number of video Youtube has, it's not hard to extract patterns of real view count. In order to hack the system, you need to provide reasonable engagement metrics like share count, comment count, view time, etc.. And it's almost impossible to fake all of them.

Web server

Many people overlook web server as it doesn't have too many things to discuss in terms of system design, However, for large systems like Youtube, there are many things you need to consider. I'd like to share a couple techniques Youtube has used.

- Youtube server was built in Python initially, which allows rapid flexible development and deployment. You might notice that many startups choose Python as their server language as it's much faster to iterate.
- Python sometimes has the performance issue, but there are many C extensions that allow you to optimize critical section, which is exactly how Youtube works.
- To scale the web server, you can simply have multiple replicas and build a load-balancer on top of them.
- The server is mainly responsible for handling user requests and return response. It should have few heavy logics and everything else should be built into separate servers. For instance, recommendation should be a separate component to let Python server fetches data from.

Summary

In this post, we try to cover as many different topics as possible and each of them can be discussed deeper in a separate post. There are still many things I'd like to talk about in the future, but readers can think about them and discuss in the comments.

For instance, Youtube recommendation is a very big topic and it drives user engagement metrics dramatically. How would you build the recommendation system? In addition, how do you identify trending videos of the day and recommend to the relevant audience?

By the way, if you want to have more guidance from experienced interviewers, you can check Gainlo (http://www.gainlo.co/?utm_source=blog&utm_campaign=blog&utm_medium=How%20to%20Design%20Youtube%20(Part%20II)) that allows you to have mock interview with engineers from Google, Facebook ,etc..

I'd like to learn more (http://www.gainlo.co/?utm_source=blog&utm_medium=footer http%3A//blog.gainlo.co/index.php/2

(http://www.facebook.com/sharer.php?u=http://blog.gainlo.co/index.php/2016/11/04/design-youtube-part-ii/) 15 (http://twitter.com/share?url=http://blog.gainlo.co/index.php/2016/11/04/design-youtube-part-ii/&text=How+to+Design+Youtube+%28Part+II%29+) (http://www.linkedin.com/shareArticle?mini=true&url=http://blog.gainlo.co/index.php/2016/11/04/design-youtube-part-ii/) 3 (http://reddit.com/submit?url=http://blog.gainlo.co/index.php/2016/11/04/design-youtube-part-ii/&title=How to Design Youtube (Part II)) 0

Subscribe

We'll email you when there are new posts here.

Enter your email address here...

Subscribe

Related Posts:

1. **How to Design Youtube (Part I) (http://blog.gainlo.co/index.php/2016/10/22/design-youtube-part/)**
2. **Design a Recommendation System (http://blog.gainlo.co/index.php/2016/05/24/design-a-recommendation-system/)**
3. **Design a Cache System (http://blog.gainlo.co/index.php/2016/05/17/design-a-cache-system/)**
4. **Design Facebook Chat Function (http://blog.gainlo.co/index.php/2016/04/19/design-facebook-chat-function/)**

LEAVE A REPLY

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

Comment

Name *

Email *

Website

Post Comment

How to Design Youtube (Part I) (http://blog.gainlo.co/index.php/2016/10/22/design-youtube-part/)

Uber Interview Questions: Weighted Random Numbers (http://blog.gainlo.co/index.php/2016/11/11/uber-interview-question-weighted-random-numbers/)

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/)

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

ebook 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/)

Gainlo - Mock Interview With Professionals

Visit Gainlo (http://www.gainlo.co/?utm_source=blog&utm_medium=site-footer http%3A//blog.gainlo.co/index