YouTube/Netflix

First talks about YouTube, then about Netflix.

Services:

1. Users should be able to upload videos

2. Users should be able to view/share other videos.

3. Users can search videos.

4. Keep record of likes and dislikes for each video.

5. User should be able to commit on some video.

Other services:

1. System should be reliable.

2. System should be consistent.

3. System should be time critical and not make the users feel too much lag.

Scenario:

Assume total 1.5 billion users, 500 million DAUs. On average, everyday DAU watches 10 videos. QPS: 500M\*10/86400 = 600k

This is a read QPS. As to write, roughly every video has 1:200 rate, so around 600k/200 = 3k

Storage estimation: every minute upload around 500 hours of video, every video one minute 50M, so 500\*60\*50 = 1.5TB

every second 25 GB.

Bandwidth estimation: assume each video upload bandwidth 10 MB/s, total bandwidth = 500\*60\*10 = 300 GB/min, 5GB/second.

So reading bandwidth = 1TB/second.

Related APIs:

1. upload video: upload (device\_key, video title, video description, tags, language, creation time, category, video contents, location)

2. search video: search (device\_key, search\_query, user\_location, maximum videos to return, page token)

Storage:

1. Processing queue
2. Encoder
3. Thumbnail generator
4. Video/thumbnail storage (HDFS)
5. User database
6. Video metadata storage

Related DB:

1. User table: user id, email, password, address, age, creation time
2. Video table: video id, owner id, thumbnail, title, description, size, number of likes, number of dislikes, number of views
3. Comment table: comment id, owner id, creation time, content, video id.

Details:

1. App server: master-slave model for write intensive workload, master is in charge of assigning the task to slaves, slave has a captain then finish the writing. Read is the same.
2. Bigtable can help store the thumbnail. Thumbnail can have a maximum size requirement, all the thumbnail from one video can be allocated into one chuck in Bigtable. We can cache them as well in the web server.

Scale:

1. Metadata sharding: shard based on what? If based on user\_id, some popular videos traffic is so high that block the network. It is hard to maintain a uniform distribution.
2. Based on consistent hashing, namely video\_id to random assign a server to store the video’s meatadata (not the video itself). The centrailized server will aggregate all the query returned results back to the user. Also cache the hot videos in the web server.
3. For video, because some videos may have exactly the same contents, so no need to store all of them. Run some video matching algorithm, choose the high-quality ones for the one in FS. Only stores the part that missing from previous ones.
4. Dynamic http redirections for the case where a busy server in one location redirect to less busy server. But this can cause extra redirections if the destination does not have the videos.
5. Memcache is enough to cache the data in web server, LRU is used for a reasonable cache eviction policy.
6. A new content delivery network (CDN) is good to replicate content in multiple places.