Instagram

Service: distributed server with load balancer; new feeds cache for each user, showing the recent ones; special design for users with large amount of followers

Push notification can be handled by iPhone/Android platform.

Different from twitter: posting images, so having a load balancer routing to an available server. Uploading to server needs to care 2 things: storing the images or short

videos to file system and posting the locations to DB, file system is HDFS. The storage will somehow update new feed table with a small image and update the new feed table.

class User {

int id;

string name;

date created\_time;

list<Photo> photos;

list<User> followers;

}

class Photo {

int id;

string description;

HashTag hashtags;

string location;

string GPS;

long size;

}

DB: NoSQL for user table and following table, and photo table.

FS: HDFS. May limit the upload size under 5MB.

This is a read-intensive system. Efficient management of storage is very important, users can upload as many as they like. No photo should be lost.

When it comes to storage, always separate into file storage and metadata storage. This will be faster than storage them together. Instagram is read-intensive system. So keeping metadata for fast search is important.

Cassandra is important for storing database like relationship.

If read and write are both busy, separate them into different servers.

We cannot allow any data lost. So, in this case, data replica is key. For both file systems to store the data and the metadata servers.

Timeline creation is a combo of epoch time and a global increasing id, this global increasing id is reset by every second.

After sharding, the photo is determined by shard id plus a shard-based increment id managed by this shard. As long as following relationship, sharding based on user id is never a good opinion, like JB, Lady Gaga.

For a global increment system, having 2 SQL auto increment, but with different offset is a good way. This can be done to the same tables like user table, comment table (YouTube comment as well).