Designing Pastebin

Let's design a Pastebin like web service, where users can store plain text. Users of the service will enter a piece of text and get a randomly generated URL to access it.

Similar Services: pasted.co, hastebin.com, chopapp.com
Difficulty Level: Easy

1. Why Instagram?

Instagram is a social networking service, which enables its users to upload and share their pictures and videos with other users. Users can share either publicly or privately, as well as through a number of other social networking platforms, such as Facebook, Twitter, Flickr, and Tumblr.

For the sake of this exercise, we plan to design a simpler version of Instagram, where a user can share photos and can also

2. Requirements and Goals of the System

We will focus on the following set of requirements while designing Instagram:

Functional Requirements

1. Users should be able to upload/download/view photos.
2. Users can perform searches based on photo/video titles.
3. Users can follow other users.
4. The system should be able to generate and display a user's timeline consisting of top photos from all the people the user follows.

Non-functional Requirements

1. Our service needs to be highly available.
2. The acceptable latency of the system is 200ms for timeline generation.
3. Consistency can take a hit (in the interest of availability), if a user doesn't see a photo for a while, it should be fine.
4. The system should be highly reliable, any photo/video uploaded should not be lost.

3. Some Design Considerations

The system would be read-heavy, so we will focus on building a system that can retrieve photos quickly.

1. Practically users can upload as many photos as they like. Efficient management of storage should be a crucial factor while designing this system.
2. Low latency is expected while reading images.
3. Data should be 100% reliable. If a user uploads an image, the system will guarantee that it will never be lost.

4. Capacity Estimation and Constraints

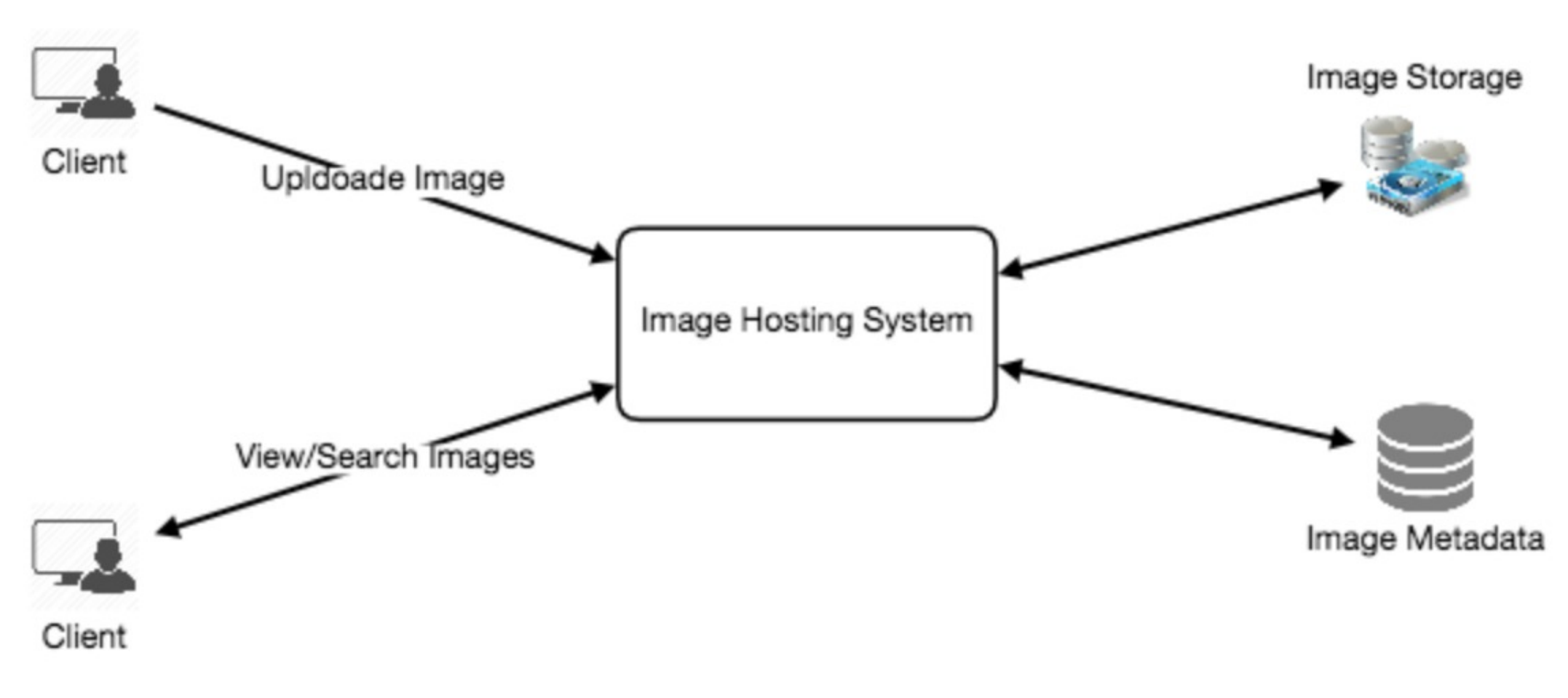
- Let's assume we have 300M total users, with 1M daily active users.
- 2M new photos every day, 23 new photos every second.
- Average photo file size => 200KB
- Total space required for 1 day of photos

2M * 200KB => 400 GB

- Total space required for 5 years:

5. High Level System Design

At a high-level, we need to support two scenarios, one to upload photos and the other to view/search photos. Our service would need some block storage servers to store photos and also some database servers to store metadata information about the photos.



Defining the DB schema in the early stages of the interview would help to understand the data flow among various components and later would guide towards the data partitioning.

We need to store data about users, their uploaded photos, and people they follow. Photo table will store all data related to a photo, we need to have an index on (PhotoID, CreationDate) since we need to fetch recent photos first.

One simple approach for storing the above schema would be to use an RDBMS like MySQL since we require joins. But relational databases come with their challenges, especially when we need to scale them. For details, please take a look at SQL vs. NoSQL.

We can store photos in a distributed file storage like HDFS or S3.

We can store the above schema in a distributed key-value store to enjoy benefits offered by NoSQL. All the metadata related to photos can go to a table, where the 'key' would be the 'PhotoID' and the 'value' would be an object containing PhotoLocation, UserLocation, CreationTimestamp, etc.

would need to store is the list of people a user follows. For both of these tables, we can use a wide-column datastore like Cassandra. For the 'UserPhoto' table, the 'key' would be 'UserID' and the 'value' would be the list of 'PhotoIDs' the user owns, stored in different columns. We will have a similar scheme for the 'UserFollow' table.

Cassandra or key-value stores in general, always maintain a certain number of replicas to offer reliability. Also, in such data stores, deletes don't get applied instantly, data is retained for certain days (to support undeleting) before getting removed from the system permanently.

7. Component Design

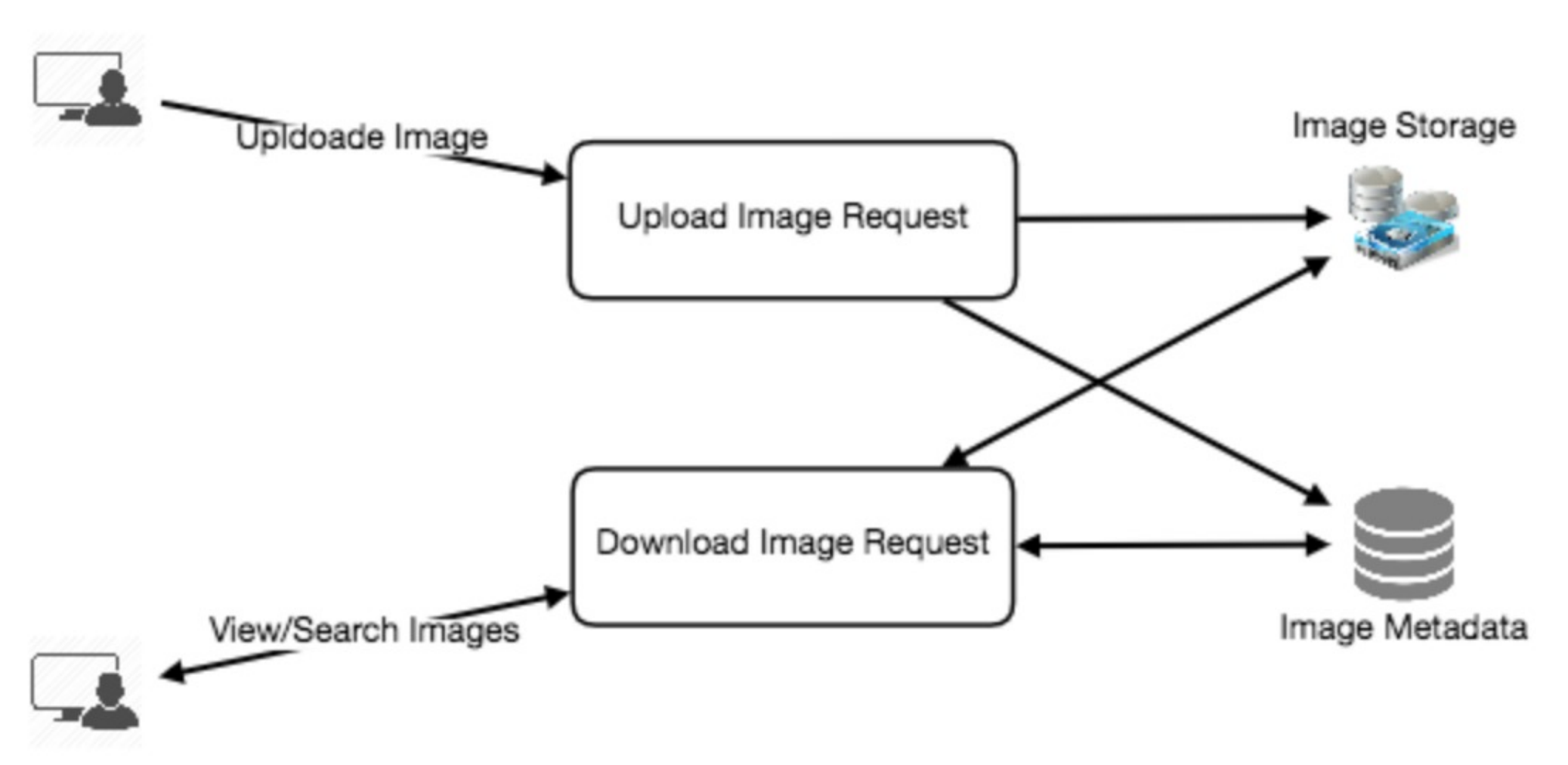
Writes or photo uploads could be slow as they have to go to the disk, whereas reads could be faster if they are being served from cache.

Uploading users can consume all the connections, as uploading would be a slower process. This means reads cannot be served if the system gets busy with all the write requests. To handle this bottleneck we can split out read and writes into separate services.

Since most of the web servers have connection limit, we should keep this thing in mind before designing our system. Synchronous connection for uploads, but downloads can be asynchronous. Let's assume if a web server can have maximum

asynchronous, the web server can serve a lot more than 500 users at any time, as it can switch between users quickly. This guides us to have separate dedicated servers for reads and writes so that uploads don't hog the system.

Separating image read and write requests will also allow us to scale or optimize each of them independently.



8. Reliability and Redundancy

dies, we can retrieve the image from the other copy present on a different storage server.

This same principle also applies to other components of the system. If we want to have high availability of the system, we need to have multiple replicas of services running in the system. So that if a few services die down, the system is still available and serving. Redundancy removes the single point of failures in the system.

Creating redundancy in a system can remove single points of failure and provide a backup or spare functionality if needed in a crisis. For example, if there are two instances of the same service running in production, and one fails or degrades, the system can failover to the healthy copy. Failover can happen automatically or require manual intervention.

✓ Completed Reset

← Previous
Designing a URL Shortening service ...

Next →
Designing Instagram

Send feedback or ask a question 12 recommendations