wuyichen24 Update Build_Photo_Sharing_System.md History

1 contributor

Build Photo Sharing System

Real-life examples

- Instagram
- Flickr
- Picasa

Requirements clarification

- **Functional requirements**
  - Upload: Users can upload photos.
  - View: Users can view photos.
  - Search: Users can search photos.
  - Follow: Users can follow other users.
  - News feed: Users can see the top/new photos from the users they follow.
- **Non-functional requirements**
  - High availability
  - High reliability (Any photo uploaded should not be lost).

- High consistency is desirable (It should be ok for a user doesn't see a photo for a while).
- Low latency is expected while viewing photos.

Estimation

- **Traffic estimation**
  - Our system will be read-heavy.
  - Users
    - 500 million users. (Assumed)
    - 1 million daily active users. (Assumed)
  - Photos
    - 2 million new photos every day (Assumed)
    - Average photo file size = 200 KB (Assumed)
- **Storage estimation**
  - Types
    - Data: Yes
    - File: Yes
  - Capacity
    - Total capacity needed every day = Number of new photos every day x Average photo file size = 2 million x 200 KB = 400 GB
- **Bandwidth estimation**

System interface definition

Data model definition

- **Schema**
  - Table 1: User
    - Description
      - Store user accounts.
    - Columns

Column Name	Column Type	PK	Description
UserId	int	PK	The user ID.
Name	string		The name of the user.

Column Name	Column Type	PK	Description
Email	string		The email of the user.
Location	string		The location of the user.
LastLogin	datetime		The last login time of the user.

    - Table 2: Photo
      - Description
        - Store the photo information.
      - Columns

Column Name	Column Type	PK	Description
PhotoId	int	PK	The photo ID.
UserId	int		The owner's user ID of the photo.
Description	string		The description of the photo.
Location	string		The location of the photo was taken.
Path	string		The URL to access the photo in distributed file system.

    - Table 3: UserFollow

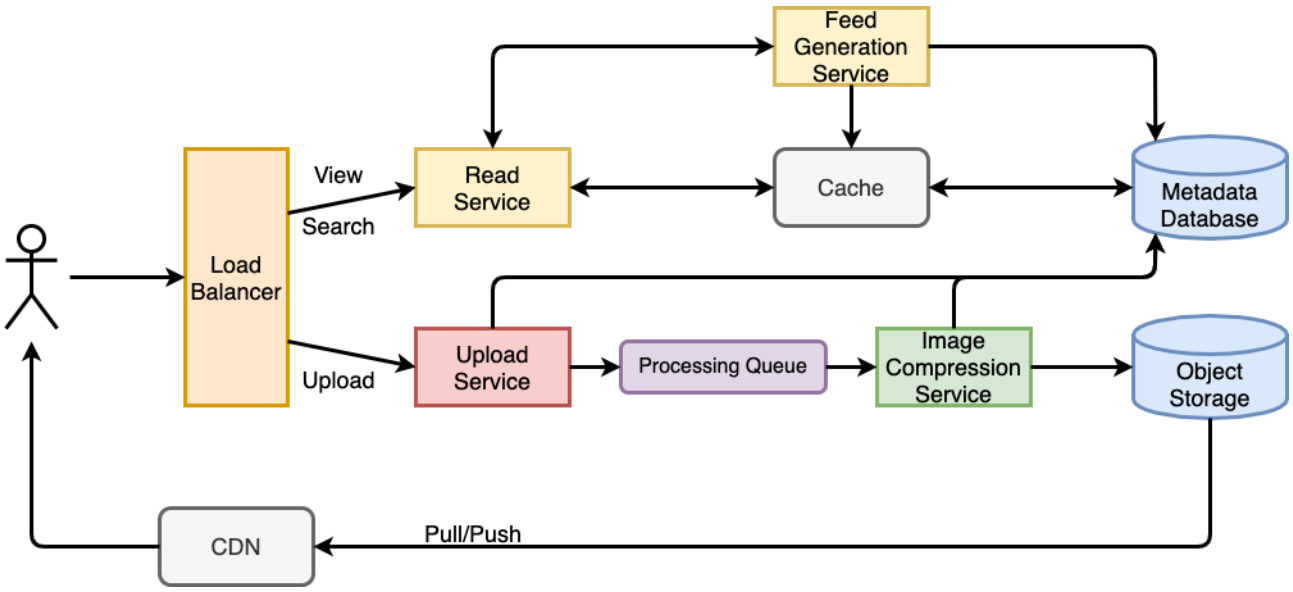
      - Columns

Column Name	Column Type	PK	Description
FollowerUserId	int		The follower user ID.
FolloweeUserId	int		The user ID who has been followed.

- Data storage
  - Database
    - SQL database (We need to do join operations).
  - File storage
    - HDFS

- Amazon S3
- GlusterFS

High-level design



- **Upload Service**
  - Handle photo upload requests.
  - Create a image compression task and push it into the processing queue.
- **Processing Queue**
  - Store all the compression tasks.
  - Decouple uploading works and compression works
  - It can act as a buffer if the image compression service is unavailable or overloaded.
- **Image Compression Service**
  - Compress images from multiple quality to a single quality for storing and transferring.
- **Read Service**
  - Handle photo view and search requests.
- **Feed Generation Service**
  - Generate news feeds based on user following relationship.
- **Metadata Database**
  - Store photo, users and user following relationship information.

Detailed design

- **Feed Generation Service**
  - Consideration 1: Generate news feed for users
    - Process
      - Get a list of people the user follows and then fetch metadata info of each user's latest 100 photos.
      - Submit all these photos to our ranking algorithm, which will determine the top 100 photos and return them to the user.
    - Problem
      - The process of generating news feed will be slow.
    - Solution
      - Pre-generate the news feed and store it in a separate table.

Key points

- Images need to be compressed.
- Use queue to decouple upload works with image compression works.
- Segregate uploads and reads into 2 different types of servers.

References

- https://www.educative.io/courses/grokking-the-system-design-interview/m2yDVZnQ8lG
- https://www.youtube.com/watch?v=VJpfO6KdyWE&ab_channel=Exponent