wuyichen24 Update Build_Video_Distribution_System.md History

1 contributor

Build Video Distribution System

Real-life examples

- Youtube
- Netflix
- Vimeo

Requirements clarification

- **Functional requirements**
  - Upload video: Users can upload videos.
  - Watch video: Users can watch videos.
  - Search video: Users can search videos.
  - Comment video: Users can leave comments to videos, also like or dislike.
- **Non-functional requirements**
  - High reliability (Any video uploaded should not be lost).
  - High availability.
  - High consistency is desirable (It should be ok for a user doesn't see a video for a while).

Estimation

- **Traffic estimation**
  - Our system will be read-heavy.
  - Read-write ratio (View-upload ratio) is 200 : 1 (Assumed)
  - Users
    - 1.5 billion users. (Assumed)
    - 150 million daily active users. (Assumed)
    - 1% of users are creators, every week will publish one new video. (Assumed)
    - Each user watches 3 videos per day. (Assumed)
  - Number of read actions and write actions per week
    - Number of writes (upload) per week = 1.5 billion x 1% = 15 million
    - Number of reads (watch) per week = 15 millions x 200 = 3 billion
  - Frequency of read actions and write actions per second (QPS)
    - Frequency of writes per second = 15 millions / (7 days x 24 hours x 3600 seconds) = 24 videos/s
    - Frequency of reads per second = 24 videos/s x 200 = 4800 videos/s
- **Storage estimation**
  - Types
    - Data: Yes
    - File: Yes
  - Capacity
    - Size of each video: 500 MB (Assumed)
    - Total capacity needed in week = Number of writes (upload) per week x Size of one record = 15 million x 500 MB = 7152 TB
- **Bandwidth estimation**
  - Size of each video: 500 MB (Assumed)
  - Write bandwidth = Frequency of writes per second x Size of one record = 24 videos/s x 500 MB = 11 GB/s

videos/s x 6 MB/s (1080p) = 28 GB/s

System interface definition

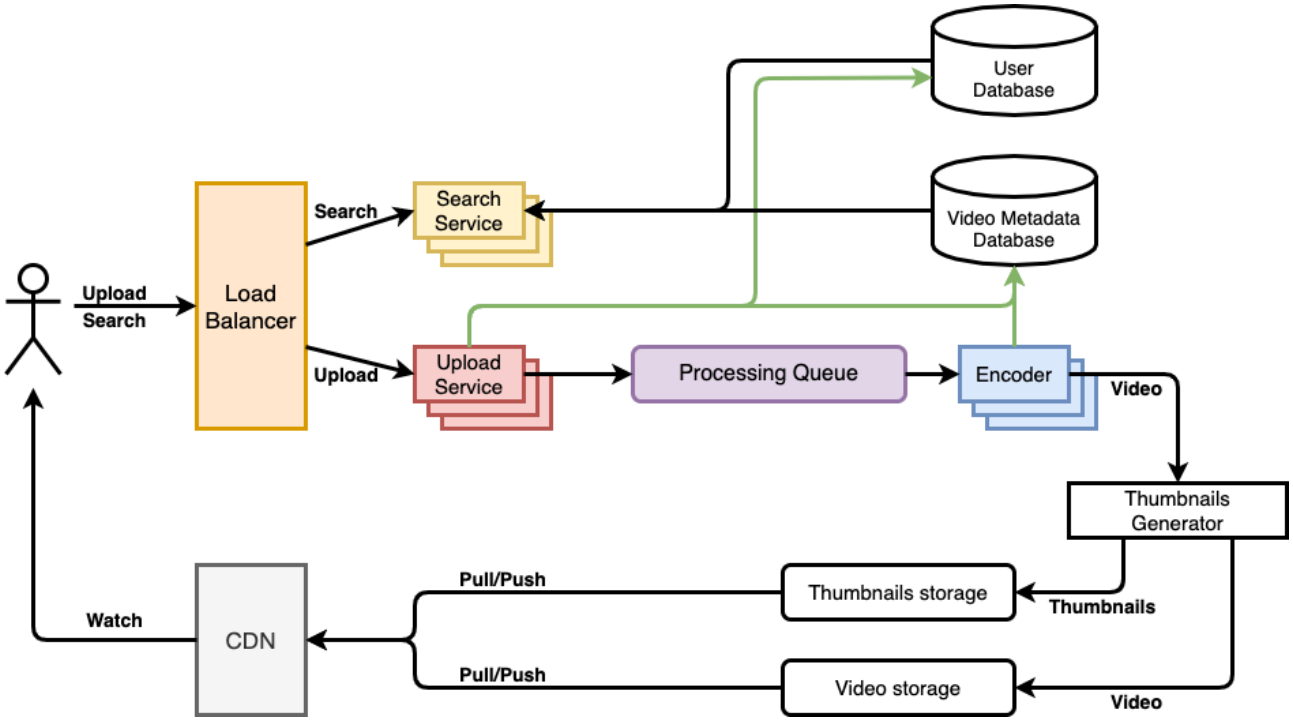
- **Interface 1**
  - `uploadVideo(api_key, video_title, video_description, video_content)`
    - Function

        - Upload a video
    - Parameters
        - api_key (string): The API developer key of a registered account.
        - video_title (string): The title of the video.
        - video_description (string): The description of the video.
        - video_content (stream): The content stream of the video.
- **Interface 2**
    - `streamVideo(api_key, video_id, offset, codec, resolution)`
        - Function
            - Watch a video
        - Parameters
            - api_key (string): The API developer key of a registered account.
            - video_id (string): The ID of the video.
            - offset (number): A playing time in seconds from the beginning of the video.
            - codec: The encoding format of the video.
            - resolution: The resolution of the video.

Data model definition

- **Schema**
    - Table 1: VideoMetadata
    - Table 2: Comment
    - Table 3: User
- **Data storage**
    - Database
    - File storage (to store video and thumbnails)
        - HDFS
        - GlusterFS
        - Amazon S3

High-level design



- **Upload Service**
  - Handle upload requests
  - Create a encoding task and push it into the processing queue.
- **Processing Queue**
  - Store all the encoding tasks.
  - Decouple uploading works and encoding works
  - It can act as a buffer if the encoder is unavailable or overloaded.
- **Encoder**
  - Encode each uploaded video into multiple formats.
- **Thumbnails generator**
  - Generate a few thumbnails for each video.
- **Video Storage**
  - Store video contents.
- **Thumbnails Storage**
  - Store thumbnails.
- **Video Metadata Database**
  - Store all the information about videos like title, file path in the system, uploading user, total views, likes, dislikes, comments.
- **User Database**
  - Store user's information.

Key points

- Use queue to decouple upload works with encoding works.
- Use proper storage for storing videos and thumbnails.
    - Object storages
    - CDN
- Read traffic for thumbnails will be huge compared to videos
    - Users will be watching one video at a time, but they might be looking at a page with 20 thumbnails of other videos.