**Design a video streaming service like Youtube/Netflix where users can upload/view/search videos. The service should be scalable where a large number of users can watch and share the videos simultaneously. It will be storing and transmitting petabytes and petabytes of data.**

**Functional Requirements:**

1. User should be able to upload the video.
2. User should be able to search the video.
3. User should be able to view the video.

**Optional Requirements:**

1. User can like/dislike the video.
2. User should be able to see the total views on the video
3. User should be able to add and view the comment.

**Non-functional Requirements:**

1. Low latency
2. High availability. Here consistency is of low importance because it is acceptable if newly added videos is not available to some user for a while.
3. System should be reliable.
4. System should be scalable.

**Estimation:**

Total users = 1Billion

Daily active users : 5 million

Daily video upload : 1 miilion

Avg 1 video size : 100MB

Total storage required : 100\*1million = 100TB(approx)

It is a ready heavy system

**Data Design Model**

**Tables**

**User**(UserId, Username, Email, createdAt)

**Videos**(VideoId, title, description, userId(Foriegn Key), upload date, duration, likeCount, dislikeCount)

**Comments**(CommentId, videoId, UserId, commentText, createdAt)

**Like**(likeId, userId, videoId,createdAt)

**Dislike**(dislikeId, userId, videoId, createdAt)

**playlist**(playlistId, userId, playlistName, createdDate)

**API Desgin**

**uploadVideo**(videoId, title, description, userId)

**searchVideo**(videoId, title, description))

**viewVideo**(videoId, title, description)

**addComment(**videoId, commentText)

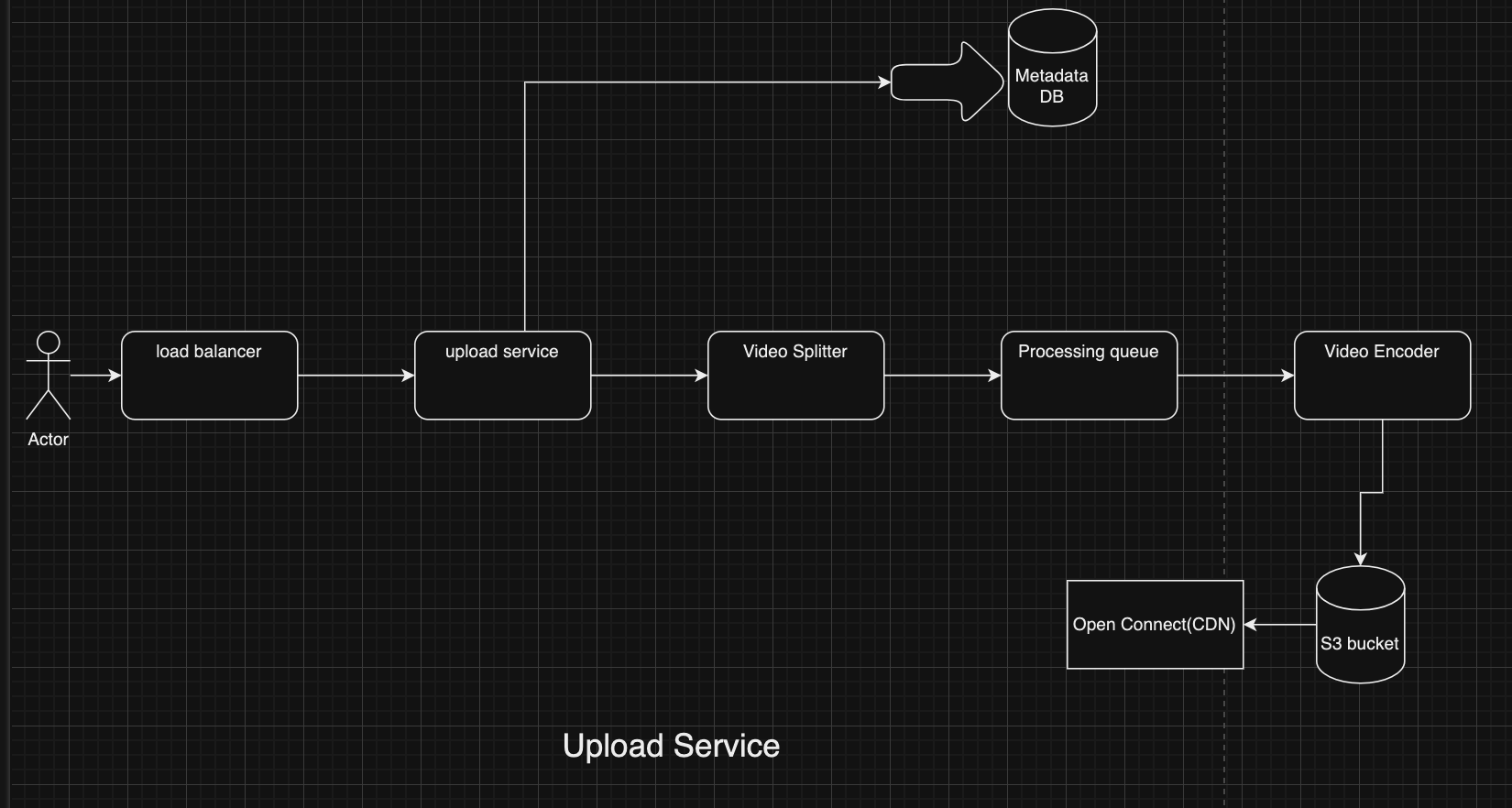
**viewComment**(videoId, commentPage)

**High Level Design**

3 basic types of request that can be made to video streaming application.

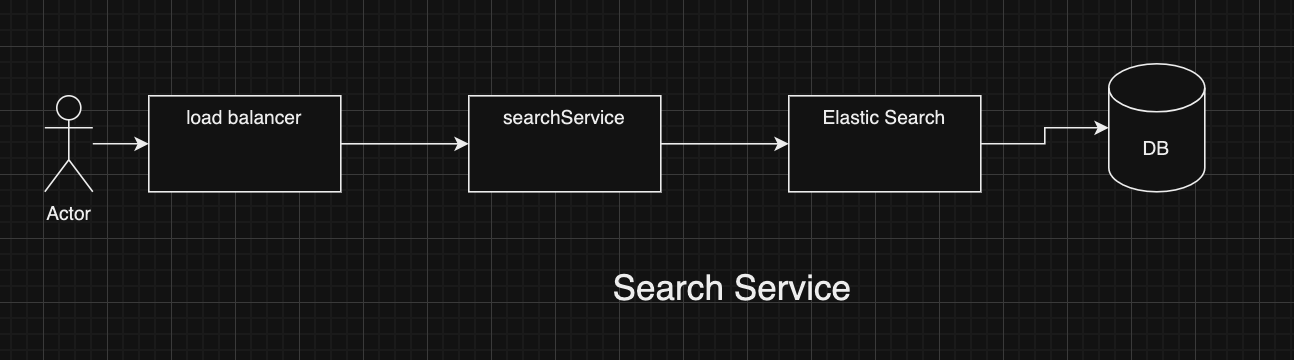
1. upload(write)
2. search(read)
3. view(read)

**Upload service**



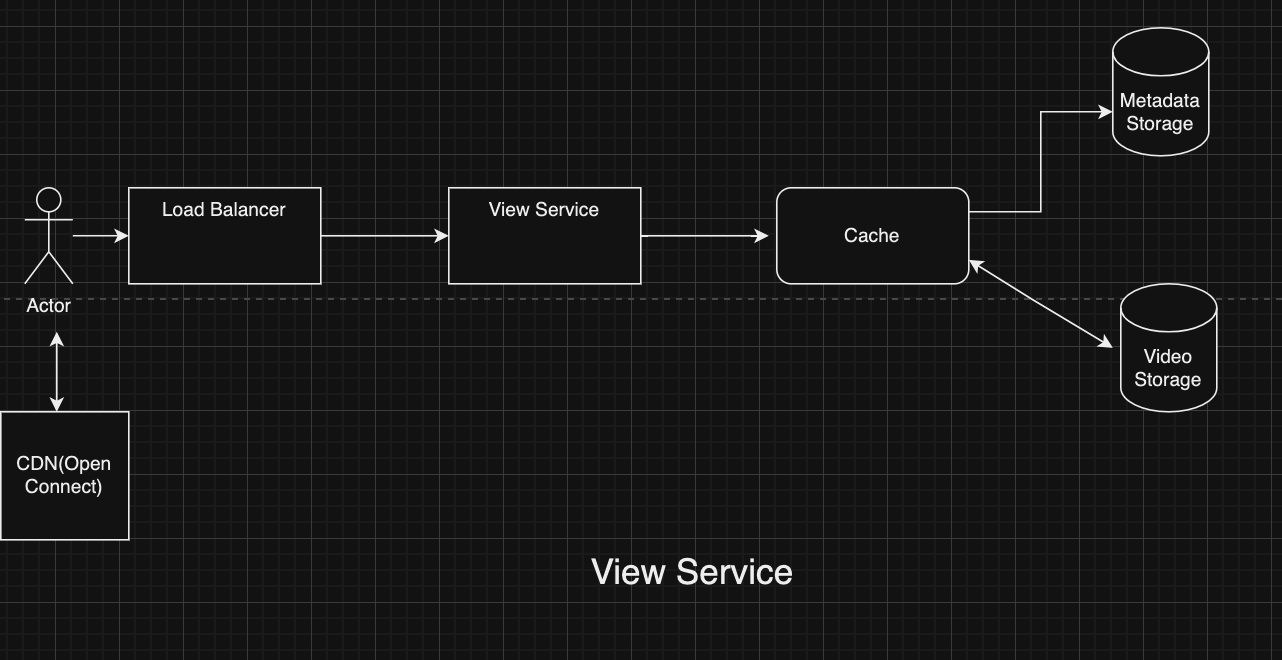
An upload request is served by the upload service that processes the video, uploads it to Open Connect servers and makes it available to all the users.

**Search Service**

****

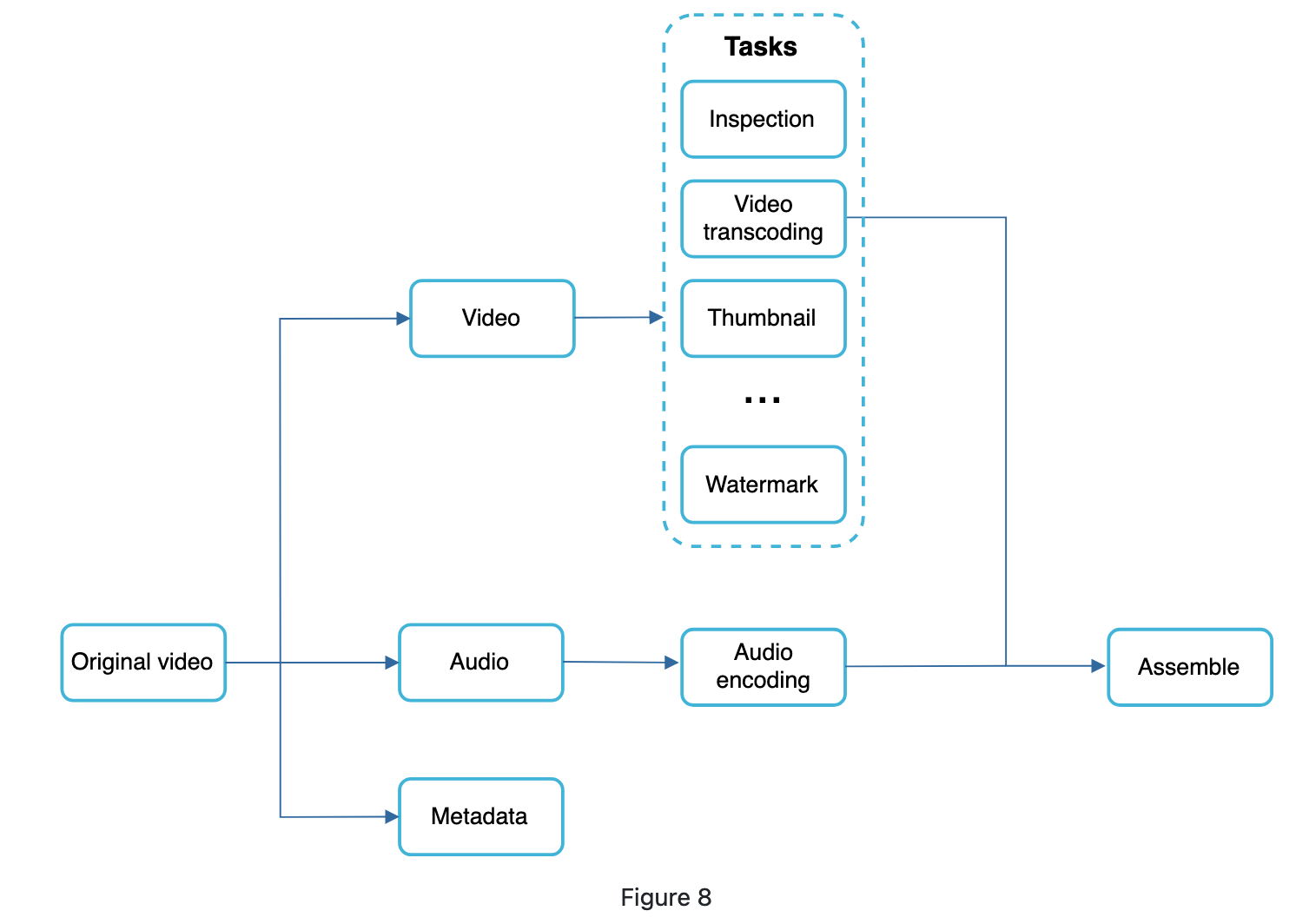
Search request is forwarded by the load balancer to the search microservice and onwards to Netflix’s Elastic search. The response of the Elastic search is returned to the client. Elastic search is a highly scalable full-text open-source search engine that handles searching for millions of videos for Netflix. Netflix also relies on Elastic search for analyzing customer services operations.

**View Service**

****

Most of the view requests will not be directed to the load balancer and Netflix’s servers. Instead, they’ll reach the local ISPs and will be served from the nearest Open Connect server directly. If the requested video isn’t available, however, it will be forwarded to the load balancer and the view microservice. From there, the video is searched in the metadata database and fetched from the path stored in the metadata and sent to the client. Of course, this approach involves delays, which is why any view request is almost always served via Open Connect.

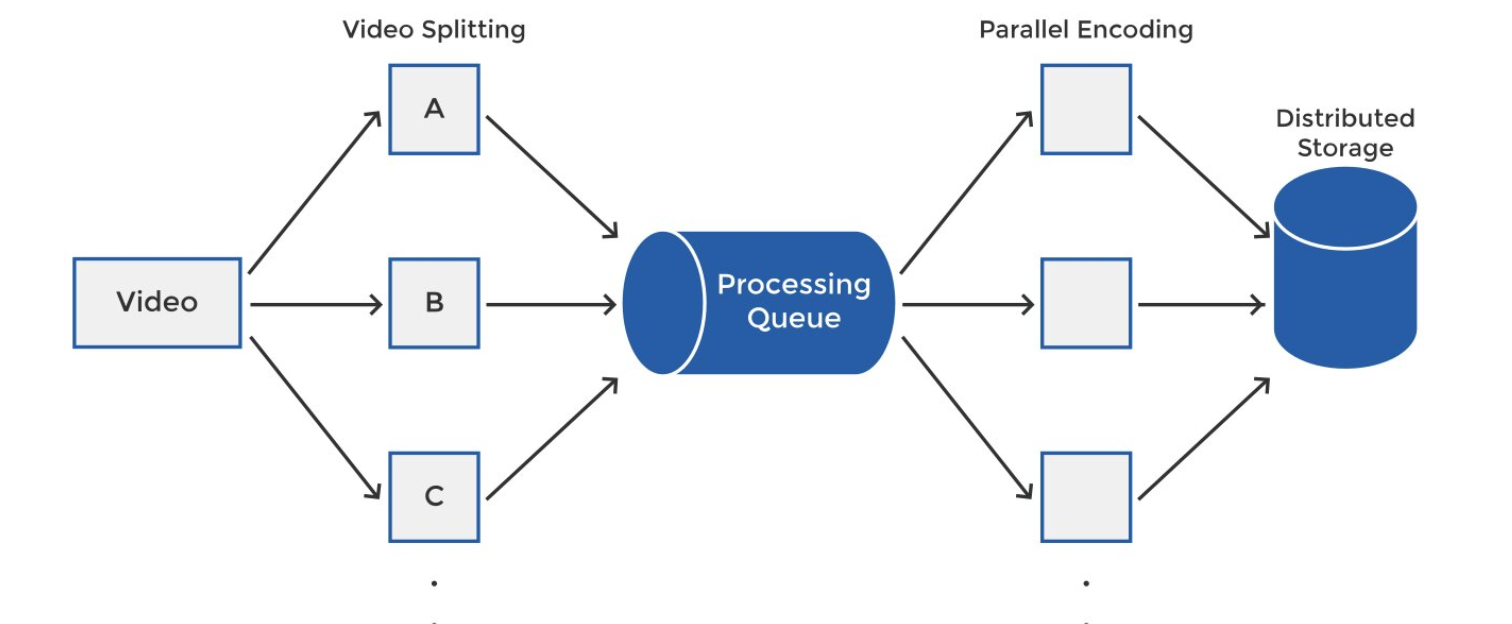
**Video Transcoding**

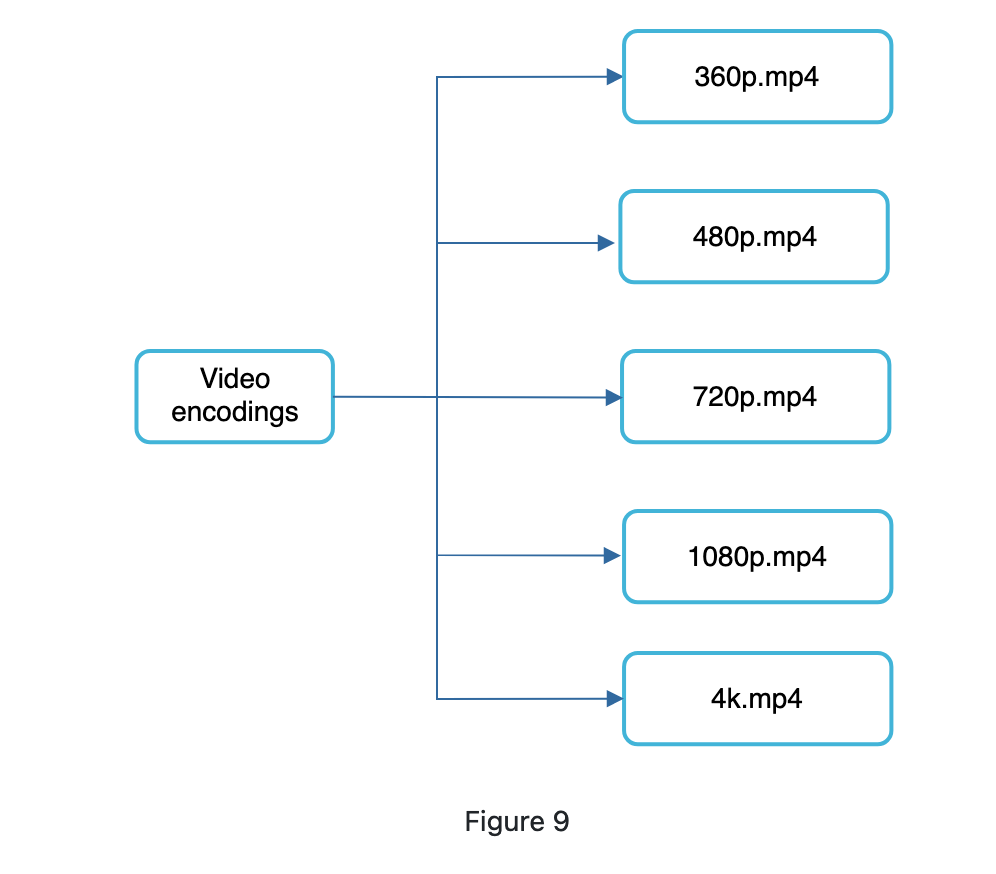
****

The original video is split into video, audio, and metadata. Here are some of the tasks that can be applied on a video file.

Instead of being stored as a single large file, each uploaded video will be stored in several chunks. This is necessary because a content creator can upload a large video. Processing or streaming a single heavy file can be time-consuming. When the video is stored and available to the viewer in chunks, the viewer will not need to download the entire video before playing it. It will request the first chunk from the server, and while that chunk is playing, the client requests for the next chunk so there’s minimum lag between the chunks and the user can have a seamless experience while watching the video.

Consider the splitting of videos as an independent microservice handled by the server, video splitter. It will divide the video into smaller chunks and put them in the processing queue. As they are de-queued, the chunks will be encoded.





**Processing Queue**

A processing queue is needed because there are several chunks for each video and Netflix will use several parallel workers to process them. This is made easier by pushing them into the queue. The workers (or the encoders which we will discuss next) will pick up the tasks from the processing queue, encode them into different formats and store them in the distributed file storage.

**Video Encoding**

An important step is to convert and store the video chunks into different formats, so different viewers can view it in the format that best suits their device and internet connection. Viewers may be watching the video through their laptop, phone, TV or some other device. Different devices have different formats that work best for them.

You will need to convert each video into different formats (mp4, avi, flv, etc), and different resolutions for each format (e.g. 1080p, 480p, 240p etc). If you have *i* number of formats to support and *j* number of resolutions, the system will generate *i\*j* number of videos for each video uploaded to the platform.

Netflix uses an intelligent solution to overcome the problem. It is called **Open Connect** (OC). Open Connect is Netflix’s customized CDN (Content Delivery Network).

**Reference**:

<https://medium.com/double-pointer/system-design-interview-video-streaming-service-e-g-netflix-or-youtube-design-adc2402e54a1>

https://bytebytego.com/courses/system-design-interview/design-youtube