Chapter 14 Design YouTube

Summary

- **举一反三**: Applicable to other interviews like designing a video sharing platform, such as Netflix, Hulu, Bilibili, Aiqiyi, etc
- **略有不同**: Short video sharing platforms, like Instagram, Tiktok, may need different considerations.

Fun facts of YouTube in 2020

- Total number of monthly active users: 2 billion.
- Number of videos watched per day: 5 billion.
- 73% of US adults use YouTube.
- 50 million creators on YouTube.
- YouTube's Ad revenue was $15.1 billion for the full year 2019, up 36% from 2018.
- YouTube is responsible for 37% of all mobile internet traffic.
- YouTube is available in 80 different languages.
  YouTube is enormous, global and makes a lot of money.

Step 1 problem and design scope

Features YouTube supports

watching a video, comment, share, or like a video, save a video to playlists, subscribe to a channel, etc

Narrow down the scope

• Ability to upload videos fast
• Smooth video streaming
• Ability to change video quality
• Low infrastructure cost
• High availability, scalability, and reliability requirements
• Clients supported: mobile apps, web browser, and smart TV

Candidate: What features are important?
Interviewer: Ability to upload a video and watch a video.

Candidate: What clients do we need to support?
Interviewer: Mobile apps, web browsers, and smart TV.

Candidate: How many daily active users do we have?
Interviewer: 5 million

Candidate: What is the average daily time spent on the product?
Interviewer: 30 minutes.

Candidate: Do we need to support international users?
Interviewer: Yes, a large percentage of users are international users.

Candidate: What are the supported video resolutions?
Interviewer: The system accepts most of the video resolutions and formats.

Candidate: Is encryption required?
Interviewer: Yes

Candidate: Any file size requirement for videos?
Interviewer: Our platform focuses on small and medium-sized videos. The maximum allowed video size is 1GB.

Candidate: Can we leverage some of the existing cloud infrastructures provided by Amazon, Google, or Microsoft?
Interviewer: That is a great question. Building everything from scratch is unrealistic for most companies, it is recommended to leverage some of the existing cloud services.

Back of the envelope estimation

Communicate w/ interviewers about assumptions.

- **Storage:**
  • Assume the product has 5M DAU.
  • Users watch 5 videos per day.
  • 10% of users upload 1 video per day.
  • Assume the average video size is 300 MB.
  • Total daily storage space needed: 5 million *10%* 300 MB = 150TB
- **CDN cost:**
  • When cloud CDN serves a video, you are charged for data transferred out of the CDN.
  • Amazon's CDN CloudFront for cost estimation (Figure 14-2) [3].
  Assume 100% of traffic is served from the United States.
  The average cost per GB is $0.02.
  For simplicity, we only calculate the cost of video streaming.
  • 7.5 PB (5 million *5 videos* 0.3GB) * $0.02 = $150,000 per day.

Per Month	United States & Canada	Europe & Israel	South Africa & Middle East	South America	Japan	Australia	Singapore, South Korea, Taiwan, Hong Kong, & Philippines	India
First 10TB	$0.085	$0.085	$0.110	$0.110	$0.114	$0.114	$0.140	$0.170
Next 40TB	$0.080	$0.080	$0.105	$0.105	$0.089	$0.098	$0.135	$0.130
Next 100TB	$0.060	$0.060	$0.090	$0.090	$0.086	$0.094	$0.120	$0.110
Next 350TB	$0.040	$0.040	$0.080	$0.080	$0.084	$0.092	$0.100	$0.100
Next 524TB	$0.030	$0.030	$0.060	$0.060	$0.080	$0.090	$0.080	$0.100
Next 4PB	$0.025	$0.025	$0.050	$0.050	$0.070	$0.085	$0.070	$0.100
Over 5PB	$0.020	$0.020	$0.040	$0.040	$0.060	$0.080	$0.060	$0.100

Figure 14-2 On-demand pricing for Data Transfer to the Internet (per GB).

Step 2 High level design

Leveraging existing cloud services: CDN and blob storage.
Blob storage: binary large objective. 是一种面向对象的存储，专注于存储非结构化数据, 比如文本、图像、视频、备份文件或日志等. 常见于云计算环境，例如 Microsoft Azure Blob Storage.
No need to build everything from scratch:

- System design interviews are not about building everything from scratch. Within limited time frame, choosing the right technology to do a job right is more important.
- Building scalable blob storage or CDN is extremely complex and costly.

High level system components

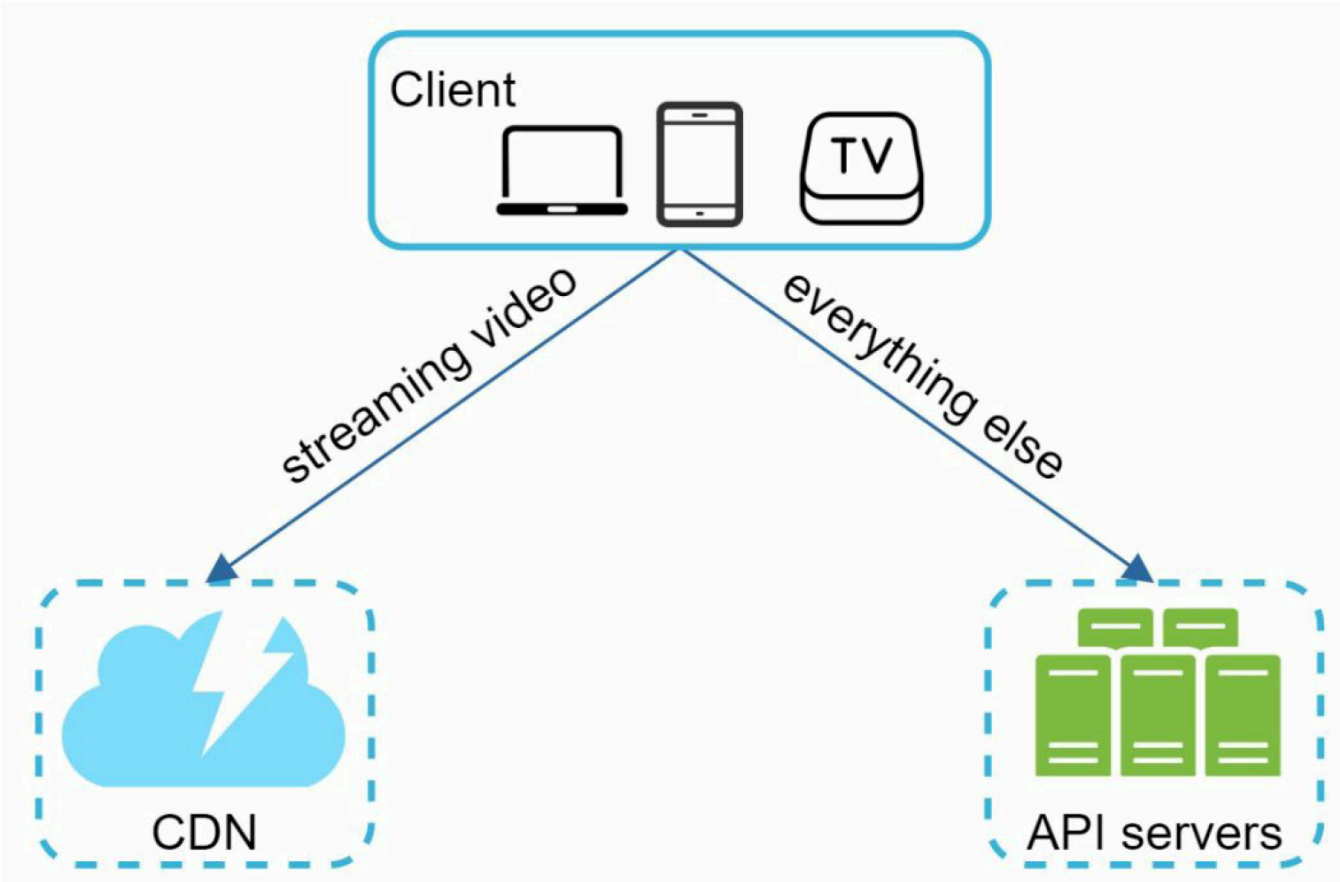

Figure 14-3

- **Client**: You can watch YouTube on your computer, mobile phone, and smartTV.
- **CDN**: Videos are stored in CDN. When you press play, a video is streamed from the CDN.
- **API servers**: Everything else except video streaming goes through API servers.
  - feed recommendation
  - generating video upload URL
  - updating metadata database and cache
  - user signup, etc.

Video uploading flow

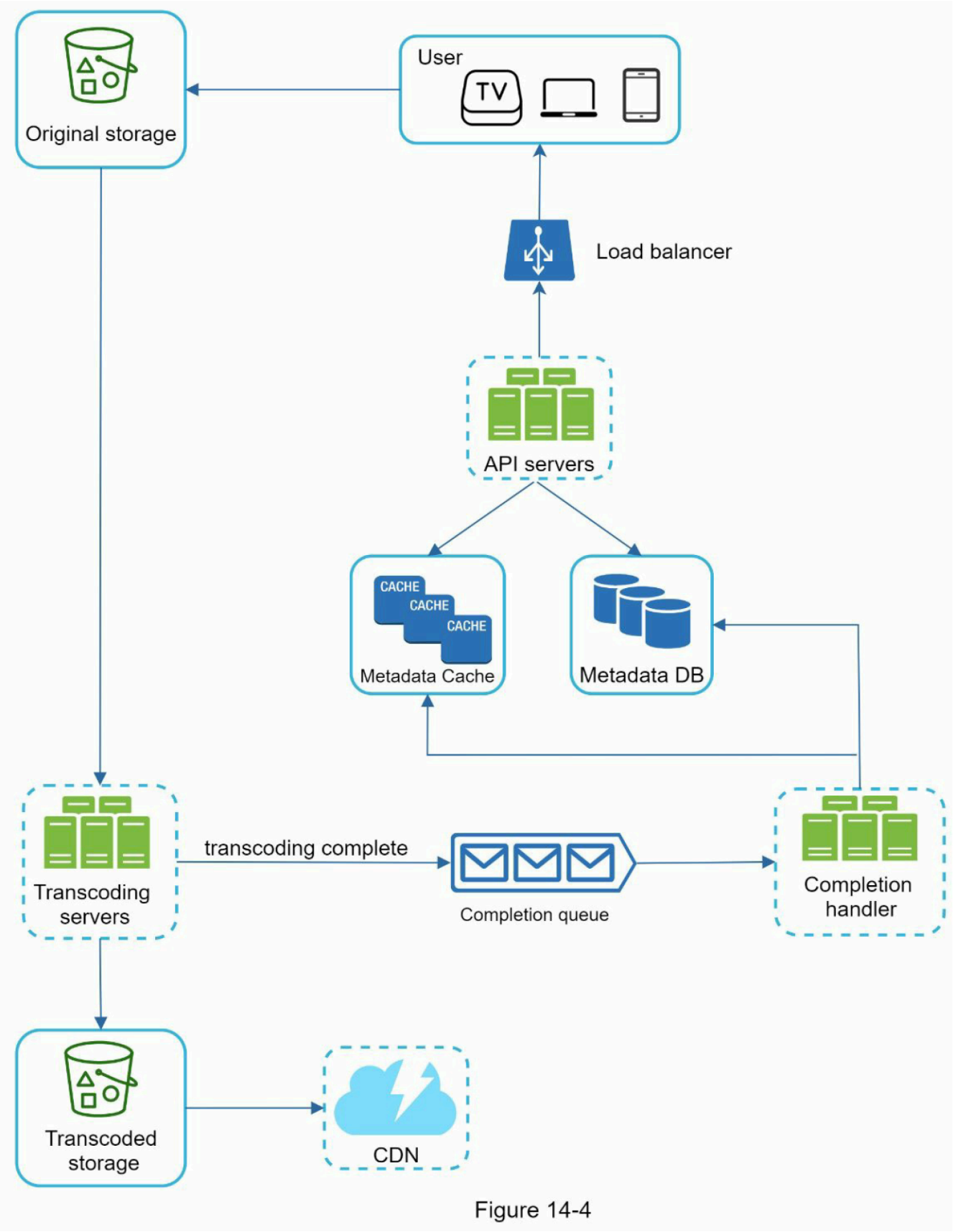

Figure 14-4

High-level design for video uploading.

- **User**: A user watches YouTube on devices such as a computer, mobile phone, or smart TV.
- **Load balancer**: A load balancer evenly distributes requests among API servers.
- **API servers**: All user requests (e.g., feed recommendation, generating video upload URL,

updating metadata database and cache, etc) go through API servers *except video streaming* (CDN).
• **Metadata DB**: Video metadata are stored in Metadata DB. It is sharded and replicated to meet performance and high availability requirements.
• **Metadata cache**: For better performance, video *metadata and user objects* are cached.
• **Original storage**: A blob storage system is used to store original videos. A quotation in Wikipedia regarding blob storage shows that: “A Binary Large Object (BLOB) is a collection of binary data stored as a single entity in a database management system” [6].
• **Transcoding servers**: Video transcoding is also called *video encoding*. It is the process of converting a video format to other formats (MPEG, HLS, etc), which provide the best video streams possible for different devices and bandwidth capabilities.
• **Transcoded storage**: It is a blob storage that stores transcoded video files.
• **CDN**: Videos are cached in CDN. When you click the play button, a video is streamed from the CDN.
• **Completion queue**: It is a message queue that stores information about video transcoding completion events.
• **Completion handler:** This consists of a list of workers that pull event data from the completion queue and *update metadata cache and database*.

Video uploading flow - two processes running in parallel:

- a. Upload the actual video
- b. Update video metadata. Metadata contains information about video URL, size, resolution, format, user info, etc.

Flow a: upload the actual video

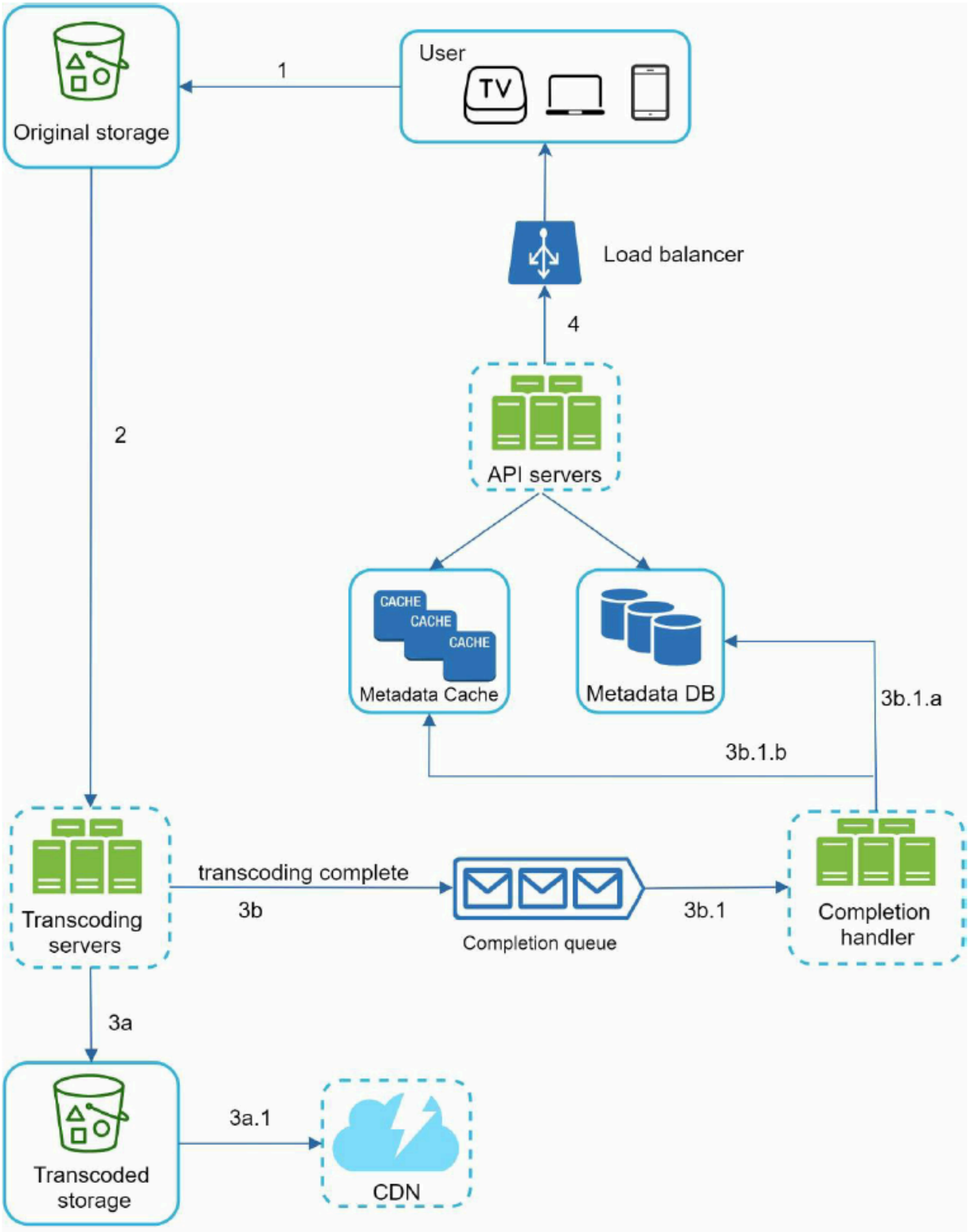

Figure 14-5

Flow of uploading the actual video:

1. Videos are uploaded to the original storage.
2. Transcoding servers fetch videos from the original storage and start transcoding.
3. Once transcoding is complete, the following two steps are executed in parallel:

3a. Transcoded videos are sent to transcoded storage.
3a.1. Transcoded videos are distributed to CDN.
3b. Transcoding completion events are queued in the completion queue.
3b.1. Completion handler contains a bunch of workers that continuously pull event data from the queue.
3b.1.a. and 3b.1.b. Completion handler updates the metadata database and cache when video transcoding is complete.
4. API servers inform the client that the video is successfully uploaded and is ready for streaming

Flow b: update the metadata

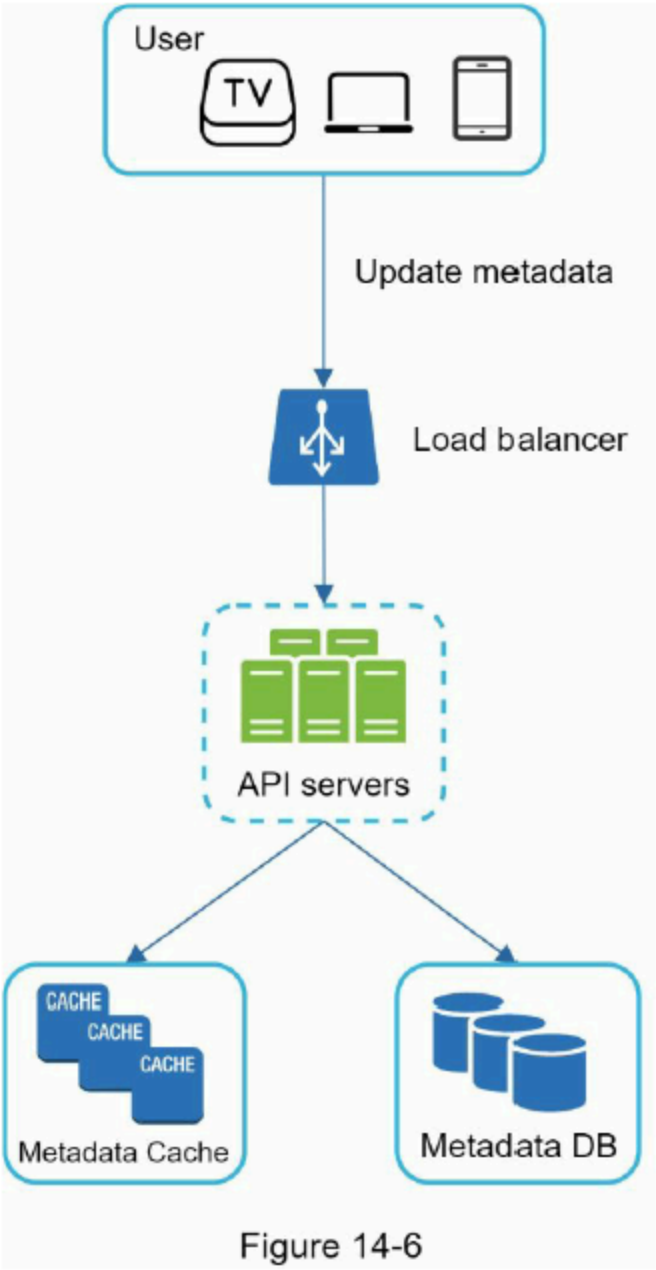

Figure 14-6

- While a file is being uploaded to the original storage, the client **in parallel** sends a request to update the video metadata.
- The request contains video metadata, including file name, size, format, etc.
- API servers update the metadata cache and database.

Video streaming flow

Steaming vs Downloading
Downloading: whole video is copied to your device
Streaming: don't need to wait for whole video is downloaded. You device continuously receives video streams from remote source videos. When you watch streaming videos, your client loads a little bit of data at a time so you can watch videos immediately and continuously.

Streaming protocol

A standardized way to control data transfer for video streaming.

Popular streaming protocols:
• MPEG–DASH. MPEG stands for “Moving Picture Experts Group” and DASH stands for "Dynamic Adaptive Streaming over HTTP".
• Apple HLS. HLS stands for “HTTP Live Streaming”.
• Microsoft Smooth Streaming.
• Adobe HTTP Dynamic Streaming (HDS).

Out-of-scope: fully understand or even remember those streaming protocol names is out of scope of system design. Read more if you are interested:
https://www.dacast.com/blog/streaming-protocols/
In-scope: understand that different streaming protocols support different video encodings and playback players. When we design a video streaming service, we have to choose the right streaming protocol to support our use cases.

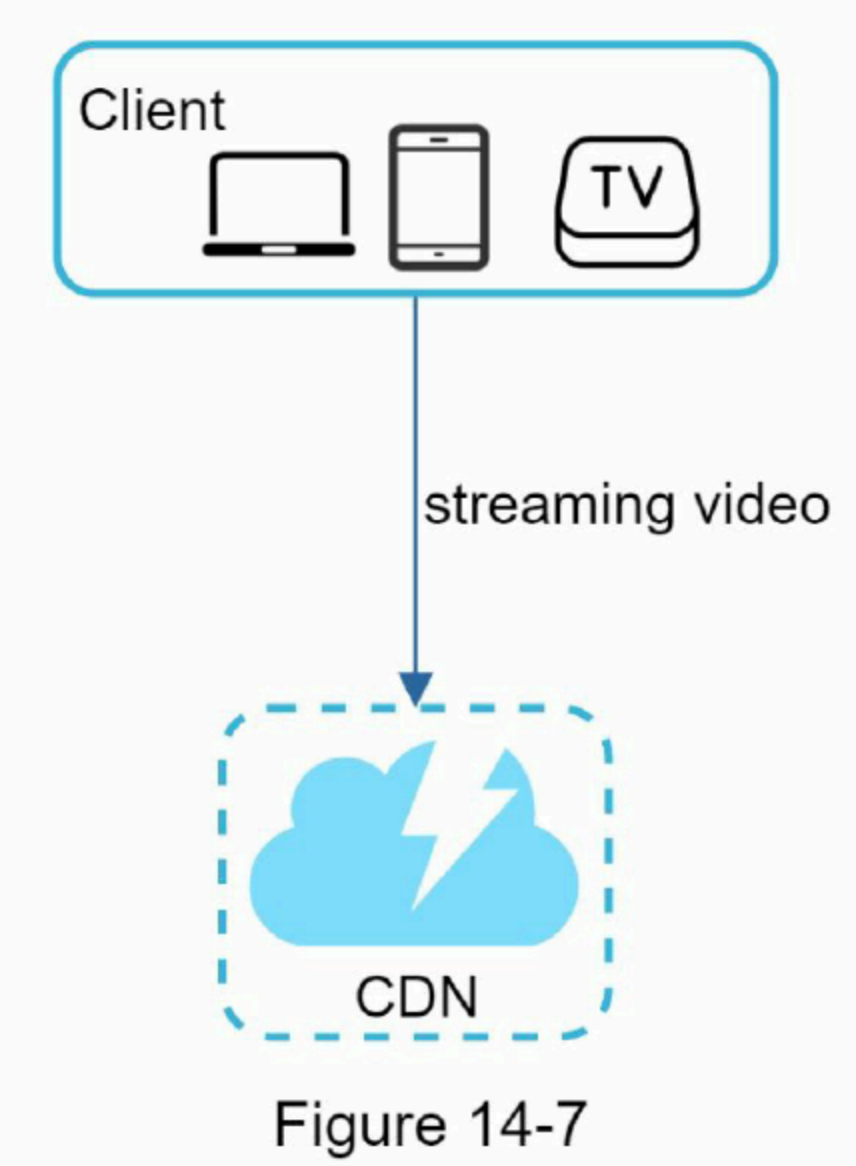

Figure 14-7

High level of design for video streaming:

- Videos are streamed from CDN directly
- The edge server closest to you will deliver the video - very little latency

Step 3: Design deep dive

Purpose of design deep dive: **refine video uploading flow and video streaming flow** respectively with **important optimizations** and introduce **error handling** mechanisms.

Video transcoding

When you record a video, the device (usually a phone or camera) gives the video file a certain format. If you want the video to be played smoothly on other devices, the video must be encoded into compatible bitrates and formats.
Bitrate: is the rate at which bits are processed over time.

A higher bitrate generally means higher video quality
High bitrate streams need more processing power and fast internet speed.

Video transcoding is important (storage, device/browser compatibility, high-quality vs smooth playback, network conditions):
• **Raw video consumes large amounts of storage space**. An hour-long high definition video recorded at 60 frames per second can take up a few hundred GB of space.
• **Many devices and browsers only support certain types of video formats.** Thus, it is important to encode a video to different formats for **compatibility** reasons.
• To ensure users **watch high-quality videos while maintaining smooth playback**, it is a good idea to deliver higher resolution video to users who have high network bandwidth and lower resolution video to users who have low bandwidth.
• **Network conditions can change**, especially on mobile devices. To ensure a video is played continuously, switching video quality automatically or manually based on network conditions is essential for smooth user experience.

Most video transcoder encoding formats have two parts:
• **Container**: This is like a basket that contains the video file, audio, and metadata. You can tell the *container format by the file extension*, such as .avi, .mov, or .mp4.
• **Codecs**: These are *compression and decompression algorithms* aim to reduce the video size while preserving the video quality. The most used video codecs are H.264, VP9, and HEVC.

Directed acyclic graph (DAG) model

Motivation:

- Transcoding a video is computationally expensive and time-consuming.
- Different content creators may have different video processing requirements, e.g., watermarks, thumbnails images, high definition video, etc

Solution - DAG model for video transcoding:

- To support different video processing pipelines and maintain high parallelism, it is important to add some level of abstraction and let client programmers define what tasks to execute.
- Facebook’s streaming video engine uses a directed acyclic graph (DAG) programming model, which defines tasks in stages so they can be executed sequentially
- We adopt a similar DAG model to achieve flexibility and parallelism

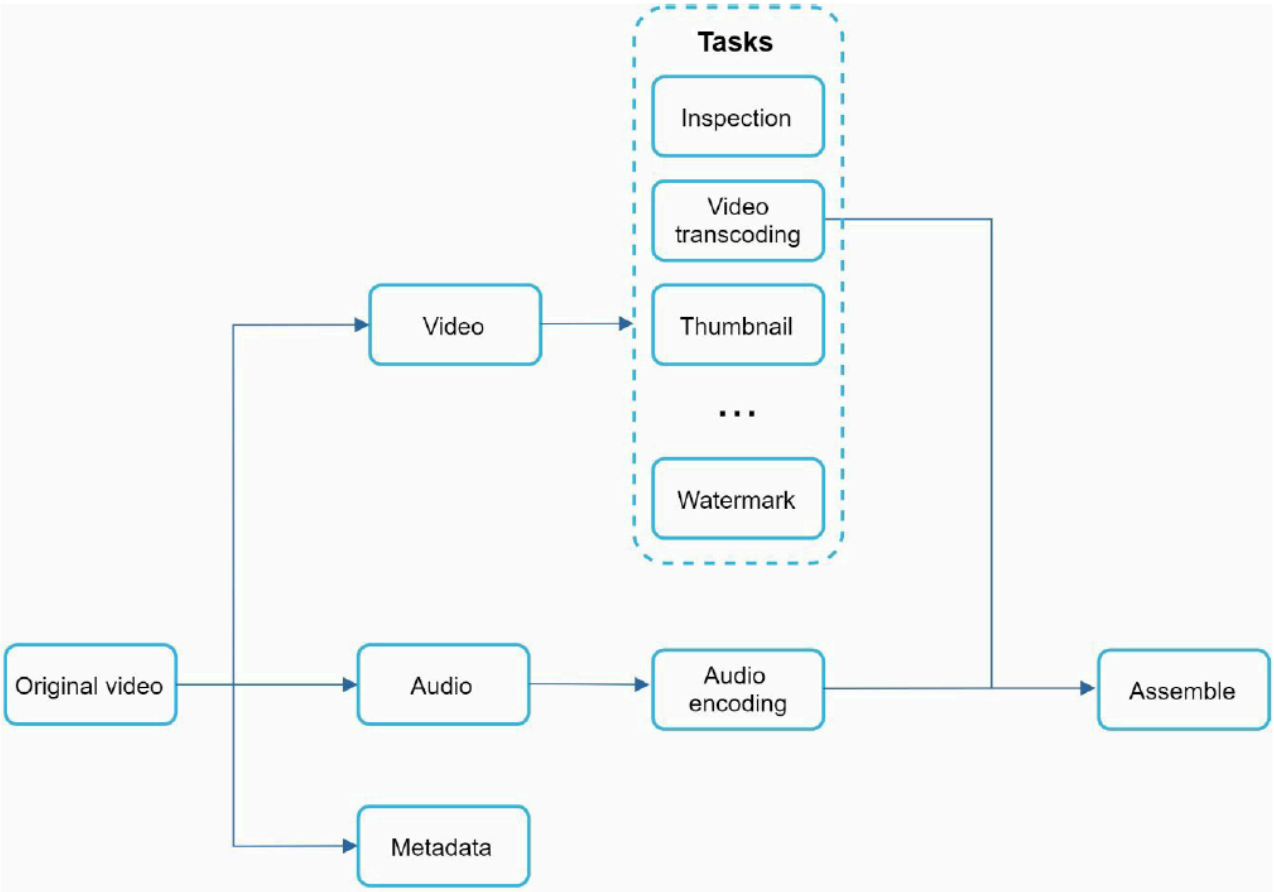

Figure 14-8

The original video is split into video, audio, and metadata. **Tasks on video file**:

• Inspection: Make sure videos have good quality and are not malformed.

• Video encodings: Videos are converted to support different resolutions, codec, bitrates, etc. Figure 14-9 shows an example of video encoded files.

• Thumbnail. Thumbnails can either be uploaded by a user or automatically generated by the system.

• Watermark: An image overlay on top of your video contains identifying information about your video.

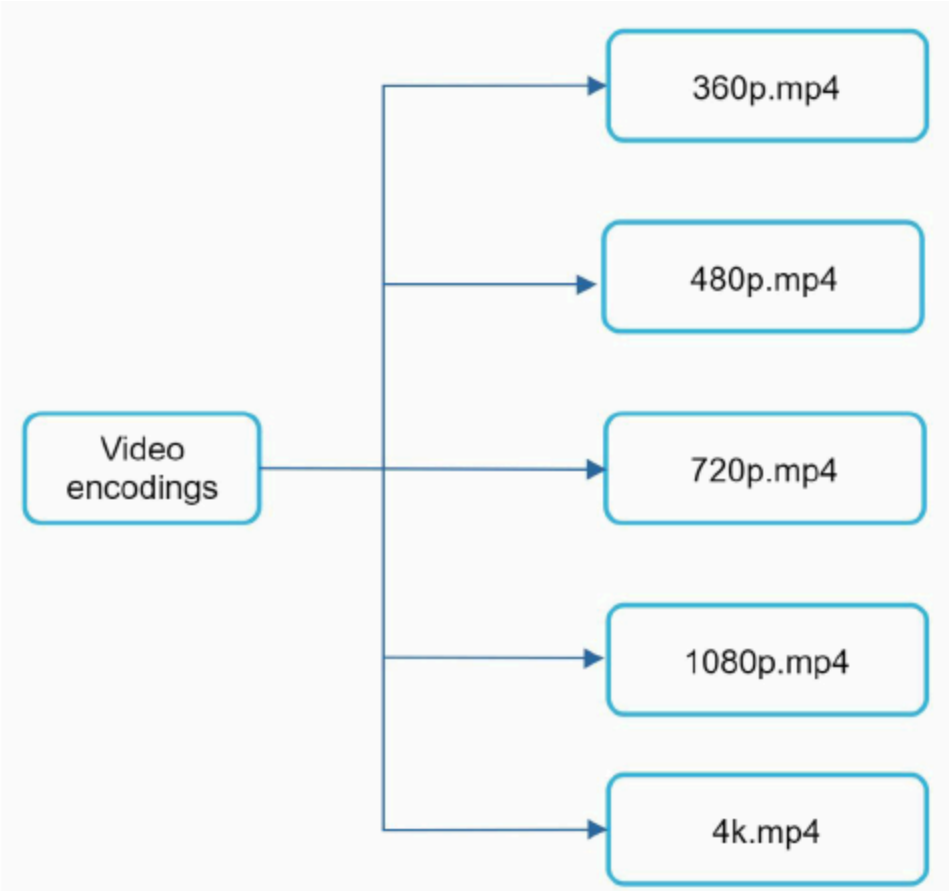

Figure 14-9

Video Transcoding Architecture

Video transcoding architecture consist of **6 components** that leverages the cloud services:

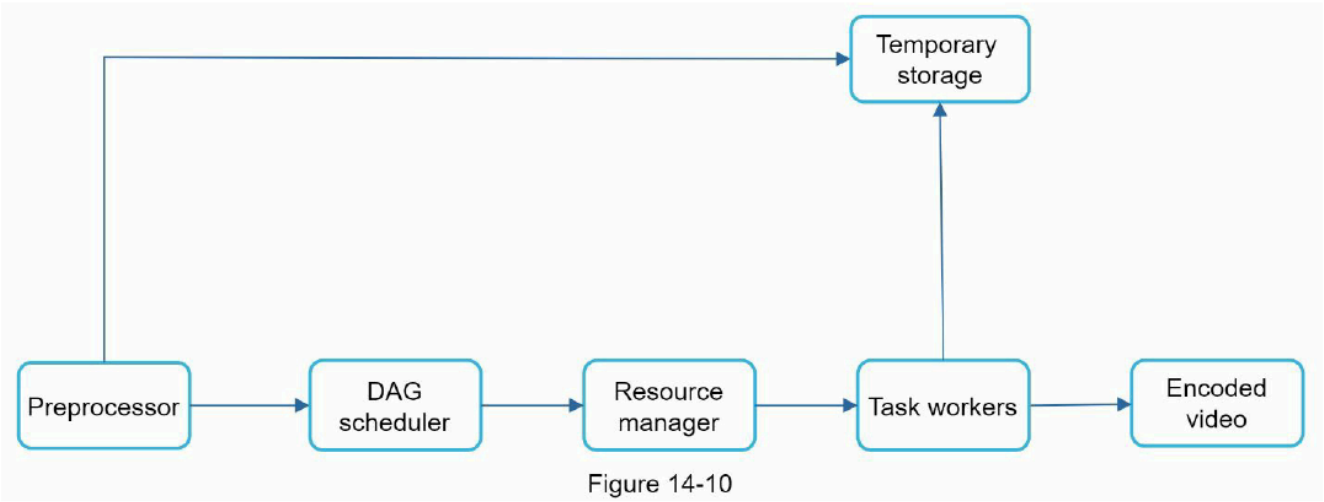

Figure 14-10

The architecture has six main components: preprocessor, DAG scheduler, resource manager, task workers, temporary storage, and encoded video as the output. Let us take a close look at each component.

1. Preprocessor

The preprocessor has 4 responsibilities:

1. **Video splitting**. Video stream is split or further split into smaller *Group of Pictures* (GOP) alignment. GOP is a group/chunk of frames arranged in a specific order. Each chunk is an independently playable unit, usually a few seconds in length.
2. Some old mobile devices or browsers might not support video splitting. **Preprocessor split videos by GOP alignment for old clients.**
3. **DAG generation**. The processor generates DAG *based on configuration files* client programmers write. Figure 14-12 is a simplified DAG representation which has 2 nodes and 1 edge.
4. **Cache data.** The *preprocessor is a cache for segmented videos*. For better reliability, the preprocessor stores GOPs and metadata in *temporary storage.* If video encoding fails, the system could use persisted data for retry operations.

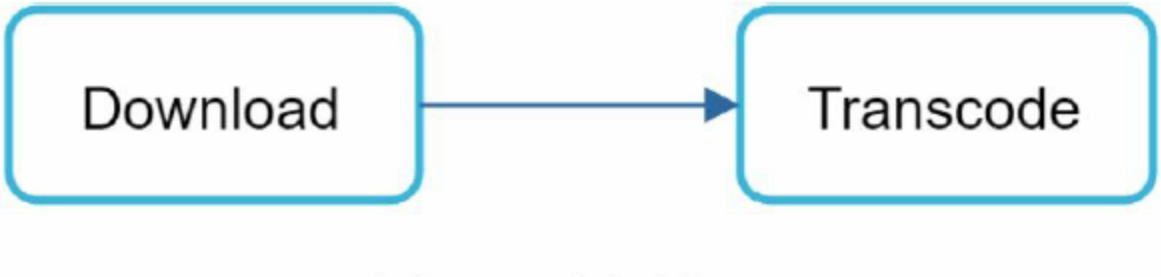

Figure 14-12

This DAG representation is generated from the two configuration files below (Figure 14-13):

```
task {
    name 'download-input'
    type 'Download'
    input {
        url config.url
    }
    output { it->
        context.inputVideo = it.file
    }
    next 'transcode'
}
```

```
task {
    name 'transcode'
    type 'Transcode'
    input {
        input context.inputVideo
        config config.transConfig
    }
    output { it->
        context.file = it.outputVideo
    }
}
```

Figure 14-13 (source: [9])

2. DAG Scheduler

The DAG scheduler splits a DAG graph into stages of tasks and puts them in the task queue in the resource manager.

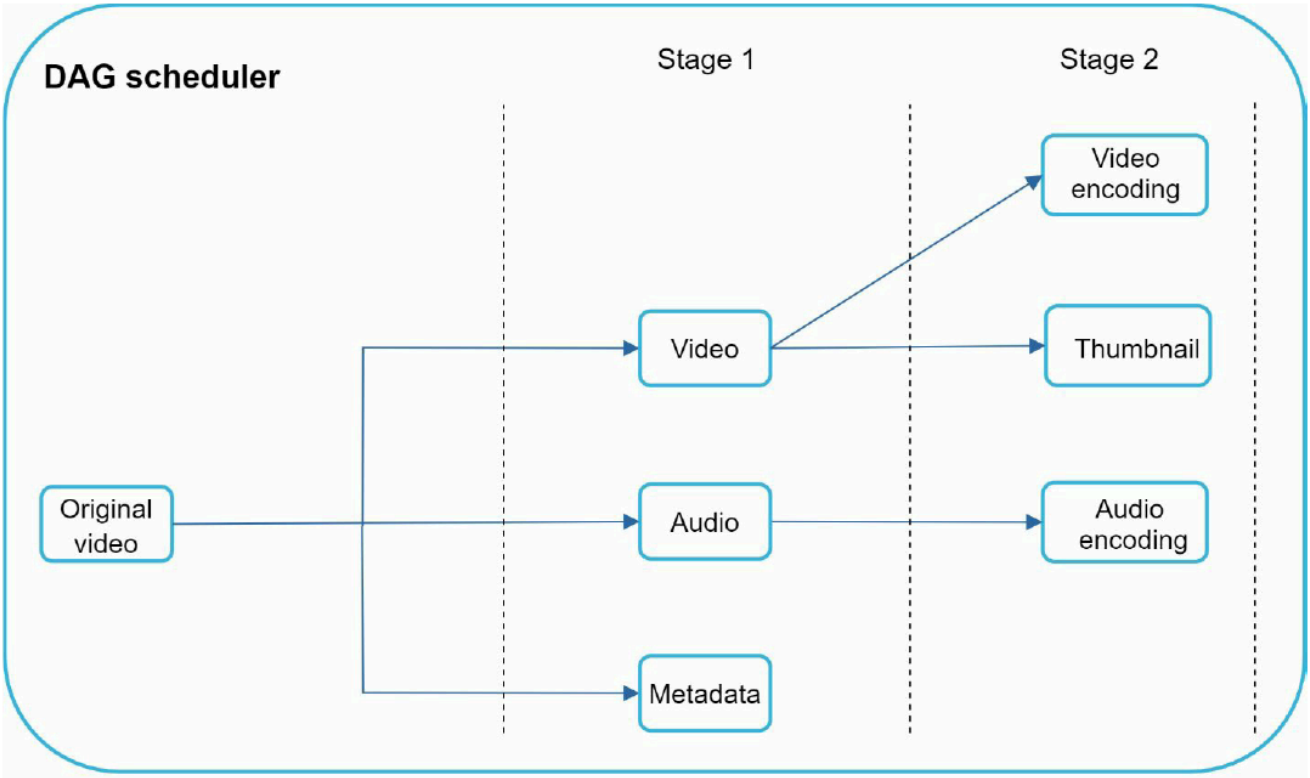

Figure 14-15

- The original video is split into three stages:
  - Stage 1: video, audio, and metadata.
    - The video file is further split into two tasks in stage 2: video encoding and thumbnail.
    - The audio file requires audio encoding as part of the stage 2 tasks.

3. Resource manager

The resource manager is responsible for managing the efficiency of resource allocation. It contains: 3 queues, and a task scheduler.

• **Task queue**: It is a *priority queue* that contains tasks to be executed.

• **Worker queue**: It is a *priority queue* that contains worker utilization info.

• **Task scheduler**: It picks the optimal task/worker, and instructs the chosen task worker to execute the job.

• **Running queue**: It contains info about the currently running tasks and workers running the tasks.

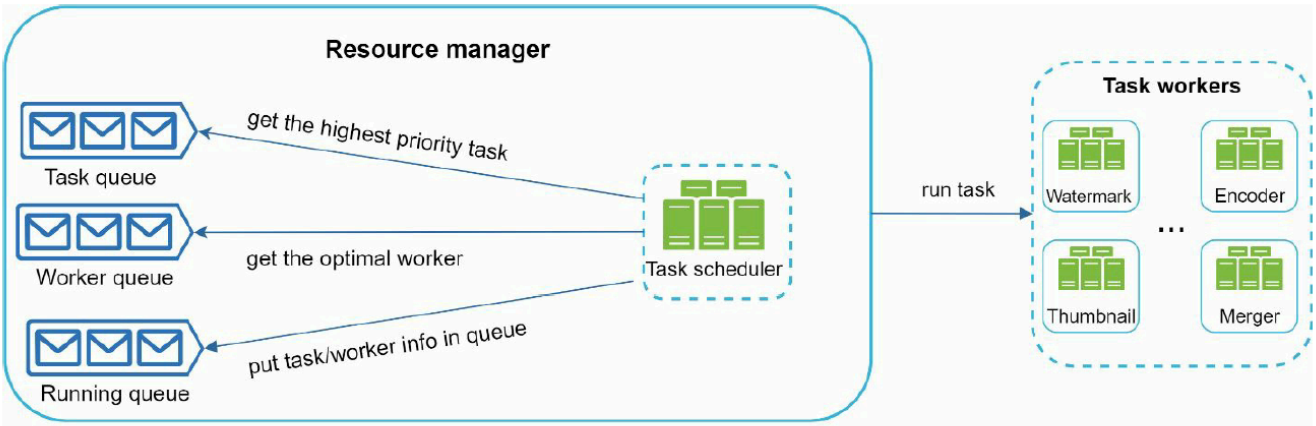

Figure 14-17

Resource manager workflow:

- The task scheduler gets the highest priority task from the task queue.
- The task scheduler gets the optimal task worker to run the task from the worker queue.
- The task scheduler instructs the chosen task worker to run the task.
- The task scheduler binds the task/worker info and puts it in the running queue.
- The task scheduler removes the job from the running queue once the job is done.

4. Task workers

Task workers run the tasks which are defined in the DAG. Different task workers may run different tasks.

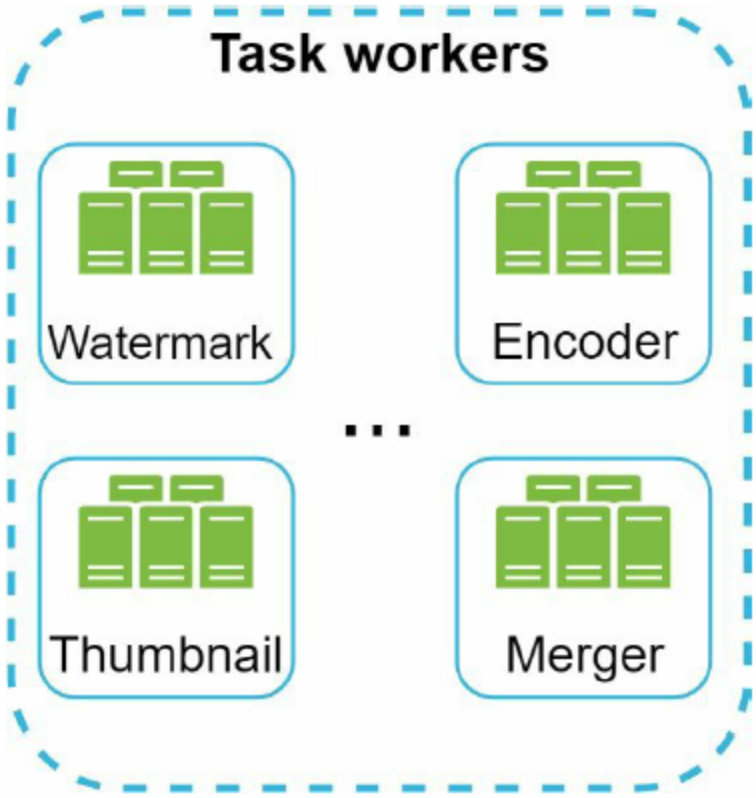

Figure 14-19

5. Temporary storage

Multiple storage systems are used here. The choice of storage system depends on factors like:
data type
data size
access frequency
data life span, etc.

- metadata is frequently accessed by workers, and the data size is usually small. Thus, caching metadata in memory is a good idea.
- For video or audio data, we put them in blob storage.
- Data in temporary storage is freed up once the corresponding video processing is complete.

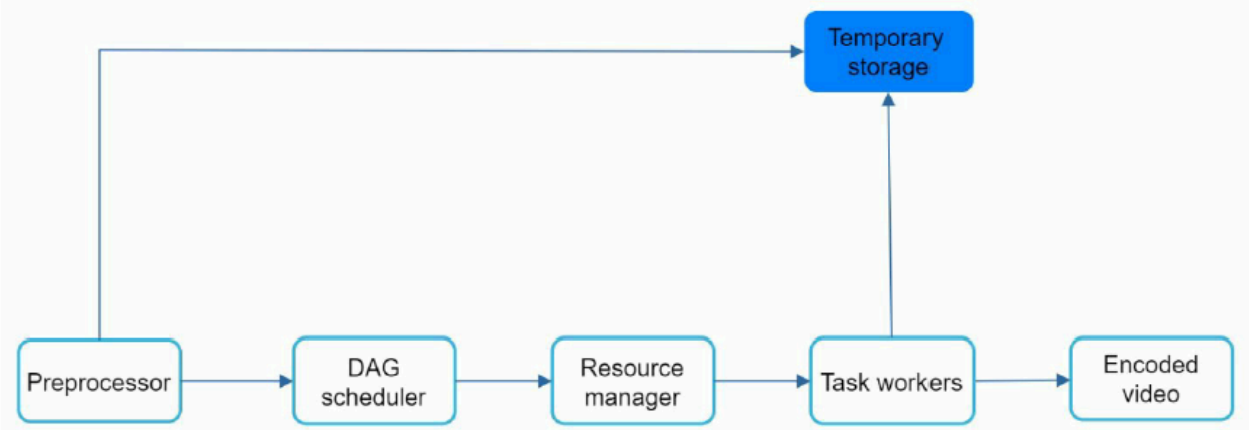

Figure 14-20

6. Encoded video

Encoded video is the final output of the encoding pipeline. An example of the output: funny_720p.mp4

System optimizations

Optimization about: **speed**, **safety**, and **cost-saving**.

Speed optimization: parallelize video uploading

- Uploading a video as a whole unit is inefficient.
- split a video into smaller chunks by GOP(group of pictures) alignment

- allows fast resumable uploads when the previous upload failed

Figure 14-22

The job of splitting a video file by GOP can be implemented by the client to improve the upload speed:

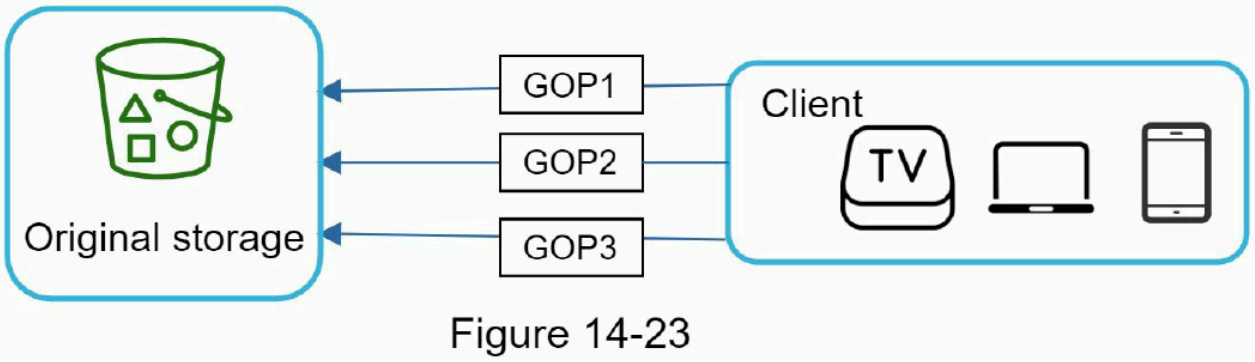

Figure 14-23

Speed optimization: place upload centers close to users

Another way to improve the upload speed is by setting up multiple upload centers across the globe. E.g., people in the United States can upload videos to the North America. To achieve this, we use CDN as upload centers.

Speed optimization: parallelism everywhere

Build a **loosely coupled system** and enable high parallelism.

Flow of how a video is transferred from **original storage to the CDN**:

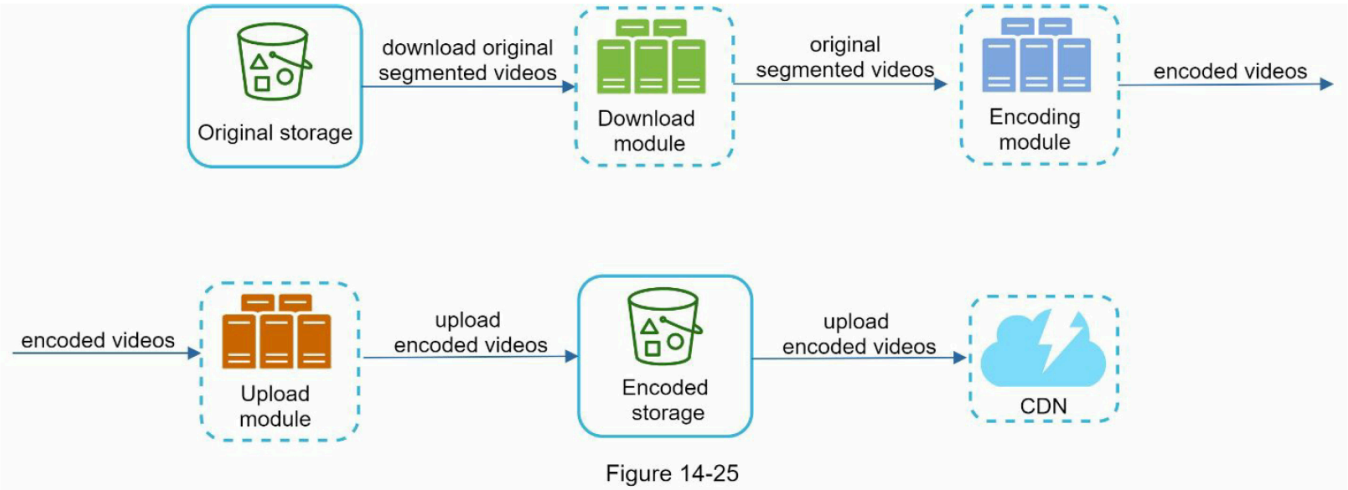

Figure 14-25

Limitations:

- output depends on the input of the previous step

- dependency makes parallelism difficult

Solution: introduced message queues

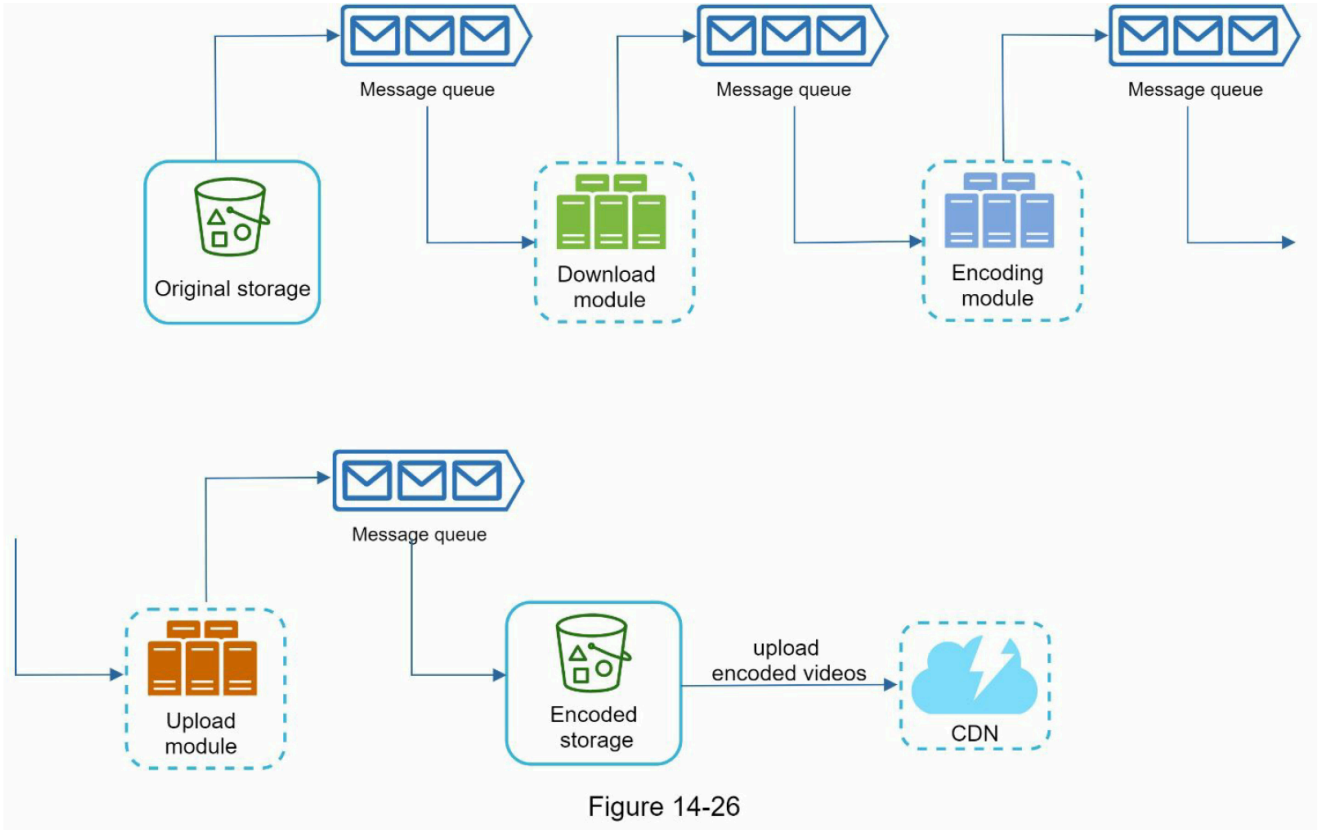

Figure 14-26

• Before the message queue is introduced, the encoding module must wait for the output of the download module.
• After the message queue is introduced, the encoding module does not need to wait for the output of the download module anymore. If there are events in the message queue, the encoding module can execute those jobs in parallel.

Safety optimization: pre-signed upload URL

Goal: Ensure only authorized users upload videos to the right location
Solution: pre-signed URLs

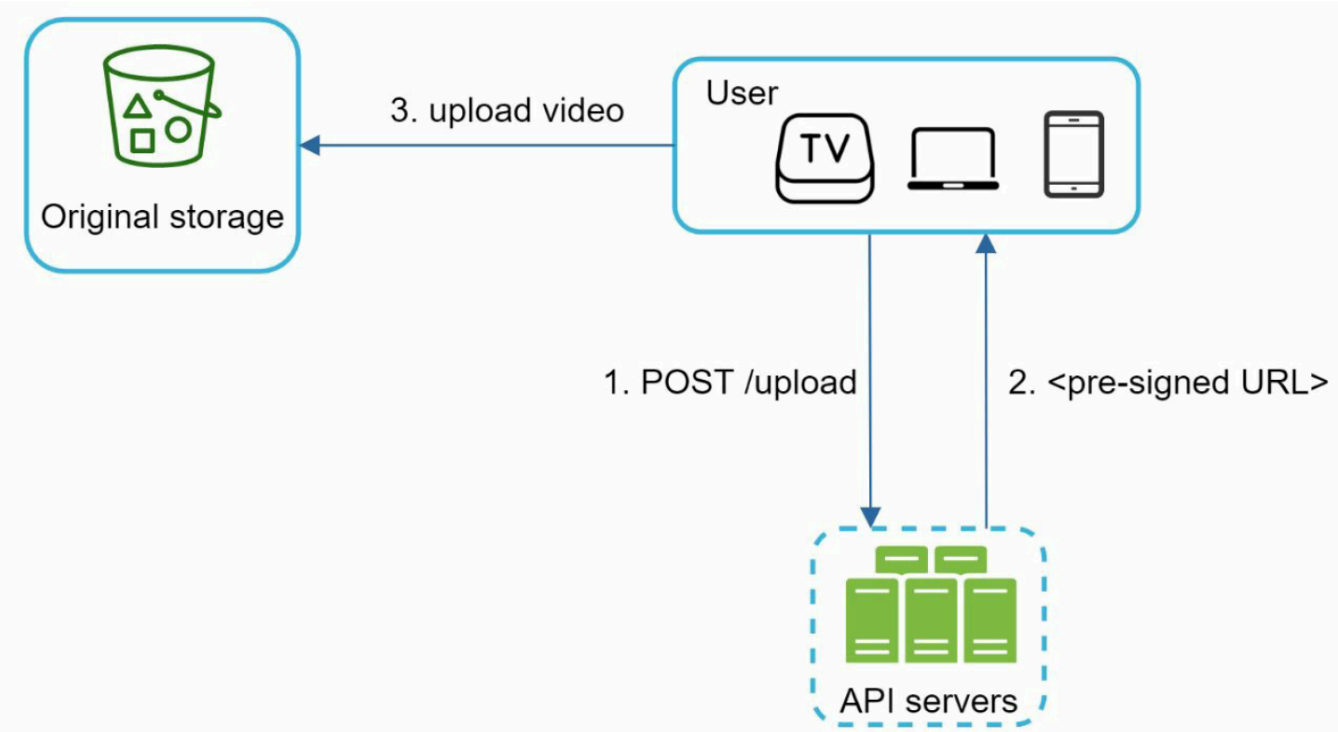

Figure 14-27

1. The client makes a post HTTP request to API servers to fetch the pre-signed URL, which gives the access permission to the object identified in the URL.
    1. The term pre-signed URL is used by uploading files to Amazon S3.
    2. Other cloud service providers might use a different name. For instance, Microsoft Azure blob storage supports the same feature, but call it “Shared Access Signature” [10].
2. API servers respond with a pre-signed URL.
3. Once the client receives the response, it uploads the video using the pre-signed URL.

Safety optimization: protect your videos

Goal: protect copyrighted videos
Solution options:
• **Digital rights management (DRM) systems**: Three major DRM systems are Apple FairPlay, Google Widevine, and Microsoft PlayReady.
• **AES encryption**: You can **encrypt** a video and **configure an authorization policy**. The encrypted video will be decrypted upon playback. This ensures that *only authorized users can watch an encrypted video*.
• **Visual watermarking**: This is an image overlay on top of your video that contains identifying information for your video. It can be your company logo or company name.

Cost-saving optimization

CDN is expensive. How can we reduce the cost?

- YouTube video streams follow **long-tail distribution**
- Only serve the **most popular videos from CDN** and **other videos from our high capacity storage video servers**.
- For **less popular content**, we may **not need to store many encoded video versions**. Short videos can be encoded on-demand.
- Some videos are popular only in certain **regions**. There is no need to distribute these videos to other regions.
- **Build your own CDN** like Netflix and **partner with Internet Service Providers (ISPs)**. Building your CDN is a giant project; however, this could **make sense for large streaming companies.** An ISP can be Comcast, AT&T, Verizon, or other internet providers. ISPs are located all around the world and are close to users. By partnering with ISPs, you can improve the viewing experience and reduce the bandwidth charges.

All those optimizations are based on content popularity, user access pattern, video size, etc. It is important to analyze historical viewing patterns before doing any optimization.

Error handling

Two types of errors :
• **Recoverable error**. For recoverable errors such as video segment fails to transcode, the general idea is to **retry** the operation a few times. If the task continues to fail and the system believes it is not recoverable, it **returns a proper error code** to the client.
• **Non-recoverable error**. For non-recoverable errors such as malformed video format, the system s**tops the running tasks associated with the video** and **returns the proper error code**
to the client.

Typical errors for each system component:
• Upload error: retry a few times.
• Split video error: if older versions of clients cannot split videos by GOP alignment, the entire video is passed to the server. The job of splitting videos is done on the server-side.
• Transcoding error: retry.
• Preprocessor error: regenerate DAG diagram.
• DAG scheduler error: reschedule a task.
• Resource manager queue down: use a replica.
• Task worker down: retry the task on a new worker.
• API server down: API servers are stateless so requests will be directed to a different API server.
• Metadata cache server down: data is replicated multiple times. If one node goes down,

you can still access other nodes to fetch data. We can bring up a new cache server to replace the dead one.
• Metadata DB server down:
• Master is down. If the master is down, promote one of the slaves to act as the new master.
• Slave is down. If a slave goes down, you can use another slave for reads and bring up another database server to replace the dead one.

Step 4: Wrap up

Extra time topics:
• **Scale the API tier**: Because API servers are stateless, it is easy to scale API tier horizontally.
• **Scale the database**: You can talk about database replication and sharding.
• **Live streaming**: It refers to the process of how a video is recorded and broadcasted in real time. Although our system is not designed specifically for live streaming, live streaming and non-live streaming have some similarities: both require uploading, encoding, and streaming. The notable differences are:
• Live streaming has a higher latency requirement, so it might need a different streaming protocol.
• Live streaming has a lower requirement for parallelism because small chunks of data are already processed in real-time.
• Live streaming requires different sets of error handling. Any error handling that takes too much time is not acceptable.
• **Video takedowns**: Videos that violate copyrights, pornography, or other illegal acts shall be removed. Some can be discovered by the system during the upload process, while others might be discovered through user flagging.