15 Design YouTube 设计 YouTube

In this chapter, you are asked to design YouTube. The solution to this question can be applied to other interview questions like designing a video sharing platform such as Netflix and Hulu. Figure 1 shows the YouTube homepage.

在本章中，要求你设计 YouTube。此问题的解决方案可应用于其他面试问题，例如设计视频共享平台，如 Netflix 和 Hulu。图 1 显示了 YouTube 主页。

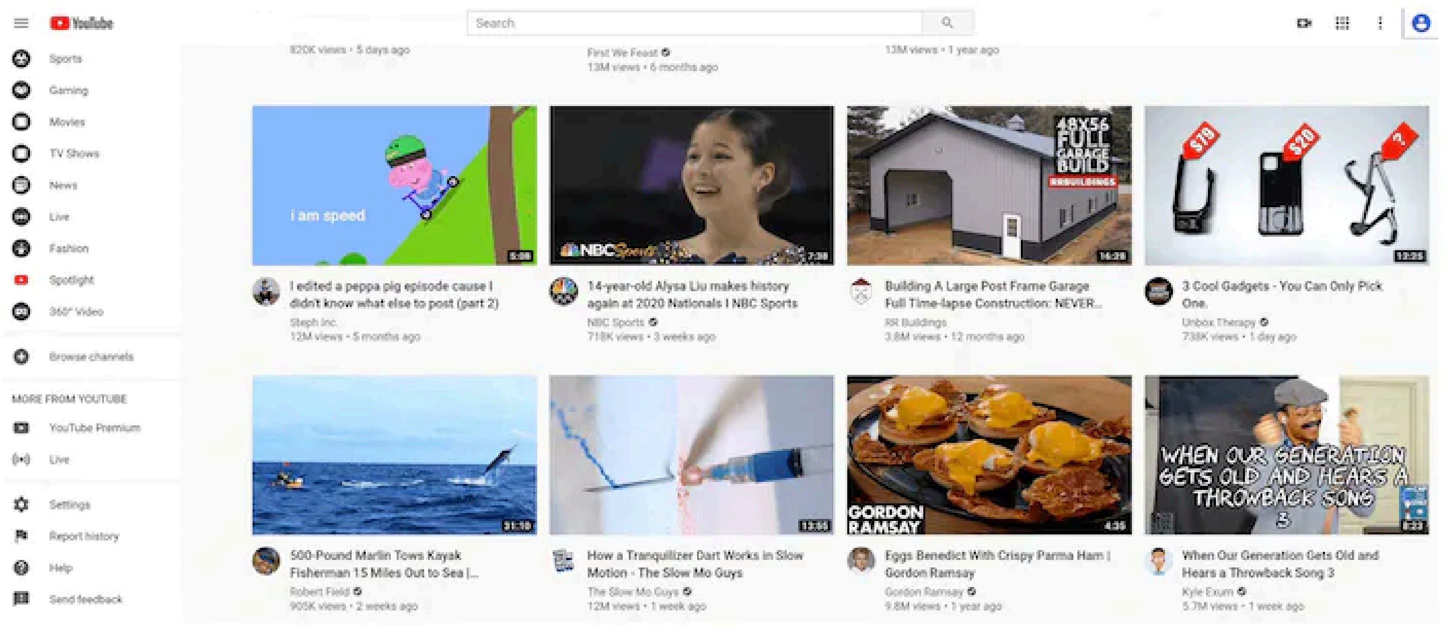

YouTube
Search
Sports
Gaming
Movies
TV Shows
News
Live
Fashion
Spotlight
360° Video
Browse channels
MORE FROM YOUTUBE
YouTube Premium
Live
Settings
Report history
Help
Send feedback
I edited a peppa pig episode cause I didn't know what else to post (part 2)
14-year-old Alysa Liu makes history again at 2020 Nationals I NBC Sports
Building A Large Post Frame Garage Full Time-lapse Construction: NEVER...
3 Cool Gadgets - You Can Only Pick One.
500-Pound Marlin Tows Kayak Fisherman 15 Miles Out to Sea |...
How a Tranquilizer Dart Works in Slow Motion - The Slow Mo Guys
Eggs Benedict With Crispy Parma Ham | Gordon Ramsay
When Our Generation Gets Old and Hears a Throwback Song 3

Figure 1 图 1

YouTube looks simple: content creators upload videos and viewers click play. Is it really that simple? Not really. There are lots of complex technologies underneath the simplicity. Let us look at some impressive statistics, demographics, and fun facts of YouTube in 2020 [1] [2].

YouTube 看起来很简单：内容创作者上传视频，观众点击播放。真的就这么简单吗？并非如此。在简单性之下隐藏着许多复杂的技术。让我们来看看 YouTube 在 2020 年的一些令人印象深刻的统计数据、人口统计数据和趣闻 [1] [2]。

- Total number of monthly active users: 2 billion.

  每月活跃用户总数：20 亿。

- Number of videos watched per day: 5 billion.

  每天观看的视频数量：50 亿。

- 73% of US adults use YouTube.

  73% 的美国成年人使用 YouTube。

- 50 million creators on YouTube

  YouTube 上有 5000 万创作者

- YouTube's Ad revenue was $15.1 billion for the full year 2019, up 36% from 2018.

  YouTube 2019 年全年的广告收入为 151 亿美元，比 2018 年增长了 36%。

- YouTube is responsible for 37% of all mobile internet traffic.

  YouTube 占所有移动互联网流量的 37%。

- YouTube is available in 80 different languages.

  YouTube 提供 80 种不同的语言版本。

From these statistics, we know YouTube is enormous, global and makes a lot of money.

从这些统计数据中，我们知道 YouTube 规模庞大、全球化且能赚很多钱。

Step 1 - Understand the problem and establish design scope

步骤 1 - 了解问题并建立设计范围

As revealed in Figure 1, besides watching a video, you can do a lot more on YouTube. For example, comment, share, or like a video, save a video to playlists, subscribe to a channel, etc. It is impossible to design everything within a 45- or 60-minute interview. Thus, it is important to ask questions to narrow down the scope.

如图 1 所示，除了观看视频，您还可以在 YouTube 上做更多的事情。例如，评论、分享或点赞视频，将视频保存到播放列表，订阅频道等。在 45 或 60 分钟的访谈中设计所有内容是不可能的。因此，提出问题以缩小范围非常重要。

Candidate: What features are important?

候选人：哪些功能很重要？

Interviewer: Ability to upload a video and watch a video.

面试官：上传视频和观看视频的能力。

Candidate: What clients do we need to support?

候选人：我们需要支持哪些客户端？

Interviewer: Mobile apps, web browsers, and smart TV.

面试官：移动应用程序、网络浏览器和智能电视。

Candidate: How many daily active users do we have?

候选人：我们有多少日活跃用户？

Interviewer: 5 million 面试官：500 万

Candidate: What is the average daily time spent on the product?

候选人：在产品上花费的平均每日时间是多少？

Interviewer: 30 minutes. 面试官：30 分钟。

Candidate: Do we need to support international users?

候选人：我们需要支持国际用户吗？

Interviewer: Yes, a large percentage of users are international users.

面试官：是的，很大一部分用户是国际用户。

Candidate: What are the supported video resolutions?

候选人：支持哪些视频分辨率？

Interviewer: The system accepts most of the video resolutions and formats.

面试官：该系统接受大多数视频分辨率和格式。

Candidate: Is encryption required?

候选人：是否需要加密？

Interviewer: Yes 面试官：是的

Candidate: Any file size requirement for videos?

候选人：视频有什么文件大小要求？

Interviewer: Our platform focuses on small and medium-sized videos. The maximum allowed video size is 1GB.

面试官：我们的平台专注于中小型视频。允许的最大视频大小为 1GB。

Candidate: Can we leverage some of the existing cloud infrastructures provided by Amazon, Google, or Microsoft?

候选人：我们能利用亚马逊、谷歌或微软提供的部分现有云基础设施吗？

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.

面试官：这是一个好问题。对于大多数公司来说，从头开始构建所有内容是不现实的，建议利用一些现有的云服务。

In the chapter, we focus on designing a video streaming service with the following features:

在本章中，我们重点设计具有以下功能的视频流服务：

- Ability to upload videos fast

  快速上传视频的能力
- Smooth video streaming 流畅的视频流
- Ability to change video quality

  更改视频质量的能力
- Low infrastructure cost 低基础设施成本
- High availability, scalability, and reliability requirements

  高可用性、可扩展性和可靠性要求
- Clients supported: mobile apps, web browser, and smart TV

  支持的客户端：移动应用程序、网络浏览器和智能电视

Back of the envelope estimation

封底估算

The following estimations are based on many assumptions, so it is important to communicate with the interviewer to make sure she is on the same page.

以下估计基于许多假设，因此与面试官沟通以确保她与你达成共识非常重要。

- Assume the product has 5 million daily active users (DAU).

  假设该产品有 500 万日活跃用户 (DAU)。

- Users watch 5 videos per day.

  用户每天观看 5 个视频。

- 10% of users upload 1 video per day.

  10% 的用户每天上传 1 个视频。

- Assume the average video size is 300 MB.

  假设平均视频大小为 300 MB。

- Total daily storage space needed: 5 million * 10% * 300 MB = 150TB

  每天所需的总存储空间：500 万 * 10% * 300 MB = 150TB

- CDN cost. CDN 成本。
- When cloud CDN serves a video, you are charged for data transferred out of the CDN.

  当云 CDN 提供视频服务时，您需要为从 CDN 传输出的数据付费。

- Let us use Amazon's CDN CloudFront for cost estimation (Figure 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.

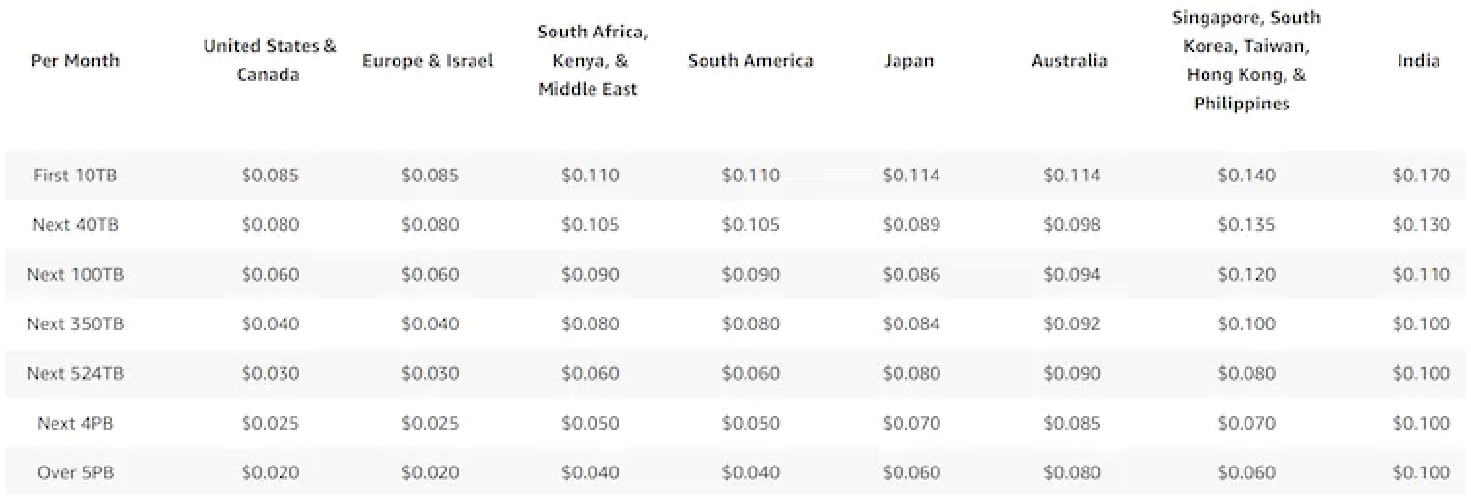
  让我们使用亚马逊的 CDN CloudFront 进行成本估算（图 2）[3]。假设 100% 的流量来自美国。每 GB 的平均成本为 0.02 美元。为简单起见，我们仅计算视频流的成本。

- 5 million * 5 videos * 0.3GB * 0.02 =150,000 per day.

  500 万 * 5 个视频 * 0.3GB * 0.02 = 150,000 每天。

From the rough cost estimation, we know serving videos from the CDN costs lots of money. Even though cloud providers are willing to lower the CDN costs significantly for big customers, the cost is still substantial. We will discuss ways to reduce CDN costs in deep dive.

从粗略的成本估算中，我们知道从 CDN 提供视频服务需要花费很多钱。即使云提供商愿意为大客户大幅降低 CDN 成本，成本仍然很大。我们将在深入探讨中讨论降低 CDN 成本的方法。



Per Month	United States & Canada	Europe & Israel	South Africa, Kenya, & 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 2 图 2

Step 2 - Propose high-level design and get buy-in

步骤 2 - 提出高层级设计并获得认同

As discussed previously, the interviewer recommended leveraging existing cloud services instead of building everything from scratch. CDN and blob storage are the cloud services we will leverage. Some readers might ask why not building everything by ourselves? Reasons are listed below:

如前所述，面试官建议利用现有的云服务，而不是从头开始构建所有内容。CDN 和 Blob 存储是我们将利用的云服务。一些读者可能会问，为什么不自己构建所有内容？原因如下：

- System design interviews are not about building everything from scratch. Within the limited time frame, choosing the right technology to do a job right is more important than explaining how the technology works in detail. For instance, mentioning blob storage for storing source videos is enough for the interview. Talking about the detailed design for blob storage could be an overkill.

  系统设计面试并不是从头开始构建一切。在有限的时间内，选择正确的技术来正确完成一项工作比详细解释该技术如何工作更重要。例如，在面试中提到用于存储源视频的 Blob 存储就足够了。讨论 Blob 存储的详细设计可能会过于繁琐。

- Building scalable blob storage or CDN is extremely complex and costly. Even large companies like Netflix or Facebook do not build everything themselves. Netflix leverages Amazon’s cloud services [4], and Facebook uses Akamai’s CDN [5].

  构建可扩展的 Blob 存储或 CDN 极其复杂且成本高昂。即使像 Netflix 或 Facebook 这样的大公司也不会自己构建所有内容。Netflix 利用亚马逊的云服务 [4]，而 Facebook 使用 Akamai 的 CDN [5]。

At the high-level, the system comprises three components (Figure 3).

在高级别上，该系统包含三个组件（图 3）。

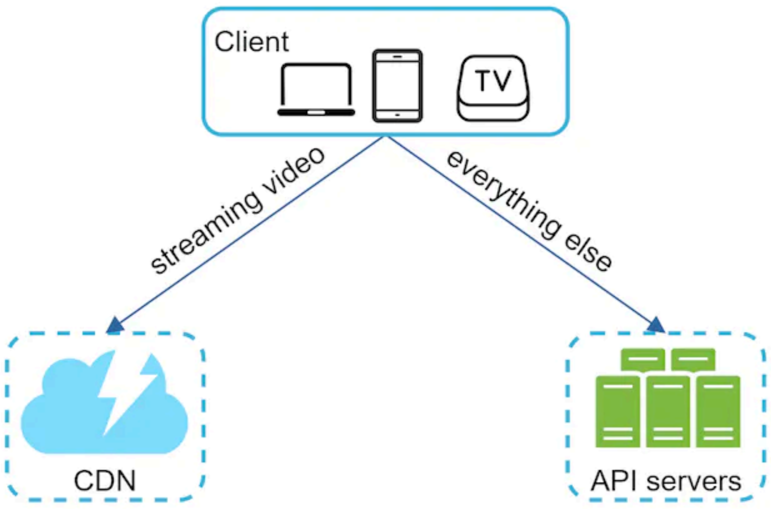


Figure 3 图 3

Client: You can watch YouTube on your computer, mobile phone, and smartTV.

客户端：您可以在计算机、移动电话和智能电视上观看 YouTube。

CDN: Videos are stored in CDN. When you press play, a video is streamed from the CDN.

CDN：视频存储在 CDN 中。当您按下播放时，视频会从 CDN 流式传输。

API servers: Everything else except video streaming goes through API servers. This includes feed recommendation, generating video upload URL, updating metadata database and cache, user signup, etc.

API 服务器：除视频流之外的所有其他内容都通过 API 服务器进行。这包括提要推荐、生成视频上传 URL、更新元数据数据库和缓存、用户注册等。

In the question/answer session, the interviewer showed interests in two flows:

在问答环节中，面试官对两个流程表现出兴趣：

- Video uploading flow 视频上传流程
- Video streaming flow 视频流流程

We will explore the high-level design for each of them.

我们将探讨每个组件的高级设计。

Video uploading flow 视频上传流程

Figure 4 shows the high-level design for the video uploading.

图 4 展示了视频上传的高级设计。

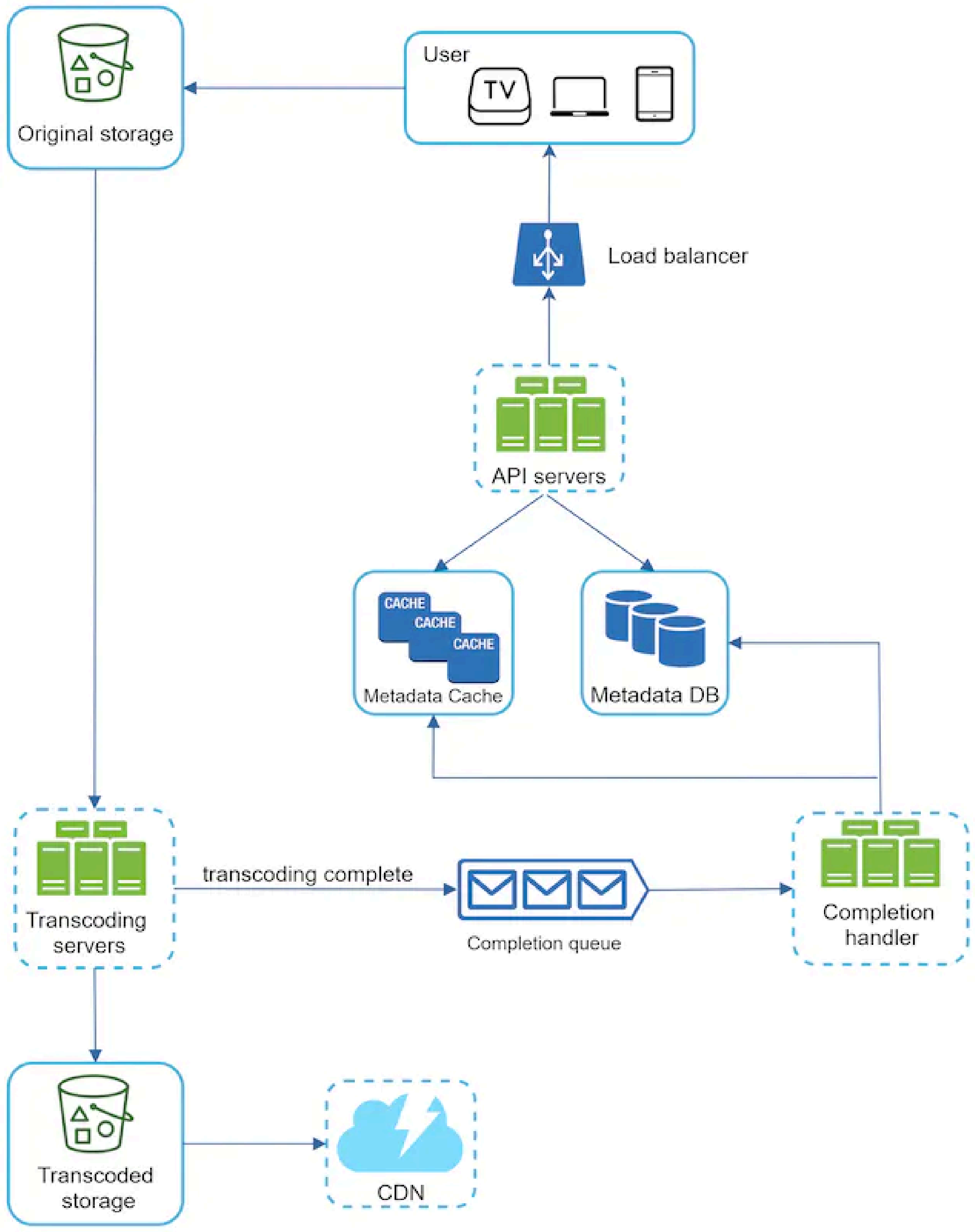


Figure 4 图 4

It consists of the following components:

它包含以下组件:

- User: A user watches YouTube on devices such as a computer, mobile phone, or smart TV.

  用户：用户在计算机、移动电话或智能电视等设备上观看 YouTube。

- Load balancer: A load balancer evenly distributes requests among API servers.

  负载均衡器：负载均衡器在 API 服务器之间均匀分配请求。

- API servers: All user requests go through API servers except video streaming.

  API 服务器：除视频流外，所有用户请求都通过 API 服务器。

- Metadata DB: Video metadata are stored in Metadata DB. It is sharded and replicated to meet performance and high availability requirements.

  元数据数据库：视频元数据存储在元数据数据库中。它被分片并复制以满足性能和高可用性要求。

- Metadata cache: For better performance, video metadata and user objects are cached.

  元数据缓存：为了获得更好的性能，视频元数据和用户对象被缓存。

- 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].

  原始存储：一个 blob 存储系统用于存储原始视频。维基百科中关于 blob 存储的一段引用显示："二进制大对象 (BLOB) 是以单个实体形式存储在数据库管理系统中的二进制数据集合"[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.

  转码服务器：视频转码也称为视频编码。它是将一种视频格式转换为其他格式（MPEG、HLS 等）的过程，这些格式为不同设备和带宽功能提供了最佳的视频流。

- Transcoded storage: It is a blob storage that stores transcoded video files.

  转码存储：它是一个 blob 存储，用于存储转码视频文件。

- CDN: Videos are cached in CDN. When you click the play button, a video is streamed from the CDN.

  CDN：视频被缓存到 CDN 中。当您单击播放按钮时，视频将从 CDN 中流式传输。

- Completion queue: It is a message queue that stores information about video transcoding completion events.

  完成队列：它是一个消息队列，用于存储有关视频转码完成事件的信息。

- Completion handler: This consists of a list of workers that pull event data from the completion queue and update metadata cache and database.

完成处理程序：它由一系列工作程序组成，这些工作程序从完成队列中提取事件数据并更新元数据缓存和数据库。

Now that we understand each component individually, let us examine how the video uploading flow works. The flow is broken down into two processes running in parallel.

现在我们已经单独了解了每个组件，让我们研究视频上传流程的工作原理。该流程被分解为两个并行运行的进程。

a. Upload the actual video.

a. 上传实际视频。

b. Update video metadata. Metadata contains information about video URL, size, resolution, format, user info, etc.

b. 更新视频元数据。元数据包含有关视频 URL、大小、分辨率、格式、用户信息等的信息。

Flow a: upload the actual video

流程 a：上传实际视频

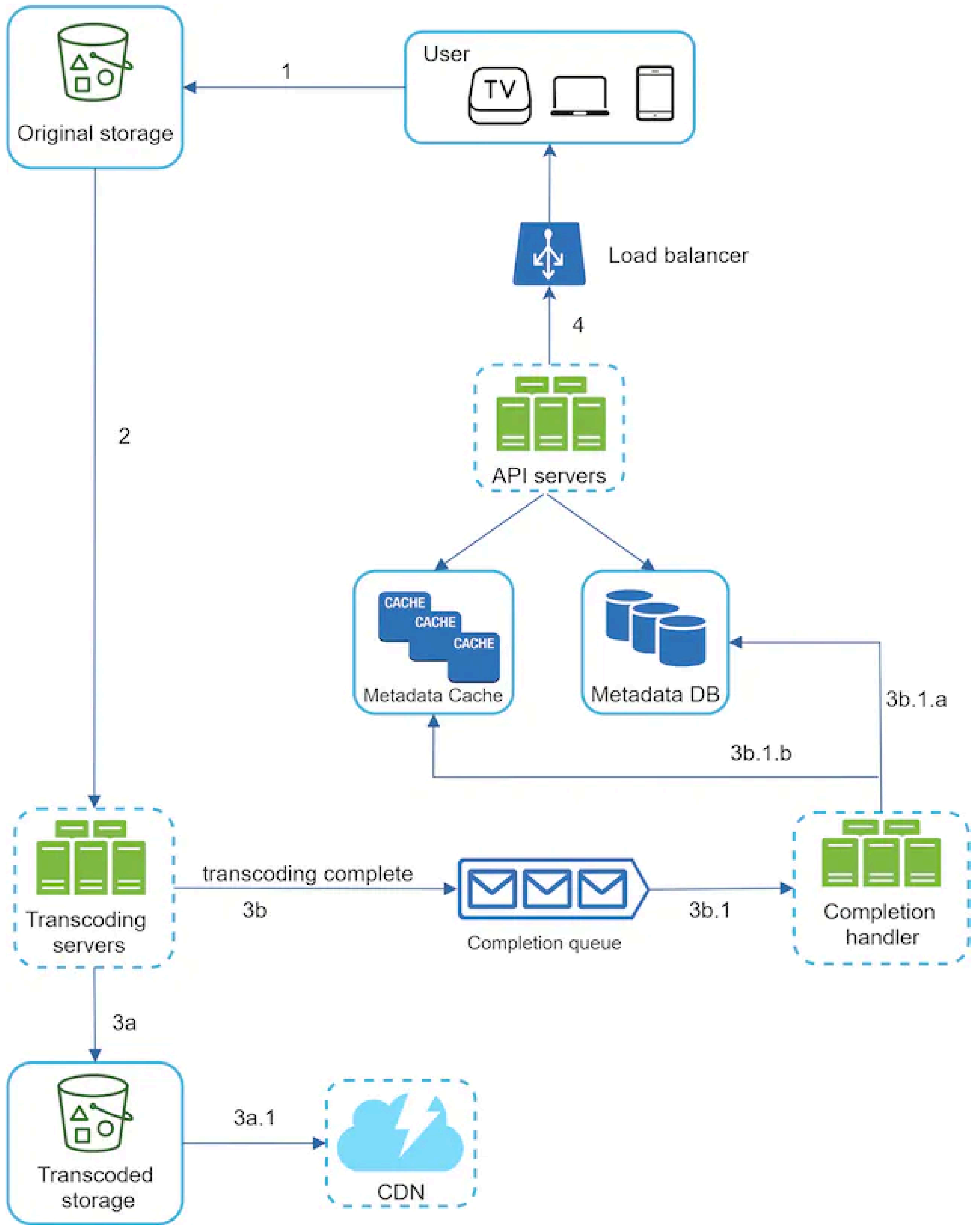


Figure 5 图 5

Figure 5 shows how to upload the actual video. The explanation is shown below:

图 5 显示了如何上传实际视频。解释如下：

1. Videos are uploaded to the original storage.

1. 视频上传到原始存储。

2. Transcoding servers fetch videos from the original storage and start transcoding.

2. 转码服务器从原始存储中获取视频并开始转码。

3. Once transcoding is complete, the following two steps are executed in parallel:

3. 转码完成后，以下两个步骤并行执行：

- 3a. Transcoded videos are sent to transcoded storage.

  3a. 转码视频发送到转码存储。

- 3b. Transcoding completion events are queued in the completion queue.

  3b. 转码完成事件在完成队列中排队。

  3a.1. Transcoded videos are distributed to CDN.

  3a.1. 转码视频分发到 CDN。

  3b.1. Completion handler contains a bunch of workers that continuously pull event data from the queue.

  3b.1. 完成处理程序包含大量持续从队列中提取事件数据的 worker。

  3b.1.a. and 3b.1.b. Completion handler updates the metadata database and cache when video transcoding is complete.

  3b.1.a. 和 3b.1.b. 视频转码完成后，完成处理程序更新元数据数据库和缓存。

4. API servers inform the client that the video is successfully uploaded and is ready for streaming.

4. API 服务器通知客户端视频已成功上传并已准备好进行流式传输。

Flow b: update the metadata

流程 b：更新元数据

While a file is being uploaded to the original storage, the client in parallel sends a request to update the video metadata as shown in Figure 6. The request contains video metadata, including file name, size, format, etc. API servers update the metadata cache and database.

当文件上传到原始存储时，客户端同时发送请求以更新视频元数据，如图 6 所示。该请求包含视频元数据，包括文件名、大小、格式等。API 服务器更新元数据缓存和数据库。

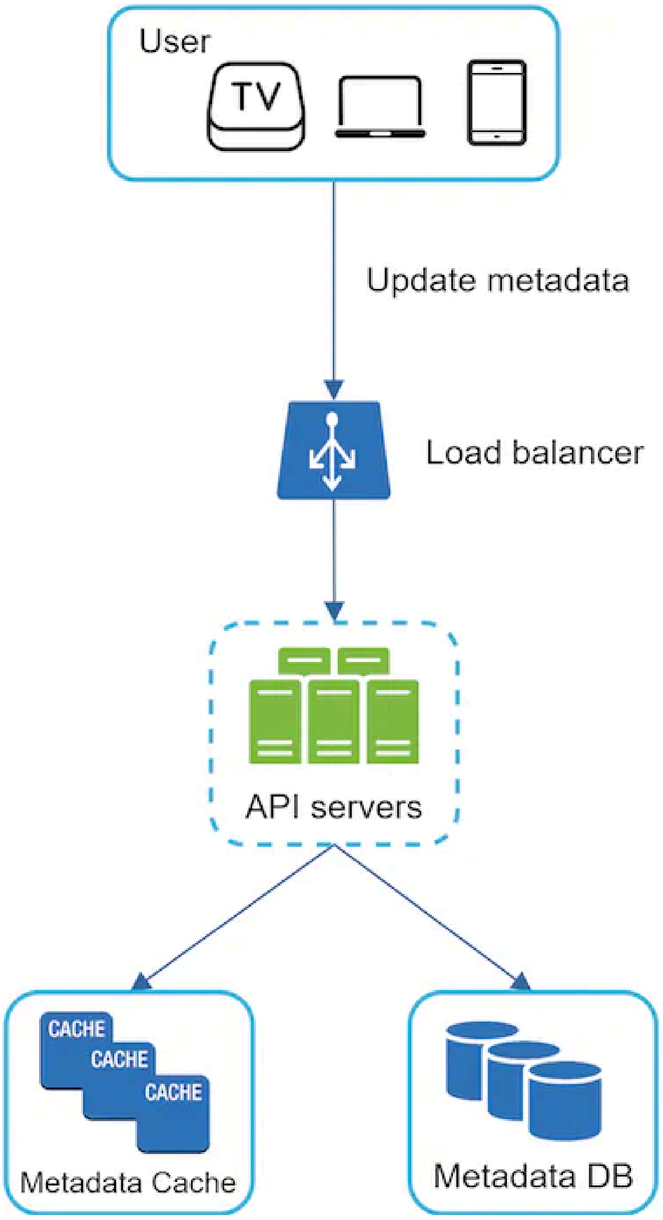


Figure 6 图 6

Video streaming flow 视频流流程

Whenever you watch a video on YouTube, it usually starts streaming immediately and you do not wait until the whole video is downloaded. Downloading means the whole video is copied to your device, while streaming means your 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.

每当您在 YouTube 上观看视频时，它通常会立即开始流式传输，而无需等到整个视频下载完毕。下载意味着整个视频已复制到您的设备，而流式传输意味着您的设备会持续从远程源视频接收视频流。当您观看流式视频时，您的客户端会一次加载少量数据，以便您可以立即且持续地观看视频。

Before we discuss video streaming flow, let us look at an important concept: streaming protocol. This is a standardized way to control data transfer for video streaming. Popular streaming protocols are:

在我们讨论视频流之前，让我们先了解一个重要的概念：流协议。这是控制视频流数据传输的标准化方式。流行的流协议有：

- MPEG–DASH. MPEG stands for “Moving Picture Experts Group” and DASH stands for "Dynamic Adaptive Streaming over HTTP".

  MPEG-DASH。MPEG 代表“动态图像专家组”，DASH 代表“通过 HTTP 进行动态自适应流”。

- Apple HLS. HLS stands for “HTTP Live Streaming”.

  Apple HLS。HLS 是“HTTP 实时流”的缩写。

- Microsoft Smooth Streaming.

  Microsoft 平滑流。

- Adobe HTTP Dynamic Streaming (HDS).

  Adobe HTTP 动态流 (HDS)。

You do not need to fully understand or even remember those streaming protocol names as they are low-level details that require specific domain knowledge. The important thing here is to 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. To learn more about streaming protocols, here is an excellent article [7].

你不必完全理解甚至记住这些流协议名称，因为它们是需要特定领域知识的低级细节。这里重要的是要理解不同的流协议支持不同的视频编码和播放器。当我们设计视频流服务时，我们必须选择正确的流协议来支持我们的用例。要了解有关流协议的更多信息，这里有一篇精彩的文章 [7]。

Videos are streamed from CDN directly. The edge server closest to you will deliver the video. Thus, there is very little latency. Figure 7 shows a high level of design for video streaming.

视频直接从 CDN 流式传输。最靠近你的边缘服务器将传送视频。因此，延迟非常小。图 7 显示了视频流的高级设计。

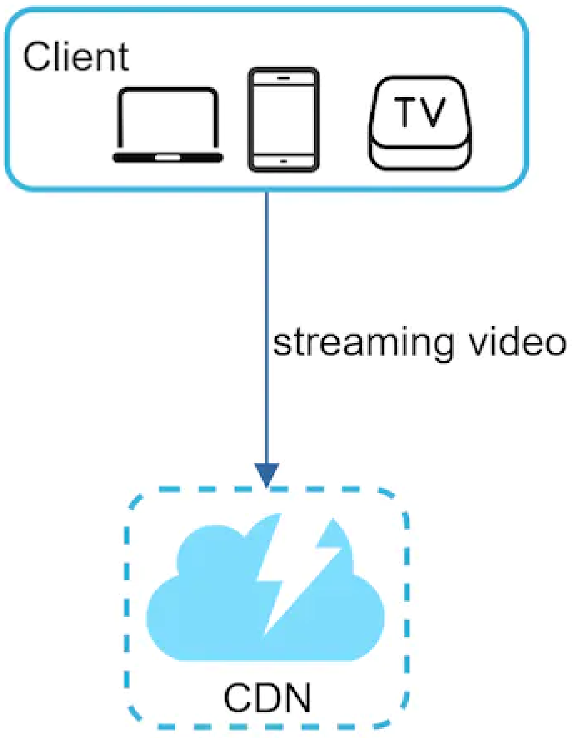


Figure 7 图 7

Step 3 - Design deep dive

步骤 3 - 深入设计

In the high-level design, the entire system is broken down in two parts: video uploading flow and video streaming flow. In this section, we will refine both flows with important optimizations and introduce error handling mechanisms.

在高级设计中，整个系统被分解为两部分：视频上传流和视频流流。在本节中，我们将使用重要的优化来优化这两个流，并引入错误处理机制。

Video transcoding 视频转码

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.

当您录制视频时，设备（通常是手机或相机）会给视频文件一个特定的格式。如果您希望视频在其他设备上流畅播放，则必须将视频编码为兼容的比特率和格式。比特率是指随着时间推移处理比特的速率。更高的比特率通常意味着更高的视频质量。高比特率流需要更多的处理能力和快速的互联网速度。

Video transcoding is important for the following reasons:

视频转码出于以下原因很重要：

- 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.

  原始视频会占用大量的存储空间。以每秒 60 帧的速度录制的一小时高清视频可能占用几百 GB 的空间。

- 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.

  许多设备和浏览器只支持特定类型的视频格式。因此，出于兼容性原因，将视频编码为不同的格式非常重要。

- 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.

  为了确保用户在保持流畅播放的同时观看高质量视频，最好向网络带宽高的用户提供更高分辨率的视频，向带宽低的用户提供较低分辨率的视频。

- 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.

  网络条件可能会发生变化，尤其是在移动设备上。为了确保视频连续播放，根据网络条件自动或手动切换视频质量对于流畅的用户体验至关重要。

Many types of encoding formats are available; however, most of them contain two parts:

有许多类型的编码格式可用；然而，它们中的大多数包含两部分：

- 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.

  容器：这就像一个包含视频文件、音频和元数据的篮子。您可以通过文件扩展名（例如 .avi、.mov 或 .mp4）来识别容器格式。

- 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.

  编解码器：这些是压缩和解压缩算法，旨在在保持视频质量的同时减小视频大小。使用最多的视频编解码器是 H.264、VP9 和 HEVC。

Directed acyclic graph (DAG) model

有向无环图 (DAG) 模型

Transcoding a video is computationally expensive and time-consuming. Besides, different content creators may have different video processing requirements. For instance, some content creators require watermarks on top of their videos, some provide thumbnail images themselves, and some upload high definition videos, whereas others do not.

转码视频在计算上很昂贵且耗时。此外，不同的内容创作者可能对视频处理有不同的要求。例如，一些内容创作者要求在视频顶部添加水印，一些内容创作者自己提供缩略图图像，而另一些内容创作者上传高清视频，而另一些内容创作者则不上传。

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. For example, Facebook’s streaming video engine uses a directed acyclic graph (DAG) programming model, which defines tasks in stages so they can be executed sequentially or parallelly [8]. In our design, we adopt a similar DAG model to achieve flexibility and parallelism. Figure 8 represents a DAG for video transcoding.

为了支持不同的视频处理管道并保持高度并行性，添加一定程度的抽象并让客户端程序员定义要执行的任务非常重要。例如，Facebook 的流媒体视频引擎使用有向无环图 (DAG) 编程模型，该模型分阶段定义任务，以便可以顺序或并行执行它们 [8]。在我们的设计中，我们采用类似的 DAG 模型来实现灵活性和并行性。图 8 表示视频转码的 DAG。

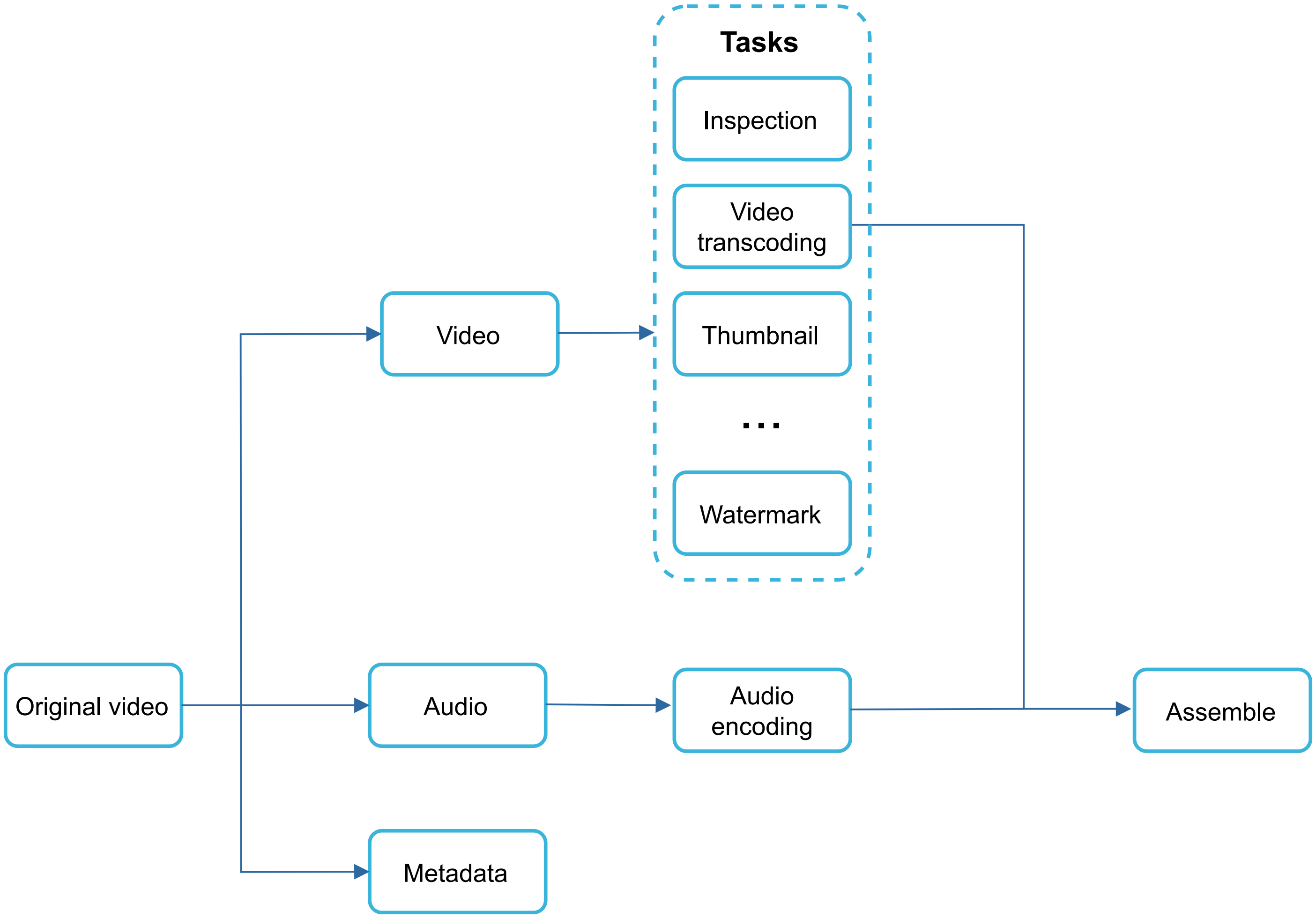


Figure 8 图 8

In Figure 8, the original video is split into video, audio, and metadata. Here are some of the tasks that can be applied on a video file:

在图 8 中，原始视频被拆分为视频、音频和元数据。以下是一些可应用于视频文件上的任务：

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

  检查：确保视频质量良好且未变形。

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

  视频编码：将视频转换为支持不同的分辨率、编解码器、比特率等。图 9 显示了视频编码文件的示例。

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

  缩略图。缩略图可以由用户上传，也可以由系统自动生成。

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

  水印：叠加在视频顶部的图像包含有关视频的识别信息。

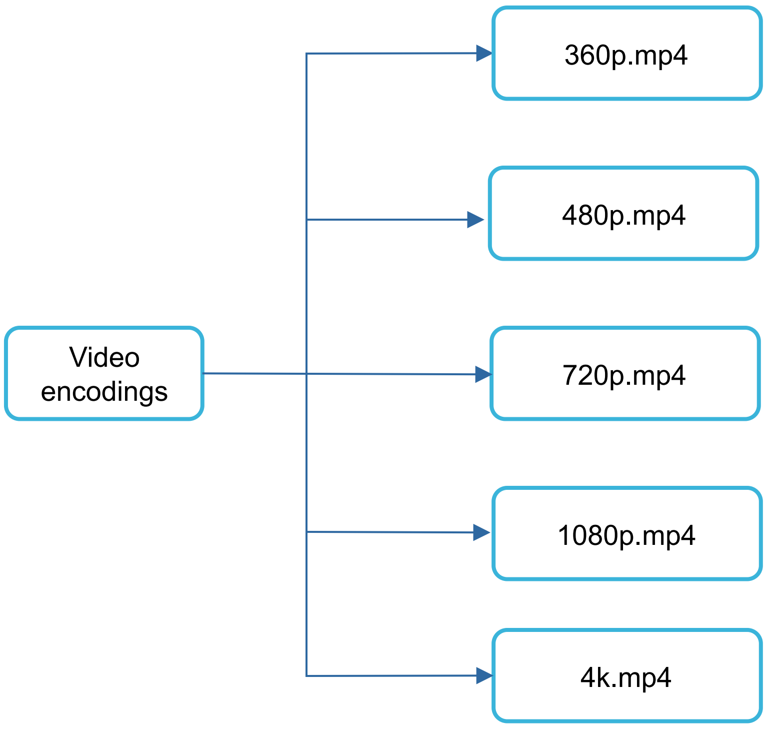


Figure 9 图 9

Video transcoding architecture

视频转码架构

The proposed video transcoding architecture that leverages the cloud services, is shown in Figure 10.

图 10 显示了利用云服务的建议视频转码架构。

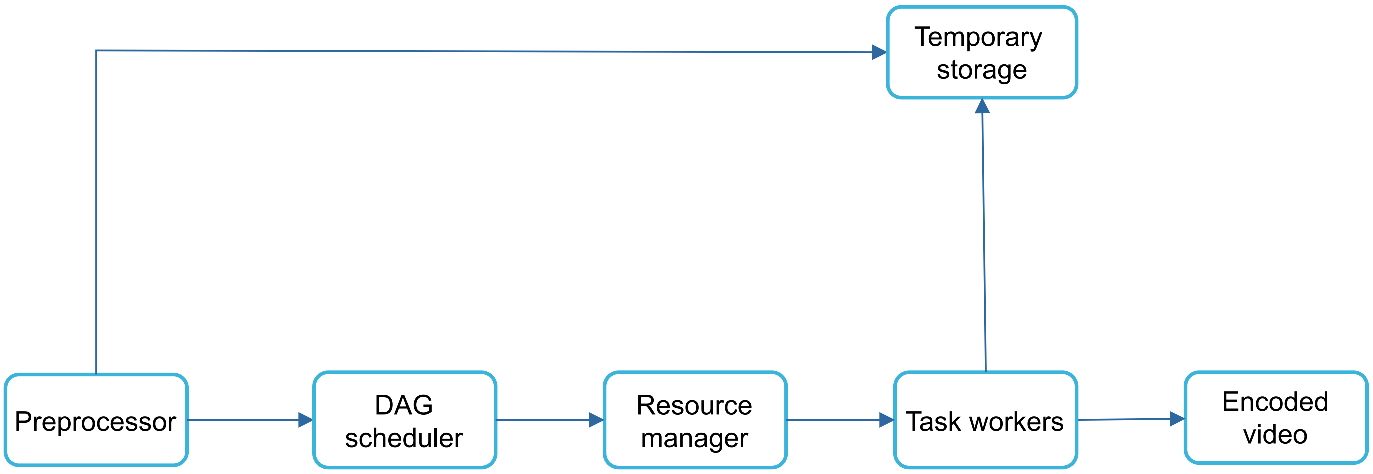


Figure 10 图 10

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.

该架构有六个主要组件：预处理器、DAG 调度程序、资源管理器、任务工作程序、临时存储以及编码视频作为输出。让我们仔细了解每个组件。

Preprocessor 预处理器

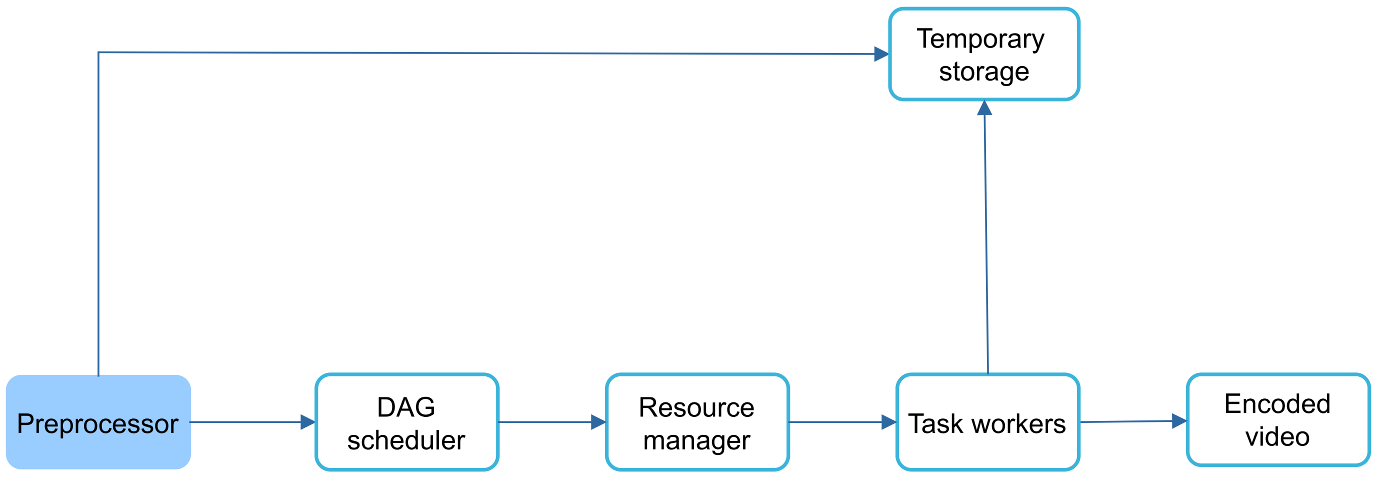


Figure 11 图 11

The preprocessor has 4 responsibilities:

预处理器有 4 项职责：

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.

1. 视频拆分。视频流被拆分或进一步拆分为较小的图片组 (GOP) 对齐。GOP 是按特定顺序排列的一组/块帧。每个块都是一个可独立播放的单元，通常长度为几秒。

2. Some old mobile devices or browsers might not support video splitting. Preprocessor split videos by GOP alignment for old clients.

2. 一些旧的移动设备或浏览器可能不支持视频拆分。预处理器按 GOP 对齐为旧客户端拆分视频。

3. DAG generation. The processor generates DAG based on configuration files client programmers write. Figure 12 is a simplified DAG representation which has 2 nodes and 1 edge:

3. DAG 生成。处理器根据客户端程序员编写的配置文件生成 DAG。图 12 是一个简化的 DAG 表示，它有 2 个节点和 1 条边：

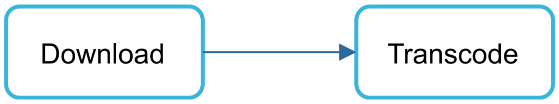
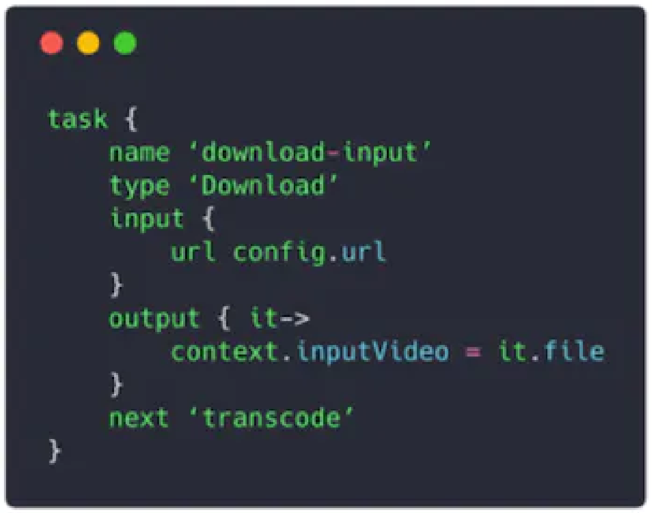


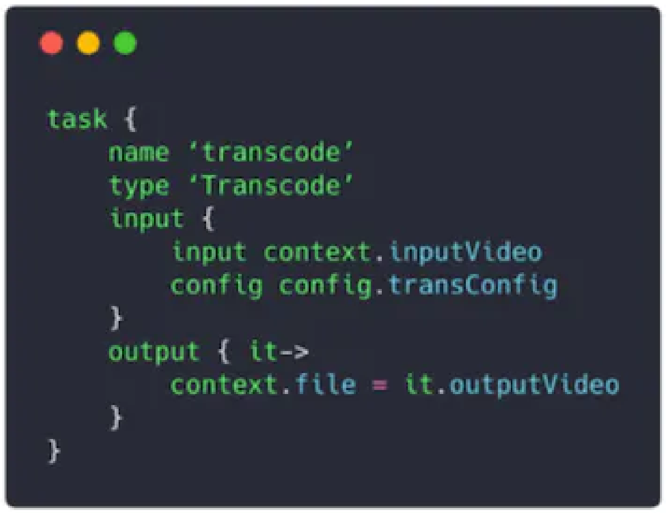
Figure 12 图 12

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

此 DAG 表示从以下两个配置文件生成（图 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 13 (source: [9]) 图 13（来源：[9]）

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.

4. 缓存数据。预处理器是分段视频的缓存。为了提高可靠性，预处理器将 GOP 和元数据存储在临时存储中。如果视频编码失败，系统可以使用持久数据进行重试操作。

DAG scheduler DAG 调度程序

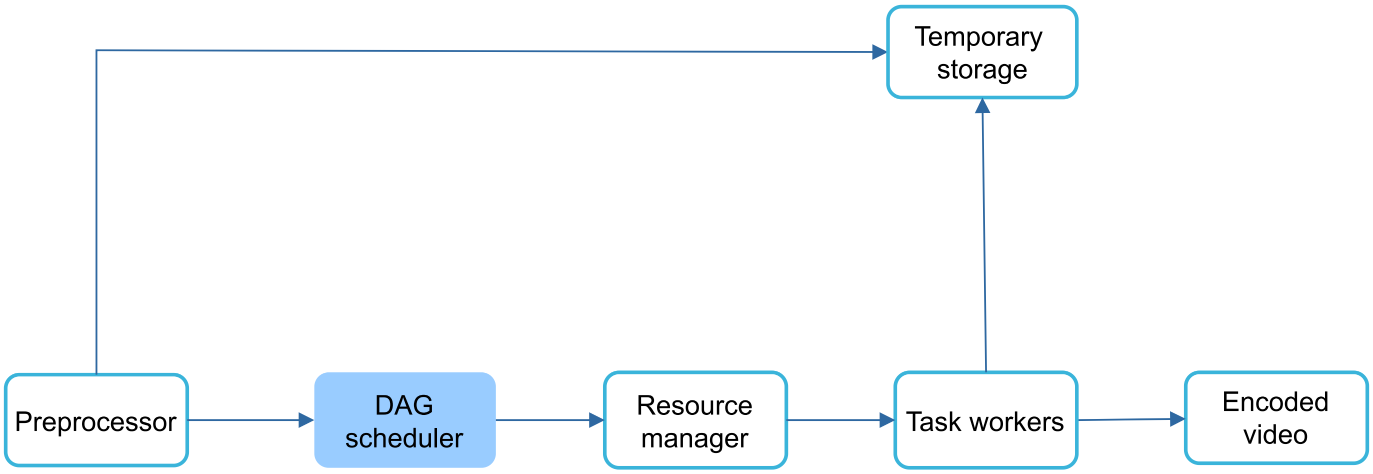


Figure 14 图 14

The DAG scheduler splits a DAG graph into stages of tasks and puts them in the task queue in the resource manager. Figure 15 shows an example of how the DAG scheduler works.

DAG 调度程序将 DAG 图表拆分为任务阶段，并将它们放入资源管理器中的任务队列中。图 15 展示了 DAG 调度程序工作方式的一个示例。

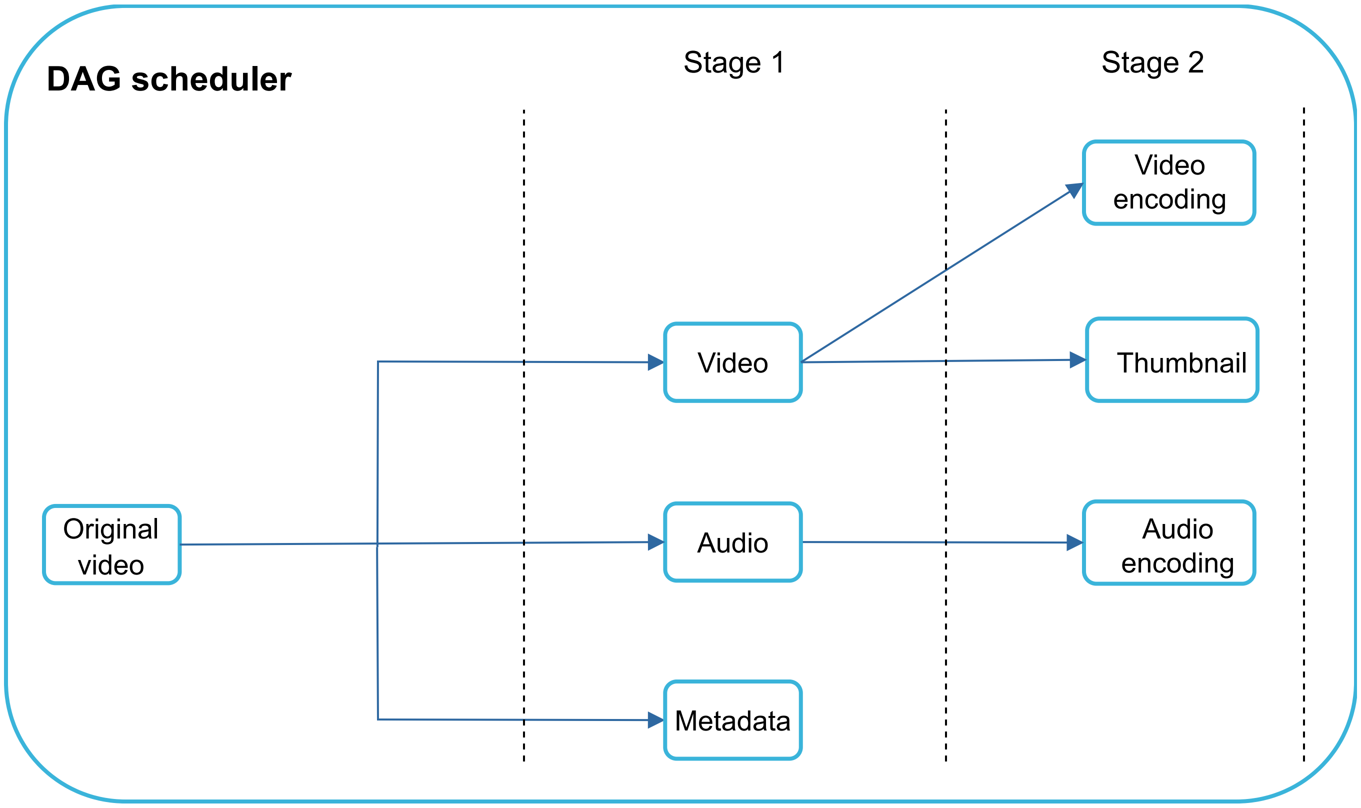


Figure 15 图 15

As shown in Figure 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.

如图 15 所示，原始视频被拆分为三个阶段：阶段 1：视频、音频和元数据。视频文件在阶段 2 中进一步拆分为两个任务：视频编码和缩略图。音频文件需要音频编码作为阶段 2 任务的一部分。

Resource manager 资源管理器

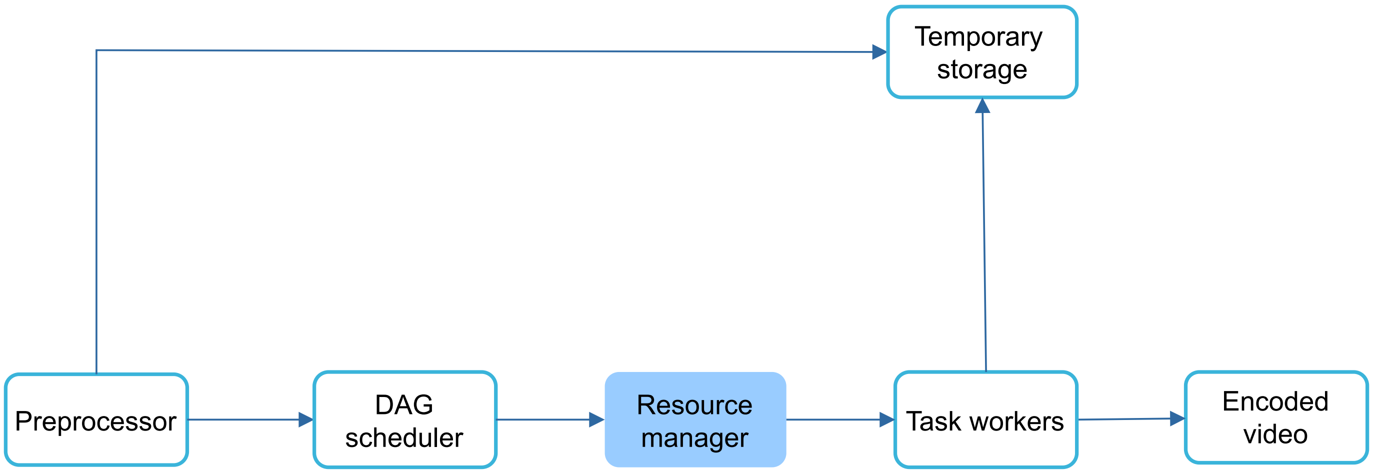


Figure 16 图 16

The resource manager is responsible for managing the efficiency of resource allocation. It contains 3 queues and a task scheduler as shown in Figure 17.

资源管理器负责管理资源分配的效率。它包含 3 个队列和一个任务调度程序，如图 17 所示。

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

  任务队列：它是一个包含要执行的任务的优先级队列。

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

  工作程序队列：它是一个包含工作程序利用率信息的优先级队列。

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

  正在运行的队列：它包含有关当前正在运行的任务和运行这些任务的工作程序的信息。

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

•任务调度程序：它选择最优任务/工作程序，并指示所选任务工作程序执行作业。

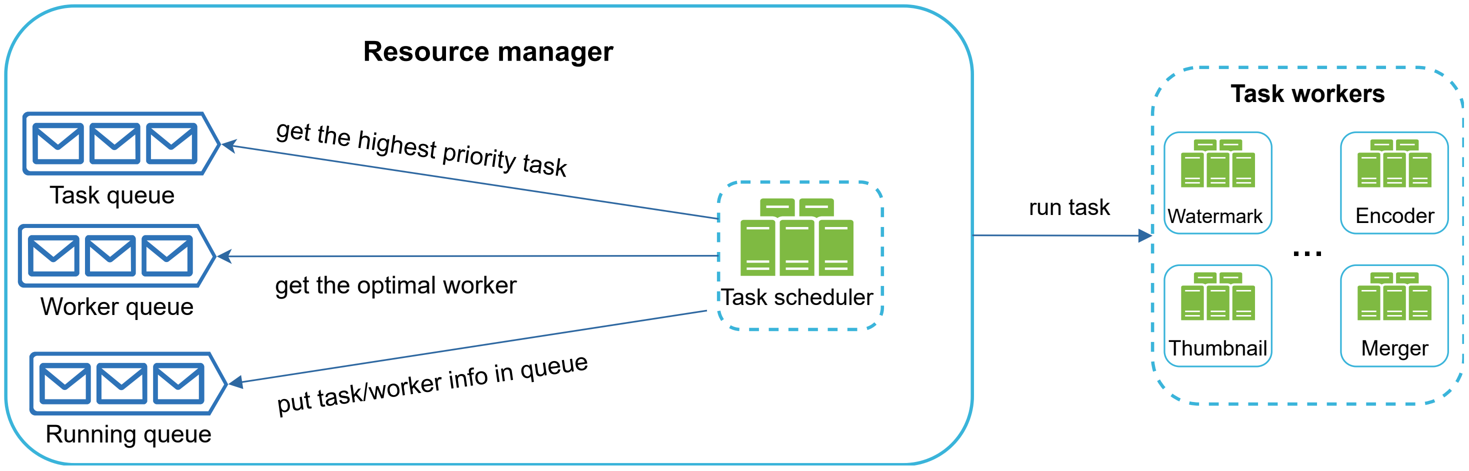


Figure 17 图 17

The resource manager works as follows:

资源管理器的工作方式如下：

- The task scheduler gets the highest priority task from the task queue.

  任务调度程序从任务队列中获取最高优先级的任务。

- The task scheduler gets the optimal task worker to run the task from the worker queue.

  任务调度器从工作队列中获取最佳任务工作器来运行任务。

- The task scheduler instructs the chosen task worker to run the task.

  任务调度器指示选定的任务工作器运行任务。

- The task scheduler binds the task/worker info and puts it in the running queue.

  任务调度器绑定任务/工作器信息，并将其放入运行队列中。

- The task scheduler removes the job from the running queue once the job is done.

  任务完成后，任务调度器会从运行队列中删除该任务。

Task workers 任务工作器

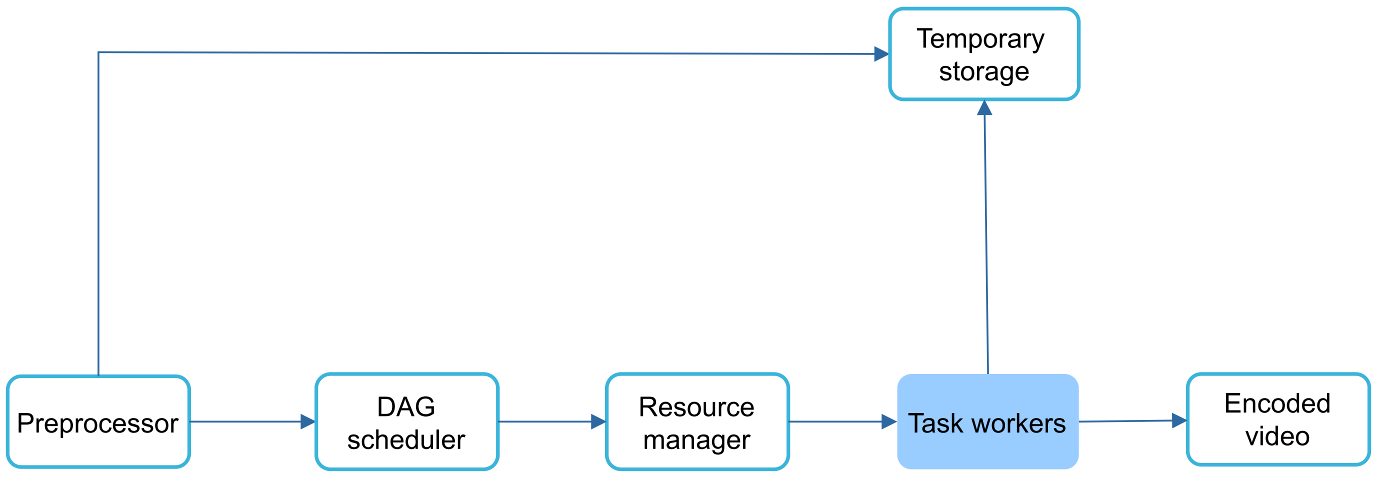


Figure 18 图 18

Task workers run the tasks which are defined in the DAG. Different task workers may run different tasks as shown in Figure 19.

任务工作器运行 DAG 中定义的任务。不同的任务工作器可能运行不同的任务，如图 19 所示。

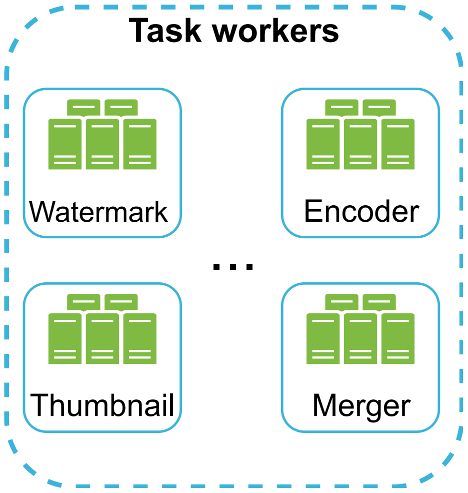


Figure 19 图 19

Temporary storage 临时存储

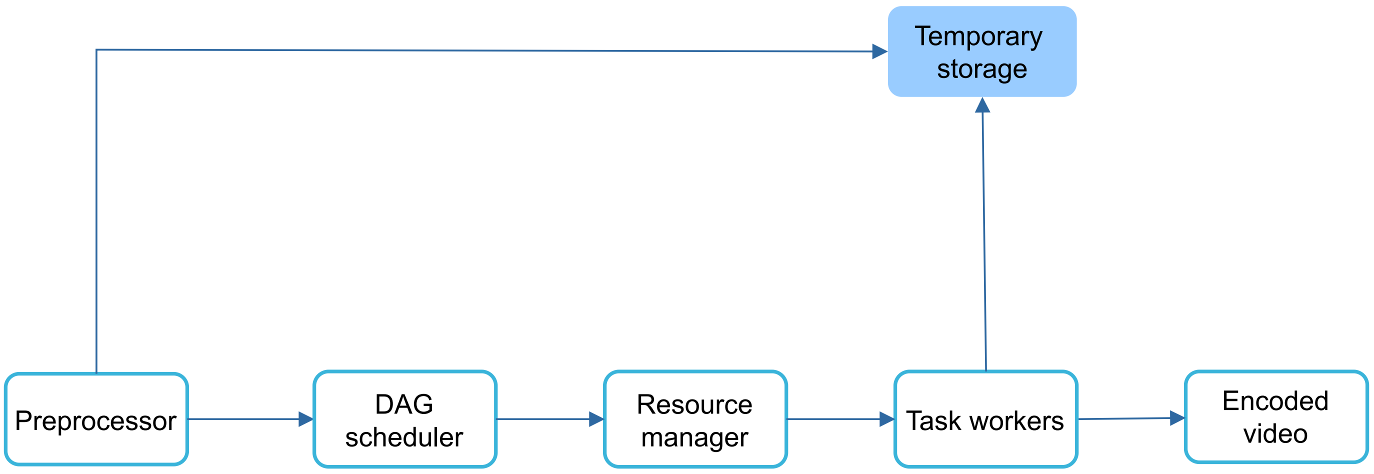


Figure 20 图 20

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. For instance, 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.

这里使用了多个存储系统。存储系统选择取决于数据类型、数据大小、访问频率、数据生命周期等因素。例如，元数据经常被工作器访问，并且数据大小通常很小。因此，将元数据缓存到内存中是一个好主意。对于视频或音频数据，我们将其放入 Blob 存储中。临时存储中的数据在相应的视频处理完成后将被释放。

Encoded video 编码视频

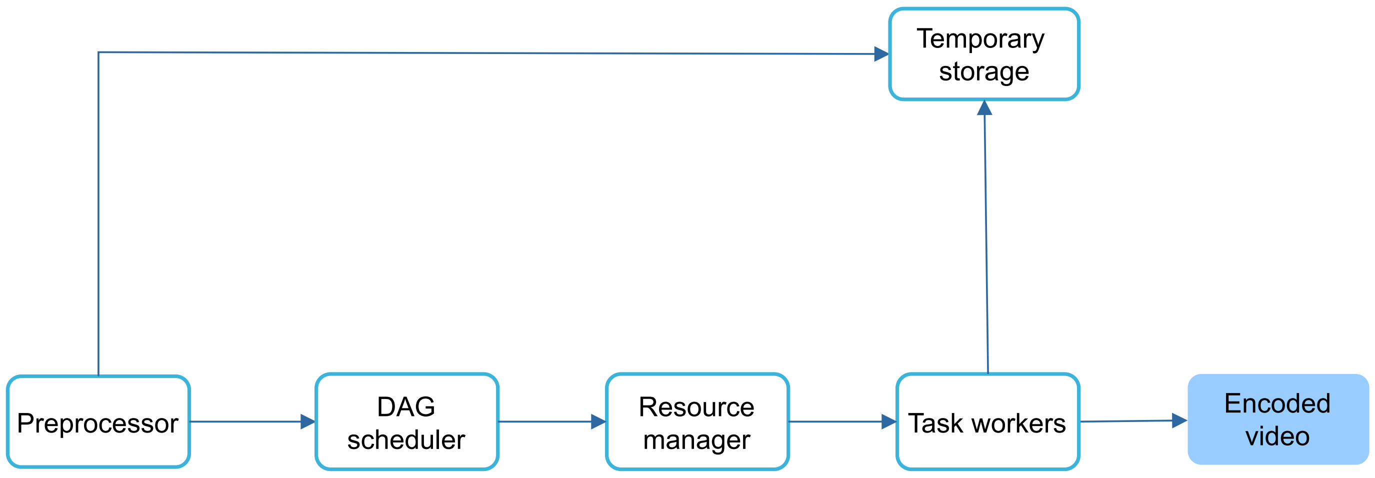


Figure 21 图 21

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

编码视频是编码管道的最终输出。以下是一个输出示例：funny_720p.mp4。

System optimizations 系统优化

At this point, you ought to have good understanding about the video uploading flow, video streaming flow and video transcoding. Next, we will refine the system with optimizations, including speed, safety, and cost-saving.

此时，你应该对视频上传流程、视频流流程和视频转码有了很好的理解。接下来，我们将通过优化来完善系统，包括速度、安全性和成本节约。

Speed optimization: parallelize video uploading

速度优化：并行化视频上传

Uploading a video as a whole unit is inefficient. We can split a video into smaller chunks by GOP alignment as shown in Figure 22.

将视频作为一个整体上传效率低下。我们可以按照 GOP 对齐将视频分割成更小的块，如图 22 所示。

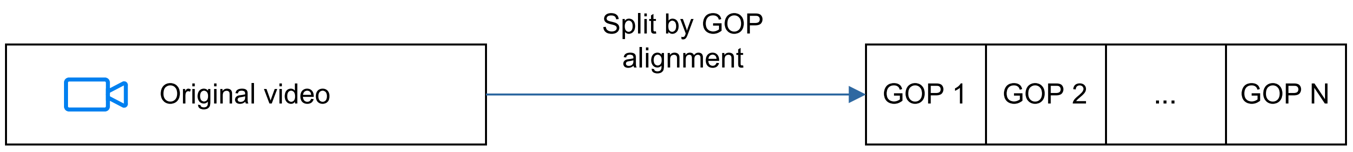


Figure 22 图 22

This allows fast resumable uploads when the previous upload failed. The job of splitting a video file by GOP can be implemented by the client to improve the upload speed as shown in Figure 23.

当之前的上传失败时，这允许快速恢复上传。通过客户端可以实现按 GOP 分割视频文件的工作，以提高上传速度，如图 23 所示。

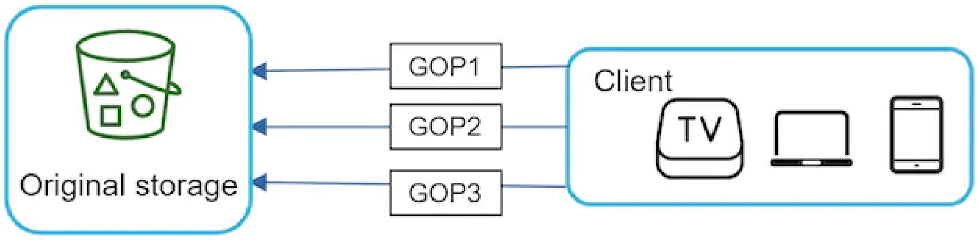


Figure 23 图 23

Speed optimization: place upload centers close to users

速度优化：将上传中心放置在靠近用户的位置

Another way to improve the upload speed is by setting up multiple upload centers across the globe (Figure 24). People in the United States can upload videos to the North America upload center, and people in China can upload videos to the Asian upload center. To achieve this, we use CDN as upload centers.

提高上传速度的另一种方法是在全球范围内设置多个上传中心（图 24）。美国的人们可以将视频上传到北美上传中心，中国的人们可以将视频上传到亚洲上传中心。为此，我们使用 CDN 作为上传中心。

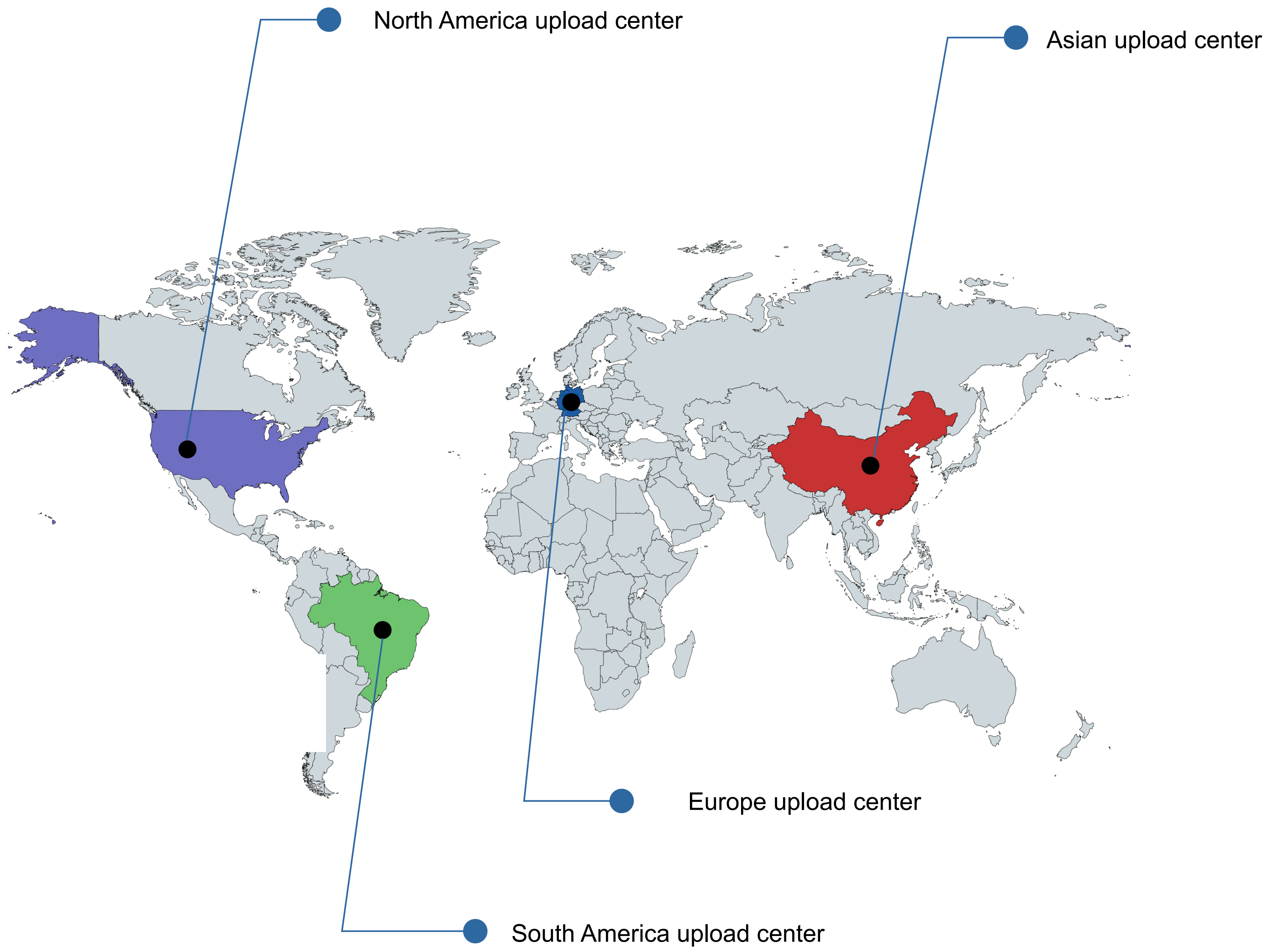


Figure 24 图 24

Speed optimization: parallelism everywhere

速度优化：无处不在的并行性

Achieving low latency requires serious efforts. Another optimization is to build a loosely coupled system and enable high parallelism.

实现低延迟需要付出巨大的努力。另一个优化是构建一个松散耦合的系统并启用高并行性。

Our design needs some modifications to achieve high parallelism. Let us zoom in to the flow of how a video is transferred from original storage to the CDN. The flow is shown in Figure 25, revealing that the output depends on the input of the previous step. This dependency makes parallelism difficult.

我们的设计需要一些修改才能实现高并行性。让我们深入了解视频如何从原始存储传输到 CDN 的流程。流程如图 25 所示，显示输出取决于前一步骤的输入。这种依赖关系使得并行性变得困难。

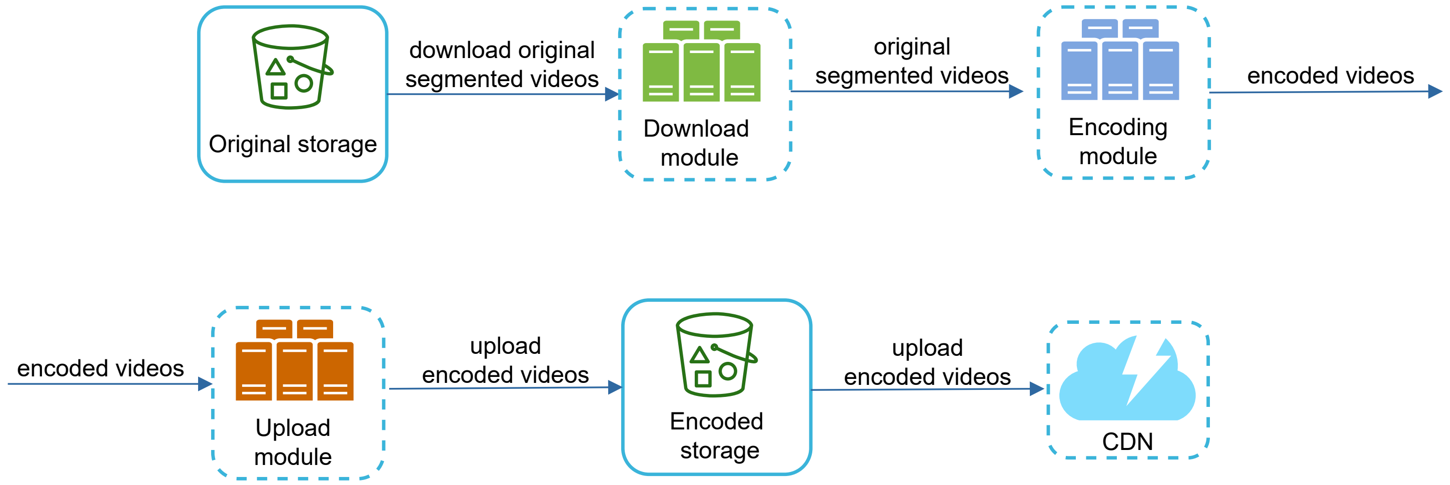


Figure 25 图 25

To make the system more loosely coupled, we introduced message queues as shown in Figure 26. Let us use an example to explain how message queues make the system more loosely coupled.

为了使系统更加松散耦合，我们引入了如图 26 所示的消息队列。让我们用一个示例来说明消息队列如何使系统更加松散耦合。

- Before the message queue is introduced, the encoding module must wait for the output of the download module.

  在引入消息队列之前，编码模块必须等待下载模块的输出。

- 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.

  在引入消息队列之后，编码模块不再需要等待下载模块的输出了。如果消息队列中有事件，则编码模块可以并行执行这些作业。

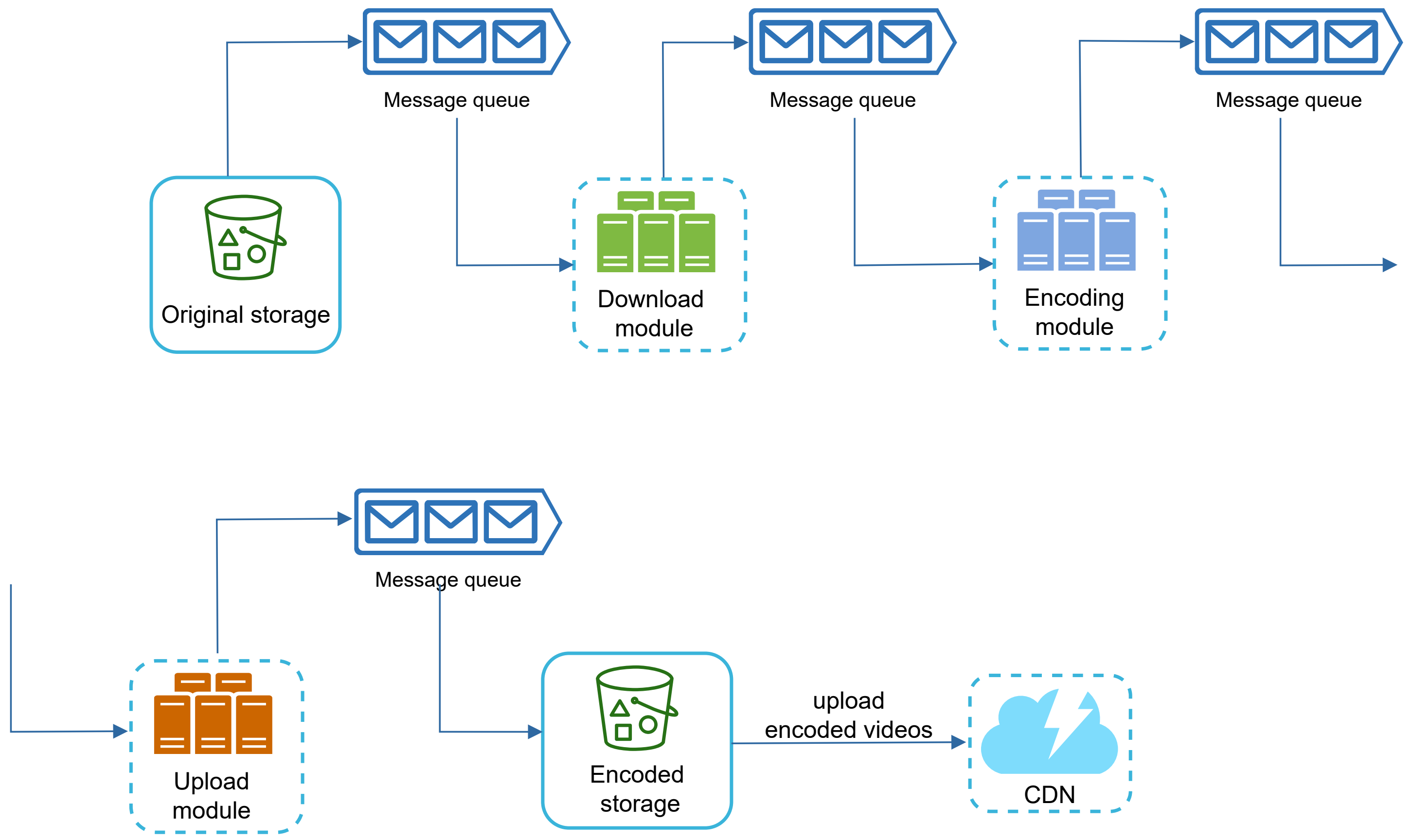


Figure 26 图 26

Safety optimization: pre-signed upload URL

安全性优化：预签名上传 URL

Safety is one of the most important aspects of any product. To ensure only authorized users upload videos to the right location, we introduce pre-signed URLs as shown in Figure 27.

安全是任何产品最重要的方面之一。为了确保只有授权用户将视频上传到正确的位置，我们引入了预签名 URL，如图 27 所示。

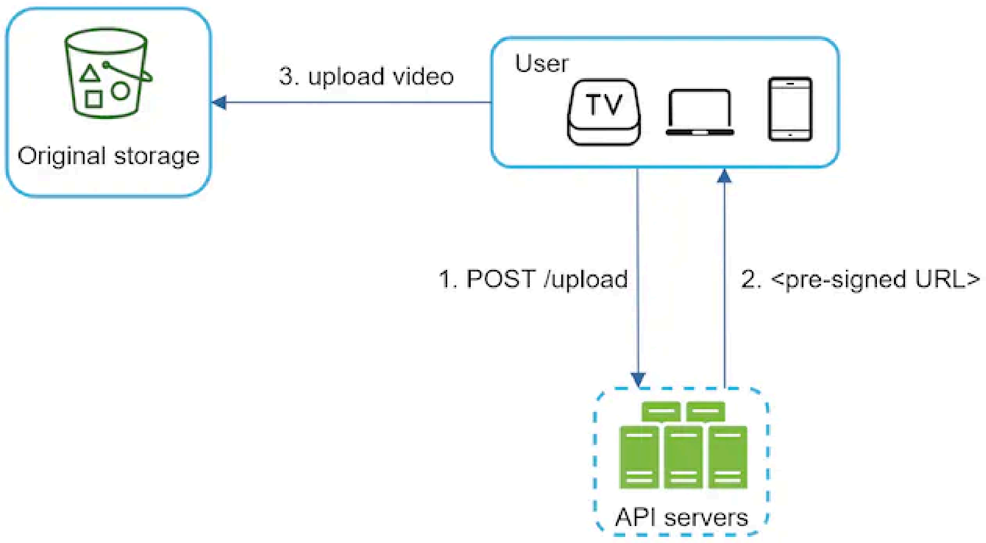


Figure 27 图 27

The upload flow is updated as follows:

上传流程更新如下：

1. The client makes a HTTP request to API servers to fetch the pre-signed URL, which gives the access permission to the object identified in the URL. The term pre-signed URL is used by uploading files to Amazon S3. 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].

1. 客户端向 API 服务器发出 HTTP 请求以获取预签名 URL，该 URL 授予对 URL 中标识的对象的访问权限。预签名 URL 一词用于将文件上传到 Amazon S3。其他云服务提供商可能使用不同的名称。例如，Microsoft Azure Blob 存储支持相同的功能，但称之为"共享访问签名"[10]。

2. API servers respond with a pre-signed URL.

2. API 服务器使用预签名 URL 响应。

3. Once the client receives the response, it uploads the video using the pre-signed URL.

3. 客户端收到响应后，使用预签名 URL 上传视频。

Safety optimization: protect your videos

安全优化：保护您的视频

Many content makers are reluctant to post videos online because they fear their original videos will be stolen. To protect copyrighted videos, we can adopt one of the following three safety options:

许多内容创作者不愿意在线发布视频，因为他们担心自己的原创视频会被盗用。为了保护受版权保护的视频，我们可以采用以下三种安全选项之一：

- Digital rights management (DRM) systems: Three major DRM systems are Apple FairPlay, Google Widevine, and Microsoft PlayReady.

  数字版权管理 (DRM) 系统：三大 DRM 系统是 Apple FairPlay、Google Widevine 和 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.

  AES 加密：您可以加密视频并配置授权策略。加密的视频将在播放时解密。这确保只有授权用户才能观看加密视频。

- 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.

  视觉水印：这是视频顶部的图像叠加，其中包含视频的识别信息。它可以是您的公司徽标或公司名称。

Cost-saving optimization

节省成本的优化

CDN is a crucial component of our system. It ensures fast video delivery on a global scale. However, from the back of the envelope calculation, we know CDN is expensive, especially when the data size is large. How can we reduce the cost?

CDN 是我们系统的一个关键组成部分。它确保在全球范围内快速交付视频。然而，从封底计算的背面，我们知道 CDN 很昂贵，尤其是在数据量大的情况下。我们如何降低成本？

Previous research shows that YouTube video streams follow long-tail distribution [11] [12]. It means a few popular videos are accessed frequently but many others have few or no viewers. Based on this observation, we implement a few optimizations:

先前的研究表明，YouTube 视频流遵循长尾分布 [11] [12]。这意味着少数热门视频被频繁访问，但许多其他视频的观看者很少或没有。基于此观察，我们实施了一些优化：

1. Only serve the most popular videos from CDN and other videos from our high capacity storage video servers (Figure 28).

1. 仅从 CDN 提供最热门的视频，从我们的高容量存储视频服务器提供其他视频（图 28）。

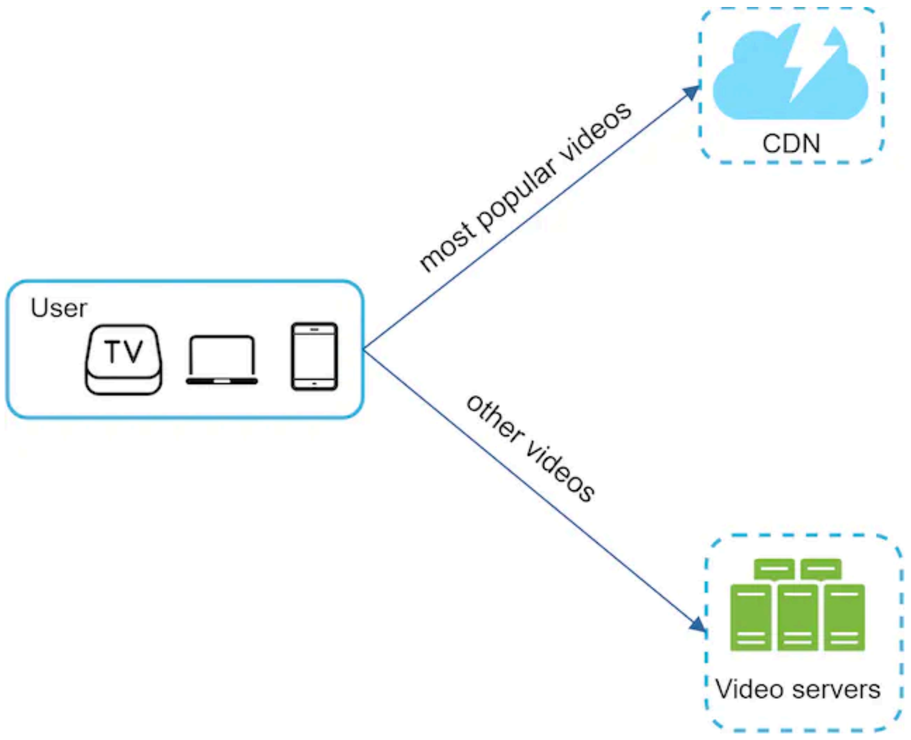


Figure 28 图 28

2. For less popular content, we may not need to store many encoded video versions. Short videos can be encoded on-demand.

2. 对于不太受欢迎的内容，我们可能不需要存储许多编码的视频版本。短视频可以按需编码。

3. Some videos are popular only in certain regions. There is no need to distribute these videos to other regions.

3. 某些视频仅在特定区域流行。无需将这些视频分发到其他区域。

4. 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.

4. 构建您自己的 CDN，如 Netflix，并与互联网服务提供商 (ISP) 合作。构建您的 CDN 是一个巨大的项目；但是，对于大型流媒体公司来说，这可能是有意义的。ISP 可以是 Comcast、AT&T、Verizon 或其他互联网提供商。ISP 遍布全球，并且靠近用户。通过与 ISP 合作，您可以改善观看体验并减少带宽费用。

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. Here are some of the interesting articles on this topic: [12] [13].

所有这些优化都基于内容流行度、用户访问模式、视频大小等。在进行任何优化之前，分析历史观看模式非常重要。以下是有关此主题的一些有趣文章：[12] [13]。

Error handling 错误处理

For a large-scale system, system errors are unavoidable. To build a highly fault-tolerant system, we must handle errors gracefully and recover from them fast. Two types of errors exist:

对于大规模系统，系统错误是不可避免的。为了构建一个高度容错的系统，我们必须妥善处理错误并快速从中恢复。存在两种类型的错误：

- 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.

  可恢复错误。对于可恢复错误（例如视频片段无法转码），一般做法是重试该操作几次。如果任务继续失败，并且系统认为无法恢复，则会向客户端返回适当的错误代码。

- Non-recoverable error. For non-recoverable errors such as malformed video format, the system stops the running tasks associated with the video and returns the proper error code to the client.

  不可恢复错误。对于不可恢复错误（例如视频格式错误），系统将停止与视频关联的正在运行的任务，并向客户端返回适当的错误代码。

Typical errors for each system component are covered by the following playbook:

以下手册涵盖了每个系统组件的典型错误：

- Upload error: retry a few times.

  上传错误：重试几次。

- 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.

  分割视频错误：如果旧版本的客户端无法按 GOP 对齐方式分割视频，则整个视频将传递给服务器。分割视频的工作在服务器端完成。

- Transcoding error: retry.

  转码错误：重试。

- Preprocessor error: regenerate DAG diagram.

  预处理器错误：重新生成 DAG 图。