Google Cloud Platform

Digital Media Asset Management and Sharing

Introduction

Digital media is one of the fastest growing areas on the internet. According to a market study by Informa Telecoms & Media conducted in 2012, the global online video market only, will reach $37 billion in 2017[1]. Other common media types include images, music, and digital documents. One driving force for this phenomena growth is the popularity of feature rich mobile devices2, equipped with higher resolution cameras, bigger screens, and faster data connections. This has led to a massive increase in media content production and consumption. Another driving force is the trend among many social networks to incorporate media sharing as a core feature in their systems[2]. Meanwhile, numerous startup companies are trying to build their own niche areas in this market.

This paper will use an example scenario to provide a technical deep-dive on how to use Google Cloud Platform to build a digital media asset management and sharing system.

1. OTT Video Revenue Forecasts, 2011-2017, by Informa Telecoms & Media, November 2012.

2. Key trends and Takeaways in Digital Media Market, by Abhay Paliwal, March 2012.

Example Scenario - Photofeed

Photofeed, a fictitious start-up company, is interested in building a photo sharing application that allows users to upload and share photos with each other. This application also includes a social aspect and allows people to post comments about photos. Photofeed's product team believes that in order for them to be competitive in this space, users must be able to upload, view, and edit photos quickly, securely and with great user experiences. Additionally, they would like this application to easily scale as the number of users and photos increases. In order for these goals to be achieved, the system must also have an efficient pipeline for photo processing capabilities, such as resizing, cropping, and thumbnail generation. As the business grows, the system must allow the development team to rapidly introduce new features.

Challenges In Building Scalable Digital Media Systems

Building a scalable digital media system from scratch that supports a large number of users and stores huge amount of media content is not a trivial task. The following list provides an overview of the common technical challenges associated with building scalable digital media systems:

Ingestion

- The system must allow end users to quickly and securely upload media objects while still providing a compelling user experience.
- Metadata of the media objects needs to be ingested and synchronized if media objects are modified or re-ingested.
- The ingestion workflow that defines the communication among all involved components needs to be managed.

Storage

- Virtually unlimited storage for the media content and the storage must be reliable, globally accessible, and cost effective[3].

Processing

- Scalable computing resources are required for media processing, such as document format conversion, image processing, and media transcoding.
- The media processing workflow needs to be managed.

Serving

- The system must allow end users to quickly and securely download media content while still providing a good user experience.
- The serving workflow needs to be managed.

Media Applications

- The system supports the integration of media metadata with application specific domain data. It also allows for the development of scalable media applications, such as asset management, and content sharing, on top of this data.

End User Experiences

- The system provides compelling user experiences for multiple clients such as browsers, mobile devices, and desktop applications.

The solution presented in this paper demonstrates how the Google Cloud Platform is able to address each of the challenges described above. The proposed system architecture is generally applicable to any media type. The solution serves as a reference for software architects and software developers for building their own digital media systems on the Google Cloud Platform.

3. Garnter: Consumers Will Drive Huge Growth for Cloud Storage, by Colleen Miller, July 2012.

Solution Overview

Google App Engine, Google Cloud Storage, and Google Compute Engine are the three features of the Google Cloud Platform. As shown in Figure 1, all of these products work together to form the basis of this digital media asset management and sharing solution.

Ingestion

Both Google Cloud Storage and Google App Engine play a critical role in media content ingestion. During

uploading, media content flows directly from the client, through the global Google network into the Google Cloud Storage. With its global reach, massive bandwidth, and integration with Google Cloud Storage, the Google network allows content to be ingested into the storage with low latency from almost anywhere. Google Cloud Storage supports two common uploading mechanisms: HTTP POST using signed URL and RESTful APIs.

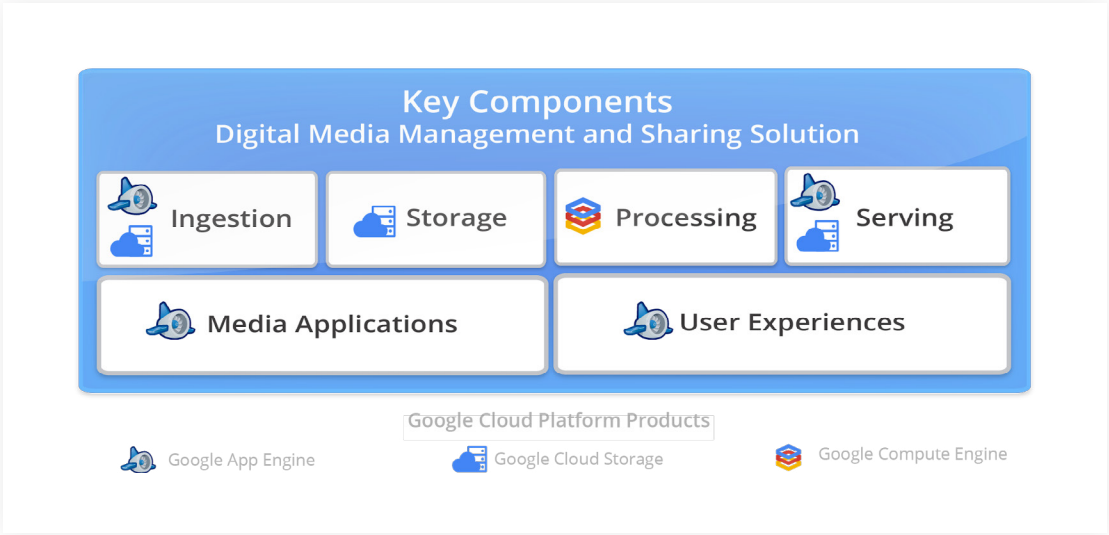


Figure 1.
Key components of proposed Digital Media Asset Management and Sharing Solution

Google App Engine is designed to power scalable web applications that handle millions of users. Front end application for content ingestion can be developed on App Engine. The application is responsible for authentication, allowing only authorized users to upload content. Meanwhile, the application manages the ingestion workflow and coordinates with the clients to upload content to Google Cloud Storage. For browser clients, the application also implements the web user interface for content uploading. For mobile or desktop clients, the user interface resides in the client application while the App Engine application exposes its functionality as RESTful APIs using Google Cloud Endpoints. The client side applications make calls to the APIs for authentication and for gaining access to Google Cloud Storage.

Another important role of the App Engine application is ingesting metadata and keeping it in sync with the media content. The metadata is stored along with the application data in the App Engine Datastore or in the Google Cloud SQL database. The decision about which storage option to choose from depends on the characteristics of your application. There are a few ways to synchronize metadata ingestion with media content ingestion, for example, (1) by using Blobstore upload callback URL, (2) by using the Cloud Storage Object Change Notification, or (3) simply by exposing appropriate APIs from the App Engine application using Google Cloud Endpoints.

Storage

Google Cloud Storage provides virtually unlimited storage for media content at low cost. The media data is replicated across data centers for redundancy. By leveraging the Google network, the content in Google Cloud Storage is globally accessible from Google App Engine, from Google Compute Engine, and from public internet outside the Google Cloud Platform.

Google Cloud Storage also provides a Durable Reduced Availability (DRA) storage option at an even lower cost. The tradeoff is lower availability compared to what standard Google Cloud Storage provides. The DRA option is useful for storing assets that are not always immediately required or can be regenerated. One example might be the output content from the media processing pipeline.

Processing

Google Compute Engine provides superior performance for batch computation. Media processing, such as document format conversion, transcoding, and image manipulation, is a perfect candidate for Compute Engine. In this case, Google Cloud Storage acts as both the input source and the output destination of the media processing pipeline. Since Google Cloud Storage is well integrated with Google Compute Engine, such as automatic authentication via service account, it can be easily accessed from Compute Engine.

The media processing workflow is also managed by the App Engine application mentioned previously. After media content is uploaded into storage, the App Engine application creates and inserts media processing tasks into TaskQueue. The enqueued tasks are pulled out by the media processing software running on Compute Engine using RESTful APIs and executed accordingly. The App Engine application can also maintain the processing status of the media content and the load information of virtual machines in order to autoscale the Compute Engine instances.

Serving

Google Cloud Storage leverages the Google network to allow media content to be served across the internet with low-latency and high-availability. The Google network automatically provides edge caching capability for public content, which can significantly lower the serving costs.

As is the case with ingestion, the Google App Engine application handles user authentication and authorization, and coordinates access to Google Cloud Storage from the clients. For browser clients, the App Engine application powers the web user interface for media content downloading. For mobile or desktop clients, the client-side applications implement the user interface and communicate with the App Engine application through APIs exposed using Google Cloud Endpoints.

Media Applications

Various media applications can be built with the availability of metadata and application data. Depending on the application domains, some common examples of media applications are asset management, content sharing, and social gaming. Google App Engine provides a scalable platform to build media applications. App Engine applications are easy to build, easy to maintain, and easy to scale as your traffic and data storage needs grow. This allows developers to focus on building their core business and bring new features to market quickly.

User Experiences

In this solution, the App Engine application plays a critical role in defining user experiences for the system. As mentioned earlier, for browser clients, the App Engine application implements the web user interface for ingestion, serving, and media applications. For mobile and desktop clients, the App Engine application exposes its functionality as APIs using Google Cloud Endpoints. The native user interface at the client side is powered by these APIs.

Implementation Details

The next section walks through the implementation details of the proposed digital media solution. It begins with a list of key components of the system and ends with a detailed presentation of the three important workflows of the system: media ingestion flow, media processing flow, and media serving flow.

System Components

Frontend and Media Applications Running on Google App Engine

- Authenticates and authorizes users and coordinate access to Google Cloud Storage.
- Implements the user interface for browser clients, and/or exposes APIs using Google Cloud Endpoints to mobile and desktop clients.
- Plays the role of system controller and is responsible for managing workflows for media ingestion, serving, and processing.
- Scalable media applications are powered by App Engine, with built-in load-balancing and auto-scaling.

Google Datastore

- Stores media content metadata and application data model.

Google Cloud SQL

- Stores media content metadata and application data model, as an alternative to Google Datastore.

App Engine Task Queue

- Integrates App Engine application with media processing software running on Google Compute Engine.

Image Services

- Provides dynamic image processing services for App Engine applications, such as thumbnail generation, resizing, and cropping.

Google Cloud Storage

- Provides scalable and highly available storage for media content. The storage can be accessed by using RESTful APIs and/or signed URLs.
- Leverages Google network for the following advantages: (1) to allow for fast and secure content ingestion into and serving from the storage and (2) for edge caching capability for public content which lowers the serving costs.

Media Processing Server

- Executes media processing on Google Compute Engine.

Media Ingestion Workflow and Media Processing Workflow

The media ingestion workflow and the media processing workflow are often tied together. Both workflows are shown in the component communication diagram in Fig 2.

1. The client accesses the Google App Engine application to start an upload. Depending on the type of clients, this request can be: (1) a simple HTTP request from the browser or (2) a call to an endpoint implemented by the App Engine application from a mobile or desktop application, such as a batch uploader. The App Engine application is responsible for authenticating the client/user and coordinate the Cloud Storage access.
2. If the client is a web browser, the application can generate a signed upload URL to the Cloud Storage embedded in an HTTP POST form. Otherwise, if the client is a mobile or desktop application, the web application returns Cloud Storage access information as an endpoint call response.
3. Regardless of the client, media files are uploaded to Google Cloud Storage directly by using either the web form or the Cloud Storage RESTful APIs.

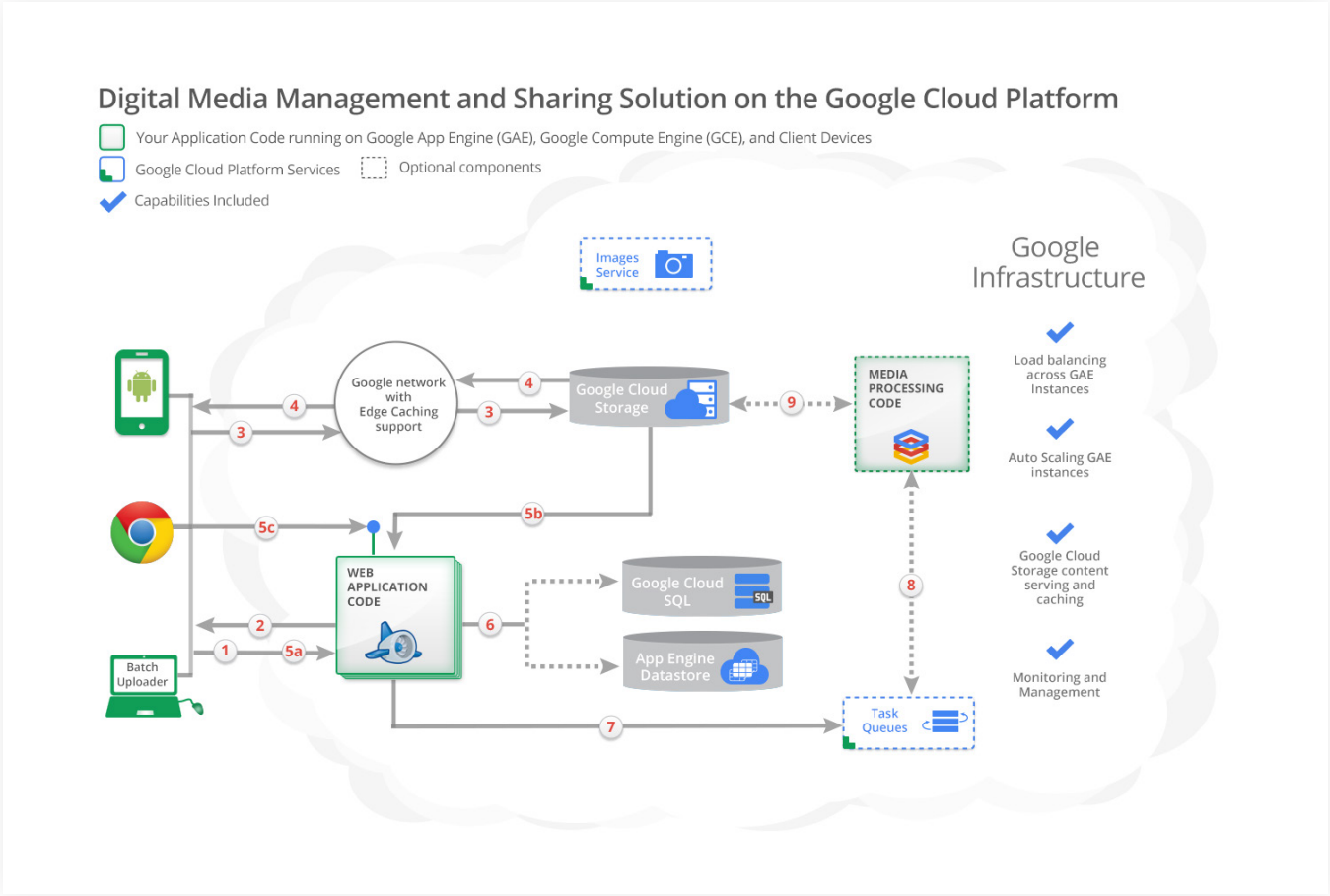


Figure 2. Media Ingestion and Media Processing Workflows

4. Google Cloud Storage returns a response back to the client. Depending on the uploading mechanism used in Step 3, the response can either be an HTTP response for form-based upload or a RESTful API response.

5. If the upload succeeds, the media metadata needs to be pushed into the App Engine application. There are a few different ways to streamline the process:

   - For browser clients using an upload form, a callback URL can be specified inside the upload URL. Based on the response, the browser can be redirected to this URL with limited metadata information embedded in the callback URL.
   - Google Cloud Storage can notify the App Engine application upon upload success using a Cloud Storage feature called Object Change Notification [1]. The notification contains metadata of the media object being uploaded.
   - Based on the content upload response from the Cloud Storage, clients can also call the App Engine application Google Cloud Endpoints directly to upload any metadata.

6. App Engine application stores the metadata in a persistent store. There are two options for the data stores depending on the application setup: (1) App Engine NoSQL Datastore, or (2) Google Cloud SQL.

7. If media processing is required, the App Engine application can create a task on the task queue in

[1] The Object Change Notification is currently still a feature under the Trusted Tester program.

order to start the media processing workflow. It is also possible for the App Engine application to spin up or bring down virtual machines based on the workload on demand.

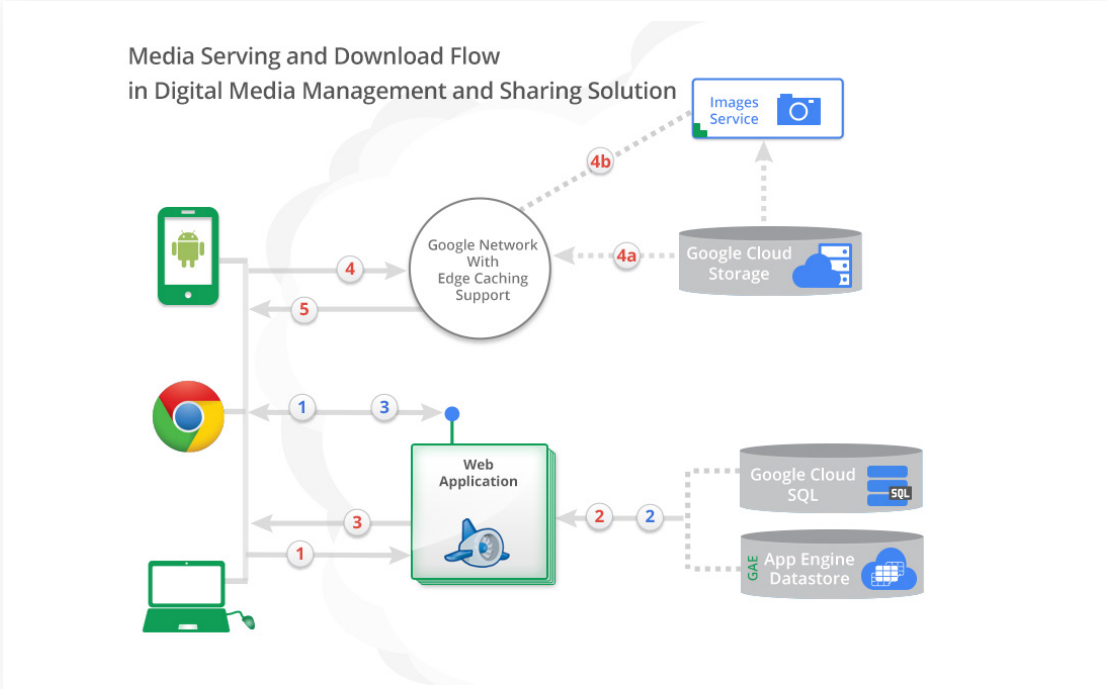
8. The media processing software, running on Google Compute Engine, pulls the task from the queue and executes the required procedures.

9. The media processing software reads the media content from Cloud Storage, processes it, and stores the output back to Cloud Storage.

Media Serving And Download Workflow

Figure 3 describes the media serving and download workflow and is accompanied by a list of detailed descriptions.

Figure 3.
Media Serving and Download Flow



1. The clients start the media download by contacting the App Engine application which authenticates and authorizes the clients and also allows for browsing and searching of specific media content. This can either be accomplished by presenting a web user interface for browser client or via a RESTful API provided by the App Engine application using Google Cloud Endpoints.

2. Based on the media metadata and application data in Datastore or Cloud SQL, the App Engine application can check the content sharing rules defined in the application, and look up access information for the content stored in Google Cloud Storage.

3. For content to be securely downloaded from Google Cloud Storage, the App Engine application can generate a signed URL or provide OAuth access token, along with the Cloud Storage bucket and object names to the client. For browsers, the information is embedded in the web user interface. For mobile and desktop clients, the information is returned in the response of the RESTful API mentioned in Step 1.

4. The clients make a request to the Google Cloud Storage to download the content by sending out HTTP or by calling the RESTful API. Google Cloud Storage can leverage the caching capability

of the Google network for public content. If the content is available in the cache, content is returned from the cache. Otherwise, the following occurs:

- The content is retrieved from the Google Cloud Storage and the cache is filled.
- App Engine application allows the content retrieved from Cloud Storage to be proxied through the App Engine Image Services for "on the fly" resizing and cropping images.

5. The media content is served to the client.

Implementation Considerations

- The metadata for the media content can be stored along with the application data, either in Google App Engine Datastore or in Google Cloud SQL. The choice depends on the size of the data, the characteristics of the overall data model, and the developer team's expertise. For example, you may want to choose Cloud SQL if you have highly relational data. Alternatively, you may want to choose Datastore if you are scaling denormalized data to a massive data set. The trade-off between the two options is well discussed in the Google IO 2012 session, SQL vs NoSQL: Battle of the Backends.
- The provided solution uses Google Compute Engine for media processing. Compute Engine allows running custom software and packages on supported operating systems. The platform is suitable for general purpose media processing. Alternatively, for simple image and photo manipulations, Google App Engine provides an Image Service that can perform image processing on the fly.

Sample Application

A sample photo sharing application[4] has been developed to demonstrate how a media asset management and sharing solution, like the one described earlier in the scenario, can be implemented. The photo sharing application allows a user to upload and make them available for other users to view. A user can also post comments for uploaded photos. The following list details the key elements of this use case scenario:

- A user is required to login with a valid Google account to use the application.
- A user uploads a photo and a description from a local disk.
- All photos uploaded into the photo sharing application are displayed in chronological order.
- The user adds comments, visible to all users, to any photo.
- When a photo is displayed, the image can be resized and cropped to fit into the user interface.

Detailed documentation of the requirements and design is provided here. The source code is hosted under Github.

4. Photo Sharing Sample Application, by Michael Tang, Oct, 2012.

Conclusion

The Google Cloud Platform enables developers to quickly build a digital media asset management and sharing solution that scales to millions of users and petabytes of data. The solution presented in this paper combines the power of Google App Engine, Google Compute Engine, and Google Cloud Storage to solve the technical challenges presented by digital media systems.