CSE 546 — Project Report

Cyriac Biju Narayamparambil

Aksa Elizabeth Sunny

Saran Prasad

1. Problem statement

Build a PaaS solution using a hybrid cloud architecture that uses Ceph (as object storage), OpenFaaS (as a private cloud) and Amazon Web Services (AWS) for performing facial recognition from video frames to recognize students and return relevant academic information for the same. Use OpenFaaS for PaaS like service, Ceph storage for persistence and DynamoDB (hosted in AWS) for storing academic information for students.

2. Design and implementation

The technical details are the following,

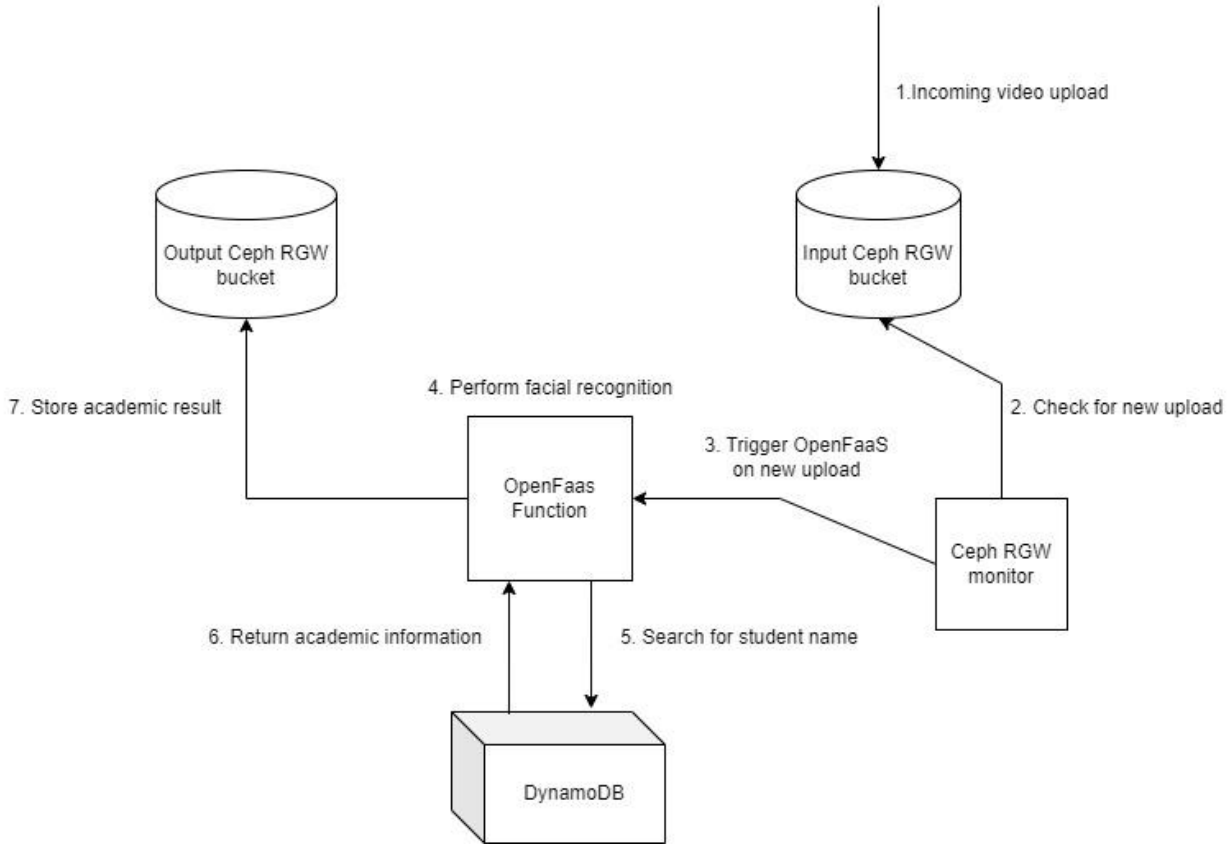
1. **Ceph RGW Buckets**
    a. Input Ceph RGW bucket is created to store input video files.
    b. Output Ceph RGW bucket is used to store the recognized students' academic information.
2. **OpenFaaS Function**
    a. The OpenFaaS function is set up as a container which is created according to specifications in the Dockerfile. This includes downloading dependencies, copying files and triggering the handler (endpoint).
    b. The handler (handler.py) receives the bucket name and uploaded file (key) name from the trigger service that monitors for uploads to Ceph RGW bucket.
    c. Using boto3, the handler downloads the .mp4 file from the input Ceph RGW bucket.
    d. After the download, the first frame is extracted using ffmpeg and stored as a .jpeg file.
    e. Face_recognition module is used to create an encoding for the first recognized face from the .jpeg file and compare against known encodings given.
    f. After the best match, the corresponding student's name is obtained.

3. **Trigger (Monitoring) Service**

a. This service is responsible for invoking OpenFaaS functions and providing the invoked function with context like bucket name and key uploaded.
b. Periodically monitors input Ceph RGW bucket to see if any new uploads have been performed.
c. Ensures not to invoke OpenFaaS function for the same upload more than once.

4. **Search using DynamoDB**
    a. Table is created in DynamoDB (Student-Data) and populated with the student data provided in student_data.json.
    b. Student name is used as the partition key for the table.
    c. Using boto3, the handler searches and retrieves the student name matched using facial recognition in DynamoDB if it exists.
    d. Only the name, major and year are retrieved from the database for the matching record.
    e. The retrieved data is stored as a csv file in the output bucket in Ceph RGW using boto3 with the name of the input file as the key.

2.1 Architecture



Steps in process pipeline:

1. A video (.mp4) is uploaded into the input Ceph RGW bucket by a client.
2. Ceph RGW bucket stores the video file.
3. A Ceph RGW trigger (monitoring) service detects the new upload and invokes an OpenFaaS function configured
4. The OpenFaaS function receives the invocation and performs facial recognition. This is done using ffmpeg to extract the first frame of the mp4 and perform facial recognition to identify the first face in the corresponding frame.
5. The identified student's (by comparing face encodings of known students against the identified face from first frame) name is searched in DynamoDB (AWS).
6. If there is a match, DynamoDB returns the corresponding student's academic information.
7. The academic information is stored in an output Ceph RGW bucket for persistence.

2.2 Autoscaling

Autoscaling is achieved using OpenFaaS function scaling parameters. We first build a dockerized image that is pushed to a public repository. The OpenFaaS function is given that prebuilt image

as the entity to be executed. We also set a trigger for the OpenFaaS function such that the function starts up when an object is created in the input bucket (using trigger/monitoring service). Therefore, whenever a new object is created in the input Ceph RGW bucket, a new OpenFaaS function instance is created and executed that does the face recognition task and sends output to the output Ceph RGW bucket. Thus, the number of functions is proportional to the number of input objects given.

2.3 Member Tasks

- **Saran Prasad**
    - Developed the trigger (monitoring) service that invokes OpenFaaS functions when new uploads are detected in Ceph RGW bucket.
    - Developed code for the handler function (handler.py) executed within OpenFaaS functions for handling incoming .mp4 file and its conversion to a .jpeg corresponding to the first frame.
    - Developed code for facial recognition and matching to a known list of students.
- **Cyriac Biju Narayamparambil**
    - Set up Ceph configurations in VM.
    - Configured RGW buckets and connected AWS CLI with it.
    - Minikube installation and setup.
    -
- **Aksa Elizabeth Sunny**
    - Setup OpenFaaS in VM
    - Created OpenFaaS function.
    - Created DynamoDB table and initialized with academic information.

3. Testing and evaluation

The code was tested both manually and using the automated workload.py tool.

3.1. Manual testing

Some video files were randomly chosen from the test_cases folder and uploaded into the input Ceph RGW bucket via local clients. The OpenFaaS functions were triggered as expected and output was verified manually in the output Ceph RGW bucket.

3.2. Automated testing using workload.py

cmd: *python3 workload.py*

This runs the entire set of test cases.

All the run test cases had corresponding outputs in Ceph RGW bucket and were verified to be correct.

4. Code

From the base (forked) project, only the following were changed

1. Dockerfile - The following COPY command was added,

   COPY encoding ${FUNCTION_DIR}

2. trigger.py - This is the monitoring service that invokes OpenFaaS functions when new uploads are seen in input Ceph RGW bucket. It,
   a. Periodically monitors the Ceph RGW bucket for new uploads by sorting the data in the bucket based on LastModifiedDate attribute.
   b. If new uploads are seen, the corresponding bucket and key names are used to invoke an OpenFaaS function using its invocation URLs.
3. handler.py **-** This is the entry point specified in Dockerfile. The python program gets invoked when a trigger is observed. From the triggered event, we obtain the bucket and key of the latest upload. Then, the following functions are done by the handler,
   a. Download video from input Ceph RGW bucket that caused the trigger
   b. Run ffmpeg on the video to obtain the first frame and store it as .jpeg
   c. Identify a face and produce its encoding using faciaL_recognition module.
   d. Compare this encoding against known encodings to identify the student name.
   e. Search this student's academic information in DynamoDB using the identified name.
   f. Retrieve and store the academic information in the output Ceph RGW bucket.
4. workload.py - The input and output bucket names were changed.