CSE 546 Cloud Computing

Project-2

Student Names:

Maharshi Patel

Sahith Doma

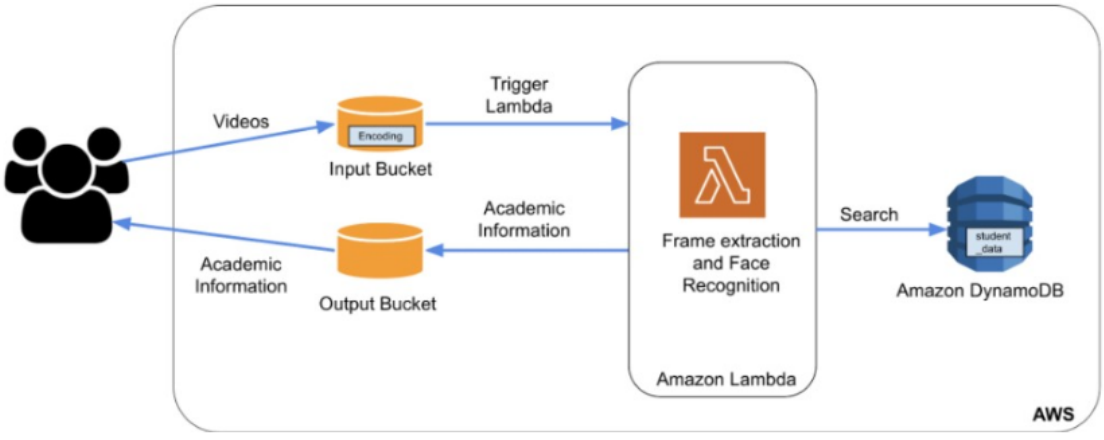
Nisarg Patel

1. Problem statement

Our objective is to create a dynamic face recognition application that can efficiently upscale and down scale as required using PaaS cloud technology. For this purpose, we will utilize AWS Lambda and other related services from Amazon Web Services. AWS Lambda is currently the prominent function-based serverless computing service available. Through this project, we aim to provide users with a valuable cloud service while also gaining valuable knowledge and skills that can be applied to future application development.

2. Design and implementation

2.1 Architecture



We have used the similar architecture that is asked in class to implement.

First the input bucket receives video from the user and stores in it. When the new video is added to the input bucket, it triggers the lambda function. This lambda function is built on a custom image recognition algorithm which identifies faces. This obtained results are used to fetch individual Academic information from the DynamoDB and returned to the output bucket. The data in the output bucket can be accessed by the user though logs or in the output bucket.

2.2 Autoscaling:

Since we are using lambda for face recognition it automatically upscales and downscales itself to handle the load. With single thread workload generator the application takes around six minutes to execute for 100 requests, while it is taking less than two minutes with the multithreaded workload generator for 100 requests.

2.3 Member Tasks:

Maharshi Patel:

- Created the initial frame of code for the project.
- Developed a logic to search the data frame that matches to generated frame in DynamoDB.
- Created the lambda function from the docker image file.
- Helped in Gathering resources for reference to configure AWS.

Nisarg Patel:

- Developed a code to load large amounts of data to the DynamoDB.
- Developed a logic to search first recognized face in DynamoDB using get item function
- Helped maharshi to debug the code and resolve issues in configuration.

Sahith Doma:

- Created input and output buckets for storage and data transmission.
- Made a detailed report by testing and noting the test results after executing the application with singlethreaded and multithreaded workload generators.
- Developed a code to upload the generated CSV file to the S3 bucket.

3. Testing and Evaluation:

When we tested the application by sending 100 requests though the single threaded workload generator, It took around 6minutes in which some time taken for clearing the input bucket.

When we tested the same application by sending 100 requests though miltithreaded workload generator, It took less than 4 minutes.

The time taken in both the cases is within the range provided in the assignment rubrics.

4. Code:

The **dockerfile** contains the implementation of an image that contains the necessary libraries to perform facial recognition on AWS. These libraries include **boto3**, which is the AWS SDK for Python, and facial_recognition, which is a Python library for facial recognition.

Most of the code in this project is in the **handler.py** file. This class is responsible to take input from the input bucket, retrieving the video file, processing video frames and matching it to the database, and returning the output in the output bucket.

When the video file is received we call the **ffmpeg** library which creates frames out of the video.

```
os.system("ffmpeg -i " + str(path_to_video) + " -r 1 " +
str(path_to_frames) + "image-%3d.jpeg")
```

This frames are matched with the existing frames in the database. Then we call the **face_recognition** library to get the encoded image of the frame. This encoded image is then matched to the known faces.

```
      final_result = fr.compare_faces(face_encoding['encoding'],
img_encoding)

      index = final_result.index(True)

      name = list(face_encoding['name'])[index]

      print(name)
```

If there is a match the academic details of the students is retrieved from **DynamoDB** and sent to the output bucket.

```
      student_table = dynamodb.Table(dynamodb_table)
      item = student_table.get_item(Key={'name': name})['Item']
```