CSE 546 Cloud Computing

Project-3

Student Names:

Maharshi Patel

Sahith Doma

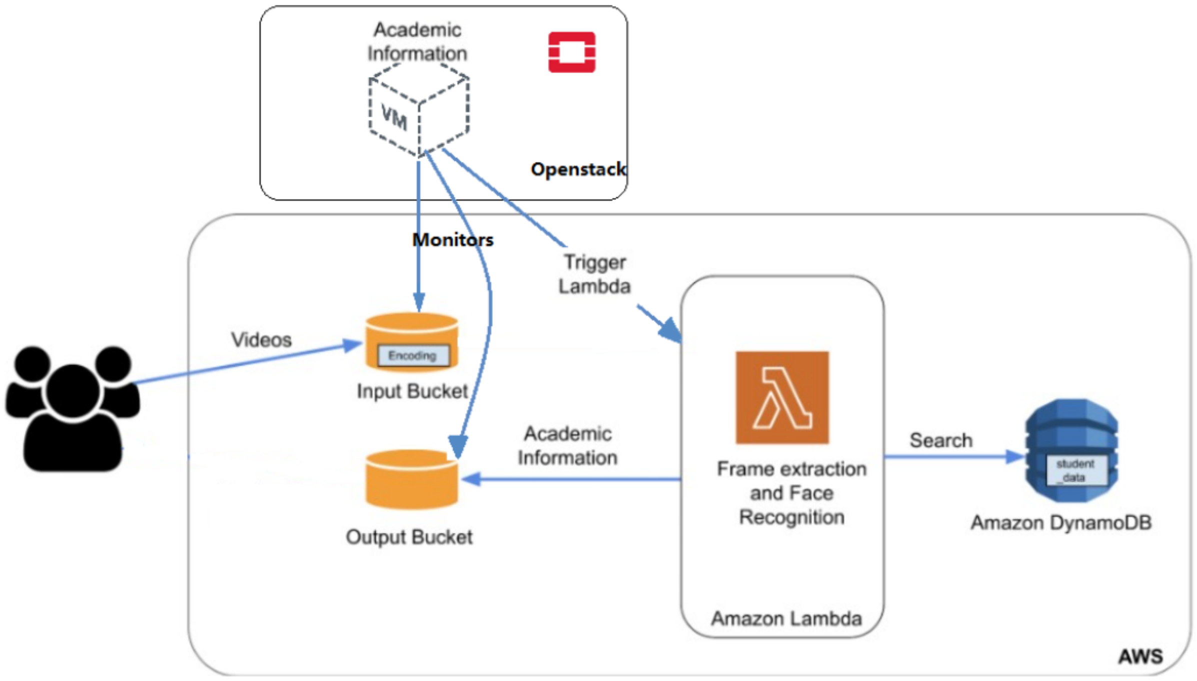
Nisarg Patel

1. Problem statement

Our goal is to cost-effectively and automatically scale up or down based on demand and deploy the elastic application developed in project-2 to a hybrid cloud environment. This application will be developed using resources from both AWS and OpenStack. AWS is a well-known cloud provider that offers a variety of computing, storage, and messaging services. OpenStack is a free and open standard cloud computing platform that is primarily used for IaaS in both public and private clouds, where users can access virtual servers and other resources. By launching this application, we intend to teach ourselves new technologies and methods that will help us create similar applications in the future while also giving users a useful cloud service.

2. Design and implementation

2.1 Architecture



We have 2 S3 buckers for input images and output data. A lambda function is created such as when passed the image path in the event datatype it will process the face recognition algorithm and compare results with DynamoDB Student data and will upload the output data to the S3 output bucket. Our Instance created in OpenStack VM will keep monitoring the S3 input bucker and when a new image is uploaded it will automatically trigger the lambda function. Our code running in an instance will also monitor the S3 output bucket and print the data on the command line.

2.2 Autoscaling:

Lambda automatically upscales and downscales itself to manage the load because it is used for face recognition. In Project 2 When using a single-thread workload generator, the application takes about six minutes to complete 100 requests, whereas when using a multithreaded workload generator, it completes 100 requests in under two minutes. Similarly, in Project 3 the application takes a few extra seconds compared to project2 to process the request as we have included OpenStack to trigger lambda when a new video is added to the S3 bucket.

2.3 Member Tasks:

Sahith:

- Installed Ubuntu in a VirtualBox for installing OpenStack.

- Worked on OpenStack configurations for running an instance.
- Made a detailed report by testing and noting the test results after executing the application.

Maharshi:

- Worked on writing scripts to trigger lambda with OpenStack.
- Helped in setting up OpenStack.
- Configured lambda function from ECR image and performed testing.

Nisarg:

- Worked on writing Python scripts to trigger to let OpenStack know whenever a video is added to S3.
- Helped in solving problems in OpenStack configuration.
- Worked on output s3 bucket monitoring.

3. Testing and Evaluation:

To test the OpenStack setup, we have created a VM in OpenStack and ensured that it is functioning correctly. We executed our code in the VM to monitor the S3 bucket and triggered the Lambda function to process the videos which are uploaded to the S3 input bucet. We also verified that the academic information generated by the Lambda function is correctly gathered by our S3 output bucket monitoring code running in the VM.

In the second part, we tested the functionality of the Lambda Function. We verified that the application can correctly recognize faces from the video using the Python face_recognition library. We also ensured that the Lambda function is correctly triggered when a new video is uploaded to the input bucket. We tested the application's ability to classify only the first detected face in the video. We verified that the Lambda function can correctly search for the person's academic information in DynamoDB using the name of the first recognized face.

We tested both test cases folder. When we tested the application by sending 100 requests through the single-threaded workload generator, It took around 7 minutes in which some time was taken for clearing the input bucket.

4. Code:

The **docker file** 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")
```

These 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']
```

main.py file is the script that will run on the OpenStack instance, will monitor s3 buckets, and also trigger the lambda function when a new item is being uploaded to the S3 input bucket.

This file is also monitoring the S3 output bucket in parallel and prints the new items data.

```
    event = {
                    'Records': [{
                       's3': {
                           'bucket': {
                               'name': bucket_name
                           },
                           'object': {
                               'key': key
                           }
                       }
                   }]
               }

lambda_client.invoke(FunctionName=lambda_function_name,
                                    InvocationType='Event',
                                    Payload=json.dumps(event))
```

We have updated **local.conf** file for OpenStack vm creation, added the following code for network configuration

```
HOST_IP=10.0.2.15
FLOATING_RANGE=10.0.2.224/27
FLAT_INTERFACE-emp0s3
```