Faculty of Engineering & Applied Science

Cloud Computing Project Proposal:

High-D Dataset Visualization

Group S4:

Monil Patel	100727400
Michael Metry	100747141
Rushin Chudasama	100740374
Thayan Sivathevan	100707190
Soham Bhavsar	100726376

Table of Contents

Introduction

Our project is centered around the HighD dataset, which was initially created by collecting data using drones and then transforming it into a large and valuable dataset. Our goal was to reverse engineer on the HighD dataset by plotting the data onto a graph, converting it into images and then after we have enough images, we can create a video.

To achieve our goal we have chosen to use Google Cloud's Dataflow, Pandas, Matplotlib, and Apache Beam. Since Dataflow is a fully-managed cloud service, it allows us to process and analyze large datasets in scalable and effective manners. Pandas is a widely known and utilized data manipulation library in Python that we used to preprocess the data. Matplotlib is a Python plotting library that allows us to visualize the data and identify any patterns or trends. Lastly, Apache Beam, is an open-source data processing framework that allows us to perform necessary calculations for creating our scheduled job flow.

By utilizing Google Cloud's powerful infrastructure and services, we can build a highly optimized pipeline that can process large amounts of data in parallel. However, for our project we are focusing on just sequential pipelining which means we can execute functions without having to wait for another one to finish. This automation of the process frees up local resources by waiting for tasks to be finished in the cloud.

With this pipeline we can ensure that our function is highly scalable and can process large amounts of data in an effective and distributed manner. This is crucial for handling the complex HighD dataset which contains over 60,000 vehicle trajectories. For our purposes, we focused on creating a video from a single csv data file, however the pipeline process can be run in parallel to help process multiple csv with the same process at once.

Methodology

Using the Pandas package, the HighD dataset was obtained and prepared. There are several files in the collection, including trajectory files, picture files, and annotation files. We used the Pandas library to read and concatenate the trajectory files into a single DataFrame that comprised information about each vehicle's position and speed at various timestamps. The OpenCV library was also used to read the picture files and transform them to arrays.

To display the trajectory data and plot the vehicle locations and speeds over time, we utilized the Matplotlib package. The picture data was also projected to ensure that it was accurately read and translated to arrays. We built a data pipeline for transforming picture data to videos using Apache Beam's Dataflow. The first function of the code simply reads the csv file to generate images. We group the data by frame, which represents a single instance of the highway. We iterate over the frames and plot each action that has happened in the dataset, which is to simply map the different vehicle objects. We save our plot as an image to our Google Cloud Storage Bucket.

The second function of our code reads our image directory, and appends them to a video writer. We specify our videowriter parameters, such as the size and frames per second. Our final result produces a video of cars moving through the highway as the drone would’ve seen it. We use openCV to append our images to our video.

We improved the speed of the data pipeline and video conversion by employing efficient data processing techniques such as lazy loading and batch processing. To minimize processing time, we also adopted a distributed computing method, running the pipeline on numerous computers. We reviewed the code and tested the pipeline on a small portion of the data to ensure that the pipeline and video output were of high quality. We also verified that the video output accurately matched the picture and trajectory data.

Performing the HighD Dataset

1. Importing necessary libraries
   For this assignment, we will need to import numerous libraries, including pandas, matplotlib, cv2, and Apache Beam.

```
import os
import shutil
import cv2
import io
import apache_beam as beam
from apache_beam.options.pipeline_options import PipelineOptions
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from google.cloud import storage
```

2. Reading in the HighD Dataset
   We will read in the HighD dataset in CSV format using pandas. The data may then be cleaned and preprocessed using pandas.

```
df = pd.read_csv('gs://highd_project/01_tracks.csv')
groups = df.groupby('frame')
```

3. Plotting the HighD Dataset
   Matplotlib may be used to generate a scatter plot of the vehicle trajectories in the HighD dataset.

```
for frame, group in groups:
    # Create a new figure and axis for the current frame
    fig, ax = plt.subplots()

    # Set the axis limits
    ax.set_xlim(0, 200)
    ax.set_ylim(0, 50)

    # Iterate over the rows in the current group and add a rectangle to the axis for each row
    for index, row in group.iterrows():
        rect = patches.Rectangle((row['x'], row['y']), row['width'], row['height'], linewidth=1, edgecolor='r', facecolor='none')
        ax.add_patch(rect)
```

4. Converting Images
   Each frame in the HighD dataset may be converted to a picture using OpenCV. To achieve this conversion, we will write a function. We need to save our plot image to a

buffer, and then read this buffer into our Google Storage with a blob, which will create our image file and then write to it.

```
for index, row in group.iterrows():
    rect = patches.Rectangle((row['x'], row['y']), row['width'], row['height'], linewidth=1, edgecolor='r', facecolor='none')
    ax.add_patch(rect)

# Save the figure as an image in the local directory
blob = bucket.blob(f'images/{frame}.png')
## Use bucket.get_blob('path/to/existing-blob-name.txt') to write to existing blobs
buf = io.BytesIO()
plt.savefig(buf, format='png')
buf.seek(0)
blob.upload_from_file(buf)
# plt.savefig(f'gs://highd_project/images/{frame}.png')
buf.close()


# Close the figure
plt.close()
```

5. Converting Images to Video

   After saving all of the images to a folder, we can use OpenCV to merge them into a video.

```
def generate_video(image_dir, video_path, fps=60):
    # Set up the OpenCV video writer
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    video_writer = cv2.VideoWriter('zmy_video.mp4', fourcc, fps, (640, 480))

    # Loop through all image files in the local image directory
    count = len(os.listdir(image_dir))
    for i in range(count):
        image_path = os.path.join(image_dir, f'{i}.png')
        img = cv2.imread(image_path)
        if video_writer is None:
            height, width, _ = img.shape
            video_writer = cv2.VideoWriter(video_path, fourcc, fps, (width, height))
        video_writer.write(img)

    video_writer.release()
```

6. Creating the Apache Beam Pipeline

   We can use Apache Beam to build a pipeline that converts each frame in the HighD dataset to an image and saves it to a folder.

```
def run_pipeline(argv=None):
    """Runs the dataflow pipeline."""
    pipeline_options = PipelineOptions(argv)
    with beam.Pipeline(options=pipeline_options) as pipeline:
        # Read the file as a bounded source
        input_pcoll = (
            pipeline
            | 'Read File' >> beam.io.ReadFromText('gs://highd_project/01_tracks.csv')
        )

        # Apply the transformations to generate the images and video
        (
            input_pcoll
            | 'Generate Images' >> beam.Map(generate_images)
            | 'Generate Video' >> beam.Map(generate_video, 'gs://highd_project/images', 'gs://highd_project/my-video.mp4', 60)
        )


if __name__ == '__main__':
    run_pipeline()
```

Conclusion

The usage of Panda, Matplotlib, Apache, and the pipeline option to read and plot data and convert images to videos was a highly successful method of displaying the HighD dataset. Plotted rectangular figures gave useful insights into the vehicle trajectories, but the video visualization provided a more complete view of the dataset. We were able to quickly and efficiently preprocess, visualize, and analyze the HighD dataset due to the combination of these technologies.

In this analysis, the usage of a data pipeline was extremely useful since it allowed us to automate the process of transforming images to videos. This method saved time and money while guaranteeing that the data was handled consistently and correctly. When new objectives and issues emerged, the pipeline architecture made it simple to adapt and improve the processing phases.

Our project was able to identify and apply cloud computing technologies that were taught throughout the course, such as using the Google Cloud Platform, developing a data pipeline job, and accessing a cloud storage. Although the project in itself was a demo, it can be easily scaled to provide a lot more features to the same dataset utilized. We can apply bigquery to iterate through the data set to gain important insights, and modify our pipeline to run as a batch job. The project has room for improvement, such as generating much more meaningful plots and visualization, and utilizing additional cloud tools such as a pub/sub to read images instead of saving them locally.

Overall, the combination of Panda, Matplotlib, Apache, and the pipeline option yields a powerful framework for analyzing and displaying huge datasets like HighD. With these methods, researchers may get significant insights into complicated phenomena such as vehicle trajectories and create visually appealing and useful visualizations. We were able to successfully incorporate cloud tools with our dataset, and produce the results specified in the project proposal.