Lab 03: Spark Streaming

Lab Instructors:

Đỗ Trọng Lễ
dtle@selab.hcmus.edu.vn

Bùi Huỳnh Trung Nam
huynhtrungnam2001@gmail.com

Abstract

In this lab, you'll be playing with Spark Streaming.

We'll be working with a dataset released by the New York City Taxi & Limousine Commission that captures over one billion individual taxi trips over the past several years. This lab is inspired by Todd Schneider's very nice blog post titled Analyzing 1.1 Billion NYC Taxi and Uber Trips, with a Vengeance (worth a read!). You can find the raw data here. The dataset is historic, but we're going to treat it as a data stream and perform simple analyses using Spark Streaming.

This lab draws inspiration from the University of Waterloo's CS 431/631: Data-Intensive Distributed Computing course.

Data

Since the dataset is 100s of GB and too large for a lab work, we're going to work with a small one-day slice, but that should be sufficient to give you a flavor of what Spark Streaming is like. The slice of the dataset is provided in the **taxi-data.zip** file.

Here's what the dataset looks like:

```
$ head -1 taxi-data/part-2015-12-01-0001.csv

green,2,2015-12-01 00:01:03,2015-12-01 00:01:29,N,1,-
73.937652587890625,40.804546356201172,-
73.940483093261719,40.805999755859375,1,.06,2.5,0.5,0.5,0,0,,0.3,3.8,2,1
```

There are two types of taxi ride records, Yellow and Green.

The schema for yellow taxi rides is as follows:

```
type,VendorID,tpep_pickup_datetime,tpep_dropoff_datetime,passenger_count,trip_distanc
e,pickup_longitude,pickup_latitude,RatecodeID,store_and_fwd_flag,dropoff_longitude,dr
opoff_latitude,payment_type,fare_amount,extra,mta_tax,tip_amount,tolls_amount,improve
ment_surcharge,total_amount
```

The schema for green taxi rides is as follows:

```
type,VendorID,lpep_pickup_datetime,Lpep_dropoff_datetime,Store_and_fwd_flag,RateCodeI
D,Pickup_longitude,Pickup_latitude,Dropoff_longitude,Dropoff_latitude,Passenger_count
,Trip_distance,Fare_amount,Extra,MTA_tax,Tip_amount,Tolls_amount,Ehail_fee,improvemen
t_surcharge,Total_amount,Payment_type,Trip_type
```

Each part file contains trips with drop-off times one minute apart, so for the entire day, there are 1440 part files. Each one of these becomes a part of a discretized stream to Spark Streaming.

Tasks

You need to write code in python/java or scala using Spark Streaming to solve the following tasks:

Task 1 (1 point):

Discover a method to simulate a stream by utilizing data sourced from files. You can refer to Dr. Thao's Streaming-Demonstration, Structured Streaming Programming Guide from the Spark homepage or any online source.

Task 2 (2 points):

Let's start with an EventCount(.py/.scala/.java) query that aggregates the number of trips by drop-off datetime for each hour. The output of each aggregation window is stored in a directory named output-xxxxx where xxxxx is the timestamp. (In this lab we will consider timestamp as time in milliseconds)

E.g.: output-360000 contains the aggregated results of trips from 00:00:00 to 01:00:00.

output-720000 contains the aggregated results of trips from 01:00:00 to 02:00:00.

There are 24 aggregation windows above, one for each hour.

Task 3 (3 points):

Create a query called RegionEventCount(.py/.scala/.java) that counts the number of taxi trips each hour that drop off at either the *Goldman Sachs* headquarters or the *Citigroup* headquarters. See the Todd Schneider blog post for context: you're replicating one a simplified version of one of his analyses. Use these coordinates for the bounding box of interest:

```
goldman = [[-74.0141012, 40.7152191], [-74.013777, 40.7152275], [-74.0141027, 40.7138745], [-74.0144185, 40.7140753]]

citigroup = [[-74.011869, 40.7217236], [-74.009867, 40.721493], [-74.010140,40.720053], [-74.012083, 40.720267]]
```

To be more precise, you are filtering for taxi trips whose drop-off locations are located within this bounding box and aggregating by hour. Write to the output-xxxx a key, value tuple (headquarters, count) with headquarters in {'goldman', 'citigroup'}.

As a hint, RegionEventCount requires minimal modifications over EventCount; basically, add a filter, and that's it.

Task 4 (4 points):

Let's build a simple "trend detector" to find out when there are lots of arrivals at either *Goldman Sachs* or *Citigroup* headquarters, defined in terms of the bounding boxes, exactly as above. We'll consider intervals of ten minutes, i.e., 6:00 to 6:10, 6:10 to 6:20, etc. The trend detector should "go

off" when there are at least twice as many arrivals in the current interval as there are in the past interval. To reduce "spurious" detections, we want to make sure the detector only "trips" if there are *ten or more* arrivals in the current interval. That is, if there are two arrivals in the last ten-minute interval and four arrivals in the current ten-minute interval, that's not particularly interesting (although the number of arrivals has indeed doubled), so we want to suppress such results.

Call this program TrendingArrivals(.py/.scala/.java). We'll run it as follows:

```
spark-submit TrendingArrivals.py --input taxi-data --checkpoint checkpoint --output output &> output.log
```

For simplicity, the detector should output its results to stdout in the form of the following:

```
The number of arrivals to Goldman Sachs has doubled from X to Y at Z!
```

or

```
The number of arrivals to Citigroup has doubled from X to Y at Z!
```

Where X and Y are the number of arrivals in the previous and current interval, and Z is the timestamp of the current interval. These timestamps are exactly the same form as above, e.g., 720000.

In other words, when we're marking, we'll be grepping output.log for the phrase "Number of arrivals to Goldman Sachs" and "Number of arrivals to Citigroup", so please make sure that the output is in the format we expect.

Also, the program should output the status for each batch to the directory specified by the output argument. Each status is stored in a separate file with the name of the format part-${timestamp}. The following command should gather the answers:

```
$ cat output/part-*/* | grep "(citigroup"
(citigroup,(${Current value},${Current timestamp},${Previous value}))
...

$ cat output/part-*/* | grep "(goldman"
(goldman,(${Current value},${Current timestamp},${Previous value}))
...
```

As a hint, TrendingArrivals is a simple modification to RegionEventCount. You'll need to change the window definition (from hourly to 10 minutes).

Report

Write a report providing instructions to the teacher on your understanding of and approach to completing this lab.

The point for each task includes implementation and report for that task.

The report should include a self-assessment of completeness.

References. Properly acknowledge any reference code utilized in your work within this section; failure to do so may be construed as academic dishonesty.