School of Computer Science

COMP47470

Project 3
Spark Streaming

Teaching Assistant:	Vsevolods Caka
Coordinator:	Dr Anthony Ventresque
Date:	Wednesday 22nd April, 2020
Total Number of Pages:	6

General Instructions

- You are encouraged to collaborate with your peers on this project, but all written work must be your own. In particular we expect you to be able to explain every aspect of your solutions if asked.
- We ask you to hand in an archive (zip or tar.gz) of your solution: your EventCount.scala file, a short pdf report of your work (no need to include your code in it but you need to explain it).
- The report should list your answers and should also contain a short introduction and conclusion.
- The report should not be longer than 10 pages (this is not a hard constraint though).
- The breakdown of marks for the project will be as follows:
  - Exercise 1 : 60%
  - Exercise 2 : 40%
- **Due date: TBC**

1 Spark Streaming

1.1 Introduction

In this exercise you will Spark Streaming to manipulate a stream of data representing taxi trips in New York City. This data is part of a dataset released by the New York City Taxi & Limousine Commission that captures over one billion individual taxi trips, but we will only be using one day's worth of data.

The dataset is in the form of CSV files that contain data about two types of trips: green and yellow taxi rides. The lines for each type of ride have a different schema shown bellow. For both types, the first column contains the type, "yellow" or "green".

The schema for yellow taxi rides is as follows:

`type, VendorID, tpep_pickup_datetime, tpep_dropoff_datetime, passenger_count, trip_distance, pickup_longitude, pickup_latitude, RatecodeID, store_and_fwd_flag, dropoff_longitude, dropoff_latitude, payment_type, fare_amount, extra, mta_tax, tip_amount, tolls_amount, improvement_surcharge, total_amount`

The schema for green taxi rides is as follows:

`type, VendorID, lpep_pickup_datetime, Lpep_dropoff_datetime, Store_and_fwd_flag, RateCodeID, Pickup_longitude, Pickup_latitude, Dropoff_longitude, Dropoff_latitude, Passenger_count, Trip_distance, Fare_amount, Extra, MTA_tax, Tip_amount, Tolls_amount, Ehail_fee, improvement_surcharge, Total_amount, Payment_type, Trip_type`

This exercise is adapted from the University of Waterloo's Data-Intensive Distributed Computing class assignment available here `https://www.student.cs.uwaterloo.ca/~cs451/F19/assignment7-451.html`.

1.2 Set up

Before we can start working on the exercise some set up is necessary. As we will be using Spark Stream you should be working inside the spark containers we have been using in the labs. See the labs on spark on how to use them. In your spark master container run the following commands to install maven and the java jdk:

```
$ apk add maven
$ apk add openjdk8
$ export JAVA_HOME="/usr/lib/jvm/java-1.8-openjdk/"
```

For this assignment we give you some code to start. This code reads the data and transforms it into a stream with a determined timing so your results are consistent. Download the code and the data:

```
$ wget http://csserver.ucd.ie/~thomas/comp47470-2020-project3.tar.gz
$ tar -xvf comp47470-2020-project3.tar.gz
$ cd comp47470-2020-project3
$ wget https://lintool.github.io/bespin-data/taxi-data.tar.gz
$ tar -xvf taxi-data.tar.gz
```

You should now be able to build the project. From inside the Project_3 directory run:

```
$ mvn clean package
```

The first build can take a bit of time the first time, as there are a few dependencies to download. You will also get a few warnings. As long as you get BUILD SUCCESS at the end, everything is fine. You can now run the code with the following command from inside the Project_3 directory:

```
$ /spark/bin/spark-submit --class ie.ucd.csl.comp47470.EventCount
↪  target/COMP47470-Project-3-1.0.0-fatjar.jar --input taxi-data
↪  --checkpoint checkpoint --output output
```

Running through the whole dataset will take a couple of minutes. Once it is finished, you should get a few directories called "output-*XXXX*", where *XXXX* is a timestamp. These directory contain the results corresponding to the different windows used in the program. To aggregate the outputs of the different windows you can use the following command:

```
find output-* -name "part*" | xargs grep 'all' | sed -E
↪  's/^output-([0-9]+)\/part-[0-9]+/\1/' | sort -n
```

This should give you the following output:

```
3600000:(all,7396)
7200000:(all,5780)
10800000:(all,3605)
14400000:(all,2426)
18000000:(all,2505)
21600000:(all,3858)
25200000:(all,10258)
28800000:(all,19007)
32400000:(all,23799)
36000000:(all,24003)
39600000:(all,21179)
43200000:(all,20219)
46800000:(all,20522)
50400000:(all,20556)
54000000:(all,21712)
57600000:(all,22016)
61200000:(all,18034)
64800000:(all,19719)
68400000:(all,25563)
72000000:(all,28178)
75600000:(all,27449)
79200000:(all,27072)
82800000:(all,24078)
86400000:(all,18806)
```

You are now ready to start working on the exercise.

1.3 Tasks

1. Have a look at the `src/main/scala/ie/ucd/csl/comp47470/EventCount.scala` file. This is the code of we are running. A lot of it is here to set up the stream from the data on disk and make sure our results are consistent. Focus on the following part:

```
val wc = stream.map(_.split(","))
       .map(tuple => ("all", 1))
       .reduceByKeyAndWindow(
         (x: Int, y: Int) => x + y, (x: Int, y: Int) => x - y,
         ↪  Minutes(60), Minutes(60))
       .persist()
```

(a) Explain what this snippet of code does. What does the output we get represent?

(b) Could something be taken out of this code without modifying the result? If so, what and why?

2. Modify the code in `RegionEventCount` (where `// Code for q2` is written) so that your new code counts the number of taxi trips each hour that drop off at either the Goldman Sachs headquarters or the Citigroup headquarters. A taxi trip drops off at one of these locations if it is inside the zones defined by the following coordinates:

   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 determine if a drop-off is inside these zones you can use the `Point` and `Polygon` classes, and the `Point.within(Polygon)` method from the geotrellis project that are provided in the project. The two polygons you need to use have been created in the code as an example of the syntax.

   We should be able to parse your output with the following command and get results in the following format (where ? are actual counts, do not worry if the counts are low):

```
$ find output-* -name "part*" | xargs grep 'goldman' | sed -E
↪  's/^output-([0-9]+)\/part-[0-9]+/\1/' | sort -n
21600000:(goldman,?)
25200000:(goldman,?)
28800000:(goldman,?)
...

$ find output-* -name "part*" | xargs grep 'citigroup' | sed -E
↪  's/^output-([0-9]+)\/part-[0-9]+/\1/' | sort -n
3600000:(citigroup,?)
7200000:(citigroup,?)
10800000:(citigroup,?)
...
```

3. Modify the code in `TrendingArrivals` (where `// Code for q3` is written) to build a trend detector to see when there are big increases in drop offs in the two zones we defines before. For this consider intervals of ten minutes, i.e., 6:00 to 6:10, 6:10 to 6:20, etc. The detector should go off when, in a 10 minute interval, there has been at least 10 arrivals in the interval and if the number of arrivals in the interval is at

least twice the one in the previous interval (e.g 10 arrivals in this interval and 5 in the previous).

(a) When the detector goes off, it should print the message `Number of arrivals to L has doubled from V1 to V2 at T!`, where L is a location ("Goldman Sachs" or "Citigroup"), V1 is the number of drop offs in the previous period, V2 is the number of drop off in this period and T is the timestamp of the current interval. This message should be printed to the standard output.

You should run your code as follows:

```
$ /spark/bin/spark-submit --class
↪  ie.ucd.csl.comp47470.TrendingArrivals
↪  target/COMP47470-Project-3-1.0.0-fatjar.jar --input
↪  taxi-data --checkpoint checkpoint --output output &>
↪  output.log
```

Hint: The function used in `mapWithState` can take an additional first argument of type Time which is the timestamp of the current RDD.

(b) Change the way the output is saved to file so that each batch's results are saved to the directory specified by the output argument. Each status is stored in a separate file with the name of the format part-$timestamp where $timestamp is a 8-digit string padded with leading zeros.

This output should be parsable with the following commands and give results in the following format:

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

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

Hint: Use `foreachRDD((rdd, time) => ...)` to print the results to files.

2 Reflection

Write a short report (1 or two pages) on one of the research papers that are available on Brighspace. The following list of sections is an indication of how to write your paper. Some of the items might not be relevant for all papers, and you might want to add some sections in your report (e.g., evaluate the posterity of a solution etc.). We will have an open mind when reading your report and we just want to see how you analyse a research paper and are able to discuss it - in short there is no one single perfect report, everything that shows you made an effort to understand and focus on the important parts (research methodology, hypotheses, etc.) will be welcome.

- identify the question/challenge the paper addresses. Explain in your own words what the motivation for the research is.

- describe briefly the related work, i.e., the other (related) solutions that the authors compare themselves to. Show the limitations of these related solutions
- Give an outline of the solution proposed by the authors (no need to go into details) showing the main components
- describe their scientific method: what are the research questions they evaluate, how do they evaluate
- describe briefly their results
- give your impression on the idea, what you liked about the paper and whether you see any limitations etc.