MET CS 755, Spring 2018

Assignment – 2 (20 points)

Hadoop MapReduce Programming

Due date: February 27, 6:00 pm

1. Description

The goal of this assignment is to implement a set of MapReduce programs in Java (using Apache Hadoop). Specifically, your MapReduce jobs will analyzing a data set consisting of New York City Taxi trip reports in the Year 2013. The dataset was released under the FOIL (The Freedom of Information Law) and made public by Chris Whong (https://chriswhong.com/open-data/foil_nyc_taxi/).

2. Dataset

The data set itself is a simple text file. Each taxi trip report is a different line in the file. Among other things, each trip report includes the starting point, the drop-off point, corresponding timestamps, and information related to the payment. The data are reported by the time that the trip ended, i.e., upon arrive in the order of the drop-off timestamps.

The attributes present on each line of the file are, in order:

Attribute	**Description**
medallion	an md5sum of the identifier of the taxi - vehicle bound (Taxi ID)
hack_license	an md5sum of the identifier for the taxi license (driver ID)
pickup_datetime	time when the passenger(s) were picked up
dropoff_datetime	time when the passenger(s) were dropped off
trip_time_in_secs	duration of the trip
trip_distance	trip distance in miles
pickup_longitude	longitude coordinate of the pickup location
pickup_latitude	latitude coordinate of the pickup location
dropoff_longitude	longitude coordinate of the drop-off location
dropoff_latitude	latitude coordinate of the drop-off location
payment_type	the payment method -credit card or cash
fare_amount	fare amount in dollars
surcharge	surcharge in dollars
mta_tax	tax in dollars
tip_amount	tip in dollars
tolls_amount	bridge and tunnel tolls in dollars
total_amount	total paid amount in dollars

The data files are in comma separated values (CSV) format. Example lines from the file are:

07290D3599E7A0D62097A346EFCC1FB5,E7750A37CAB07D0DFF0AF7E3573AC141,2013-01-01, 00:00:00,2013-01-01 00:02:00,120,0.44,-73.956528,40.716976,-73.962440, 40.715008,CSH,3.50,0.50,0.50,0.00,0.00,4.50

22D70BF00EEB0ADC83BA8177BB861991,3FF2709163DE7036FCAA4E5A3324E4BF,2013-01-01, 00:02:00,2013-01-01 00:02:00,0,0,0.00,0.000000,0.000000,0.000000,0.000000, CSH,27.00,0.00,0.50,0.00,0.00,27.50

0EC22AAF491A8BD91F279350C2B010FD,778C92B26AE78A9EBDF96B49C67E4007,2013-01-01, 00:01:00,2013-01-01 00:03:00,120,0.71,-73.973145,40.752827,-73.965897 73.965897,40.760445,CSH,4.00,0.50,0.50,0.00,0.00,5.00

3. Obtaining the Dataset

We are providing two versions of the data set. The first is a small one that you can use for debugging. We strongly recommend that you write your code and execute it on your laptop using this data set first. Only once you are sure that your implementation is working on your laptop should you try the larger data set. This is going to save money, but more importantly, it is going to save time!

- Small data set. (93 MB compressed, uncompressed 384 MB) for implementation and testing purposes (roughly 2 million taxi trips).

This is available at Amazon S3: https://s3.amazonaws.com/metcs755/taxi-data-sorted-small.csv.bz2

or as direct S3 address, so you can use it in a MapReduce job:

s3://metcs755/taxi-data-sorted-small.csv.bz2

- Larger data set. (8.8 GB compressed, uncompressed 33.3 GB) for your final data analyzing (roughly 173 million taxi trips).

This is available at Amazon S3: https://s3.amazonaws.com/metcs755/taxi-data-sorted-large.csv.bz2

or as direct S3 address, so you can use it in a MapReduce job:

s3://metcs755/taxi-data-sorted-large.csv.bz2

4. Assignment Tasks:

4.1. Task 1 (5 points)

GPS positions are missing in some of the trip reports (they are zero). We will treat these as GPS device failures. Such failures happen when the taxi GPS device cannot receive GPS signals (for example driving close to NYC skyscrapers) or the Taxi device is damaged, or it can happen that the device is manually disabled.

We want to know what time of days as the most errors, to see if errors are evenly distributed throughout the day (consider the dropoff time to be the time use to generate your error report). So your task is to write a MapReduce job that computes a list of **`(hour of day, number of errors)`** pairs.

Hour of day can be an integer number from 1 to 24.

NOTE - 1: You should consider that this is a real world data set that might include wrongly formated data lines. You should clean up the data before the main processing, a line might not include all of the fields. If a data line is not correctly formated, you should drop that line and do not consider it.

NOTE - 2 : You should consider the pickup date for the hour of day. This is the position 3 of the comma separated line. The pick up hour might be different than the drop-off. You need to consider the pick up hour.

4.2. Task 2 (10 Points)

We like to know the five worst taxis in terms of the fraction of the data associated with the taxi that contains GPS errors (note that the first data field, “medallion”, identifies the taxi). The more errors, the worse the taxi. Computing this is going to require you to write a sequence of at least a couple of MapReduce jobs.

One job is going to need to compute the fraction of errors for each taxi. This is a pretty simple MapReduce job. The second job is going to need to compute the five worst taxis. There are a lot of ways to do this, but the most efficient (and a rather simple way) is to have a priority queue as a private member within your Mapper class. Every call to map just inserts the next taxi into the priority queue, organized by error fraction. Whenever the queue has more than five entries in it, throw out the worst entries so that you only have five. But your map method will not actually emit any data. Then, your Mapper class will implement the cleanup method:

`public void cleanup(Context context) throws IOException, InterruptedException ...`

This is a method that any Mapper class is free to implement, that is automatically called when the mapper is about to be finished. Here, you'll simply emit the contents of the priority queue, writing them to the Context. In this way, your Mapper class will only write out the five worst taxis processed by that mapper. Essentially, we are using the mapper as a filter. The reducer then computes the top five out of the top five found by every mapper.

In the end, we are interested in computing a set of **(taxi, percent of errors).**

NOTE - 3: GPS positions include 4 data values, two for drop-off and two for pick-up locations, if any of them is equal to zero, you should consider that line to be an error data line for the specific taxi.

A GPS location error might be also an empty string.

4.3. Task 3 (5 Points)

Finally, we would like to figure out who the ten best drivers are in terms of their average earned money per minute spent carrying a customer. The total amount field is the total money earned on a trip. In the end, we are interested in computing a set of `(driver, money per minute)` pairs.

4.4. Task 4 (For Advanced Student Groups)

Here are two further tasks for advanced groups.

- How many percent of taxi customers pay with cash and how many percent using electronic cards? Analyze these payment methods for different time of the day and provide a list of percents for each day time? As a result provide two numbers for total percentages and a list like

(hour of day, percent paid card)

- We would like to measure the efficiency of taxis drivers by finding out their average earned money per mile. (Consider the total amount which includes tips, as their earned money)

Implement a MapReduce job that can find out the top-10 efficient taxi divers.

5. Important Considerations

5.1. Machines to Use

One thing to be aware of is that you can choose virtually any configuration for your EMR cluster - you can choose different numbers of machines, and different configurations of those machines. And each is going to cost you differently! Pricing information is available at:

http://aws.amazon.com/elasticmapreduce/pricing/

Since this is real money, it makes sense to develop your code and run your jobs locally, on your laptop, using the small data set. Once things are working, you'll then move to Amazon EMR. We are going to ask you to run your Hadoop jobs over the "real" data using five **c3.2xlarge** machines as workers. This provides **8 cores per machine (40 cores total)** so it is quite a bit of horsepower.

As you can see on EC2 Price list , this costs around 50 cents per hour. That is not much, but **IT WILL ADD UP QUICKLY IF YOU FORGET TO SHUT OFF YOUR MACHINES**. Be very careful, and stop your machine as soon as you are done working. You can always come back and start your machine or create a new one easily when you begin your work again. Another thing to be aware of is that Amazon charges you when you move data around. To avoid such charges, do everything in the **N. Virginia** region. That's where data is, and that's where you should put your data and machines.

- You should document your code very well and as much as possible.

- You should use the Google Java Style Guide (https://google.github.io/styleguide/javaguide.html)

- You code should be compilable on a unix-based operating system like Linux or MacOS.

5.2. Academic Misconduct Regarding Programming

In a programming class like our class, there is sometimes a very fine line between "cheating" and acceptable and beneficial interaction between peers. Thus, it is very important that you fully understand what is and what is not allowed in terms of collaboration with your classmates. We want to be 100% precise, so that there can be no confusion.

The rule on collaboration and communication with your classmates is very simple: you cannot transmit or receive code from or to anyone in the class in any way---visually (by showing someone your code), electronically (by emailing, posting, or otherwise sending someone your code), verbally (by reading code to someone) or in any other way we have not yet imagined. Any other collaboration is acceptable.

The rule on collaboration and communication with people who are not your classmates (or your TAs or instructor) is also very simple: it is not allowed in any way, period. This disallows (for example) posting any questions of any nature to programming forums such as **StackOverflow**.

As far as going to the web and using Google, we will apply the **"two line rule"**. Go to any web page you like and do any search that you like. But you cannot take more than two lines of code from an external resource and actually include it in your assignment in any form. Note that changing variable names or otherwise transforming or obfuscating code you found on the web does not render the "two line rule" inapplicable. It is still a violation to obtain more than two lines of code from an external resource and turn it in, whatever you do to those two lines after you first obtain them.

Furthermore, you should cite your sources. Add a comment to your code that includes the URL(s) that you consulted when constructing your solution. This turns out to be very helpful when you're looking at something you wrote a while ago and you need to remind yourself what you were thinking.

6. Turnin

Create a single document that has results for all three tasks. For each task, copy and paste the result that your last MapReduce job wrote to Amazon S3 (for example, the ten results you obtained for task 3). Also for each task, for each MapReduce job you ran, include a screen shot of the summary the web page provides you of the job. An example of what we are looking for is found at:

http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.1.3/bk_using-apache-hadoop/content/figures/1/figures/mapreduce_job_finished.png

HortonWorks also has a nice description of how to use the web interface at

http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.1.3/bk_using-apache-hadoop/content/running_mapreduce_examples_on_yarn.html

in case you are interested.

For your soft copy, please zip up all of your code and your document (use .gz or .zip only, please!), or else attach each piece of code as well as your document to your submission individually.