Big Data – Fall 2021
Assignment 1 – MapReduce

Goals

In this assignment, you will explore New York City taxi data using MapReduce and Hadoop Streaming. You will write map and reduce python programs the tasks outlined below.

Suggestion: first, run your programs on Peel using a sample (smaller) dataset to test and debug your code (you can use the lab2 data, which can be found under the `/user/CS-GY-6513/lab2-data` directory), and then use the large dataset to generate the results for your submission.

Data

The data `(fare_data_week1.csv, trip_data_week1.csv)` can be found in HDFS under the `/user/CS-GY-6513/hw1`.
The directory includes a subdirectory called `samples` which contains two small datasets for you to test your code. Use the following command to copy the small datasets to your Peel account:

```
hfs -get /user/CS-GY-6513/hw1/samples
```

Hint:
Parsing can be challenging with big data. We need to be particularly careful with special characters since they can cause your scripts to crash, and when this happens, your Hadoop job simply dies. To avoid such crashes, you can add a *try-except* block in your code to skip these characters.

Submission Instructions

You will submit the map and reduce programs for all tasks in a single .zip file with the following structure:

1. One directory per task, named `taskX`, where X is the task's number.
2. If a task has a sub-task, use `taskX-Y,` where Y identifies the sub-task. E.g.: "task2-a" refers to sub-task "a" of Task 2. If you do an “`ls`” on your homework directory, it should contain the following sub-directories:
```
task1
task2-a
task2-b
task2-c
task2-d
```

```
task2-e
task2-f
task3 (for extra credit)
```

Each task directory should include a “map.py” file and a “reduce.py” file.
Note:

- Files names must not have spaces, commas, or special characters.
- You should not include any input files or output files in your submission.

You may use combiners and partitioners, but they will not be tested, i.e., your program should work correctly *without* combiners or partitioners.

You should use 2 reducers for all tasks.

The code for your mappers (`map.py`) can access the `mapreduce_map_input_file` environment variable to determine which input (.csv) file is being read. The CSV filenames will contain the substrings "trip" and "fare". You cannot use any other environment variables besides mapreduce_map_input_file.

Tester

We provide a *tester* that you can use to make sure that your submission has the proper structure and that your code outputs have the correct format and that . The tester uses smaller datasets and scripts to run your code automatically.

Type the following command to obtain the tester folder:

```
hfs -get /user/CS-GY-6513/hw1-test/tester1.tar.gz
tar -xzvf tester1.tar.gz
```

Run the following command:

```
chmod -R +x tester
```

To run the tester (Input path should be directory with the structure mentioned above):

```
./testall_hadoop.sh <INPUTPATH>
```

The tester will generate a directory called “results” containing the running results (taskX_hadoop.res) indicating whether your code passes the test. You can view the difference between your output and the solution from the file taskX_hadoop.diff

Note: Passing the tests does not guarantee your code is correct. Make sure your code runs over the complete datasets.

Tasks

Task 1: Write one map-reduce job that joins the 'trips' and 'fare' data (taxi data).

The 'fares' and 'trips' data share 4 attributes:

`medallion, hack_license, vendor_id, pickup_datetime.`

The join MUST BE a reduce-side inner join.

Output: A key-value pair per line. Use a “tab” to separate key and value, a comma between multiple keys (and values)

key: `medallion, hack_license, vendor_id, pickup_datetime`
value: the remaining attributes of 'trips' data in their original order
and
the remaining attributes of 'fare' data in their original order

You must respect this ordering requirement!

Here’s a sample output with 2 key-value pairs:
00005007A9F30E289E760362F69E4EAD,2C584442C9DC6740767CDE5672C12379,CMT,2013-08-07 00:55:11	1,N,2013-08-07 00:25:38,1,990,8.9,-73.981972,40.764397,-73.927887,40.865353,CRD,26.5,0.5,0.5,5.5,0,33
00005007A9F30E289E760362F69E4EAD,2C584442C9DC6740767CDE5672C12379,CMT,2013-08-07 02:01:47	1,N,2013-08-07 01:06:05,1,653,3.3,-73.983887,40.780346,-73.991646,40.744511,CSH,12.5,0.5,0.5,0,0,13.5

The contents of `task1` subdirectory looks like:

```
ls -F task1/
map.py*        reduce.py*
```

Here is the command you should use to run task1:

```
hjs -D mapreduce.job.reduces=2 -file ~/<your HW directory>/task1
-mapper task1/map.py -reducer task1/reduce.py -input /user/CS-
GY-6513/hw1/fare_data_week1.csv  -input /user/CS-GY-
6513/hw1/trip_data_week1.csv -output /user/netid/task1.out
```

Task 2: Write map-reduce jobs for each of the following sub-tasks, using the output of Task 1 (joined data) as input:

(Similar to Task 1, you must use a tab to separate the key and the value in the output tuples.)

a) Find the distribution of fare amounts (`fare_amount`) for each of the following ranges:
   [0, 20],
   [20.01, 40],
   [40.01, 60],
   [60.01, 80],
   [80.01, infinite],

   Thus, for each range, give the number of trips whose fare amount falls in that range.

   Output: A key-value pair per line, where the key is the range, and the value is the number of trips. For example,

```
0,20	100
20.01,40	300
…
```

b) Find the number of trips whose cost is *less than or equal* to $15 (total_amount).

   Output: The number of trips.

c) Find the distribution of the number of passengers, i.e., for each number of passengers A, the number of trips that had A passengers.

   Output: A key-value pair per line, where the key is the number of passengers, and the value is the number of trips. For example,

d) Find the *total revenue* (for all taxis) and the *total amount of tips*, **per day** (from `pickup_datetime`). Revenue should include the fare amount, tips, and surcharges.

   Output: A key-value pair per line, where the key is the day `YYYY-MM-DD`, and the value contains the total revenue and the total tips for that day, in this order.

Use two decimal digits, e.g., 3.02245 should be represented as 3.02. For example,

```
2016-01-01    100000.02,11000.00
2016-01-02    202000.00,1000.00
```

e) For each taxi (`medallion`) find the **total** number of trips, and the **average** number of trips **per day**. For the average trips per day, use 2 decimal digits.

   Your average should be over all days that the taxi drove, e.g., if the input data has entries for a given taxi on 6 different days, your average should be over 6 days.

   Output: A key-value pair per line, where the key is the medallion, and the value contains the total number of trips and the average number of trips per day.

f) Find the number of different taxis (`medallion`) used by each driver (`license`).

   Output: A key-value pair per line, where the key is the driver, and the value is the number of different taxis used by that driver.

Task 3 Extra Credit: Try to optimize your map reduce program for Task 1 and report: the strategy you used, statistics about the original and optimized task, and discuss the results (e.g., why the optimization worked, why it did not work)

If you choose to do the extra credit, include in the zip file one additional folder named "task3" that contains a text/pdf/docx file with your answer.

The End