Assignment 3
Taxi trip analysis

Toon Van Craenendonck
`toon.vancraenendonck@cs.kuleuven.be`

Jessa Bekker
`jessa.bekker@cs.kuleuven.be`

Collaboration policy:
Projects are independent: no working together! You must come up with how to solve the problem independently. Do not discuss specifics of how you structure your solution, etc. You cannot share solution ideas, pseudocode, code, reports, etc. If you are unsure about the policy, ask the professor in charge or the TAs.

Before you start... note that you will have an exercise session on Hadoop on the 3rd of March. If you would like to start working on this assignment before, check the exercise sheet of session 5 for more details about working with Hadoop.

1 Datasets

For this assignment, you will be working on GPS tracks of taxis in San Francisco. We provide several datasets, which are briefly described in this section. More details about their format are given later, in the corresponding sections. The datasets are available online[1] and will be available on the clusters in the `/data` directory.

- `all.segments`: The complete GPS tracks (including intermediary points) from May 2008 to January 2011. Each record is a segment with two end points (start position, end position), two time stamps (start date, end date) and two taxi states (empty/full) for the beginning and for the end of the segment. Consecutive segments can be concatenated to form complete trips. The dataset contains about 306 million segments and is 28GiB in size.
- `2010_03.segments`: Follows the same specification as the previous dataset, but only contains the segments started or finished in March 2010. The dataset still contains about 19 million segments and is 1.2 GiB in size.
- `2010_03.trips`: Contains trips constructed from the segments in the previous dataset. Each trip is simply represented by its two end points (i.e. there is no intermediary position).
- `Taxi_706.segments`: This dataset contains 19237 segments only involving taxi number 706. This dataset is useful for debugging.

The datasets are provided (almost) as such: they are not sorted, and they contain errors and misformatted records. Dealing with this type of data is part of the assignment. All datasets will be available on the

[1] `http://people.cs.kuleuven.be/~toon.vancraenendonck/bdap_files/`

distributed file system (*DFS*). We recommend that you leave the largest dataset on the file system as there is no need to process it locally.

2 Trip length distribution

First, we are interested in computing a simple statistic: the distribution of trip lengths. We will compute this distribution for the trips in the `2010_03.trips` preprocessed dataset.

For this exercise only, we will assume that the trip distance is the distance between the two end points of the trip. This information is easy to compute from `2010_03.trips`, as it contains descriptions of the trips without the intermediary segments. In this dataset, each line has the following format (represented here on two lines):

```
<taxi-id> <start date> <start pos (lat)> <start pos (long)>...
...<end date> <end pos (lat)> <end pos (long)>
```

To compute the geographical distance between two coordinates, you can use a simple flat-surface formula[2], which will give a reasonable approximation for this dataset because the distances are not too large (but remember that these formulæ are not always appropriate for larger distances[3]).

First, implement a simple algorithm in the language of your choice to compute the trip distance distribution. Then propose a Spark [4] program that solves the same problem. You only have to run your Spark code locally. You can do that on your own system, or on one of the computers at the CS department. In any case, your code should also run on the ones on the CS department, with the Spark installation that is available in `/home/u0098478/spark-2.1.0-bin-hadoop2.7`. The WordCount example that is available on Toledo can get you started. For example, execute the following commands to compile and run the WordCount example (omit the dots in the last command):

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/
export SPARK_INSTALL=/home/u0098478/spark-2.1.0-bin-hadoop2.7
mvn package
$SPARK_INSTALL/bin/spark-submit --class "WordCount" --master local[1] target/WordCount-1.0.jar ..
                                .. /path/to/file /path/to/output
```

Which implementation is faster? Does it match your expectations? Include a description of your implementation and a discussion about efficiency in your report.

Plot the trip length distribution and interpret the results. For plotting, you can use a mainstream scripting/plotting language of your choice.

3 Computing airport ride revenue

A significant number of taxi rides pass through the San Francisco airport. Assume that taxi companies have to pay for an expensive license for this airport access. A company may then be interested in knowing exactly how much they earn from these airport rides, to know whether paying the license is actually worth it.

[2] `http://en.wikipedia.org/wiki/Geographical_distance`
[3] After all, the earth is not really flat.
[4] `http://spark.apache.org/`

The goal of this part of the assignment is to compute an estimate of the revenue coming from airport rides based on the GPS tracking data. The estimate should be as accurate as possible, which means that you must analyze all data, i.e. sampling is not an option.

This part of the assignment consists of two steps, which we describe next: (1) reconstructing trips from segments, (2) computing the revenue obtained from these trips.

3.1 Reconstructing trips

First, you will be required to reconstruct complete trips from the ride segments. The `.segments` files contain the complete GPS tracks decomposed into segments. A segment is simply a pair of geographical coordinates. The sampling rate is generally 1 minute, although there can be larger gaps. Each line from these datasets has the following format (represented here on two lines):

```
<taxi-id>, <start date>, <start pos (lat)>, <start pos (long)>, <start status> ...
...<end date> <end pos (lat)> <end pos (long)> <end status>
```

Propose a design of a Map/Reduce application to construct trips. Depending on your implementation, you may or may not need multiple Map/Reduce jobs. Implement your application and test it locally on the `2010_03.segments` dataset.

Note on erroneous data-points: GPS points and recording devices are far from 100% reliable. Erroneous records will ultimately lead to erroneous results. One possible way to eliminate trips including erroneous data points, is to use a simple heuristic. For example, you can eliminate trips that include at least one segment with an average speed above $200km/h$. You should think about what other possible errors could arise, and how you could correct them. Please discuss your techniques in the report. Plotting trips can be helpful to test the validity of your implementation in general. This can for example be done with the Google maps API (copying the link in the footnote in your browser should for example plot a simple fictitious trip between two points [5]).

Once you have developed and tested your implementation on the sample, you should run it on the cluster on the complete dataset. However before you do so, you should think about the scalability of your approach. Execution times on Hadoop clusters are usually very variable and are thus not very relevant. Instead it is better to reason about efficiency in terms of number of input records and output records for the various components (mapper, combiner, reducer ...), and in terms of individual task complexity and maximum memory usage for each mapper and reducer. With the clusters you are given, how should you choose the number of mappers and the number of reducers? Based on this analysis, propose a set of changes to improve scalability of your approach. Test and observe the impact of your changes locally, and run your approach on the cluster to construct all trips.

3.2 Computing the revenue

In this step, you will use the output of the previous component to compute the total revenue obtained from airport trips. We consider airport trip rides as those that pass through a circle with the airport as center, and a radius of 1km. The airport is located at 37.62131° N, -122.37896° W. To calculate trip revenue, you can use a simple formula that combines a starting fee of \$3.5 with an additional \$1.71 per kilometer.

[5] `https://maps.google.com/maps/api/staticmap?&size=640x640&markers=color:green%7Clabel:S%7C37.762573,-122.437477&markers=color:red%7Clabel:E%7C37.7452,-122.458076&path=geodesic:true|color:0x0000ff|weight:5|37.762573,-122.437477|37.7452,-122.458076`

Report the total revenue that has been earned from airport trips, and also make a plot that shows the evolution of this revenue over time.

4 If you want to go beyond . . .

More interesting insights can be learned from this dataset, if you're interested in searching for more information feel free to do so and to comment in the report. As usual, any interesting insight about the problem or the data can potentially result in bonus points. (Nevertheless, you should focus on the previous sections first).

5 (Very) Important remarks

- Here is a (non exhaustive) list of potential issues you may encounter when you're dealing with the datasets:
  - Date and timezone: to parse dates and compute trip durations, make sure you use the correct time zone (which can be fetched with `TimeZone.getTimeZone("America/San_Francisco")`).
  - Do not make wrong assumptions about the order or the format of the input record, your program should be robust enough to cope with broken records in the data without crashing.
- Do **NOT** run anything on the clusters before making sure it runs locally (at least on a smaller dataset). The clusters are shared resources.
- **Kill your jobs** on the cluster, if you think they are going to crash or run for too long. For that use `hadoop job -list` and `hadoop job -kill <job id>`. Not following this policy will have unimaginable consequences :)
- When you describe your Hadoop implementation, make sure to describe the inputs and the outputs of each mapper / reducer / combiner . . .
- Also discuss the runtime of your implementation on the full dataset for reconstructing trips and computing revenues.
- Scaling up to the large dataset is not trivial and may require some work. Make sure you have a decent implementation and a report for the smaller dataset (`2010_03.segments`) before working on the scalability on the large dataset.

6 Report

Your report should contain the following information:

1. A description of your trip length distribution algorithm, and a discussion of the results that addresses the questions given in Section 2 of the assignment. This should be no more than half a page, excluding any tables or figures you include.
2. A discussion about how you cleaned the data to remove erroneous data points (see Subsection 3.1) of no more than half a page, excluding any tables or figures.

3. A brief description of your approach to reconstructing the trips, with an emphasis on discussing any design decisions related to scalability of the approach. This should be no more than one page of text.
4. A report on the observed efficiency of your approach on the cluster (e.g., run time) as well as addressing the questions in the last paragraph of Subsection 3.1.
5. Be sure to include the calculations and plots requested in Subsection 3.2.

7 Turning in your code and report

- The assignment should be handed in on Toledo before the **Tuesday the** 18th **of April**. As usual, there will be a 10% penalty per day, starting from the due day.
- You must upload an archive (`.zip` or `.tar.gz`) containing the following files
  - the report (as pdf)
  - a runnable jar file called `Exercise1.jar` for the first exercise
  - a runnable jar file called `Exercise2.jar` for the second exercise
  - your source code in a sub directory called `src` (which may contain more subdirectories if necessary)
  - a `README` including the command lines to compile and execute your code (on the cluster).

Good luck!