CS 6240: Assignment 3

Goal: For a more complex problem, compare implementation effort and performance of plain MapReduce and Pig.

This homework is to be completed individually (i.e., no teams). You have to create all deliverables yourself from scratch. In particular, it is not allowed to copy someone else's code or text and modify it. (If you use publicly available code/text, you need to cite the source in your report!)

All deliverables for this HW are due as stated on Canvas. For late submissions you will lose one percentage point per hour after the deadline. This HW is worth 100 points and accounts for 10% of your overall homework score. To encourage early work, you will receive a 2-point bonus if you submit your solution on or before the early submission deadline stated on Canvas. (Notice that your total score cannot exceed 100 points, but the extra points would compensate for any deductions.) Please submit your solution through the Canvas assignment submission system and make sure your report is a **PDF** file. Always package all your solution files, including the report, into a single ZIP file. Make sure you are using standard ZIP, i.e., the format also used by Windows archives.

Flight Data and Computation Challenge

We are working with a new data set containing information about real flights, e.g., origin, destination, delays, and distance. You can find the data set as a zip file in this assignment. Unzip the big data file (data03.zip) and upload all contained files to S3 so that you can access them from EMR. For convenience we also included with this assignment the following files: small data sample for October 1988, list of attributes extracted from the data header, and an html file describing the attributes.

Our goal is to compute the **average delay** for all **two-leg flights** from airport **ORD** (Chicago) to **JFK** (New York) where both legs have a flight date that falls into the 12-month period between **June 2007 and May 2008** (including these two months). More precisely:

- A two-leg flight from ORD to JFK consists of two flights F1 and F2 with the following properties:
  - F1 has origin ORD and some destination X that is different from JFK.
  - F2 originates from that airport X where F1 ended; its destination is JFK.
  - F1 and F2 have the same flight date. (Use the **FlightDate** attribute.)
  - The departure time of F2 is later than the arrival time of F1. (Use the actual arrival time **ArrTime** and the actual departure time **DepTime**.)
  - Neither of the two flights was *cancelled* or *diverted*. (Find the attributes containing this information.)

- The delay of the entire two-leg flight should be computed as the delay of F1 plus the delay of F2. Use attribute **ArrDelayMinutes**, which sets the delay for early arrivals to zero.
- Compute a simple average of the delays of all flight pairs (F1, F2) that match the conditions and report it.

Hints and notes:

- To find all qualifying pairs (F1, F2), we have to JOIN the flight data set with itself. Joins tend to be expensive operators.
- To reduce computation cost, it is good practice to apply data-reduction operators as early as possible. For instance, when processing an input record, we can remove attributes that are not needed for later processing steps. (Database speak: apply projections early.) Similarly, simple filters (Database speak: selection operators) should be applied early. E.g., for the first leg, we only need to consider flights originating from ORD.
- Think carefully about how to select only flights with a flight date between June 2007 and May 2008. You can use attributes Month and Year with the appropriate conditions.
- Not all input data files have the same schema. However, all attributes we need for our computation do exist in all of them and are in the exact same column.
- **To help you debug your code, we provide the following information**: In the two-month period consisting of December 2007 and January 2008 there are 73273 two-leg flights that satisfy the above conditions. The average delay is approximately 57.527479 minutes.

Plain MapReduce Program: PLAIN

Write a plain Java MapReduce program PLAIN to compute the average flight delay as described above. Try to write the best and most efficient code you can come up with. Notice that the join computation can be very expensive when not implemented carefully. Test your code on small samples before running it on the full data and check if your results for December 2007—January 2008 match ours. Hints:

- Think carefully about the intermediate keys for the join implementation in M-R. Recall that two flights F1 and F2 are only considered a two-leg flight if F1's destination matches F2's origin, F1 and F2 have the same flight date, and F2 departs after F1's arrival time. You can enforce some join conditions through the appropriate key choice; others might have to be enforced in the Reduce function.
- Make good use of early projections and selections to eliminate irrelevant data as soon as possible.
- You might have to create more than just one M-R job to solve the problem. However, a good rule of thumb in MapReduce is: the fewer jobs, the better.
- Since the input file is in CSV format, we recommend using existing CSV parsers, e.g., OpenCSV (http://opencsv.sourceforge.net/), but you can also write your own.

PigLatin Programs: JOINFIRST and FILTERFIRST

Write three PigLatin programs. JOINFIRST has two different versions. It starts with the join and then filters the results based on the time-range constraint, while FILTERFIRST does the filtering first. For all PigLatin programs, we recommend using the CSVLoader() instead of PigStorage(). It can be found in the piggybank.jar file, which should automatically be located in 'file:/home/hadoop/lib/pig/piggybank.jar' when you start a Pig job in EMR. Add the following two lines to your program to be able to use it:

```
REGISTER file:/home/hadoop/lib/pig/piggybank.jar
DEFINE CSVLoader org.apache.pig.piggybank.storage.CSVLoader;
```

You can also install Pig on your development machine. Then you need to change the file path for piggybank.jar accordingly. Notice that Pig functionality tends to change a bit from one version to another. Try to work with the same version that is available on EMR.

JOINFIRST

1. Load the flights file as Flights1.
2. Load the flights file again as Flights2.
3. Remove as many records and attributes from Flights1 and Flights2 as possible, but do NOT yet use the condition that the flight date has to be between June 2007 and May 2008.
4. Join Flights1 and Flights2, using the condition that the destination airport in Flights1 matches the origin in Flights2 and that both have the same flight date.
5. Filter out those join tuples where the departure time in Flights2 is not after the arrival time in Flights1.
6. Filter out those tuples whose flight date is not between June 2007 and May 2008.
    a. Version 1: check that *both* the flight date of Flights1 and Flights2 are in the range.
    b. Version 2: only check that the flight date of Flights1 is in the range.
7. Compute the average delay over all the tuples produced in the previous step.

FILTERFIRST

1. Load the flights file as Flights1.
2. Load the flights file again as Flights2.
3. Remove as many records and attributes from Flights1 and Flights2 as possible, this time make sure you also use the condition that the flight date has to be between June 2007 and May 2008.
4. Join Flights1 and Flights2, using the condition that the destination airport in Flights1 matches the origin in Flights2 and that both have the same flight date.
5. Filter out those join tuples where the departure time in Flights2 is not after the arrival time in Flights1.
6. Compute the average delay over all the tuples produced in the previous step.

Note: Make sure your Pig programs store the final output. Otherwise the lazy execution strategy will not do any data processing.

Report

Write a brief report about your findings, using the following structure.

Header

This should provide information like class number, HW number, and your name.

Source Code (60 points total)

Show the source code for programs PLAIN (30 points), JOINFIRST version 1 and version 2 (15 points total), and FILTERFIRST (15 points). **Make sure your code is clean and well-documented. Messy and hard-to-read code will result in point loss.** In addition to correctness, efficiency is also a criterion for the code grade.

We strongly recommend that you also include the **pseudo-code of program PLAIN** in your report.

For the PigLatin programs, you can also explore different ways of loading the data. In particular, since we are computing a self-join of the flight data, you can load the flight data just once and then extract Flights1 and Flights2 from it. See if this makes a performance difference compared to using two LOAD commands.

Performance Comparison (30 points total)

Run each of the four programs in Elastic MapReduce (EMR) on the entire big flight data set, using the following parameters:

- 11 small machines (1 master and 10 workers)
- Make sure your program creates at least 10 reduce tasks for all computation steps. This is not required for the final average-delay computation step: If you created a file with the delay information for all flights of interest, you can compute the final average flight delay in whatever way you want, e.g., using MapReduce or just a simple sequential program.

Report the total runtime of each program. (5 points each)

Notice that EMR already supports Pig, making it easy to run a PigLatin program. For details, look at the EMR documentation.

Critically evaluate the runtime results by comparing them against what you had expected to see and **discuss your findings**. Make sure you address the following questions: Did your PLAIN program beat Pig? How did the differences in the Pig programs affect runtime? Can you explain why these runtime results happened? (8 points)

Report the average flight delay each of your four programs computed. (2 points)

Deliverables

1. The report as discussed above. (1 PDF file)
2. The source code for each program. (5 points)
3. The syslog file for a successful run of PLAIN. And the stderr files for a successful run of each of the three PigLatin programs. (4 plain text files) (5 points)