CPSC 5330 -- Spring 2023
Week 2 Lab: Hadoop Streaming and Sqoop

Part 1 (5 points)

Hadoop for Log Processing

The file `Hadoop_2k.log` (from https://github.com/logpai/loghub) contains log entries from a Hadoop run sampled for ten minutes (Oct 18, 2015, between 6:01PM and 6:10PM). Each log entry has a severity level, which is one of INFO, WARN, ERROR, and FATAL.

Use Hadoop Streaming to process these log lines to record, for each minute, the number of log entries in each category.

Your script, named `process-log-file`, will run Hadoop streaming to produce one line for each minute in the log, and each line should have these 6 fields:

1. Minute number
2. Total number of log entries for that minute
3. Number of log entries for that minute with severity INFO
4. Number of log entries for that minute with severity WARN
5. Number of log entries for that minutes with severity ERROR
6. Number of log entries for that minutes with severity FATAL

Your script will then retrieve the output, and print the lines to standard output, in ascending order of minute number.

Use our Hadoop Streaming idiom from lab to structure your solution: create the directory log-files containing your mapper and reducer, then your script process-log-files will call run-hadoop-streaming to kick off the log processing job, and then display the properly sorted output on standard output.

To hand in:

1. Your script `process-log-file`
2. The folder `log-processing`, containing your mapper and reducer

Part 2 (5 points)

Sqoop for On-Time Airline Tracking

In the `/tmp` directory of the Docker image you will find a SQL dump file, `airline.sql`, taken from this repository: https://relational.fit.cvut.cz/dataset/Airline.

Use the database, Sqoop, and Hadoop to produce output for each airline (Carrier): the minimum, maximum, and average flight delay in minutes.

Your goal is to write a script `report-airline-delays` that writes one record for each carrier, in descending order of average flight delay. Your script needs to set up the database, run Sqoop to import the data to HDFS, run the MapReduce job to aggregate the flight records, then produce the output records.

For example, here is one output line:

AA 0.000000 1659.000000 10.695254

For this question submit:

1. Your script `report-airline-delays`
2. A folder `airline-delays` containing your mapper and reducer code

Hint: the database table has lots of columns, most of which you don't need, and you don't want to define the full schema to Sqoop. Refer to the Sqoop documentation and the class slides to figure out how to import only the columns you need.

Part 3 (5 points)

Secondary Sorting

When we did word count in Hadoop Streaming, we got the results back sorted by term. That was not by accident -- Hadoop Streaming guarantees that the reducers get their keys in sorted order.

Suppose instead we wanted the output sorted in descending order of word count instead. We saw how to do the sort using Linux tools, in particular sorting the Hadoop output records by the the second field (count) in reverse order.

But do the sort in MapReduce instead. Make it so the output from Hadoop has the word counts in descending order of count.

You will write a script `terms-by-count` that takes as a parameter the name of a directory containing the text files of interest and writes to standard output *all* of the terms from the text, and their counts, in descending order of count.

Here is some example output. This output is taken from a different document set, so the numbers and terms might be different. Also the debug messages have been edited out.

```
# terms-by-count folder-containing-my-documents | head -n 6

the   136537
and   97569
of    73741
to    50373
a     35164
in    34773
```

You will start with the version of Word Count we did in Lab 2 using Hadoop Streaming, which has a mapper that "cleans" the tokens by down casing and removing punctuation from each word.

Hint #1 -- do the job in TWO map reduce jobs, not one. The first job generates the word counts in the form <term> <count> then the second job takes output from the first map reduce job and sorts the records by count.

You will package the two jobs together into the shell script named `terms-by-count`. The shell script will run the two map reduce jobs, and also print the output to standard output

Hint #2 -- as you saw, sorting ON THE RECORD KEY is trivial in Hadoop -- if you make <count> the key, then you get sorting by count for free.

Hint #3 -- but the problem is when Hadoop sorts, it acts as though its keys are strings, not numbers, so 9 sorts after 899. That's the tricky part.

For this question submit:

1. Your script `get-frequent-terms`
2. Directories containing the Hadoop streaming jobs that the script uses

---

Part 0, Style Points (2 points)

(These are easy points to get, just by following instructions and handing in good clean work!)

Was the output clean and well presented? Was the text and explanations well written?
Were all the files there and named according to instructions?

To Hand In

A Zip file containing (only) these files

- The files and directories called out in each of the three questions.
- A retrospective report in a file `retrospective.pdf` – a reflection on the assignment, with the following components
    - Your name

- How much time you spent on the assignment
- If parts of the assignments are not fully working, which parts and what the problem(s) are
- Were there aspects of the assignment that were particularly challenging? Particularly confusing?
- What were the main learning take-aways from this lab – that is, did it introduce particular concepts or techniques that might help you as an analyst or engineer in the future?