Module 3 Homework

New York University

Courant Institute of Mathematical Sciences

Homework

1. Wordcount without using *MapReduce framework*

This is a WordCount-based problem - the goal is to find the *number of lines* containing a specific search term.
Write a Java (or Python or C/C++) program, **without using Hadoop MapReduce**, that:

    a. Searches for all of the following strings in the input file containing tweet data (you can provide the search terms as parameters, or hardcode them):   `hackathon, Dec, Chicago, Java`

    b. Accepts a small input file to be searched containing lines of the form: ***Date,Time;Name,Tweet***
       Here is the ***exact data*** to type or copy-paste into your input file:

```
09-Dec-18,6:00PM;#Hackatopia,Tribeca Film Hackathon: Code As A New Language For Content Creators Hackathon
28-Dec-18,7:00PM;#NYCHadoop,Hadoop-NYC Strata/Hadoop World Meetup at Google NYC
31-Dec-18,3:00PM;#Hackatopia,Designers, Developers, Doers, don't miss this upcoming Chicago hackathon
```

    c. Your code will search for all of the search strings in the input file and output the ***number of lines*** that contained each search string (not the number of occurrences of a search string). The matching is **not case sensitive**, i.e. if searching for the search string *hackathon*, all of the following are a match: `hackathon, Hackathon, hACKathon` (and any other combination of upper and lower case characters).

    d. Your code should output the number of lines that contained each search string. Using the input data above, the resulting counts will be:

```
Chicago 1
Dec 3
Java 0
hackathon 2
```

    e. Upload homework to NYU Classes. To receive full credit, please hand in all of the following items on or before the due date:
       - Your program, your input file, and job output.
       - Evidence that the program ran successfully - screenshots

Homework

2. Wordcount using *MapReduce framework*

In this next program you will use the *MapReduce framework* and develop an algorithm based on WordCount.

Hint: You need a parallelizable WordCount algorithm. In general, this is how a Hadoop WordCount solution works:

Mapper
 - Outputs every word it encounters followed by '1' - e.g.:
    ```
    zoo 1
    hello 1
    georgia 1
    goodbye 1
    chicago 1
    zoo 1
    georgia 1
    ```

Reducer
 - Receives (in Java) values grouped by key (keys come into reducer task sorted):
    ```
    chicago {1}
    georgia {1, 1}
    goodbye {1}
    hello {1}
    zoo {1, 1}
    ```

 - Sums 1's by key and outputs word and sum to an HDFS output file:
    ```
    chicago 1
    georgia 2
    goodbye 1
    hello 1
    zoo 2
    ```

Homework

2. Wordcount using *MapReduce framework*   (continued)

This program accomplishes what your program from step 1 did, but this time you will use the *MapReduce framework* and develop an algorithm based on WordCount that takes advantage of the cluster resources (a parallelizable algorithm). Unlike your step 1 solution, this solution will scale with the size of the input data, and it works with Big Data (step 1 program does not). **This algorithm is very different from your previous one** - it needs to split the algorithm into work that the map phase can do, and work that the reduce phase can do. (Study the previous slide to learn about this approach.)

Your program:

   a.   Searches for all of the following strings in the input file containing tweet data (you can provide the search terms as parameters, or hardcode them):   `hackathon, Dec, Chicago, Java`

   b. Accepts the *same small input file* you used in step 1 and searches it for the search strings.

   c. Has Mapper code that will search the input file line by line to find matches. The matching is not case sensitive (same as before).

   The mapper code should not do any summing or buffering (no storing data in a map or array).

    The summing must happen in the reducer code.

   d. You are required to use a Reducer. The Reducer code will input the key-value pairs generated by the map phase and output the number of lines that contained each search string. Using the input file, the resulting counts will be:

```
Chicago 1
Dec 3
Java 0
hackathon 2
```

   e. Upload homework to NYU Classes. To receive full credit, please hand in all of the following items:
      - Your program, your input file, and job output.
      - Evidence that the program ran successfully - screenshots

Note: Your plain old Java algorithm (from step 1) is not what you want to use in this MapReduce solution. Think about the example covered in the book where temperatures (values) were sorted by year (the key). These key-value pairs are guaranteed to arrive at the reducer(s) sorted by *key*. The reducer can iterate through the *values* associated with a given key and process them. In the MaxTemperature example, that *processing* was to select the max temperature by iterating through the values. Think about what a reducer should do for a WordCount algorithm to be efficient in a distributed system - write your algorithm so the summing happens in the reducer.