Hadoop MapReduce Hands-on programming tasks

General instruction: *You may discuss the questions with your classmates, but all the codes and analysis should be done independently. The points will be given based on the correctness and the running time of your programs. Therefore, try to find all correct results and reduce the running time of your programs.*

Hadoop guide: https://gist.github.com/xupefei/e2c7d230529d1234f4b622aef8ce098e

Problem 1.

The WordCount example is a good starting point for learning Hadoop and MapReduce. Now you are going to have some practice. First, obtain the txt file of Lewis Carroll's famous *Through the Looking-Glass* from here. Modify the WordCount code to process this file, and

- Removing all leading and trailing spaces
- Removing as many kinds of punctuations as possible, e.g., ',', '.', '-'

Now answer the following questions:

1. What are the top 10 most frequent words (case-insensitive) excluding "the", "am", "is", and "are"? (10 points)
2. What are the top 5 most frequent words (case-insensitive) after the word "the"? (5 points)
3. What are the top 10 most frequent two-word phrases (case-insensitive)? A two-word phrase looks like "I am", "am a", "a potato", etc. (10 points)
4. Try to run your program on multiple VMs. Do you notice any performance gain? (10 points)

Problem 2.

Implement three executable Hadoop MapReduce programs to perform the inner join of two tables based on "*Student ID"* to satisfy the following two filtering conditions simultaneously:

1. The year of birth is greater than (>) 1989 and;
2. Each scores of the courses 1,2,3 is greater than (>) 80.

Sample data of Student table:

Student ID	Name	Year of Birth
20170126453	Kristalee Copperwaite	2000
20170433596	Roeberta Naden	1997

Sample data of Score table:

Student ID	Score for course 1	Score for course 2	Score for course 3
20170126453	93	97	80
20170140241	86	85	87
20170433596	82	81	90

Join result:

Student ID	Name	Year of Birth	Score for course 1	Score for course 2	Score for course 3
20170433596	Roeberta Naden	1997	82	81	90

Download datasets: https://www.cs.helsinki.fi/u/jilu/dataset/BigData2018.zip
There are four files in the unzipped folder: three *Score* tables and one *Student* table.
Considering the following three cases, design three **different** MapReduce programs to efficiently handle them.

1. Both *Student* table and *Score* table are big tables with thirty million entries. That is, "*score_5000000.txt*" and "*student_5000000.txt*".
2. *Student* table is a big table, but *Score* table is small with three thousand entries. That is, "*score_5000.txt*" and "*student_5000000.txt*". (Hint: Use the distributed cache in this case)
3. *Student* table is still the same one, but *Score* table is a medium-size table with three million entries. That is, "*score_500000.txt*" and "*student_5000000.txt*". (Hint: Use bloom filter in this case)

Run your three different MapReduce programs to each combination of datasets, and then report the result size and the performance, e.g. the elapsed time by filling the following two tables.

Result size (10 points):

Data combination	Result size
"score_5000000.txt" and "student_5000000.txt"	
"score_5000.txt" and "student_5000000.txt"	
"score_500000.txt" and "student_5000000.txt"	

Note that if your three MapReduce programs return the different result size for the same datasets, please check your codes carefully and make sure that their answer sizes are consistent.

Total running time (ms) (10 points)

Data combination	Program (1)	Program (2)	Program (3)
“score_5000000.txt” and “student_5000000.txt”			
“score_5000.txt” and “student_5000000.txt”			
“score_500000.txt” and “student_5000000.txt”			

Analyze your results by answering the following questions:

1. How many computer nodes did you use to run the program? Have you tried to reduce the running time by using more nodes in Ukko cluster? (5 points)
2. Upload your source codes and analyze the performance of your codes. What optimization have you tried to reduce the running time for the large dataset? (20 points)
3. What is the elapsed time for the construction of bloom filter in Program 3? (5 points)

Problem 3.

Hadoop can also be used for *approximate string processing*. Implement Hadoop MapReduce programs to perform the similarity string join with n-gram Jaccard similarity.

Download datasets: https://www.cs.helsinki.fi/u/jilu/dataset/Gram2018.zip, where you will find two Wikipedia samples. Each of them contains 10k lines of Wikipedia categories like this. These datasets are now inconsistent due to misspellings.

Can you find similar pairs of records, one from each dataset, which have Jaccard similarity **no less than 0.85 but not 1.0**?

1. Fill the following table: (20 points)

Gram size	Result size	Running time on a single VM (s)	Running time on a multiple VMs (s), specify how many VMs
2 (bi-gram)			
3 (tri-gram)			

2. Describe your algorithm for this problem, and upload your source codes and analyze the performance of your codes. (20 points)

Hint 1: You may need to use two MapReduce procedures: first find potential pairs and then verify them (i.e., "filtering and verification"). Ref: here.
Hint 2: You may feed the output of one Reducer directly to another Mapper by using the following lines:

```
job1.setOutputFormatClass(SequenceFileOutputFormat.class)
```

and

```
job2.setInputFormatClass(SequenceFileInputFormat.class)
```

References

- Miner D, Shook A. MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems [M]. " O'Reilly Media, Inc.", 2012.
- Lam C. Hadoop in action [M]. Manning Publications Co., 2010.