Cloud Computing Labs: Big Data Programming

The primary purpose of this assignment is to get familiar with Hadoop, MapReduce, and Spark programming. You will be asked to work through a few tutorials. The first part of the assignment does not involve actual coding, but requires a lot of activities on the command line (running Hadoop jobs, copying files around, etc.). In the second part, you'll enhance a preliminary version of MapReduce kmeans clustering algorithm. And the third part includes a couple of relational analytical tasks that you should use Spark to answer.

Part 1: Getting Familiar with Hadoop and Spark

For this project, you will start with your single-node Hadoop system in your linux system (native linux, WSL, or virtual machine). You have been asked to install the single-node Hadoop system in the previous exercise. If you have problems, please let us know.You can also skip the yarn setup and run mapreduce with the local mode.

After you set up the Hadoop system, work through the HDFS shell commands and the MapReduce tutorial with the WordCount source code that can be found in the directory "share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-*-sources.jar" of your hadoop installation directory.

Now, answer the following questions:

Question 1.1 Please follow the *commandline commands* in the lecture slides to compile and run the WordCount program. You can use the HDFS command to upload a text file for your test run. Post the screenshot of your run.

The output files can be conveniently checked with a command like this:

```
hadoop fs -cat your_output_directory/part-r-00000 | less
```

Follow the instructions in the slides to download and setup Spark. You can use the local mode, skipping the yarn setup.

Question 1.2 run the Scala or Python version of Spark Wordcount, respectively, and post the screenshots.

Question 1.3 How many hours approximately did you spend on these tasks (including the time you spent previously installing the Hadoop system)?

Question 1.4 How useful is this task to your understanding of the Hadoop and Spark environments?

Part 2: MapReduce KMeans Algorithm

In this task, you will need to complete and optimize a MapReduce version of the KMeans Algorithm. We have discussed the KMeans algorithm in the class. Briefly, the KMeans algorithm has two steps in each iteration: (1) determine the cluster label for each data point with the current centroids and (2) re-calculate the centroids based on the updated cluster labels. You can find a detailed description of KMeans.

For the part 2, first, you should review the simple (incomplete) version MapReduce KMeans that includes only one iteration of KMeans. You can test it with the sample input data data.hdfs and initial centroids. You can compile and run the KMeans program with the following commands.

```
javac -cp `yarn classpath` -d working_directory KMeans.java
jar -cvf km.jar -C working_directory .
```

Now you get the km.jar file. Assume your HDFS runs at localhost:9000 and run the program as follows.

```
   hadoop jar km.jar KMeans hdfs://localhost:9000/user/your_home_directory/centroids da
```

After you successfully run the program, check the output: output/part-r-00000 and verify whether the program works correctly as expected. Note that the sketch program may have bugs. You should be able to take care of the possible bugs.

The basic idea of MapReduce KMeans algorithm is to cast the two steps to the Mapper and Reducer functions, respectively. The first step of KMeans applies to each data point, which fits Mapper processing. The second step is a typical aggregation process, where each Reducer can take care of the re-calculation of one or a few centroids. In the following, you will see the basic design of MapReduce KMeans algorithm.

We store the centroids in a file in the HDFS, where each line represents the cluster ID and its centroid in the following form.

```
0\t1.0,1.0
1\t10.0,10.0
```

where "\t" is the tab character. The data records are stored in the data file as comma seperated values. Check the sample centroids file and the data file for the MapReduce KMeans algorithm. The data file is a simple CSV file, where each line is a record.

```
1,1
0.5,0.5
2,2
10,10
11,11
```

You may upload both files to your HDFS directory.

In the Mapper class, the centroids will be loaded from a HDFS file via the setup() function. The map function then processes the text input (e.g., a line in the Data.hdfs file), and convert each line to a vector. By checking the distance between the vector and the centroids, you can determine the cluster membership. The output key-value pair will be (ClusterID, record_in_string_representation).

In the Reducer class, all records with the same ClusterID will be aggregated to the same Reducer. The reduce function will convert the string records to numerical records, and compute the updated centroids. The output key-value pair will be (ClusterID, centroid_in_string_representation). Note that in this implementation we only update the centroids.

You may check the FileUtil class API if you need file/directory operations in your program.

Answer the following questions after you understand the code.

Question 2.1 The current program only run one iteration of the KMeans algorithm. Please revise it (in the main function) to implement iterative processing, paste your code here, and briefly describe how it works. Were you able to successfully compile and run your program (yes/no)?

Question 2.2 It appears that a Combiner class can be used in K-Means to reduce the Shuffle traffic. Please carefully describe the design of the Combiner class (no need to implement it). Explain the amount of data being passed to the Reduce phase in two settings (1) without the combiner and (2) with the combiner, in the big-O representation of the parameters: N records, k clusters, M mappers, and R reducers.

Question 2.3 Note that in the previous MapReduce program you only get the final cluster centroids, not the cluster label for each record. Now you want to use the Hadoop streaming tool with Python-coded Mapper/Reducer programs to assign cluster labels to the records. You only need a Mapper program to assign labels. Assume that the centroids are stored in a local file, encoded in the previously mentioned format. Your Mapper program will take the centroid filename as a parameter and the MapReduce output format is "cluster_label \t record" per line. For example, the first two records in the sample data file are labeled with cluster 0.

```
0 \t1.0,1.0
0 \t0.5,0.5
```

Please implement the Mapper.py program. Paste your code and the commandline command using Hadoop streaming here.

Question 2.4 How many hours approximately did you spend on this task?

Question 2.5 How useful is this task to your understanding of MapReduce programming? (not useful, somewhat useful, very useful)

Part III: Spark Programming

In this task, you will use Spark to solve the analytic problems for a set of linked tables. Now we have three data files with the following schema: customer(cid, name, age, city, sex), book(isbn, name), and purchase(year, cid, isbn, seller, price), where purchase.cid is the foreign key to customer.cid and purchase.isbn is the foreign key to book.isbn. The fields in the dataset are separated by "\t".

The Spark local model is also recommended for easier setup. You can use PySpark or Scala Spark to answer the questions. You should carefully comment your code.

Question 3.1 How much did each seller earn? Develop the Spark RDD (Spark RDD means using only the transformations and actions, no DataFrame/SQL) version to solve the query.

Question 3.2 Find the names of the books that Amazon gives the lowest price among all sellers, excluding the cases that other sellers also give the lowest price. Develop the DataFrame (using Spark SQL) version to solve this query.

Question 3.3 How many hours approximately did you spend on this task?

Question 3.4 How useful is this task to your understanding of MapReduce programming? (not useful, somewhat useful, very useful)

Final Survey Questions

Question 4.1 Your level of interest in this lab exercise (high, average, low);

Question 4.2 How challenging is this lab exercise? (high, average, low);

Question 4.3 How valuable is this lab as a part of the course (high, average, low).

Question 4.4 Are the supporting materials and lectures helpful for you to finish the project? (very helpful, somewhat helpful, not helpful);

Quertion 4.5 Do you feel confident on applying the skills learned in the lab to solve other problems with big data? (low, average, high)

Deliverables

- Note this is NOT a group project. You can discuss with your classmates but cannot share solutions
- Turn in FOUR files to Pilot - THREE PDF files and one zipped source code file. *DO NOT zip the four files into one file*. (1) The PDF file 1 answers ONLY the questions 1.1-1.2, 2.1-2.3, 3.1-3.2 - do not rephrase the questions, just the answers labeled with question numbers, (2) the PDF file 2 answers the remaining survey questions: 1.3-1.4, 2.4-2.5, 3.3-3.4, and 4.1-4.5, (3) copy and paste all source code for Part II and III to the PDF file 3, (4) and zip the source Java/Python/SparkScript files and a README file describing how to compile and run your programs.

This page, last updated: Sept 21, 2019