LAB 2 - WEEK 3

MAPREDUCE WORD COUNT LAB – FOR VM

This document is divided into two parts.

1. Practice Lab

- MapReduce Java WordCount Implementation

  Just try to run through all the steps and see if they work properly for you. It's very essential to get the word count program run properly on your machine.

No need to submit this part.

2. Homework Questions

a. Do some research and find out what code needs to be added to the Word Count program for automatic removal of "output" directory before job execution.

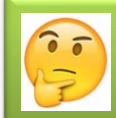
b. Run the above basic Word Count program in pseudo-distributed mode with 2 reducers. Use `setNumReduceTask` method of `Job` object. Paste the screenshot of the two `part-r-*` files created in HDFS. (Note: multiple reducers work in pseudo-distributed mode and not in local mode if you are using the given VM)

c. Modify the *WordCount* program to output the counts of only the words "Hadoop" and "Java". (This is case-insensitive count. For example, *"Hadoop"* and *"hadoop" should be counted* as same word!)
Submit your output file along with the java program.

d. Modify the WordCount program to output the counts of only those words which appear in the document at least 25 times. (Use case-insensitivity)
Submit your output file along with java program.

e. Write a MapReduce program to find out how many distinct (unique) words are there in the input file. (Use case-insensitivity)
(Hint: Recall that there are setup and cleanup methods in the Reducer class)
Submit your output file along with java program.

In your lab submission, I should be able to find the java programs for all (a), (b), (c), (d) and (e) with commands to run these programs in pseudo-distributed mode.
For (b), I'll need a screenshot as mentioned.



Food for thought

What if you want to produce an output file which will be sorted on Word Counts and not on Words?

Practice Lab - MapReduce Java WordCount Implementation

The purpose of this practice lab is to give you a feel of running MapReduce programs in Hadoop environment on the Cloudera Quickstart VM.

Make sure that you can run the given java Word Count program both locally and in pseudo-distributed mode.

1. Create a new *WordCount* project in Eclipse with the given *WordCount.java* file.
2. You'll see lots of errors now. To get rid of these errors, you need to properly configure the build path of the project by adding external jars from the following locations.

   ```
   File system/usr/lib/hadoop/client-0.20
   File system/usr/lib/hadoop
   File system/usr/lib/hadoop/lib
   ```

3. Once all the errors are gone, follow the following steps to run the word count program in local mode first and then in pseudo-distributed mode.

Local Mode:

- Create a directory in your eclipse project as "input" and copy the given input file "Lab2-WC-Input.txt" there.
- You'll need to supply runtime arguments to your program as "input" and "output". These are the paths which Hadoop will use to take input from and store output to, respectively. (alternatively, you can choose to have your own names for these folders.)
- Run the Java program in Eclipse and see that the output directory got created in your project path and there's the output file named "part-r-00000" which has the counts of all the words in your input file.

Pseudo-distributed Mode:

- Create a jar file of your Word Count program and save it on "Desktop".
- Then, create "input" folder in HDFS
  (under /user/cloudera, check it out using HDFS browser)

  ```
  hadoop fs -mkdir input
  ```

- Put the given input file in this newly created HDFS input folder.

  ```
  hadoop fs -put /home/cloudera/Desktop/wc-input.txt input
  ```

- Run your MR word count job using the command given below.

  ```
  hadoop jar /home/cloudera/Desktop/wordcount.jar WordCount input output
  ```

- After running the program, check the output in HDFS.