CE807 – Lab on HADOOP

Prerequisites

- Download the resource files for this lab from the CE807 website.
- Tip: Try to stick to the naming conventions I used to avoid any issues.

A. Hadoop Setup

This lab will make use of a Hadoop distribution provided by Hortonworks. More specifically, we are using the sandbox version, a self-contained virtual machine. This is useful for purposes such as learning about the Hadoop environment, building a proof of concept or testing new functionality before production deployment.

A step by step installation guide:

1. Open Oracle Virtual Box which is already installed on your computers. To avoid file permission issues please navigate to File->Preferences and change the “Default Machine Folder” to the following folder (which you have to create):
   C:\Users\your_username\VM\
2. Hit the OK button so changes from Step 1 are saved and then go to “File->Import Appliance” and select the Open Virtual Appliance provided for you in the “C:\Hortonworks” directory. The only appliance setting that needs to be changed is the RAM. Configure the RAM to 2048 instead of the default value of 4096. This setting will make sure your computers are still usable ☺
3. The “Hortonworks Sandbox 2.1” VM should appear on the left side of the Oracle Virtual Box window. Right click on the VM and hit the “Start” option.
4. Congratulations! You have successfully installed the Hadoop environment consisting of a single node cluster.

B. Prepare the local environment for development

In most cases Hadoop is installed on a separate machine – in this case in a VM. It is desirable to be able to write the code locally and submit it to the Hadoop environment.

1. This lab will use WinSCP for file manipulations between your local environment and the VM. Open WinSCP and connect to the VM as follows:
    a. Use the SCP File protocol.
    b. Default hostname is 127.0.0.1 with 2222 port number. You can check these details are correct by inspecting the Hortonworks sandbox window.
    c. The username is root while the password is hadoop.
2. This lab will make use of Eclipse as an IDE for Hadoop related development. Please open Eclipse and create a new Java project.
3. We need to have access to the appropriate libraries in order to successfully compile Hadoop-related code in a local environment. The libraries will be obtained from the Hadoop installation on the VM. It is good practice to have a local copy of all the Hadoop libraries, but for our purposes we are interested only in 2 files:
    1. "hadoop-common-2.4.0.2.1.1.0-385.jar" – for the Hadoop specific code. Copy this file to your local machine using WinSCP. The file is located in the "usr\lib\hadoop" directory of the VM.
    2. "hadoop-mapreduce-client-core-2.4.0.2.1.1.0-385.jar" – for the MapReduce specific code. Copy this file to your local machine using WinSCP. The file is located in the "usr\lib\hadoop-mapreduce" directory of the VM.
4. After copying the above 2 files to your local directory, please add the libraries to the Java build path of your Eclipse project.
5. This version of Hadoop still uses java 1.7 so please change the compiler compliance level to 1.7. You do that from project properties.

C. Prepare the VM environment for the exercises

1. Using WinSCP create a directory in the root folder called "ce807". Copy into the just created directory the input folder from the downloaded lab resources.
2. Make another folder in /root/ce807/ called "output" – this is where we will later transfer the result of the MapReduce jobs

3. Open a SSH connection to the VM. I suggest you use PuTTY for this and for any other command line jobs for this lab.
   1. Set hostname to 127.0.0.1
   2. Set port to 2222
   3. Make sure the SSH connection type is selected and hit Open.
   4. When prompted, the user is root, while the password is hadoop.

   Note: You can find the hostname IP and port details by checking the VM window.
4. We need to copy the input files from the Linux local file system to the Hadoop Distributed File System. Type in the following commands:
   a. Create an input directory in HDFS
      ```
      hadoop fs -mkdir -p /ce807/input/
      ```
   b. Copy the input files from LFS to HDFS
      ```
      hadoop fs -copyFromLocal /root/ce807/input/* /ce807/input/
      ```
5. You are now ready to jump to some exercises! Don't worry, the code is provided ☺

D. Exercises

Exercise 1

1. Create a new package in your java project called "exercise1" and copy all the files from the resources folder with the same name you have downloaded at the beginning of this lab.
2. Let us study the code for a while…

   There are 3 classes which are the core of any MapReduce program:
   - WordMapper – the **Mapper** class.
     - Generic explanation: the role of this class is to segment the data into key value pairs.
     - In the case of our exercise it processes a file line by line. The key is the line offset while the value is the text from that line.
     - The output is a set of key-value pairs corresponding to a word and the integer 1 – the logic behind this is that every time a word is encountered, its appearance is taken into account once.
   - WordReducer – the **Reducer** class.
     - Generic explanation: the role of this class is to collect all the values associated with a key and perform the desired processing on the set of values.

        - In this exercise, the Reducer counts all the 1s associated with a word and outputs the word and its frequency as a pair.
    - WordCount – the **Driver** class. This class takes care of the symbiotic relationship between the MapReduce components. It bounds all components together and acts as an abstract configuration panel.
3. Now that we are aware of the basic components of MapReduce, we can move forward and run our first Hadoop program.
4. We first need to pack our code into a ".jar" file. Use Eclipse to export the "exercise1" package to a jar with the default configurations. Please name your jar "exercise1.jar".
5. Copy the above created jar file into /root/ce807/ folder from the VM using the WinSCP tool.
6. Let's launch now our first MapReduce job! Please follow the steps below using PuTTY:
    a. Execute the jar
       hadoop jar /root/ce807/exercise1.jar exercise1.WordCount /ce807/input/ /ce807/output/exercise1/
    b. Transfer the data from HDFS to LFS
       hadoop fs -copyToLocal /ce807/output/exercise1/ /root/ce807/output/
7. Now go to /root/ce807/output/exercise1/ and examine the output of the MapReduce job. You can open the produced files using WinSCP.

For the brave

Study the output from the PuTTY console. Look for things such as launched map and reduce tasks, map input/output records, combine input/output records, reduce input groups and input/output records. Try to understand how those numbers came up.

Exercise 2 – the Combiner

This exercise was created to introduce the notion of "Combiner" and to see it in action. The role of the combiner is to aggregate locally the data in order to decrease the overall network cost. The intelligent use of combiners has a big effect on the run time of a large scale program.

The task for this exercise is to repeat all of the steps from Exercise 1 by carefully making the appropriate amendments such as replacing exercise1 with exercise2 everywhere you encounter it.

You will notice the code is the same except for an extra class called WordCombiner and an extra line in the driver class which maps the MapReduce Combiner to the appropriate class. If you take a look at the

WordCombiner class you will see it has the same code as WordReducer. As mentioned before, the role of the combiner is to locally aggregate the data. So, instead of sending over the network from each mapper pairs of words and 1s we **potentially** send pairs of words and their occurrence encountered into the files processed by the mappers in question. The word "potentially" was underlined to reflect the fact that the Hadoop environment **does not guarantee** it will make use of the provided combiner.

Question

Can you predict how the numbers from the output of the MapReduce job look like? Are they any different from the ones from the previous exercise? Based on your prediction calculate how much the network cost is reduced.

Exercise 3 – a first design pattern

It was mentioned in Exercise 2 the Hadoop environment does not guarantee it will call the combiner. What if we want to make sure our program benefits from local aggregation? Luckily there is a very simple design pattern that helps one overcome this challenge: the in-mapper combiner.

The task of this exercise is to repeat all of the steps from Exercise 1 by carefully making the appropriate amendments related to exercise 3. The only difference from exercise 1 is in the WordCombiner class in which you can see how an associative array is used to locally aggregate the data.

Question

Can you predict how the numbers from the output of the MapReduce job look like? Based on your prediction calculate how much the network cost is reduced.

Exercise 4 – enhanced in-mapper combiner

Exercise 3 introduced a very useful design pattern for local aggregation but we can do better. We can achieve maximum local aggregation by exploiting the fact that MapReduce jobs are written in Java. By completing the previous exercise we noticed the "map()" method of the mapper class is executed for every line from a file – that means the previous method can aggregate the data only at line level. The trick to achieve maximum local aggregation is to move the associative array outside the map() method, initialize it at the beginning of the Map task, collect the data in the map() method and output the aggregated key-value pairs at the end of the task after all the data from an input split (in our case a whole file) that is associated with the Map task has been processed.

The task of this exercise is to repeat all of the steps from Exercise 1 by carefully making the appropriate amendments related to exercise 4. The code differences from the previous exercises are found in the WordMapper class.

Question

Can you predict how the numbers from the output of the MapReduce job look like? Are they different from the output from exercises 2 or 3? Based on your prediction calculate how much the network cost is reduced.