CSC 3331: Analysis of Algorithms
Parallel Programming and MapReduce Framework
Lab Tutorial

Part 1: Preparing Hadoop Work Environment with VirtualBox and Cloudera VM

The purpose of this part is (1) to get you started with Hadoop and (2) to get you acquainted with the process of compiling, and executing a simple MapReduce program in Hadoop. We will utilize Cloudera Quickstart VM in VirtualBox environment which has all the necessary packages installed and configured. Please follow the following steps meticulously.

1. Download and install VirtualBox on your machine: https://www.virtualbox.org/wiki/Downloads Make sure to choose appropriate distribution for your machine (i.e. Windows, OS X etc.)
2. Download the Cloudera Quickstart VM at https://www.cloudera.com/downloads/quickstart_vms/5-8.html Make sure to choose Virtual Box under platform before downloading. You may be asked to complete the product interest form before downloading. Mention the educational purpose there.
3. Uncompress the VM archive. It is compressed with 7-zip. If needed, you can download a tool to uncompress the archive at http://www.7-zip.org/.
4. Start VirtualBox and click Import Appliance in the File dropdown menu. Click the folder icon beside the location field. Browse to the uncompressed archive folder, select the .ovf file, and click the Open button. Click the Continue button. Click the Import button.
5. Your virtual machine should now appear in the left column. Select it and click on Start to launch it. It may take a while to start the VM.
6. To verify that the VM is running and you can access it, open a browser to the URL: http://localhost:8088. You should see the Hadoop resource manager UI (Figure 1).

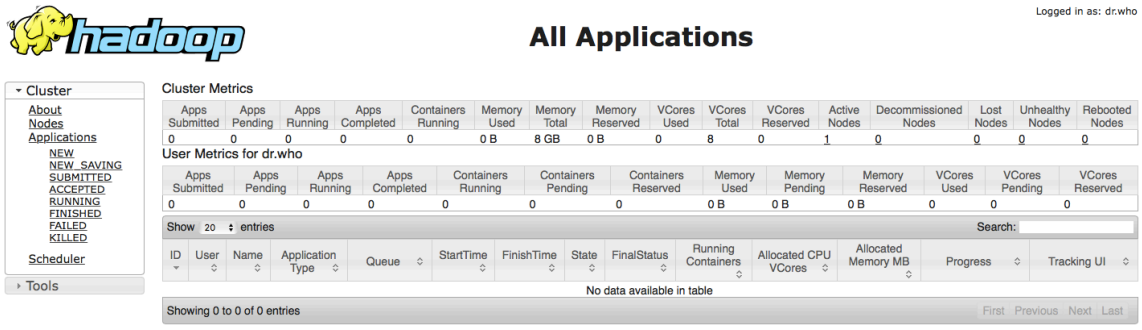

Figure 1: Hadoop Resource Manager UI

7. The virtual machine includes the following software
   - CentOS 6.4
   - JDK 7 (1.7.0 67)
   - Hadoop 2.5.0

The virtual machine runs best with 4096MB of RAM, but has been tested to function with 1024MB. Note that at 1024MB, while it did technically function, it was very slow to start up.

8. Take screenshots/pictures of your started VM and Hadoop Resource manager (step 6) and show that to the instructor.

Part 2: Compiling and executing your first MapReduce Application

In this section we will try to execute a MapReduce application (MaxTemperature.java) in Hadoop running on Cloudera quickstart VM. The required files are in Blackboard for you to download and save in the correct places as instructed below.

1. In your started Cloudera VM, open Terminal (Applications --> System Tools --> Terminal). If you just launched the VM, you may have to close the Firefox window to see the Cloudera desktop. You should see the following prompt
   ```
   [cloudera@quickstart ~]$
   ```
2. Create a folder `MaxT` in workspace folder by entering the following command
   ```
   $ mkdir workspace/MaxT
   ```
3. Change the current directory to MaxT
   ```
   $ cd workspace/MaxT
   ```
4. Using VM's browser (Firefox), download and save the following files in MaxT folder.
   `MaxTemperature.java, MaxTemperatureMapper.java, MaxTemperatureReducer.java, temp.txt`
5. Before you execute a MapReduce application, you must create input and output locations in HDFS. Use the following commands to create the input directory `/user/cloudera/wordcount/input` in HDFS.
   ```
   $ sudo su hdfs
   $ hadoop fs –mkdir /user/cloudera
   $ hadoop fs –chown cloudera /user/cloudera
   $ exit
   $ sudo su cloudera
   $ hadoop fs –mkdir /user/cloudera/MaxT /user/cloudera/MaxT/input
   ```
6. Move the input text file `temp.txt` to the `/user/cloudera/MaxT/input` directory in HDFS.
   ```
   $ hadoop fs –put temp.txt /user/cloudera/MaxT/input
   ```
7. Compile the java files. The second command should be all in the same line.
   ```
   $ mkdir –p build
   $ javac -cp /usr/lib/hadoop/*:/usr/lib/hadoop-mapreduce/*
   MaxTemperature.java MaxTemperatureMapper.java
   MaxTemperatureReducer.java -d build -Xlint
   ```
8. A folder named as `build` should appear in `MaxT` folder.
9. Create a Jar file for `MaxTemperature` Application
   ```
   $ jar -cvf maxtemperature.jar -C build/ .
   ```
10. `MaxT` should have a `maxtemperature.jar`
11. Run the `MaxTemperature` application from the JAR file, passing the paths to the input and output directories in HDFS.
    ```
    $ hadoop jar maxtemperature.jar MaxTemperature
    /user/cloudera/MaxT/input /user/cloudera/MaxT/output
    ```
    where `MaxTemperature` is the name of the class that contains the main function.
12. Ignore any warning, however you need to take care of the errors if any.
13. Executing the program won't show you the output directly. To view the output
    ```
    $ hadoop fs -cat /user/cloudera/MaxT/output/*
    ```
14. This should print the following two lines containing the maximum temperature of each respective year in our input data.
    ```
    1990    24
    1991    27
    ```
15. If you want to run the sample again, you first need to remove the output directory.
    ```
    $ hadoop fs -rm -r /user/cloudera/MaxT/output
    ```

16. Change the temp.txt to add few more years and temperature, Follow step 6 to 14 to see whether your program is working for added input data. If it is working correctly, take screenshots/pictures of your executed program and show that to the instructor.

Part 3: Compiling and executing your second MapReduce Application

In this section we will try to execute another MapReduce application (WordCount.java) in Hadoop running on Cloudera quickstart VM. The required files are in Blackboard for you to download and save in the correct places as instructed below.

1. In your started Cloudera VM, open Terminal (Applications --> System Tools --> Terminal). If you just launched the VM, you may have to close the Firefox window to see the Cloudera desktop. You should see the following prompt
   `[cloudera@quickstart ~]$`
2. Create a folder `WordCount` in workspace folder by entering the following command
   ```
   $ mkdir workspace/WordCount
   ```
3. Change the current directory to `WordCount`
   ```
   $ cd workspace/WordCount
   ```
4. Using VM's browser (Firefox), download and save the following files in `WordCount` folder. `WordCount.java, pg100.txt.` pg100.txt contains the Complete Works of William Shakespeare from Project Gutenberg.
5. Before you execute a MapReduce application, you must create input and output locations in HDFS. Use the following commands to create the input directory `/user/cloudera/wordcount/input` in HDFS.
   ```
   $ sudo su hdfs
   $ hadoop fs –mkdir /user/cloudera
   $ hadoop fs –chown cloudera /user/cloudera
   $ exit
   $ sudo su cloudera
   $ hadoop fs –mkdir /user/cloudera/WordCount
   /user/cloudera/WordCount/input
   ```
6. Move the input text file `temp.txt` to the `/user/cloudera/WordCount/input` directory in HDFS.
   ```
   $ hadoop fs –put pg100.txt /user/cloudera/WordCount/input
   ```
7. Compile the java files. The second command should be all in the same line.
   ```
   $ mkdir –p build
   $ javac -cp /usr/lib/hadoop/*:/usr/lib/hadoop-mapreduce/*
   WordCount.java -d build -Xlint
   ```
8. A folder named as `build` should appear in `WordCount` folder.
9. Create a Jar file for `WordCount` Application
   ```
   $ jar -cvf wordcount.jar -C build/ .
   ```
10. `WordCount folder` should have a `wordcount.jar`
11. Run the `WordCount` application from the JAR file, passing the paths to the input and output directories in HDFS.
    ```
    $ hadoop jar wordcount.jar WordCount
    /user/cloudera/WordCount/input /user/cloudera/WordCount/output
    ```
    where `WordCount` is the name of the class that contains the main function.
12. Ignore any warning, however you need to take care of the errors if any.
13. Executing the program won't show you the output directly. To view the output
    ```
    $ hadoop fs -cat /user/cloudera/WordCount/output/*
    ```
14. If you want to run the sample again, you first need to remove the output directory.
    ```
    $ hadoop fs -rm -r /user/cloudera/WordCount/output
    ```
15. Take screenshots/pictures of your executed program and show that to the instructor.