Hadoop MapReduce Tutorial

This tutorial will help you to get familiar with Hadoop and run a simple WordCount Example in Map Reduce.

It will teach you how to write, compile, debug and execute a simple Hadoop program. First part of this document serves as a tutorial and the second part asks you to write your own Hadoop program. You have to complete the tutorial individually.

1. Setting up a virtual machine

- Download and install VirtualBox on your machine
- The latest version of Cloudera Quickstart VM is placed at \\sandata\Xeon\ Zareen Alamgir\BIG DATA\TOOLS. Uncompress the VM archive.
- In VirtualBox and click Import Appliance in the File menu. The virtual machine should now appear in the left column. Select it and click on Start to launch it.

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.

2. Running Hadoop jobs

Generally Hadoop can be run in three modes.

1. **Standalone (or local) mode:** There are no daemons used in this mode. Hadoop uses the local file system as a substitute for HDFS file system. The jobs will run as if there is 1 mapper and 1 reducer.

2**. Pseudo-distributed mode:** All the daemons run on a single machine and this setting mimics the behavior of a cluster. All the daemons run on your machine locally using the HDFS protocol. There can be multiple mappers and reducers.

3. **Fully-distributed mode**: This is how Hadoop runs on a real cluster.

In this tutorial we will show you how to run Hadoop jobs in Standalone mode

(very useful for developing and debugging) and also in Pseudo-distributed mode (to mimic the behavior of a cluster environment).

2.1. Example project

In this section you will create a new Eclipse Hadoop project, compile, and execute it. The program will count the frequency of all the words in a given large text file. In your virtual machine, Hadoop, Java environment and Eclipse have already been pre-installed.

• Open Eclipse by clicking its icon on desktop.

• Right-click on the training node in the Package Explorer and select Copy. See Figure1.

Figure 1: Creating a Hadoop Project

• Right-click on the training node in the Package Explorer and select Paste. See

Figure 2. *Figure 2: Creating a Hadoop Project*

- In the pop-up dialog, enter the new project name in the Project name field and click OK. See Figure 3.

Figure 3: Creating a Hadoop Project

- Create a new package by right-clicking on the src node and selecting New -> Package. See Figure 4.

Figure 4: Creating a Hadoop Project

- Enter bigdata.wordcount in the Name field and click Finish.

- Create a new class in that package called WordCount by right-clicking on the bigdata.wordcount node and selecting New -> Class. See Figure 5.

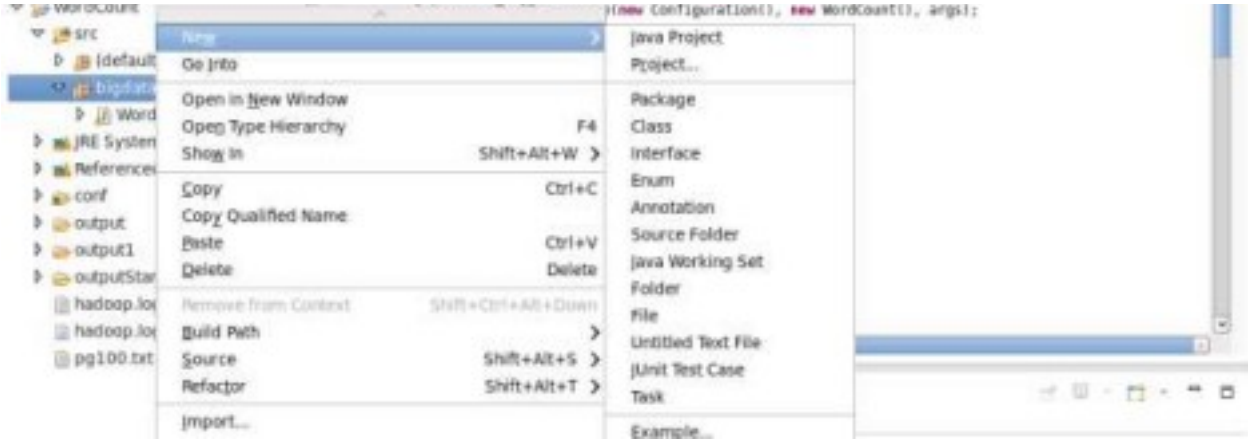

Figure 5: Creating a Java Class

• In the pop-up dialog, enter WordCount as the Name.

• In the Superclass field, enter Configured and click the Browse button. From the popup window select Configured - org:apache:hadoop:conf and click the OK button. See Figure 6.

Figure 6: Creating a Java File

• In the Interfaces section, click the Add button. From the pop-up window

select Tool org:apache:hadoop:util and click the OK button. See Figure 7.

Figure 7: Creating a Java File

- Check the boxes for public static void main(String args[]) and Inherited abstract methods and click the Finish button. See Figure 8.

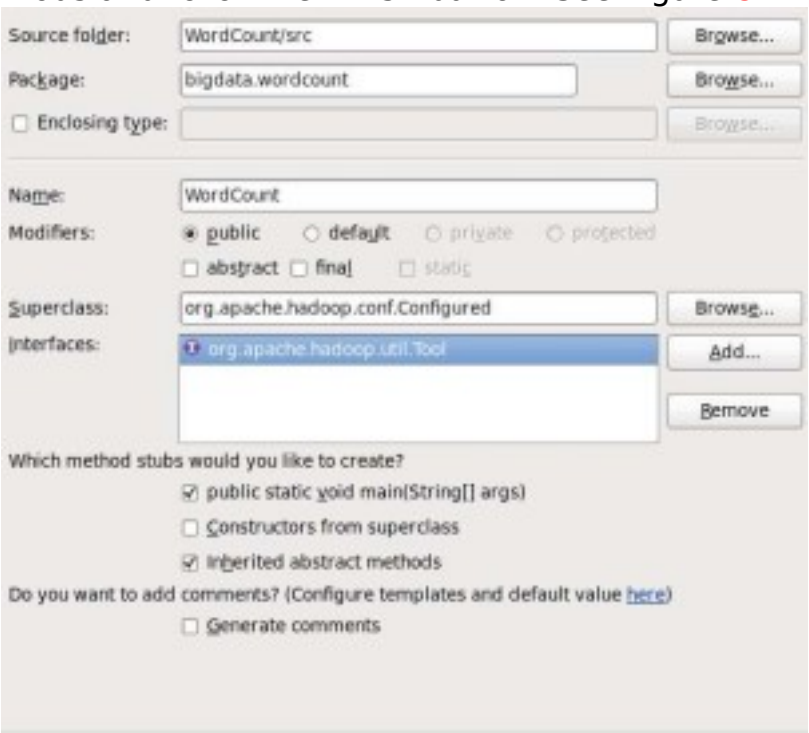

Figure 8: Creating a Java Class

• You will now have a rough skeleton of a Java file. You can now add code to this class to implement your Hadoop job.

• A sample WordCount.java file is provided with this tutorial.

2.1.1. Running Hadoop jobs in standalone mode

Once you've created your project and written the source code, to run the project in standalone mode, do the following:

• Right-click on the project and select Run As -> RunConfigurations. See Figure 9.

Figure 9: Run WordCount.java

• In the pop-up dialog, select the Java Application node and click the New launch configuration button in the upper left corner. See Figure 10.

Figure 10: Run WordCount.java

• Enter a name in the Name field and WordCount in the Main class field. • **Switch to the Arguments tab and put input.txt output in the Program arguments field. Click Apply and Close.**

• Place the input.txt file in the ~/workspace/WordCount folder.

• Right-click on the project and select Run As ->Java Application. See Figure 11.

Figure 11: Run WordCount.java

• In the pop-up dialog select WordCount – bigdata.wordcount from the selection list and click OK.

• You will see the command output in the console window, and if the job succeeds, you'll find the results in the ~/workspace/WordCount/output directory. If the job fails complaining that it cannot find the input file, make sure that the input.txt file is located in the ~/workspace/WordCount directory.

• If you get errors regarding the Jackson-mapper dependencies. Follow the screen shots in the zip file named “screenshots for adding dependencies to eclipse”.

Running Hadoop in pseudo-distributed mode

Once you've created your project and written the source code, to run the project in pseudo distributed mode, do the following

• Right-click on the project and **Refresh**. After this right-click on the project and select Export. See Figure 12.

Figure 12

• In the pop-up dialog, expand the Java node and select JAR file. See Figure 13. Click Next >

Figure 13: Export a Hadoop Project

Enter /home/cloudera/wordcount.jar in the JAR file field and click Finish. See Figure 14.

Figure 14

• If you see an error dialog warning that the project compiled with warnings, you can simply click OK.

• Place input.txt file in the folder /home/cloudera/. Now open a terminal and run the following commands:

```
hadoop fs -put workspace/WordCount/input.txt
hadoop jar wordcount.jar bigdata.wordcount.WordCount
```

input.txt output • Run the command: hadoop fs -ls output

- You should see an output file for each reducer. Since there was only one reducer for this job, you should only see one part-* file. Note that sometimes the files will be called part-NNNNN, and sometimes they'll be called part-r-NNNNN. See Figure.
- To view the job's logs, open the browser in the VM and point it to http://localhost: 8088 as in Figure 15

Figure 15

Debugging Hadoop jobs

To debug an issue with a job, the easiest approach is to run the job in stand-alone mode and use a debugger. To debug your job, do the following steps:

1. Right-click on the project and select Debug As -> Java Application. See Figure 16.

Figure 16: Debug a Hadoop Project

2. In the pop-up dialog select the main class from the selection list and click OK. See Figure 17.

Figure 17: Debug a Hadoop Project

You can use the Eclipse debugging features to debug your job execution. See the additional Eclipse tutorials at the end of section for help using the Eclipse debugger.

When running your job in pseudo-distributed mode, the output from the job is logged in the task tracker's log files, which can be accessed most easily by pointing a web browser to port 8088 of the server, which will the localhost. From the job tracker web page, you can drill down into the failing job, the failing task, the failed attempt, and finally the log files. Note that the logs for stdout and stderr are separated, which can be useful when trying to isolate specific debugging print statements.

This tutorial is adapted from the Hadoop Tutorial given in the course Mining of Massive Dataset (Stanford).

Further Hadoop tutorials

• Yahoo! Hadoop Tutorial: http://developer.yahoo.com/hadoop/tutorial/ •

Cloudera Hadoop Tutorial: http://www.cloudera.com/content/cloudera-content/cloudera docs/HadoopTutorial/CDH4/Hadoop-Tutorial.html

• How to Debug MapReduce Programs:

http://wiki.apache.org/hadoop/HowToDebugMapReducePrograms

Further Eclipse tutorials

• Genera Eclipse tutorial:

http://www.vogella.com/articles/Eclipse/article.html. • Tutorial on how to use the Eclipse debugger:
http://www.vogella.com/articles/EclipseDebugging/article.html