Hadoop MapReduce Code for WordCount

1. sudo su – Hadoop

We first SSH into our master node. This helps ensure that our users is changed from root to Hadoop.

1. start-all.sh

We then entered this line of command to ensure that all the essential components of the Hadoop cluster are running and ready to process the dataset.

1. ping slave1 and slave2

We then ping both the slave1 and slave2 slave nodes to ensure that it is up and running.

1. wget “https://1500kdataset.s3.amazonaws.com/1500k\_Books.csv”

We the used the wget method to put the dataset into the Hadoop server.

1. cd IST3134/

We then changed our directory to the IST3134 directory.

1. unzip wordcount.zip

Once in this directory, we then unzip the wordcount.zip folder in order to obtain the three portions of the wordcount which is the driver code (WordCount.java), the mapper (WordMapper.java) and the reducer (SumReducer.java).

1. mkdir ~/workspace

We then created a workspace directory in our home directory.

1. cp -r wordcount/ ~/workspace/

We then copied the wordcount directory to the workspace directory.

1. hadoop fs -put 1500k\_Books.csv /user/hadoop/1500k\_Books.csv

We then upload the dataset 1500k\_Books.csv into the HDFS server.

1. cd ~/workspace/wordcount/src

We then changed our directory back to the workspace/wordcount directory.

1. javac -classpath `hadoop classpath` stubs/\*.java

We then compiled the three Java classes.

1. jar cvf wc.jar stubs/\*.class

Afterwards, we then compiled the Java files into a JAR file.

1. hadoop jar wc.jar stubs.WordCount 1500k\_Books.csv wordcounts121

We then submitted the MapReduce job to Hadoop using the JAR file to count the occurrences of each word in 1500k\_Books.csv file.

1. hadoop fs -ls wordcounts121

We then used this to review the results of our MapReduce job.

1. hadoop fs -cat wordcounts121/part-r-00000 | less

We then used this line of code to view the content of the output for our MapReduce job.