**CSC14118**

**Introduction to Big Data**

**Group 03**

**Lab02: MapReduce Programming**

|  |  |
| --- | --- |
| **Student ID** | **Full name** |
| 20127449 | Trần Quốc Bảo |
| 20127452 | Hồ Đăng Cao |
| 20127476 | Đỗ Đức Duy |

**Project Version Date**

PLAN001 v1.0 2023-07-22

# Summary

|  |  |  |
| --- | --- | --- |
| **Section** | **Completed percentage** | **Issues** |
| S01 | 100% |  |
| S02 | 100% |  |
| S03 | 100% |  |
| S04 | 100% |  |
| S05 | 100% |  |
| S06 | 100% |  |
| S07 | 100% |  |
| S08 | 100% |  |
| S09 | 100% |  |
| S10 | 10% |  |

Contents

[Summary 2](#_Toc140578954)

[1 Wordcount Program 4](#_Toc140578955)

[2 WordSizeWordCount Program 5](#_Toc140578956)

[3 WeatherData Program 8](#_Toc140578957)

[4 Patent Program 9](#_Toc140578958)

[5 MaxTemp Program 10](#_Toc140578959)

[6 AverageSalary Program 11](#_Toc140578960)

[7 De Identify HealthCare Program 12](#_Toc140578961)

[8 Music Track Program 13](#_Toc140578962)

[9 Telecom Call Data Record Program 18](#_Toc140578963)

[10 Count Connected Component Program 19](#_Toc140578964)

[References 19](#_Toc140578965)

# 1 Wordcount Program

**Note:** The source code is taken from the source code provided at lab 01.

Step 1: Type the following command to export the hadoop classpath into bash.

export HADOOP\_CLASSPATH=$(hadoop classpath)

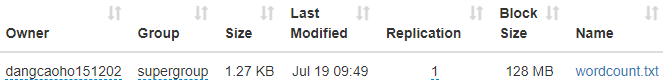
echo $HADOOP\_CLASSPATH

Step 2: Create directories on hdfs and put the input data file to hdfs.

hadoop fs -mkdir /WordCount

hadoop fs -mkdir /WordCount/Input

hadoop fs -put <input file’s path> /WordCount/Input



Step 3: Compile the WeatherData.java file

javac -classpath $HADOOP\_CLASSPATH -d <classes folder’s path> <source’s path>



Step 4: Put the output files in a jar file.

jar -cvf <.jar file’s path> -C <classes folder’s path> .

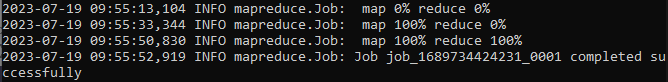
A computer screen with numbers and symbols

Description automatically generated

Step 5: Run the jar file on Hadoop.

hadoop jar <.jar file’s path> WordCount /WordCount/Input /WordCount/Output





Step 6: Result

hadoop dfs -cat /WordCount/Output/\*

A screen shot of a computer

Description automatically generated

# 2 WordSizeWordCount Program

**Note**: the source code is based on the provided requirements file and [link](https://github.com/Rkrahul04/Word_size_Count_Mapreduce).

Step 1: Type the following command to export the hadoop classpath into bash.

export HADOOP\_CLASSPATH=$(hadoop classpath)

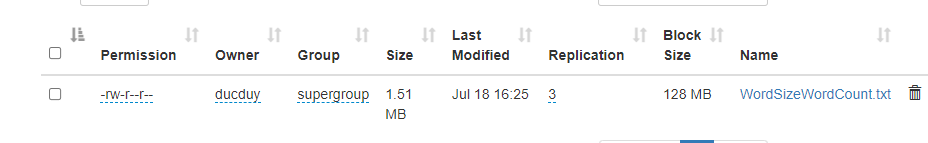
echo $HADOOP\_CLASSPATH

Step 2: Create directories on hdfs and put the input data file to hdfs.

hadoop fs -mkdir /WordSizeWordCount

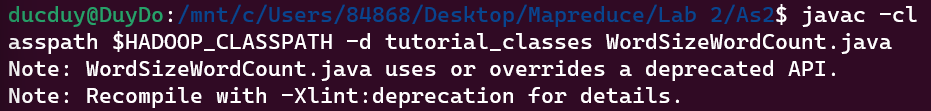
hadoop fs -mkdir /WordSizeWordCount/Input

hadoop fs -put <input file’s path> /WordSizeWordCount/Input



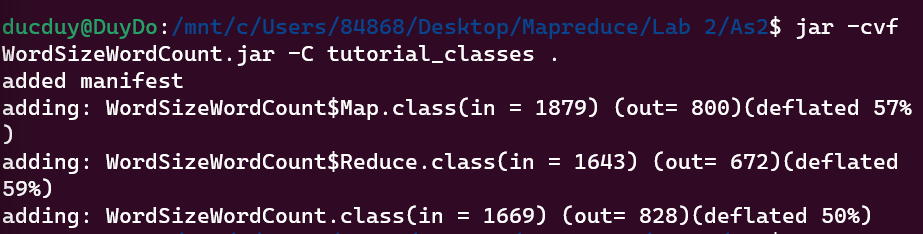
Step 3: Compile the WordSizeWordCount.java file

javac -classpath $HADOOP\_CLASSPATH -d <classes folder’s path> <source’s path>



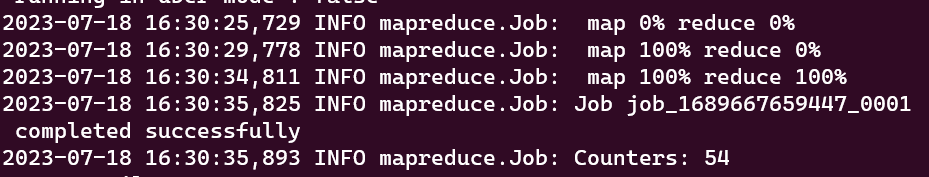
Step 4: Put the output files in a jar file.

jar -cvf <.jar file’s path> -C <classes folder’s path> .



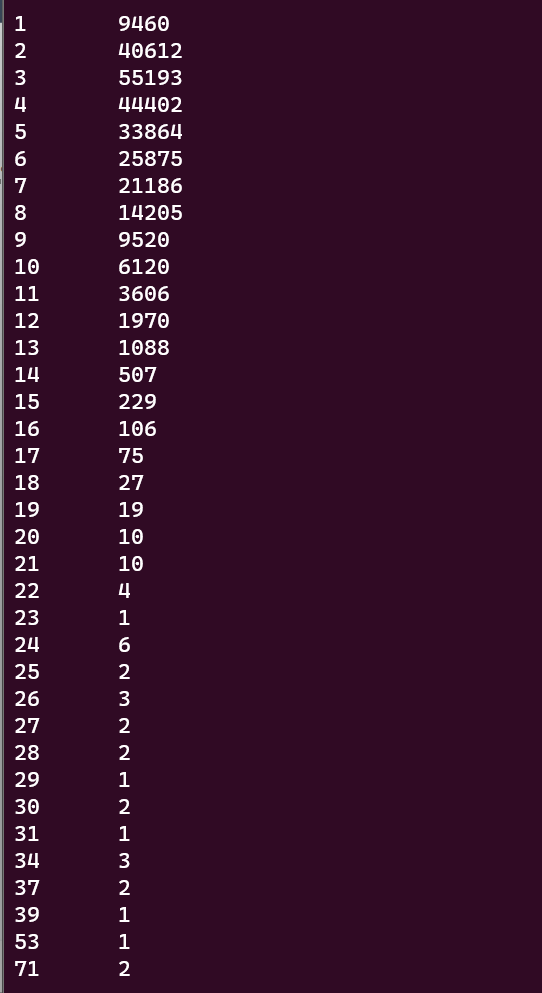
Step 5: Run the jar file on Hadoop.

hadoop jar <.jar file’s path> WordSizeWordCount /WordSizeWordCount/Input /WordSizeWordCount/Output



Step 6: Result

hadoop dfs -cat /WordSizeWordCount/Output/\*



# 3 WeatherData Program

**Note:** the source code is referenced from the provided requirements file.

Step 1: Type the following command to export the hadoop classpath into bash.

export HADOOP\_CLASSPATH=$(hadoop classpath)

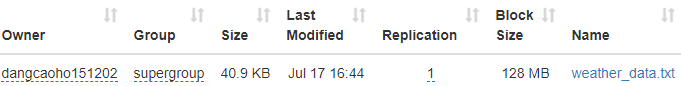
echo $HADOOP\_CLASSPATH

Step 2: Create directories on hdfs and put the input data file to hdfs.

hadoop fs -mkdir /WeatherData

hadoop fs -mkdir /WeatherData/Input

hadoop fs -put <input file’s path> /WeatherData/Input



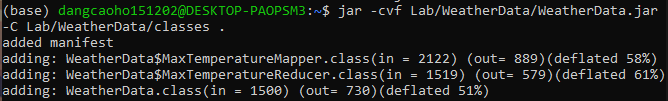
Step 3: Compile the WeatherData.java file

javac -classpath $HADOOP\_CLASSPATH -d <classes folder’s path> <source’s path>



Step 4: Put the output files in a jar file.

jar -cvf <.jar file’s path> -C <classes folder’s path> .



Step 5: Run the jar file on Hadoop.

hadoop jar <.jar file’s path> WeatherData /WeatherData/Input /WeatherData/Output

A black background with white text

Description automatically generated

Step 6: Result

hadoop dfs -cat /WeatherData/Output/\*

A screenshot of a computer program

Description automatically generated

# 4 Patent Program

**Note**: the source code is based on the provided requirements file and [link](https://github.com/Rkrahul04/Sub-Patents_count_mapreduce/blob/master/patent.java).

Step 1: Type the following command to export the hadoop classpath into bash.

export HADOOP\_CLASSPATH=$(hadoop classpath)

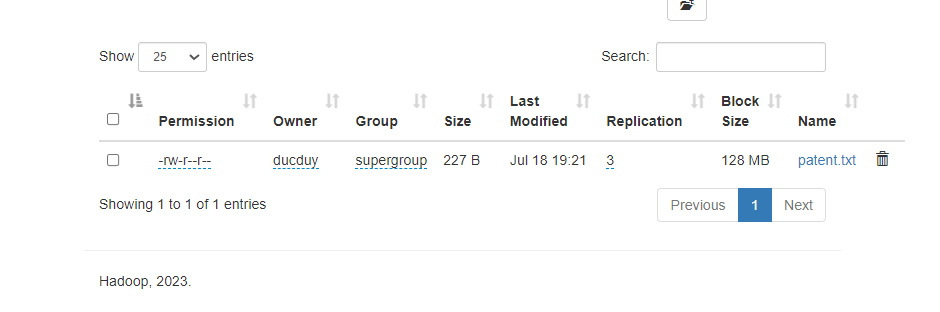
echo $HADOOP\_CLASSPATH

Step 2: Create directories on hdfs and put the input data file to hdfs.

hadoop fs -mkdir /Patent

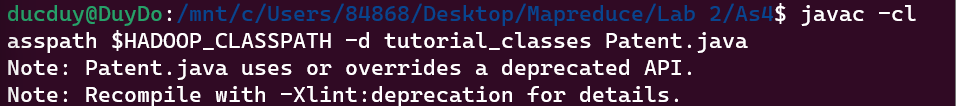
hadoop fs -mkdir /Patent/Input

hadoop fs -put <input file’s path> /Patent/Input



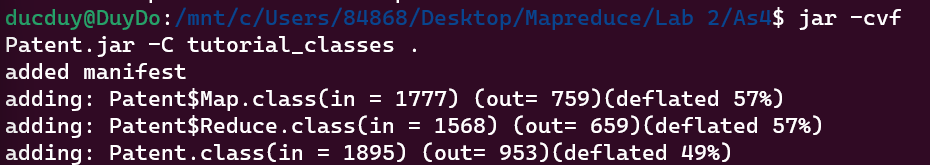
Step 3: Compile the Patent.java file

javac -classpath $HADOOP\_CLASSPATH -d <classes folder’s path> <source’s path>



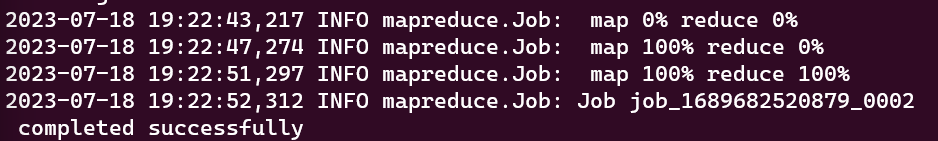
Step 4: Put the output files in a jar file.

jar -cvf <.jar file’s path> -C <classes folder’s path> .



Step 5: Run the jar file on Hadoop.

hadoop jar <.jar file’s path> Patent /Patent/Input /Patent/Output



Step 6: Result

hadoop dfs -cat /Patent/Output/\*

A screenshot of a computer

Description automatically generated

# 5 MaxTemp Program

**Note**: the source code is based on the provided requirements file.

Step 1: Type the following command to export the hadoop classpath into bash.

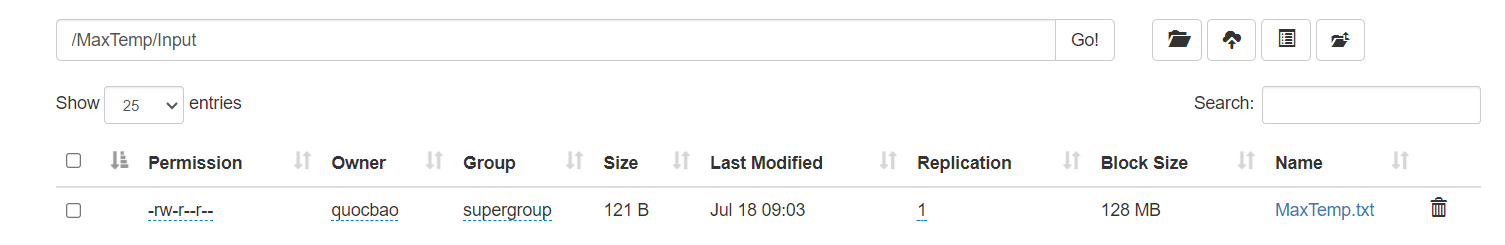
export HADOOP\_CLASSPATH=$(hadoop classpath)

Step 2: Create directories on hdfs and put the input data file to hdfs.

hadoop fs -mkdir /MaxTemp

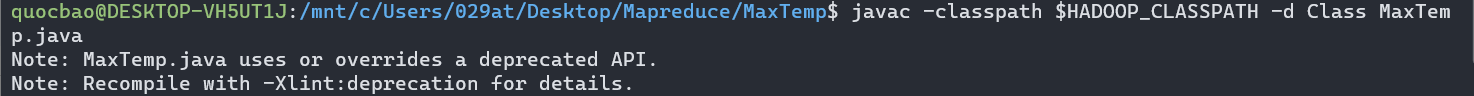
hadoop fs -mkdir /MaxTemp/Input

hadoop fs -put MaxTemp.txt /MaxTemp/Input



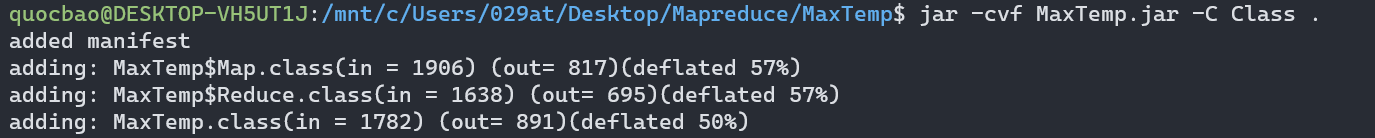
Step 3: Compile the MaxTemp.java file

javac -classpath $HADOOP\_CLASSPATH -d <classes folder’s path> <source’s path>



Step 4: Put the output files in a jar file.

jar -cvf <.jar file’s path> -C <classes folder’s path> .



Step 5: Run the jar file on Hadoop.

hadoop jar MaxTemp.jar MaxTemp MaxTemp/Input /MaxTemp/Output

A black screen with white text

Description automatically generated

Step 6: Result

hadoop dfs -cat /MaxTemp/Output/\*

A screen shot of a computer

Description automatically generated

# 6 AverageSalary Program

**Note**: the source code is based on the provided requirements file.

Step 1: Type the following command to export the hadoop classpath into bash.

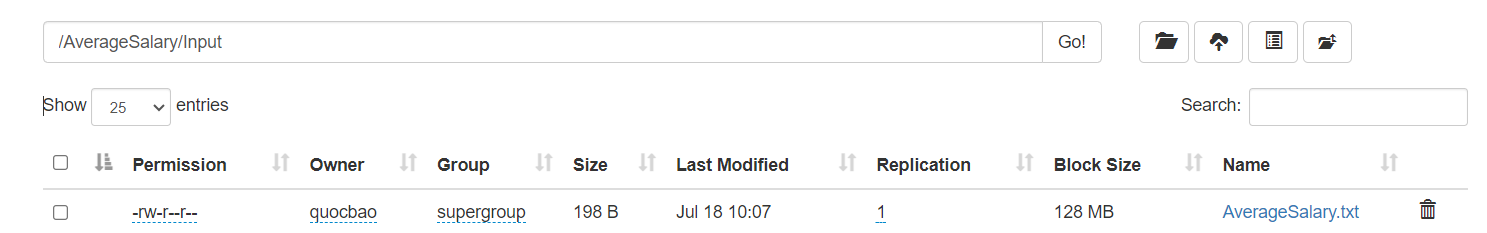
export HADOOP\_CLASSPATH=$(hadoop classpath)

Step 2: Create directories on hdfs and put the input data file to hdfs.

hadoop fs -mkdir /AverageSalary

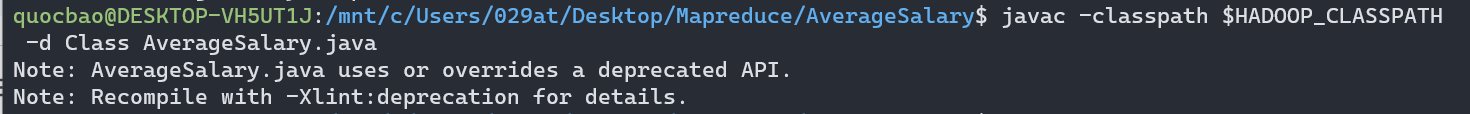
hadoop fs -mkdir /AverageSalary/Input

hadoop fs -put AverageSalary.txt /AverageSalary/Input



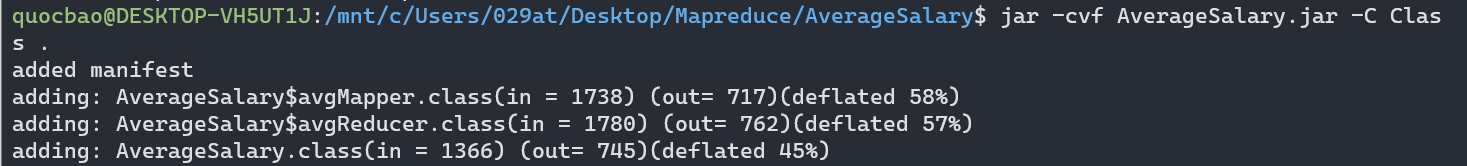
Step 3: Compile the AverageSalary.java file

javac -classpath $HADOOP\_CLASSPATH -d <classes folder’s path> <source’s path>



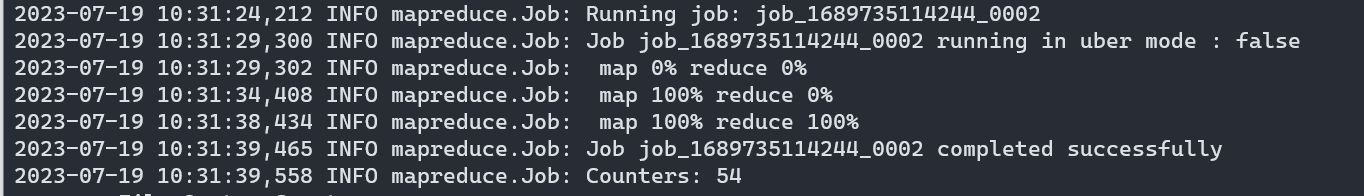
Step 4: Put the output files in a jar file.

jar -cvf <.jar file’s path> -C <classes folder’s path> .



Step 5: Run the jar file on Hadoop.

hadoop jar  AverageSalary.jar  AverageSalary AverageSalary/Input /AverageSalary/Output



Step 6: Result

hadoop dfs -cat /AverageSalary/Output/\*

A screen shot of a computer

Description automatically generated

# 7 De Identify HealthCare Program

**Note**: the source code is based on the provided requirements file.

Step 1: Type the following command to export the hadoop classpath into bash.

export HADOOP\_CLASSPATH=$(hadoop classpath)

Step 2: Create directories on hdfs and put the input data file to hdfs.

hadoop fs -mkdir /DeIdentifyData

hadoop fs -mkdir /DeIdentifyData/Input

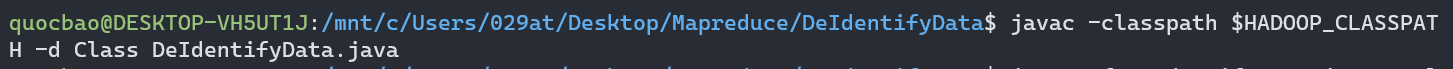
hadoop fs -put DeIdentifyData.txt  /DeIdentifyData

A screenshot of a computer

Description automatically generated

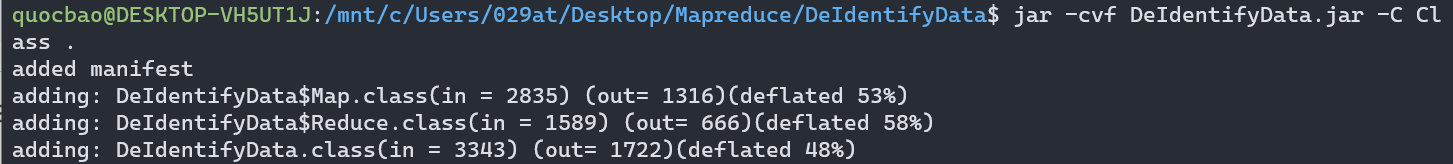
Step 3: Compile the DeIdentifyData.java file

javac -classpath $HADOOP\_CLASSPATH -d <classes folder’s path> <source’s path>



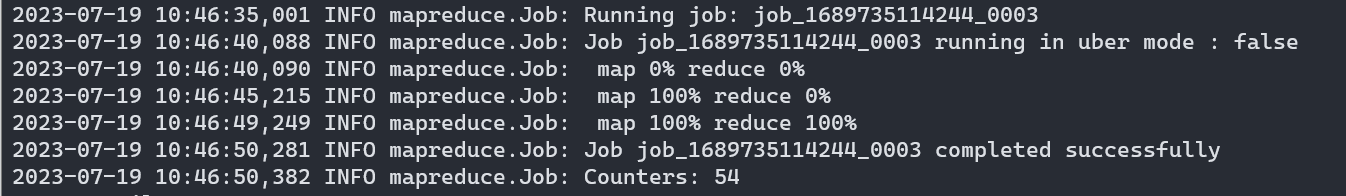
Step 4: Put the output files in a jar file.

jar -cvf <.jar file’s path> -C <classes folder’s path> .



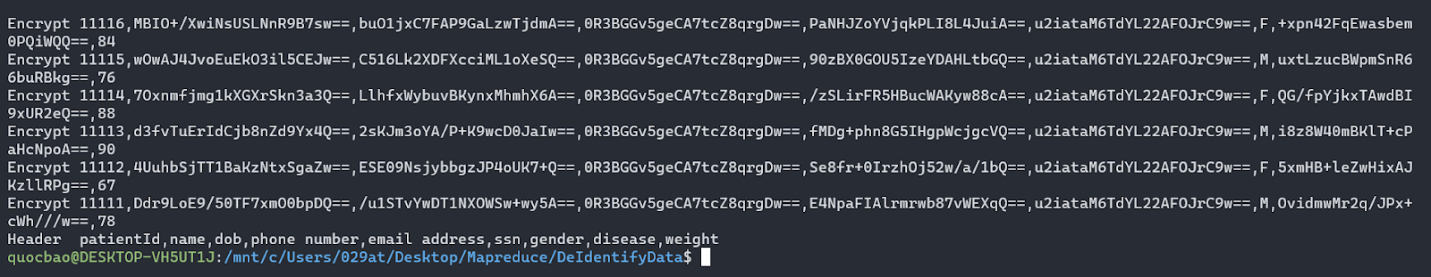
Step 5: Run the jar file on Hadoop.

hadoop jar  DeIdentifyData.jar  DeIdentifyData DeIdentifyData/Input /DeIdentifyData/Output



Step 6: Result

hadoop dfs -cat /DeIdentifyData/Output/\*



# 8 Music Track Program

**Note:** Statement 1 is based on the provided requirements file.

The statements from 2 to 5 are based on the [Blogs](https://my-learnings-about-hadoop.blogspot.com/2017/08/mapreduce-real-timeexamples.html).

Step 1: Type the following command to export the hadoop classpath into bash.

export HADOOP\_CLASSPATH=$(hadoop classpath)

echo $HADOOP\_CLASSPATH

Step 2: Create directories on hdfs and put the input data file to hdfs.

hadoop fs -mkdir /MusicTrack

hadoop fs -mkdir /MusicTrack/Input

hadoop fs -put <input file’s path> /MusicTrack/Input

A screen shot of a computer

Description automatically generated

**Statement 1:** Number of unique listeners

Step 3: Compile the UniqueListeners.java file

javac -classpath $HADOOP\_CLASSPATH -d <classes folder’s path> <source’s path>



Step 4: Put the output files in a jar file.

jar -cvf <.jar file’s path> -C <classes folder’s path> .

A computer screen with text on it

Description automatically generated

Step 5: Run the jar file on Hadoop.

hadoop jar <.jar file’s path> UniqueListeners /MusicTrack/Input /MusicTrack/UniqueListeners/Output



A black background with white text

Description automatically generated

Step 6: Result

hadoop dfs -cat /MusicTrack/UniqueListeners/Output/\*

A screen shot of a computer

Description automatically generated

**Statement 2:** Number of times the track was shared with others

Step 3: Compile the SharedOthers.java file

javac -classpath $HADOOP\_CLASSPATH -d <classes folder’s path> <source’s path>



Step 4: Put the output files in a jar file.

jar -cvf <.jar file’s path> -C <classes folder’s path> .

A computer screen with white text

Description automatically generated

Step 5: Run the jar file on Hadoop.

hadoop jar <.jar file’s path> SharedOthers /MusicTrack/Input /MusicTrack/SharedOthers/Output



A black background with white text

Description automatically generated

Step 6: Result

hadoop dfs -cat /MusicTrack/SharedOthers/Output/\*

A screen shot of a computer

Description automatically generated

**Statement 3:** Number of times the track was listened to on the radio

Step 3: Compile the ListenedRadio.java file

javac -classpath $HADOOP\_CLASSPATH -d <classes folder’s path> <source’s path>



Step 4: Put the output files in a jar file.

jar -cvf <.jar file’s path> -C <classes folder’s path> .

A computer screen shot of a program

Description automatically generated

Step 5: Run the jar file on Hadoop.

hadoop jar <.jar file’s path> ListenedRadio /MusicTrack/Input /MusicTrack/ListenedRadio/Output



A black background with white text

Description automatically generated

Step 6: Result

hadoop dfs -cat /MusicTrack/ListenedRadio/Output/\*

A screenshot of a computer

Description automatically generated

**Statement 4:** Number of times the track was listened to in total

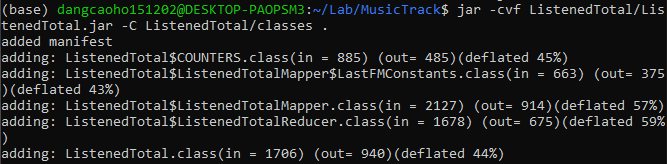
Step 3: Compile the ListenedTotal.java file

javac -classpath $HADOOP\_CLASSPATH -d <classes folder’s path> <source’s path>



Step 4: Put the output files in a jar file.

jar -cvf <.jar file’s path> -C <classes folder’s path> .



Step 5: Run the jar file on Hadoop.

hadoop jar <.jar file’s path> ListenedTotal /MusicTrack/Input /MusicTrack/ListenedTotal/Output



A black background with white text

Description automatically generated

Step 6: Result

hadoop dfs -cat /MusicTrack/ListenedTotal/Output/\*

A screen shot of a computer

Description automatically generated

**Statement 5:** Number of times the track was skipped on the radio

Step 3: Compile the SkippedRadio.java file

javac -classpath $HADOOP\_CLASSPATH -d <classes folder’s path> <source’s path>



Step 4: Put the output files in a jar file.

jar -cvf <.jar file’s path> -C <classes folder’s path> .

A computer screen with white text

Description automatically generated

Step 5: Run the jar file on Hadoop.

hadoop jar <.jar file’s path> SkippedRadio /MusicTrack/Input /MusicTrack/SkippedRadio/Output



A black background with white text

Description automatically generated

Step 6: Result

hadoop dfs -cat /MusicTrack/SkippedRadio/Output/\*

A screen shot of a computer

Description automatically generated

# 9 Telecom Call Data Record Program

**Note**: the source code is referenced from the provided requirements file.

Step 1: Type the following command to export the hadoop classpath into bash.

export HADOOP\_CLASSPATH=$(hadoop classpath)

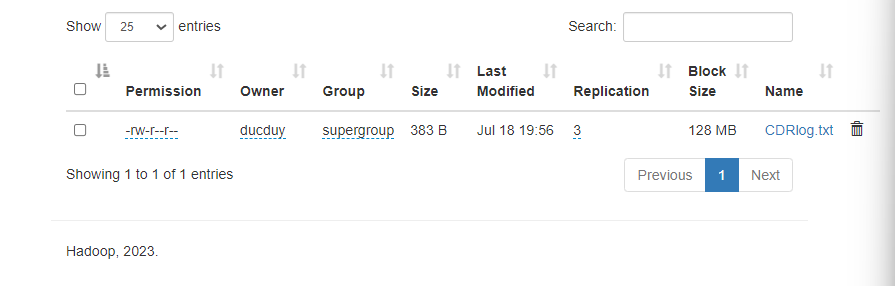
echo $HADOOP\_CLASSPATH

Step 2: Create directories on hdfs and put the input data file to hdfs.

hadoop fs -mkdir /CDRlog

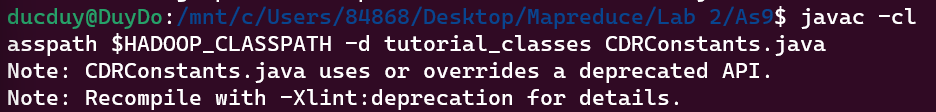
hadoop fs -mkdir /CDRlog/Input

hadoop fs -put <input file’s path> /CDRlog/Input



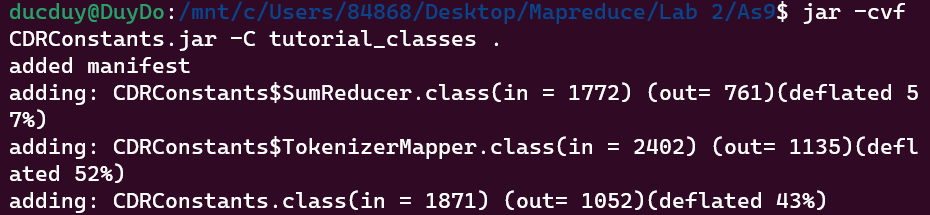
Step 3: Compile the CDRConstants.java file

javac -classpath $HADOOP\_CLASSPATH -d <classes folder’s path> <source’s path>



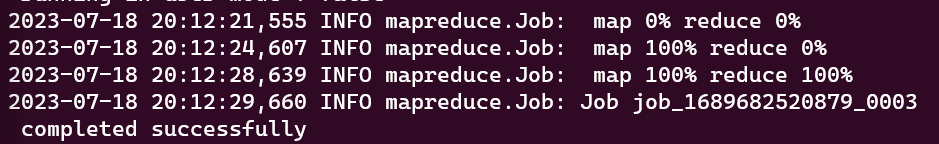
Step 4: Put the output files in a jar file.

jar -cvf <.jar file’s path> -C <classes folder’s path> .



Step 5: Run the jar file on Hadoop.

hadoop jar <.jar file’s path> CDRlog /CDRlog/Input /CDRlog/Output



Step 6: Result

hadoop dfs -cat /CDRlog/Output/\*

A screenshot of a computer

Description automatically generated

# 10 Count Connected Component Program

Step 1: Type the following command to export the hadoop classpath into bash.

export HADOOP\_CLASSPATH=$(hadoop classpath)

echo $HADOOP\_CLASSPATH

Step 2: Create directories on hdfs and put the input data file to hdfs.

hadoop fs -mkdir /CountConnectedComponentProgram

hadoop fs -mkdir /CountConnectedComponentProgram/Input

hadoop fs -put <input file’s path> /CountConnectedComponentProgram/Input

A screen shot of a computer

Description automatically generated

# References

The provided requirements file.

Manohar, 2 August 2017, [MapReduce Real time](https://my-learnings-about-hadoop.blogspot.com/2017/08/mapreduce-real-timeexamples.html).

Rkrahul04, May 8, 2017, [Word\_size\_Count\_Mapreduce](https://github.com/Rkrahul04/Word_size_Count_Mapreduce).

Rkrahul04, 2017, [Sub-Patents\_count\_mapreduce](https://github.com/Rkrahul04/Sub-Patents_count_mapreduce/blob/master/patent.java).