****

**Athens University of Economy and Business**

**School of Information Sciences & Technology: Department of Informatics**

**Master of Science in “Data Science”**

**Course: Large Scale Data Management**

**“Project-1 – Hadoop MapReduce”**

**Student**: Dimitris Sttathopoulos**:**

**Email:** [demetresstathopoulos8@gmail.com](mailto:demetresstathopoulos8@gmail.com)

A.M: f3352318

**Part 1:**

Firstly,

we downloaded a book in plain text format from the following link: <https://www.gutenberg.org/cache/epub/72849/pg72849.txt>

Following is the terminal input to acquire this (happened inside vagrant virtual enviroment)

INPUT:

$ wget [https://www.gutenberg.org/cache/epub/72849/pg72849.txt -O private\_chivarly.txt](https://www.gutenberg.org/cache/epub/72849/pg72849.txt%20-O%20private_chivarly.txt)

OUTPUT:

A computer screen shot of a black screen

Description automatically generated

We copy the file to the namenode of our hdfs system by using the following command:

INPUT:

$ docker cp private\_chivarly.txt namenode:/

OUTPUT:

Successfully copied 593kB to namenode:/

We are specifying where exactly the file should be imported, that is , /user/hdfs/input/

INPUT:

$ docker exec namenode hdfs dfs -put private\_chivarly.txt /user/hdfs/input/

OUTPUT:

2024-02-01 17:18:21,184 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false

We can also see the file by executing:

INPUT:

$docker exec namenode hdfs dfs -ls /user/hdfs/input/private\_chivarly.txt

OUTPUT:

-rw-r--r-- 3 root supergroup 591620 2024-02-01 17:18 /user/hdfs/input/private\_chivarly.txt

Next, we are editing the java/driver.java file to accept the private\_chivarly.txt instead of the initial value of “MobyDick.txt”.

Specifically, we changed:  


Then, after rebooting the vagrant virtual environment we are ready to execute the jar file by :

INPUT:

$ docker cp /vagrant/hadoop-mapreduce-examples/target/hadoop-map-reduce-examples-1.0-SNAPSHOT-jar-with-dependencies.jar namenode:/

OUTPUT:

Successfully copied 24.4MB to namenode:/

INPUT:

$ docker exec namenode hadoop jar /hadoop-map-reduce-examples-1.0-SNAPSHOT-jar-with-dependencies.jar

OUTPUT:

A computer screen with text on it

Description automatically generated

We can observe the word count happened to the new improted file (private\_chivarly.txt) by executing:

INPUT:

docker exec namenode hdfs dfs -text /user/hdfs/output/part-r-00000 | head -2

OUPUT:

Word counts for the top 20 words.

A screen shot of a computer

Description automatically generated

**Part 2:**

**Our mapreduce java class:**

package gr.aueb.panagiotisl.mapreduce.wordcount;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

import javax.naming.Context;

public class mapreduce\_queries {

    public static class CountMapper extends Mapper<LongWritable, Text, Text, Text> {

        // declare outputKey and outputValue as class members

        private final Text outputKey = new Text();

        private final Text outputValue = new Text();

        /\*\*

         \* @param key

         \* @param value

         \* @param context

         \* @throws IOException

         \* @throws InterruptedException

         \*/

        @Override

        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

            // ignore the header(first line)

            if (key.get() == 0){ // if line number(index) == 0 pass

                return;

            }

            // split a line into words

            String[] tokens = value.toString().split(",(?=(?:[^\"]\*\"[^\"]\*\")\*[^\"]\*$)", -1);

            // variables for storing values

            String song\_name = null;

            String country = null;

            String date = null;

            String dance = null;

            // extract value from the string "value"

            tokens[6] = tokens[6].replaceAll("\"", "");

            // check first if the word is missing from 6th position -> country

            if(tokens[6] != null && !tokens[6].isEmpty()){

                // output (word, 1)

                // run down all the array(tokens) and hold only those string values that

                // are in the positions 1, 6, 7, 13 referencing back to the original csv distribution of columns

                for (int i = 0; i < tokens.length; i++) {

                    // here we cut down the not needed columns based on position on the array

                    // this happens for each line of the csv at a time!

                        // for the song\_name

                        if (i == 1){song\_name = tokens[i];}

                        // for example (word(country), 6)}

                        if( i == 6 ) {country = tokens[i];}

                        // get the date from the csv

                        if (i == 7){date = tokens[i];}

                        // get the danceability number from the csv

                        // import it as string

                        if (i == 13){dance = tokens[i];}

                    }

                // i.e. derive the format 2020-01 from 2020-01-10

                date = date.substring(1, 8);

                // build the keys

                outputKey.set(new Text(country + ":" + date));

                // build the output values

                outputValue.set(new Text(song\_name + "," + dance));

                // build the list that the reducer will utilize

                context.write(outputKey, outputValue);

            }

            // else pass the current line (country = empty-string)

            else{return;}

        }

    }

    public static class CountReducer extends Reducer<Text, Text, Text, Text> {

         // declare outputKey and outputValue as class members

         private final Text outputKey = new Text();

         private final Text outputValue = new Text();

        @Override

        public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

            float sum = 0;

            int total = 0;

            // variables for storing values

            String song\_name = null;

            Float dance = null;

            // variables for storing values

            String mostDanceableSong = null;

            float maxDanceability = Float.MIN\_VALUE; // very small float value

            // variable to track whether the most danceable song has been found

            Boolean mostDanceableFound = null;

            // iterate throw the list values

            for (Text value: values){

                // separate the values by commas as the input format suggests

                String[] tokens = value.toString().split(",(?=(?:[^\"]\*\"[^\"]\*\")\*[^\"]\*$)", -1);

                // parse the dance value to float and increment the sum

                // but first remove any unnecessary quotes inside the tokens[1]

                float danceValue = Float.parseFloat(tokens[1].replaceAll("\"", ""));

                song\_name = tokens[0];

                sum += danceValue;

                total++; // increment the total count

                // update most danceable song if current song is more danceable

                if (danceValue > maxDanceability) {

                    mostDanceableSong = song\_name;

                    maxDanceability = danceValue;

                }

            }

            // emit output only if the most danceable song has been found

            // calculate average only if there are valid values

            float avg = (total > 0) ? sum / total : 0;

            // build the ouputvalue for average and max danceability

            outputValue.set(new Text(mostDanceableSong + ": " + maxDanceability + ", avg: " + avg));

            // output (country-date pair, outputValue)

            context.write(key, outputValue);

        }

    }

}

**Our Driver java class:**

package gr.aueb.panagiotisl.mapreduce.wordcount;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Driver {

    public static  void main(String[] args) throws Exception {

        System.setProperty("hadoop.home.dir", "/");

        // instantiate a configuration

        Configuration configuration = new Configuration();

        // instantiate a job

        Job job = Job.getInstance(configuration, "Map-reduce Queries");

        // set job parameters

        job.setJarByClass(mapreduce\_queries.class);

        job.setMapperClass(mapreduce\_queries.CountMapper.class);

        job.setReducerClass(mapreduce\_queries.CountReducer.class);

        job.setOutputKeyClass(Text.class);

        job.setOutputValueClass(Text.class);

        // set io paths

        FileInputFormat.addInputPath(job, new Path("/user/hdfs/input/universal\_top\_spotify\_songs.csv"));

        FileOutputFormat.setOutputPath(job, new Path("/user/hdfs/output/"));

        System.exit(job.waitForCompletion(true)? 0 : 1);

    }

}