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

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

OUTPUT:

```
--2024-02-01 17:13:55--  https://www.gutenberg.org/cache/epub/72849/pg72849.txt
Resolving www.gutenberg.org (www.gutenberg.org)... 152.19.134.47, 2610:28:3090:3000:0:bad:cafe:47
Connecting to www.gutenberg.org (www.gutenberg.org)|152.19.134.47|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 591620 (578K) [text/plain]
Saving to: 'private_chivarly.txt'

private_chivarly.txt          100%[=================================================>] 577.75K  1005KB/s    in 0.6s

2024-02-01 17:13:56 (1005 KB/s) - 'private_chivarly.txt' saved [591620/591620]
```

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:

```
// set io paths
FileInputFormat.addInputPath(job, new Path("/user/hdfs/input/private_chivarly.txt"));
```

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:

```
2024-02-01 20:36:13,037 INFO client.RMProxy: Connecting to ResourceManager at resourcemanager/172.18.0.4:8032
2024-02-01 20:36:13,226 INFO client.AHSProxy: Connecting to Application History server at historyserver/172.18.0.2:102002024-02-01 20:36:13,475 WARN mapreduce.JobResourceUploader:
nt the Tool interface and execute your application with ToolRunner to remedy this.
2024-02-01 20:36:13,508 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/root/.staging/job_1706808463730_0015
2024-02-01 20:36:13,690 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-02-01 20:36:14,064 INFO input.FileInputFormat: Total input files to process : 1
2024-02-01 20:36:14,112 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-02-01 20:36:14,152 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-02-01 20:36:14,166 INFO mapreduce.JobSubmitter: number of splits:1
2024-02-01 20:36:14,409 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-02-01 20:36:14,451 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1706808463730_0015
2024-02-01 20:36:14,452 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-02-01 20:36:14,705 INFO conf.Configuration: resource-types.xml not found
2024-02-01 20:36:14,706 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-02-01 20:36:15,265 INFO impl.YarnClientImpl: Submitted application application_1706808463730_0015
2024-02-01 20:36:15,348 INFO mapreduce.Job: The url to track the job: http://resourcemanager:8088/proxy/application_1706808463730_0015/
2024-02-01 20:36:15,350 INFO mapreduce.Job: Running job: job_1706808463730_0015
2024-02-01 20:36:23,514 INFO mapreduce.Job: Job job_1706808463730_0015 running in uber mode : false
2024-02-01 20:36:23,515 INFO mapreduce.Job:  map 0% reduce 0%
2024-02-01 20:36:31,714 INFO mapreduce.Job:  map 100% reduce 0%
2024-02-01 20:36:37,784 INFO mapreduce.Job:  map 100% reduce 100%
2024-02-01 20:36:38,804 INFO mapreduce.Job: Job job_1706808463730_0015 completed successfully
2024-02-01 20:36:38,960 INFO mapreduce.Job: Counters: 54
```

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

<u>INPUT:</u>

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

<u>OUPUT:</u>

Word counts for the top 20 words.

```
        3083
#72849] 1
$       1
$1.00.  1
$1.00;  3
$1.25.  4
$1.25.= 2
$1.50.  13
$1.50.= 2
$5,000) 1
&       3
($1     1
(801)   1
(Ill.)  1
(This   1
(_Nearly        1
(a)     1
(and    2
(any    1
(b)     1
```

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);
    }
}
```