Name: Kunal Patil

Roll No:2183117

Enroll No: MITU18BTCS0187

Class: CSE LY IS 1

Experiment No. 5: Execute Map Reduce program for the weather forecasting data and word count

```
import java.io.IOException; import
java.util.StringTokenizer;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper; import
org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.output.MultipleOutputs; import
org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
```

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


public class CalculateMaxAndMinTemeratureWithTime {
public static String calOutputName = "California"; public
static String nyOutputName = "Newyork"; public static
String njOutputName = "Newjersy"; public static String
ausOutputName = "Austin"; public static String
bosOutputName = "Boston"; public static String
balOutputName ="Baltimore";


public static class WhetherForcastMapperextends
  Mapper<Object, Text, Text, Text> {


 public void map(Object keyOffset, Text dayReport, Context con)
   throws IOException, InterruptedException { StringTokenizer
  strTokens = new StringTokenizer(
    dayReport.toString(), "\t");
  int counter = 0;
Float currnetTemp = null;
Float   minTemp   =   Float.MAX_VALUE;
Float   maxTemp   =   Float.MIN_VALUE;
String date = null;
String   currentTime   =   null;   String
minTempANDTime = null;
String maxTempANDTime = null;


while (strTokens.hasMoreElements()) {
 if (counter == 0) {
  date = strTokens.nextToken();
 } else {
  if (counter % 2 == 1) {
```

```
    currentTime = strTokens.nextToken();
   } else {
    currnetTemp = Float.parseFloat(strTokens.nextToken());
    if (minTemp > currnetTemp) {
     minTemp = currnetTemp;
     minTempANDTime = minTemp + "AND" + currentTime;
    }
    if (maxTemp < currnetTemp) {
     maxTemp = currnetTemp;
     maxTempANDTime = maxTemp + "AND" + currentTime;
    }
   }
  }
  counter++;
 }
 // Write to context - MinTemp, MaxTemp and corresponding time Text temp
 = new Text();
 temp.set(maxTempANDTime);
 Text dateText = new Text();
 dateText.set(date);
 try {
  con.write(dateText, temp);
 } catch (Exception e) {
   e.printStackTrace();
 }

 temp.set(minTempANDTime);
 dateText.set(date);
 con.write(dateText, temp);

 }
}
```

```
public static class WhetherForcastReducer extends
Reducer<Text, Text, Text, Text> { MultipleOutputs<Text, Text>
mos;

public void setup(Context context) {
 mos = new MultipleOutputs<Text, Text>(context);
}

public void reduce(Text key, Iterable<Text> values,Context context)
  throws IOException, InterruptedException {
 int counter = 0;
 String reducerInputStr[] = null; String
 f1Time = "";
 String f2Time = ""; String f1
 = "", f2 = ""; Text result =
 new Text();
 for (Text value : values) {

  if (counter == 0) {
   reducerInputStr = value.toString().split("AND"); f1 =
   reducerInputStr[0];
   f1Time = reducerInputStr[1];
 else {
  reducerInputStr = value.toString().split("AND"); f2 =
  reducerInputStr[0];
  f2Time = reducerInputStr[1];
 }

 counter = counter + 1;
}
```

```
if (Float.parseFloat(f1) > Float.parseFloat(f2)) {

 result = new Text("Time: " + f2Time + " MinTemp: " + f2 + "\t"
   + "Time: " + f1Time + " MaxTemp: " + f1);
} else {

 result = new Text("Time: " + f1Time + " MinTemp: " + f1 + "\t"
   + "Time: " + f2Time + " MaxTemp: " + f2);
}
String fileName = "";
if (key.toString().substring(0, 2).equals("CA")) {
 fileName = CalculateMaxAndMinTemeratureTime.calOutputName;
} else if (key.toString().substring(0, 2).equals("NY")) {
 fileName = CalculateMaxAndMinTemeratureTime.nyOutputName;
} else if (key.toString().substring(0, 2).equals("NJ")) {
 fileName = CalculateMaxAndMinTemeratureTime.njOutputName;
} else if (key.toString().substring(0, 3).equals("AUS")) {
 fileName = CalculateMaxAndMinTemeratureTime.ausOutputName;
} else if (key.toString().substring(0, 3).equals("BOS")) {
 fileName = CalculateMaxAndMinTemeratureTime.bosOutputName;
} else if (key.toString().substring(0, 3).equals("BAL")) {
 fileName = CalculateMaxAndMinTemeratureTime.balOutputName;
}
```

```
} else if (key.toString().substring(0, 2).equals("NJ")) {
 fileName = CalculateMaxAndMinTemeratureTime.njOutputName;
} else if (key.toString().substring(0, 3).equals("AUS")) {
 fileName = CalculateMaxAndMinTemeratureTime.ausOutputName;
} else if (key.toString().substring(0, 3).equals("BOS")) {
 fileName = CalculateMaxAndMinTemeratureTime.bosOutputName;
} else if (key.toString().substring(0, 3).equals("BAL")) {
 fileName = CalculateMaxAndMinTemeratureTime.balOutputName;
}
String strArr[] = key.toString().split("_"); key.set(strArr[1]); //Key
is date value mos.write(fileName, key, result);
}

@Override
public void cleanup(Context context) throws IOException,
  InterruptedException {
 mos.close();
}
```

```
public static void main(String[] args) throws IOException,
 ClassNotFoundException, InterruptedException { Configuration conf
 = new Configuration();
 Job job = Job.getInstance(conf, "Wheather Statistics of USA");
 job.setJarByClass(CalculateMaxAndMinTemeratureWithTime.class);

 job.setMapperClass(WhetherForcastMapper.class);
 job.setReducerClass(WhetherForcastReducer.class);

 job.setMapOutputKeyClass(Text.class);
 job.setMapOutputValueClass(Text.class);

 job.setOutputKeyClass(Text.class);
 job.setOutputValueClass(Text.class);

 MultipleOutputs.addNamedOutput(job, calOutputName, TextOutputFormat.class,
   Text.class, Text.class);
 MultipleOutputs.addNamedOutput(job, nyOutputName, TextOutputFormat.class,
   Text.class, Text.class);
 MultipleOutputs.addNamedOutput(job, njOutputName, TextOutputFormat.class,
   Text.class, Text.class);

MultipleOutputs.addNamedOutput(job, njOutputName, TextOutputFormat.class,
  Text.class, Text.class);
MultipleOutputs.addNamedOutput(job, bosOutputName, TextOutputFormat.class,
  Text.class, Text.class);
MultipleOutputs.addNamedOutput(job, ausOutputName, TextOutputFormat.class,
  Text.class, Text.class);
MultipleOutputs.addNamedOutput(job, balOutputName, TextOutputFormat.class,
  Text.class, Text.class);

// FileInputFormat.addInputPath(job, new Path(args[0]));
```

```
// FileOutputFormat.setOutputPath(job, new Path(args[1])); Path
pathInput = new Path(
  "hdfs://192.168.213.133:54310/weatherInputData/input_temp.txt"); Path
pathOutputDir = new Path(
  "hdfs://192.168.213.133:54310/user/hduser1/testfs/output_mapred3");
FileInputFormat.addInputPath(job, pathInput);
FileOutputFormat.setOutputPath(job, pathOutputDir);


try {
 System.exit(job.waitForCompletion(true) ? 0 : 1);
} catch (Exception e) {
 // TODO Auto-generated catch block e.printStackTrace();
}
}
}
```

Execution:

Copy a input file form local file system to HDFS

```
hdoop@benoi:~/hadoop-3.2.1/bin$ ./hadoop fs -put
/home/zytham/input_temp.txt /weatherInputData/
```

Give write permission to all user for creating output directory

```
hdoop@benoi:~/hadoop-3.2.1/bin$ ./hadoop fs -chmod -R 777
/user/hduser1/testfs/
```

Output:

```
hdoop@benoi:~/hadoop-3.2.1/bin$ ./hadoop fs -ls
/user/hduser1/testfs/output_mapred3 Found 8
items
-rw-r--r--   3 zytham supergroup          438 2020-12-11 19:21
/user/hduser1/testfs/output_mapred3/Austin-r-00000
-rw-r--r--   3 zytham supergroup          219 2020-12-11 19:21
/user/hduser1/testfs/output_mapred3/Baltimore-r-00000
-rw-r--r--   3 zytham supergroup          219 2020-12-11 19:21
/user/hduser1/testfs/output_mapred3/Boston-r-00000
-rw-r--r--   3 zytham supergroup          511 2020-12-11 19:21
/user/hduser1/testfs/output_mapred3/California-r-00000
-rw-r--r--   3 zytham supergroup          146 2020-12-11 19:21
/user/hduser1/testfs/output_mapred3/Newjersy-r-00000
-rw-r--r--   3 zytham supergroup          219 2020-12-11 19:21
/user/hduser1/testfs/output_mapred3/Newyork-r-00000
-rw-r--r--   3 zytham supergroup            0 2020-12-11 19:21
/user/hduser1/testfs/output_mapred3/_SUCCESS
-rw-r--r--   3 zytham supergroup            0 2020-12-11 19:21
/user/hduser1/testfs/output_mapred3/part-r-00000
```

Open one of the file and verify expected output schema, execute following command for the same.

```
hdoop@benoi:~/hadoop-3.2.1/bin$ ./hadoop fs -cat
/user/hduser1/testfs/output_mapred3/Austin-r-00000
25-Jan-2020 Time: 12:34:542 MinTemp: -22.3 Time: 05:12:345 MaxTemp:
35.7
```

```
hdoop@benoi:~/hadoop-3.2.1/bin$ ./hadoop fs -cat
/user/hduser1/testfs/output_mapred3/Austin-r-00000
25-Jan-2020 Time: 12:34:542 MinTemp: -22.3 Time: 05:12:345 MaxTemp:
35.7
26-Jan-2020 Time: 22:00:093 MinTemp: -27.0 Time: 05:12:345 MaxTemp:55.7
27-Jan-2020 Time: 02:34:542 MinTemp: -22.3 Time: 05:12:345 MaxTemp:55.7
29-Jan-2020 Time: 14:00:093 MinTemp: -17.0 Time: 02:34:542 MaxTemp:62.9
30-Jan-2020 Time: 22:00:093 MinTemp: -27.0 Time: 05:12:345 MaxTemp:49.2
31-Jan-2020 Time: 14:00:093 MinTemp: -17.0 Time: 03:12:187 MaxTemp:56.0
```