**DSBDA GROUP B - Practical 3**

**WeatherDatasetMapper.java :**

package Weather;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.\*;

public class WeatherDatasetMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {

private static final int MISSING = 9999;

public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {

String line = value.toString();

String year\_temp = line.substring(15, 19) + "\_temp";

String year\_dew = line.substring(15, 19) + "\_dew";

String year\_wind = line.substring(15, 19) + "\_wind";

int airTemperature, dewpoint, windspeed;

if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs

airTemperature = Integer.parseInt(line.substring(88, 92));

} else {

airTemperature = Integer.parseInt(line.substring(87, 92));

}

if (line.charAt(93) == '+') {

dewpoint = Integer.parseInt(line.substring(94, 98));

} else {

dewpoint = Integer.parseInt(line.substring(93, 98));

}

windspeed = Integer.parseInt(line.substring(60, 63));

String quality = line.substring(92, 93);

if (airTemperature != MISSING && quality.matches("[01459]")) {

output.collect(new Text(year\_temp), new IntWritable(airTemperature));

output.collect(new Text(year\_dew), new IntWritable(dewpoint));

output.collect(new Text(year\_wind), new IntWritable(windspeed));

}

}

}

**WeatherDatasetReducer.java :**

package Weather;

import java.io.IOException;

import java.util.\*;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.\*;

public class WeatherDatasetReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {

public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException {

int tempValue = 0, dewValue = 0, windSpeedValue = 0, countTemp = 0, countDew = 0, countWindSpeed = 0, tempAvg, dewAvg, windSpeedAvg;

String key\_str = key.toString();

if (key\_str.contains("\_temp")) {

while (values.hasNext()) {

// replace type of value with the actual type of our value

IntWritable value = (IntWritable) values.next();

tempValue += value.get();

countTemp += 1;

}

tempAvg = tempValue / countTemp;

output.collect(key, new IntWritable(tempAvg));

} else if (key\_str.contains("\_dew")) {

while (values.hasNext()) {

// replace type of value with the actual type of our value

IntWritable value = (IntWritable) values.next();

dewValue += value.get();

countDew += 1;

}

dewAvg = dewValue / countDew;

output.collect(key, new IntWritable(dewAvg));

} else {

while (values.hasNext()) {

// replace type of value with the actual type of our value

IntWritable value = (IntWritable) values.next();

windSpeedValue += value.get();

countWindSpeed += 1;

}

windSpeedAvg = windSpeedValue / countWindSpeed;

output.collect(key, new IntWritable(windSpeedValue));

}

}

}

**WeatherDatasetDriver.java :**

package Weather;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapred.\*;

public class WeatherDatasetDriver {

public static void main(String[] args) {

JobClient my\_client = new JobClient();

// Create a configuration object for the job

JobConf job\_conf = new JobConf(WeatherDatasetDriver.class);

// Set a name of the Job

job\_conf.setJobName("WeatherDataset");

// Specify data type of output key and value

job\_conf.setOutputKeyClass(Text.class);

job\_conf.setOutputValueClass(IntWritable.class);

// Specify names of Mapper and Reducer Class

job\_conf.setMapperClass(Weather.WeatherDatasetMapper.class);

job\_conf.setReducerClass(Weather.WeatherDatasetReducer.class);

// Specify formats of the data type of Input and output

job\_conf.setInputFormat(TextInputFormat.class);

job\_conf.setOutputFormat(TextOutputFormat.class);

// Set input and output directories using command line arguments,

//arg[0] = name of input directory on HDFS, and arg[1] = name of output directory to be created to store the output file.

FileInputFormat.setInputPaths(job\_conf, new Path(args[0]));

FileOutputFormat.setOutputPath(job\_conf, new Path(args[1]));

my\_client.setConf(job\_conf);

try {

// Run the job

JobClient.runJob(job\_conf);

} catch (Exception e) {

e.printStackTrace();

}

}

}



