
Group B

Assignment No: 2

Title of the Assignment: Locate dataset (e.g., sample_weather.txt) for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed.

Objective of the Assignment: Students should be able to install and Hadoop MapReduce framework on local-standalone set-up and they should able to Write a code in JAVA for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed.

Prerequisite:

1. **JAVA-Java JDK (installed)**
2. **HADOOP-Hadoop package (Downloaded, Installed, Configured, Tested)**

Contents for Theory:

1. **Hadoop MapReduce framework Procedures to be Follow**
2. **Installation of Hadoop on Windows**
3. **Installation of Hadoop on Linux/Unix/Fedora.etc**
4. **Running Java Program for average for temperature, dew point and wind speed Mapreduce Example using weather dataset.**
5. **Java Code for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed.**

---

1. Hadoop MapReduce framework Procedures to be Follow: -

1. Install Java
2. install Hadoop
3. Configure hadoop
4. Test hadoop installation
5. Create mapreduce program
6. Input file to mapreduce
7. Display the output

A. **Set up Hadoop:** Install and configure Hadoop on your system or use a Hadoop cluster.

B. **Prepare the dataset:** Ensure your weather dataset (e.g., "sample_weather.txt") is available and accessible to Hadoop. You may need to upload the dataset to the Hadoop Distributed File System (HDFS) or make it available through other means.

C. **Write a MapReduce program:** Create a Java program that implements the MapReduce paradigm to process the weather data and calculate the average values.

D. **Map function:** In the map function, you'll parse each input record (line) from the dataset and extract the temperature, dew point, and wind speed values. Emit key-value pairs with the key set to a constant value (to aggregate all values) and the values set to the extracted temperature, dew point, and wind speed.

E. **Reduce function:** In the reduce function, you'll receive the key-value pairs emitted by the map function. Iterate over the values and calculate the sums of temperature, dew point, and wind speed, as well as the count of observations.

F. **Output:** Emit a single key-value pair with the key set to a constant value (to ensure all values are reduced together) and the value set to a string representation of the calculated averages.

G. **Submit the MapReduce job:** Use the Hadoop command-line interface (CLI) or a job submission framework (e.g., Apache Oozie) to submit the MapReduce job to the Hadoop cluster.

H. **Retrieve the results:** Once the MapReduce job completes, you can retrieve the output file containing the calculated averages from the Hadoop cluster.

2. How to Install Single Node Cluster Hadoop on Windows?

Step 1: Verify the Java installed
Step 2: Installing Hadoop
Step 3: Hadoop Configuration:
Step 4: Testing Hadoop Installation
Step 5: Create program for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed.

3. Installation of Hadoop on Linux/Unix/Fedora.etc

Step 1: Verify the Java installed
Step 2: Installing Hadoop
Step 3: Hadoop Configuration:
Step 4: Testing Hadoop Installation

4. Hadoop – Running Mapreduce Example

Step 1: Store the dataset file, such as "sample_weather.txt,"

Store the dataset file, such as "sample_weather.txt," in the Hadoop Distributed File System (HDFS) using the hadoop fs command.

For example:

```
hadoop fs -put /path/to/sample_weather.txt /input/
```

This command will copy the sample_weather.txt file from the local file system to the /input/ directory in HDFS.

Step 2: Write a MapReduce program in Java

Write a MapReduce program in Java to read and process the data in the sample_weather.txt file, and calculate the average temperature, dew point, and wind speed.

Here is an example code for the mapper and reducer classes:

i. WeatherDataMapper.java

ii. WeatherDataReducer.java

Step 3: Write a MapReduce program in Java for Hadoop job configuration

Create a Hadoop job configuration and specify the input and output paths, as well as the mapper and reducer classes:

WeatherDataAnalyzer.java

Step 4: Compile the Java code and package it into a JAR file.

Step 5: Run the Hadoop job using the following command:

```
hadoop jar /path/to/WeatherDataAnalyzer.jar WeatherDataAnalyzer
```

This will execute the Hadoop job, which will read the input file from HDFS, process the data using the mapper and reducer classes, and write the output to the /output directory in HDFS.

Step 6: Output:

View the output using the hadoop fs -cat command:

```
hadoop fs -cat /output/part-r-00000
```

This will display the average temperature, dew point, and wind speed values in the Hadoop console.

5. Java Code for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed.

/* WeatherDataMapper.java*/

```
import java.io.IOException;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;

public class WeatherDataMapper extends Mapper<LongWritable, Text, Text,
DoubleWritable> {
    @Override
    public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        String line = value.toString();
        String[] data = line.split(",");
```

```
            double temperature = Double.parseDouble(data[2]);
            double dewPoint = Double.parseDouble(data[3]);
            double windSpeed = Double.parseDouble(data[4]);
            context.write(new Text("temperature"), new
    DoubleWritable(temperature));
            context.write(new Text("dewPoint"), new DoubleWritable(dewPoint));
            context.write(new Text("windSpeed"), new
    DoubleWritable(windSpeed));
        }
    }
```

/* WeatherDataReducer.java*/

```
    import java.io.IOException;
    import org.apache.hadoop.io.*;
    import org.apache.hadoop.mapreduce.*;

    public class WeatherDataReducer extends Reducer<Text, DoubleWritable, Text,
    DoubleWritable> {
        @Override
        public void reduce(Text key, Iterable<DoubleWritable> values, Context
    context) throws IOException, InterruptedException {
            double sum = 0.0;
            int count = 0;
            for (DoubleWritable value : values) {
                sum += value.get();
                count++;
            }
            double average = sum / count;
            context.write(key, new DoubleWritable(average));
        }
    }
```

/* WeatherDataAnalyzer.java*/

```
    import org.apache.hadoop.conf.Configuration;
    import org.apache.hadoop.fs.Path;
    import org.apache.hadoop.io.*;
    import org.apache.hadoop.mapreduce.*;

    public class WeatherDataAnalyzer {
        public static void main(String[] args) throws Exception {
            Configuration conf = new Configuration();
            Job job = Job.getInstance(conf, "Weather Data Analyzer");

            job.setJarByClass(WeatherDataAnalyzer.class);
            job.setMapperClass(WeatherDataMapper.class);
            job.setReducerClass(WeatherDataReducer.class);

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

            FileInputFormat.addInputPath(job, new
    Path("/input/sample_weather.txt"));
            FileOutputFormat.setOutputPath(job, new Path("/output"));

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

Implement a Hadoop MapReduce program:

- Create a Java class that defines the Mapper and Reducer functions for the MapReduce job. Here's an example implementation:

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WeatherDataAnalyzer {
    public static class WeatherDataMapper extends Mapper<LongWritable,
Text, Text, DoubleWritable> {
        private Text keyOut = new Text();
        private DoubleWritable valueOut = new DoubleWritable();

        public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
            String line = value.toString();
            String[] data = line.split(",");

            // Extract temperature, dew point, and wind speed
            double temperature = Double.parseDouble(data[2]);
            double dewPoint = Double.parseDouble(data[3]);
            double windSpeed = Double.parseDouble(data[4]);

            // Emit key-value pairs
            keyOut.set("average_temperature");
            valueOut.set(temperature);
            context.write(keyOut, valueOut);

            keyOut.set("average_dew_point");
            valueOut.set(dewPoint);
            context.write(keyOut, valueOut);

            keyOut.set("average_wind_speed");
            valueOut.set(windSpeed);
            context.write(keyOut, valueOut);
        }
    }

    public static class WeatherDataReducer extends Reducer<Text,
DoubleWritable, Text, DoubleWritable> {
        private DoubleWritable result = new DoubleWritable();

        public void reduce(Text key, Iterable<DoubleWritable> values,
Context context)
                throws IOException, InterruptedException {
            double sum = 0;
            int count = 0;

            // Calculate the sum and count for each key
            for (DoubleWritable value : values) {
                sum += value.get();
                count++;
```

```
                }

                // Calculate the average
                double average = sum / count;

                result.set(average);
                context.write(key, result);
            }
        }

        public static void main(String[] args) throws Exception {
            Configuration conf = new Configuration();
            Job job = Job.getInstance(conf, "Weather Data Analysis");

            job.setJarByClass(WeatherDataAnalyzer.class);
            job.setMapperClass(WeatherDataMapper.class);
            job.setReducerClass(WeatherDataReducer.class);

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

            FileInputFormat.addInputPath(job, new Path(args[0]));
            FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

Conclusion- In this way we have Located dataset (e.g., sample_weather.txt) for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed using mapreduce java program.