EX.NO:11 REG.NO:210701294

DATE:

HADOOP IMPLEMENT THE MAX TEMPERATURE MAPREDUCE PROGRAM TO

IDENTIFY THE YEAR WISE MAXIMUM TEMPERATURE FROMSENSOR DATA

AIM

To implement the Max temperature MapReduce program to identify the year-wise maximum temperature from the sensor data.

Description

Sensors senses weather data in big text format containing station ID, year, date, time, temperature, quality etc. from each sensor and store it in a single line. Suppose thousands of data sensors are there, then we have thousands of records with no particular order. We require only a year and maximum temperature of particular quality in that year.

For example:

Input string from sensor:

0029029070999991902010720004+64333+023450

FM-12+

000599999V0202501N027819999999N0000001N9-00331+

99999098351ADDGF102991999999999999999999

Here: 1902 is year

0033 is temperature

1 is measurement quality (Range between 0 or 1 or 4 or 5 or 9)

Here each mapper takes the input key as "byte offset of line" and value as "one weather sensor read i.e one line". and parse each line and produce an intermediate key "year" and intermediate value as "temperature of certain measurement qualities" for that year.

The combiner will form set values of temperature. Year and set of values of temperatures is

given as input <key, value> to reducer and Reducer will produce year and maximum temperature for that year from the set of temperature values.

PROGRAM

*/

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
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;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
importorg.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
importorg.apache.hadoop.mapreduce.Reducer;
//Mapper class
class MaxTemperatureMapper
extends Mapper<LongWritable, Text, Text, IntWritable> { private static final int MISSING
= 9999;
@Override
public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
String line = value.toString(); String year = line.substring(15, 19); int airTemperature;
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));
}
String quality = line.substring(92, 93);
if (airTemperature != MISSING && quality.matches("[01459]")) { context.write(new
Text(year), new IntWritable(airTemperature));
}
}
}
//Reducer class
class MaxTemperatureReducer
extends Reducer<Text, IntWritable, Text, IntWritable> {
@Override
public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {
int maxValue = Integer.MIN_VALUE; for (IntWritable value : values) {
maxValue = Math.max(maxValue, value.get());
}
context.write(key, new IntWritable(maxValue));
}
}
//Driver Class
public class MaxTemperature {
public static void main(String[] args) throws Exception { if (args.length != 2) {
System.err.println("Usage: MaxTemperature <input path=""> <output path>"); System.exit(-
1);
```

```
}
Job job = Job.getInstance(new Configuration()); job.setJarByClass(MaxTemperature.class);
job.setJobName("Max temperature");
FileInputFormat.addInputPath(job, new Path(args[0])); FileOutputFormat.setOutputPath(job,
new Path(args[1]));
job.setMapperClass(MaxTemperatureMapper.class);
job.setReducerClass(MaxTemperatureReducer.class);
job.setOutputKeyClass(Text.class); job.setOutputValueClass(IntWritable.class);
job.submit();
}
}
```

OUTPUT:

```
Input for String : 0029029070999991902010720004+64333+023450FM-12+
000599999V0202501N027819999999N0000001N9-00331+
99999098351ADDGF102991999999999999999999
```

RESULT:

Output Text contain year and maximum temperature in that year as 1902 33.