## Ex.No.3 Implement a MR program that process a weather dataset

#### Aim

To implement a MR program that process a weather dataset.

#### Pre-lab Discussion

**Inputs and Outputs (Java Perspective)**

The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework. Input and Output types of a **MapReduce job** − (Input) <k1, v1> → map → <k2, v2> → reduce → <k3, v3>(Output).

|  |  |  |
| --- | --- | --- |
|  | **Input** | **Output** |
| **Map** | <k1, v1> | list (<k2, v2>) |
| **Reduce** | <k2, list(v2)> | list (<k3, v3>) |

#### Terminology

* **PayLoad** − Applications implement the Map and the Reduce functions, and form the core of the job.
* **Mapper** − Mapper maps the input key/value pairs to a set of intermediate key/value pair.
* **NamedNode** − Node that manages the Hadoop Distributed File System (HDFS).
* **DataNode** − Node where data is presented in advance before any processing takes place.
* **MasterNode** − Node where JobTracker runs and which accepts job requests from clients.
* **SlaveNode** − Node where Map and Reduce program runs.
* **JobTracker** − Schedules jobs and tracks the assign jobs to Task tracker.
* **Task Tracker** − Tracks the task and reports status to JobTracker.
* **Job** − A program is an execution of a Mapper and Reducer across a dataset.
* **Task** − An execution of a Mapper or a Reducer on a slice of data.
* **Task Attempt** − A particular instance of an attempt to execute a task on a SlaveNode.

#### Program AverageMapper.java

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

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

public class AverageMapper extends Mapper <LongWritable, Text, Text, IntWritable>

{

public static final int MISSING = 9999;

public void map(LongWritable key, Text value, Context context) throwsIOException, InterruptedException

{

String line = value.toString(); String year = line.substring(15,19); int temperature;

if (line.charAt(87)=='+')

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

temperature = Integer.parseInt(line.substring(87, 92)); String quality = line.substring(92, 93);

if(temperature != MISSING && quality.matches("[01459]")) context.write(new Text(year),new IntWritable(temperature));

}

}

#### AverageReducer.java

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

public class AverageReducer extends Reducer <Text, IntWritable,Text, IntWritable >

{

public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException

{

int max\_temp = 0; int count = 0;

for (IntWritable value : values)

{

max\_temp += value.get(); count+=1;

}

context.write(key, new IntWritable(max\_temp/count));

} }

**AverageDriver.java** import.org.apache.hadoop.io.\*; import org.apache.hadoop.fs.\*;

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

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class AverageDriver

{

public static void main (String[] args) throws Exception

{

if (args.length != 2)

{

System.err.println("Please Enter the input and output parameters"); System.exit(-1);

}

Job job = new Job(); job.setJarByClass(AverageDriver.class); job.setJobName("Max temperature");

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

job.setMapperClass(AverageMapper.class); job.setReducerClass(AverageReducer.class); job.setOutputKeyClass(Text.class); job.setOutputValueClass(IntWritable.class) System.exit(job.waitForCompletion(true)?0:1);

}

}

#### Result

Thus, the implementation of an MR program that process a weather dataset was executed successfully.