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.mapreduce.Mapper;

import org.apache.hadoop.util.StringUtils;

public class MyMap extends Mapper<LongWritable, Text, IntWritable, TempWritable> {

private IntWritable key = new IntWritable();

private TempWritable value = new TempWritable();

@Override

protected void map(LongWritable ikey, Text ivalue,

Mapper<LongWritable, Text, IntWritable, TempWritable>.Context context)

throws IOException, InterruptedException {

// TODO Auto-generated method stub

String line = ivalue.toString();

String[] tokens = StringUtils.split(line,'|');

if (tokens.length == 5){

int year = Integer.parseInt(tokens[1]);

float temp = Float.parseFloat(tokens[4]);

String station\_no = tokens[0];

key.set(year);

value.set(temp,station\_no);

context.write(key, value);

}

}

}

=================================================================================

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class MyReduce extends Reducer<IntWritable, TempWritable, IntWritable, Text> {

private Text value = new Text();

@Override

protected void reduce(IntWritable key, Iterable<TempWritable> values,

Reducer<IntWritable, TempWritable, IntWritable, Text>.Context context)

throws IOException, InterruptedException {

// TODO Auto-generated method stub

Iterator<TempWritable> iterator = values.iterator();

float max\_Temp=0;

String maxTempStation = null;

if(iterator.hasNext()){

TempWritable temp = iterator.next();

max\_Temp = temp.getTemp();

maxTempStation = temp.getStationNo();

}

while(iterator.hasNext()){

TempWritable temp = iterator.next();

if(temp.getTemp()>max\_Temp){

max\_Temp = temp.getTemp();

maxTempStation = temp.getStationNo();

}

}

value.set(max\_Temp+" "+maxTempStation);

context.write(key, value);

}

}

=================================================================================

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.Writable;

public class TempWritable implements Writable {

private Text station\_No;

private FloatWritable temp;

public TempWritable() {

temp = new FloatWritable();

station\_No = new Text();

}

public void set(float tem, String stn\_no) {

temp.set(tem);

station\_No.set(stn\_no);

}

@Override

public void readFields(DataInput input) throws IOException {

temp.readFields(input);

station\_No.readFields(input);

}

public float getTemp() {

return temp.get();

}

public String getStationNo() {

return station\_No.toString();

}

@Override

public void write(DataOutput output) throws IOException {

temp.write(output);

station\_No.write(output);

}

}

=================================================================================

import java.io.IOException;

import org.apache.hadoop.fs.FsConstants;

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.input.TextInputFormat;

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

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

public class Driver {

public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {

@SuppressWarnings("deprecation")

Job job = new Job();

job.setJarByClass(Driver.class);

job.setMapOutputKeyClass(IntWritable.class);

job.setMapOutputValueClass(TempWritable.class);

job.setOutputKeyClass(IntWritable.class);

job.setOutputValueClass(Text.class);

job.setMapperClass(MyMap.class);

job.setReducerClass(MyReduce.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

Path outputDir = new Path( args[1] );

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

FileOutputFormat.setOutputPath(job, outputDir);

FileSystem fs = FileSystem.get(new Configuration());

//Make sure fs related classes are imported--

//import org.apache.hadoop.fs.FileSystem;

fs.delete(outputDir, true);

if (!job.waitForCompletion(true))

return;

;

}

}

===================