Assignment 13

Program :

package com.org.vasanth.weather;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.FileInputFormat;

import org.apache.hadoop.mapred.FileOutputFormat;

import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.mapred.KeyValueTextInputFormat;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

/\*\*

\*

\*

\* This is an Hadoop Map/Reduce application for Working on weather data It reads

\* the text input files, breaks each line into stations weather data and finds

\* average for temperature , dew point , wind speed. The output is a locally

\* sorted list of stations and its 12 attribute vector of average temp , dew ,

\* wind speed of 4 sections for each month.

\*

\* To run: bin/hadoop jar target/weather-1.0.jar [-m <i>maps</i>] [-r

\* <i>reduces</i>] <i>in-dir for job 1</i> <i>out-dir for job 1</i> <i>out-dir

\* for job 2</i>

\*/

public class Weather extends Configured implements Tool {

final long DEFAULT\_SPLIT\_SIZE = 128 \* 1024 \* 1024;

/\*\*

\* Map Class for Job 1

\*

\* For each line of input, emits key value pair with

\* station\_yearmonth\_sectionno as key and 3 attribute vector with

\* temperature , dew point , wind speed as value.Map method will strip the

\* day and hour from field and replace it with section no (

\* <b>station\_yearmonth\_sectionno</b>, <b><temperature,dew point , wind

\* speed></b>).

\*/

public static class MapClass extends MapReduceBase

implements Mapper<LongWritable, Text, Text, Text> {

private Text word = new Text();

private Text values = new Text();

public void map(LongWritable key, Text value,

OutputCollector<Text, Text> output,

Reporter reporter) throws IOException {

String line = value.toString();

StringTokenizer itr = new StringTokenizer(line);

int counter = 0;

String key\_out = null;

String value\_str = null;

boolean skip = false;

loop:while (itr.hasMoreTokens() && counter<13) {

String str = itr.nextToken();

switch (counter) {

case 0:

key\_out = str;

if(str.contains("STN")){//Ignoring rows where station id is all 9

skip = true;

break loop;

}else{

break;

}

case 2:

int hour = Integer.valueOf(str.substring(str.lastIndexOf("\_")+1, str.length()));

str = str.substring(4,str.lastIndexOf("\_")-2);

/\*if(hour<=5){

str = str.concat("\_section4");

}else if(hour>5 && hour<=11){

str = str.concat("\_section1");

}else if(hour>11 && hour<=17){

str = str.concat("\_section2");

}else if(hour>17 && hour<=23){

str = str.concat("\_section3");

}\*/

if(hour>4 && hour<=10){

str = str.concat("\_section1");

}else if(hour>10 && hour<=16){

str = str.concat("\_section2");

}else if(hour>16 && hour<=22){

str = str.concat("\_section3");

}else{

str = str.concat("\_section4");

}

key\_out = key\_out.concat("\_").concat(str);

break;

case 3://Temperature

if(str.equals("9999.9")){//Ignoring rows where temperature is all 9

skip = true;

break loop;

}else{

value\_str = str.concat(" ");

break;

}

case 4://Dew point

if(str.equals("9999.9")){//Ignoring rows where dew point is all 9

skip = true;

break loop;

}else{

value\_str = value\_str.concat(str).concat(" ");

break;

}

case 12://Wind speed

if(str.equals("999.9")){//Ignoring rows where wind speed is all 9

skip = true;

break loop;

}else{

value\_str = value\_str.concat(str).concat(" ");

break;

}

default:

break;

}

counter++;

}

if(!skip){

word.set(key\_out);

values.set(value\_str);

output.collect(word, values);

}

}

}

/\*\*

\* Map Class for Job 2

\*

\* For each input, emits key value pair with station\_yearmonth as key and 3

\* attribute vector with temperature , dew point , wind speed as value by

\* stripping the section no from key and adding section no into vector value

\* ( <b>station\_yearmonth</b>, <b><temperature,dew point , wind speed></b>).

\*/

public static class MapClassForJob2 extends MapReduceBase

implements Mapper<Text, Text, Text, Text> {

private Text key\_text = new Text();

private Text value\_text = new Text();

@Override

public void map(Text key, Text value,

OutputCollector<Text, Text> output, Reporter reporter) throws IOException {

String str = key.toString();

String station = str.substring(str.lastIndexOf("\_")+1, str.length());

str = str.substring(0,str.lastIndexOf("\_"));

key\_text.set(str);

StringTokenizer itr = new StringTokenizer(value.toString());

String str\_out = station.concat("<");

while (itr.hasMoreTokens()) {

String nextToken = itr.nextToken(" ");

str\_out = str\_out.concat(nextToken);

str\_out = ((itr.hasMoreTokens()) ? str\_out.concat(",") : str\_out.concat(">"));

}

value\_text.set(str\_out);

output.collect(key\_text,value\_text);

}

}

/\*\*

\* Reducer Class for Job 1

\*

\* A reducer class that just emits 3 attribute vector with average

\* temperature , dew point , wind speed for each of the section of the month

\* for each input

\*/

public static class Reduce extends MapReduceBase

implements Reducer<Text, Text, Text, Text> {

private Text value\_out\_text = new Text();

public void reduce(Text key, Iterator<Text> values,

OutputCollector<Text, Text> output, Reporter reporter) throws IOException {

double sum\_temp = 0;

double sum\_dew = 0;

double sum\_wind = 0;

int count = 0;

while (values.hasNext()) {

String str = values.next().toString();

StringTokenizer itr = new StringTokenizer(str);

int count\_vector = 0;

while (itr.hasMoreTokens()) {

String nextToken = itr.nextToken(" ");

if(count\_vector==0){

sum\_temp += Double.valueOf(nextToken);

}

if(count\_vector==1){

sum\_dew += Double.valueOf(nextToken);

}

if(count\_vector==2){

sum\_wind += Double.valueOf(nextToken);

}

count\_vector++;

}

count++;

}

double avg\_tmp = sum\_temp / count;

double avg\_dew = sum\_dew / count;

double avg\_wind = sum\_wind / count;

System.out.println(key.toString()+" count is "+count+" sum of temp is "+sum\_temp+" sum of dew is "+sum\_dew+" sum of wind is "+sum\_wind+"\n");

String value\_out = String.valueOf(avg\_tmp).concat(" ").concat(String.valueOf(avg\_dew)).concat(" ").concat(String.valueOf(avg\_wind));

value\_out\_text.set(value\_out);

output.collect(key, value\_out\_text);

}

}

/\*\*

\* Reducer Class for Job 2

\*

\* A reducer class that just emits 12 attribute vector with average

\* temperature , dew point , wind speed for all section of the month

\* for each input

\*/

public static class ReduceForJob2 extends MapReduceBase

implements Reducer<Text, Text, Text, Text> {

private Text value\_out\_text = new Text();

public void reduce(Text key, Iterator<Text> values,

OutputCollector<Text, Text> output, Reporter reporter) throws IOException {

String value\_out = "";

while (values.hasNext()) {

value\_out = value\_out.concat(values.next().toString()).concat(" ");

}

value\_out\_text.set(value\_out);

output.collect(key, value\_out\_text);

}

}

static int printUsage() {

System.out.println("weather [-m <maps>] [-r <reduces>] <job\_1 input> <job\_1 output> <job\_2 output>");

ToolRunner.printGenericCommandUsage(System.out);

return -1;

}

/\*\*

\* The main driver for weather map/reduce program.

\* Invoke this method to submit the map/reduce job.

\* @throws IOException When there is communication problems with the

\* job tracker.

\*/

public int run(String[] args) throws Exception {

Configuration config = getConf();

// We need to lower input block size by factor of two.

/\*config.setLong(

FileInputFormat.SPLIT\_MAXSIZE,

config.getLong(

FileInputFormat.SPLIT\_MAXSIZE, DEFAULT\_SPLIT\_SIZE) / 2);\*/

JobConf conf = new JobConf(config, Weather.class);

conf.setJobName("Weather Job1");

// the keys are words (strings)

conf.setOutputKeyClass(Text.class);

// the values are counts (ints)

conf.setOutputValueClass(Text.class);

conf.setMapOutputKeyClass(Text.class);

conf.setMapOutputValueClass(Text.class);

conf.setMapperClass(MapClass.class);

//conf.setCombinerClass(Combiner.class);

conf.setReducerClass(Reduce.class);

List<String> other\_args = new ArrayList<String>();

for(int i=0; i < args.length; ++i) {

try {

if ("-m".equals(args[i])) {

conf.setNumMapTasks(Integer.parseInt(args[++i]));

} else if ("-r".equals(args[i])) {

conf.setNumReduceTasks(Integer.parseInt(args[++i]));

} else {

other\_args.add(args[i]);

}

} catch (NumberFormatException except) {

System.out.println("ERROR: Integer expected instead of " + args[i]);

return printUsage();

} catch (ArrayIndexOutOfBoundsException except) {

System.out.println("ERROR: Required parameter missing from " +

args[i-1]);

return printUsage();

}

}

// Make sure there are exactly 2 parameters left.

/\*if (other\_args.size() != 3) {

System.out.println("ERROR: Wrong number of parameters: " +

other\_args.size() + " instead of 3.");

return printUsage();

}\*/

FileInputFormat.setInputPaths(conf, other\_args.get(0));

FileOutputFormat.setOutputPath(conf, new Path(other\_args.get(1)));

JobClient.runJob(conf);

JobConf conf2 = new JobConf(config, Weather.class);

conf2.setJobName("Weather Job 2");

// the keys are words (strings)

conf2.setOutputKeyClass(Text.class);

// the values are counts (ints)

conf2.setOutputValueClass(Text.class);

conf2.setInputFormat(KeyValueTextInputFormat.class);

conf2.setMapOutputKeyClass(Text.class);

conf2.setMapOutputValueClass(Text.class);

conf2.setMapperClass(MapClassForJob2.class);

//conf.setCombinerClass(Combiner.class);

conf2.setReducerClass(ReduceForJob2.class);

FileInputFormat.setInputPaths(conf2, new Path(other\_args.get(1)));

FileOutputFormat.setOutputPath(conf2, new Path(other\_args.get(2)));

JobClient.runJob(conf2);

return 0;

}

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

int res = ToolRunner.run(new Configuration(), new Weather(), args);

System.exit(res);

}

}

