**Advanced Database Management System Project**

**Scope:**

To analyze the Airlines dataset for period (1987 to 2008). The dataset has information about the flights departure and arrival times, place, delays if any across USA. We can analyze the reason for delays, type of delays for the flights in various time frames and destination.

With the data, available for over 2 decades, we can find the trend in flight delays in various locations.

The different kinds of flight delays captured are:

**Carrier Delay, Weather Delay, NAS Delay, Security Delay, Late Aircraft Delay**

* We can also find the most crowded location in terms of flights leaving and arriving (Flight traffic) and we can use the summarization patterns to identify the trend in maximum occurring delay over the years.
* Delays pertaining to the location
* Delays which occur more frequently in a year
* Longest distance travelled by the flights to reach the destination.

By analyzing the trend, we can arrive at the delays we need to concentrate to prevent. Determining the locations that need to be monitored for better traffic management.

**DataSet Link**

Air Transport Data

<http://stat-computing.org/dataexpo/2009/the-data.html>

<https://www.bts.gov/>

**Summary of the Analysis Performed**

* Analyzed the flight information form varying data for each year to get the distinct flights information from at Origin Airports
* There are five different types of Delay as discussed which causes flight delay at the departure. Using MR patterns analyzed which type of delay is causing most of the problems at Origin Airports.
* In addition to the five type of delays, the flight may get cancelled due to variety of reasons. Analyzed cancelled flights details at an airport with the reason stating the reason for the cancellation at the airport also binned the output monthwise.
* Analyzed the frequency of flights getting delayed at the airport.
* Using Distributed Cached derived the Individual Source, Destination and distance between them and airport information in addition to the airport code from another file.
* For the given Dataset calculated the top 10 most delayed airports.
* With the join patterns analyzed and derived airports information which have departed and arrived on time without delay.
* Using Custom Practitioner derived and segregated the airports having most delays based on the type /category of the delay.
* Using Bloom filter with distributed cache analyzed the airports which have are punctual and on time for given set of hot values.

**Source Codes**

1. package finalproject\_1;

import java.io.IOException;

import java.util.ArrayList;

import java.util.HashSet;

import java.util.List;

import java.util.Set;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

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 FinalProject\_1 {

public static class horizMap extends Mapper<Object, Text, Text, DestFlightNoPojo> {

private Text origin = new Text();

private Text passenger = new Text();

private DestFlightNoPojo custompojo = new DestFlightNoPojo();

@Override

protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {

String input = new String(value.toString());

String[] token = input.split(",");

if (!token[0].contains("Year")) {

origin.set(token[16]);

// passenger.set(token[9]);

custompojo.setDestination(token[17]);

custompojo.setFlightNo(Integer.parseInt(token[9]));

context.write(origin, custompojo);

}

}

}

public static class ReducerHorix extends Reducer<Text, DestFlightNoPojo, Text, Text> {

// private LongWritable result = new LongWritable();

private Text result = new Text();

// private Set<String> dist = new HashSet<String>();

private List<String> dist = new ArrayList<String>();

@Override

protected void reduce(Text key, Iterable<DestFlightNoPojo> values, Context context) throws IOException, InterruptedException {

String destination = "";

Integer flightNo = 0;

boolean flag = false;

StringBuilder sb = new StringBuilder();

/\*

for (Text val : values) {

dist.add(val.toString());

}

for (String a : dist){

sb.append(a + " ");

}

//result.set(sb.toString());\*/

for (DestFlightNoPojo val : values) {

destination = val.getDestination();

flightNo = val.getFlightNo();

//sb.append(destination+","+flightNo);

for (String s : dist) {

if (s.contains(destination)) {

s = s + ',' + flightNo;

flag = true;

}

}

if (!flag) {

dist.add(destination + ' ' + flightNo);

}

flag = false;

}

for (String s : dist) {

sb.append(s);

sb.append(" ");

}

result.set(sb.toString());

context.write(key, result);

dist.clear();

}

}

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic here

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Horiz1");

job.setJarByClass(FinalProject\_1.class);

job.setMapperClass(horizMap.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(DestFlightNoPojo.class);

job.setReducerClass(ReducerHorix.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

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

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

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

}

}

1. /\*
2. \* To change this license header, choose License Headers in Project Properties.
3. \* To change this template file, choose Tools | Templates
4. \* and open the template in the editor.
5. \*/
6. package finalproject\_2;
7. import java.io.IOException;
8. import java.util.ArrayList;
9. import java.util.Collections;
10. import java.util.List;
11. import org.apache.hadoop.conf.Configuration;
12. import org.apache.hadoop.fs.Path;
13. import org.apache.hadoop.io.Text;
14. import org.apache.hadoop.mapreduce.Job;
15. import org.apache.hadoop.mapreduce.Mapper;
16. import org.apache.hadoop.mapreduce.Reducer;
17. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
18. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
19. //import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
20. //import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
21. /\*\*
22. \*
23. \* @author akil
24. \*/
25. public class FinalProject\_2 {
26. /\*\*
27. \* @param args the command line arguments
28. \*/
29. public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {
30. // TODO code application logic here
31. Configuration conf = new Configuration();
32. Job job = Job.getInstance(conf, "Horiz1");
33. job.setJarByClass(FinalProject\_2.class);
34. job.setMapperClass(DelayMapper.class);
35. job.setMapOutputKeyClass(Text.class);
36. job.setMapOutputValueClass(DelaysPojo.class);
37. job.setReducerClass(DelayReducer.class);
38. job.setOutputKeyClass(Text.class);
39. job.setOutputValueClass(Text.class);
40. FileInputFormat.addInputPath(job, new Path(args[0]));
41. FileOutputFormat.setOutputPath(job, new Path(args[1]));
42. System.exit(job.waitForCompletion(true) ? 0 : 1);
43. }
44. public static class DelayMapper extends Mapper<Object, Text, Text, DelaysPojo> {
45. private Text origin = new Text();
47. static int count = 0;
48. private DelaysPojo custompojo = new DelaysPojo();
49. @Override
50. protected void map(Object key, Text value, Mapper.Context context) throws IOException, InterruptedException {
51. String input = new String(value.toString());
52. String[] token = input.split(",");
53. count++;

56. if (!token[0].trim().contains("Year")) {
58. origin.set(token[16]);
59. // passenger.set(token[9]);
60. /\*if (token[24].trim().contains("NA")){
61. custompojo.setCarrierDelay(0);
62. }
63. else {
64. custompojo.setCarrierDelay(Integer.parseInt(token[24]));
65. }
66. if (token[25].trim().contains("NA")){
67. custompojo.setCarrierDelay(0);
68. }
69. else {
70. custompojo.setWeatherDelay(Integer.parseInt(token[25]));
71. }
72. if (token[26].trim().contains("NA")){
73. custompojo.setCarrierDelay(0);
74. }
75. else {
76. custompojo.setNasDelay(Integer.parseInt(token[26]));
77. }
78. if (token[27].trim().contains("NA")){
79. custompojo.setCarrierDelay(0);
80. }
81. else {
82. custompojo.setSecurityDelay(Integer.parseInt(token[27]));
83. }
84. if (token[28].trim().contains("NA")){
85. custompojo.setCarrierDelay(0);
86. }
87. else {
88. custompojo.setLateAircraftDelay(Integer.parseInt(token[28]));
89. }\*/
90. if (!(token[24].trim().contains("NA"))){
91. custompojo.setCarrierDelay(Integer.parseInt(token[24]));
92. }
93. if (!(token[25].trim().contains("NA"))){
94. custompojo.setWeatherDelay(Integer.parseInt(token[25]));
95. }
96. if (!(token[26].trim().contains("NA"))){
97. custompojo.setNasDelay(Integer.parseInt(token[26]));
98. }
99. if (!(token[27].trim().contains("NA"))){
100. custompojo.setSecurityDelay(Integer.parseInt(token[27]));
101. }
102. if (!(token[28].trim().contains("NA"))){
103. custompojo.setLateAircraftDelay(Integer.parseInt(token[28]));
104. }
106. context.write(origin, custompojo);
107. }
108. }
109. }
110. public static class DelayReducer extends Reducer<Text, DelaysPojo, Text, Text> {
111. private Text result = new Text();
112. // private Set<String> dist = new HashSet<String>();
113. private List<String> dist = new ArrayList<String>();
114. @Override
115. protected void reduce(Text key, Iterable<DelaysPojo> values, Context context) throws IOException, InterruptedException {
116. int carrierDelayTotal = 0;
117. int weatherDelayTotal = 0;
118. int nasDelayTotal = 0;
119. int securityDelayTotal = 0;
120. int lateAircraftDelayTotal = 0;
121. for (DelaysPojo val : values) {
122. carrierDelayTotal = val.getCarrierDelay() + carrierDelayTotal;
123. weatherDelayTotal = val.getWeatherDelay() + weatherDelayTotal;
124. nasDelayTotal = val.getNasDelay() + nasDelayTotal;
125. securityDelayTotal = val.getSecurityDelay() + securityDelayTotal;
126. lateAircraftDelayTotal = val.getLateAircraftDelay() + lateAircraftDelayTotal;
127. }
128. dist.add(0, String.valueOf(carrierDelayTotal));
129. dist.add(1, String.valueOf(weatherDelayTotal));
130. dist.add(2, String.valueOf(nasDelayTotal));
131. dist.add(3, String.valueOf(securityDelayTotal));
132. dist.add(4, String.valueOf(lateAircraftDelayTotal));
134. int max = dist.indexOf(Collections.max(dist));
135. String delay = "";
136. if (max==0) {
137. delay = "CarrierDelay " + dist.get(max);
138. }
139. if (max == 1) {
140. delay = "WeatherDelay " + dist.get(max);
141. }
142. if (max == 2) {
143. delay = "NASDelay " + dist.get(max);
144. }
145. if (max == 3) {
146. delay = "SecurityDelayTotal " + dist.get(max);
147. }
148. if (max == 4) {
149. delay = "LateAircraftDelayTotal " + dist.get(max);
150. }
151. result.set(delay);
152. context.write(key, result);
153. dist.clear();
154. }
155. }
156. }

3) /\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalproject\_3;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.NullWritable;

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.input.MultipleInputs;

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

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

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

/\*\*

\*

\* @author akil

\*/

public class FinalProject\_3 {

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic here

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Monthwise Delay");

job.setJarByClass(FinalProject\_3.class);

job.setMapperClass(MonnthlyDelayMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(CancelledFlightsPojo.class);

job.setReducerClass(MonnthlyDelayReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

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

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

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

boolean success = job.waitForCompletion(true);

// return success ? 0 : 2;

Configuration conf2 = new Configuration();

Job job2 = Job.getInstance(conf2, "Month wise binning output");

if (success) {

job2.setJarByClass(FinalProject\_3.class);

MultipleOutputs.addNamedOutput(job2, "bins", TextOutputFormat.class, Text.class, NullWritable.class);

MultipleOutputs.setCountersEnabled(job2, true);

job2.setMapperClass(MonnthlyDelayMapper2.class);

job2.setNumReduceTasks(0);

FileInputFormat.addInputPath(job2, new Path(args[1]));

FileOutputFormat.setOutputPath(job2, new Path(args[2]));

System.exit(job2.waitForCompletion(true) ? 0 : 2);

}

return success ? 0 : 2;

}

public static class MonnthlyDelayMapper2 extends Mapper<Object, Text, Text, NullWritable> {

private MultipleOutputs<Text, NullWritable> mos = null;

@Override

protected void setup(Context context) throws IOException, InterruptedException {

mos = new MultipleOutputs(context); //To change body of generated methods, choose Tools | Templates.

}

@Override

protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {

// super.map(key, value, context); //To change body of generated methods, choose Tools | Templates.

//String input = new String(value.toString());

String[] token = value.toString().split("\\t");

String month = token[0];

//for (int i = 0; i < 12; i++) {

mos.write("bins", value, NullWritable.get(), month);

//}

//To change body of generated methods, choose Tools | Templates.

}

}

public static class MonnthlyDelayMapper extends Mapper<Object, Text, Text, CancelledFlightsPojo> {

private CancelledFlightsPojo custompojo = new CancelledFlightsPojo();

private Text source = new Text();

@Override

protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {

String input = new String(value.toString());

String[] token = input.split(",");

if (!token[0].trim().contains("Year")) {

source.set(token[1]);

custompojo.setFlightNo(Integer.parseInt(token[9]));

//custompojo.setMonth(Integer.parseInt(token[1]));

custompojo.setSource(token[16]);

custompojo.setDestination(token[17]);

custompojo.setCancellationCode(token[22]);

if (token[21].equalsIgnoreCase("1"))

context.write(source, custompojo);

}

}

}

public static class MonnthlyDelayReducer extends Reducer<Text, CancelledFlightsPojo, Text, Text> {

private Text result = new Text();

@Override

protected void reduce(Text key, Iterable<CancelledFlightsPojo> values, Context context) throws IOException, InterruptedException {

Integer flightNo;

Integer month;

String source;

String destination;

String cancellationCode;

int delayCountA = 0;

int delayCountB = 0;

int delayCountC = 0;

for (CancelledFlightsPojo val : values) {

flightNo = val.getFlightNo();

month = val.getMonth();

source = val.getSource();

destination = val.getDestination();

cancellationCode = val.getCancellationCode();

String cancelledFlightDetails = "";

String canCodeDes = "";

if (cancellationCode.matches("A")) {

canCodeDes = "Carrier Delay";

}

if (cancellationCode .contains("B")) {

canCodeDes = "Weather Delay";

}

if (cancellationCode.contains("C")) {

canCodeDes = "NAS Delay";

}

if (cancellationCode == "D") {

canCodeDes = "Weather Delay";

}

cancelledFlightDetails = source + " "+ flightNo + " " + destination + " " + canCodeDes;

result.set(cancelledFlightDetails);

context.write(key, result);

}

}

}

}

4) /\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalproject\_4;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.DoubleWritable;

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;

/\*\*

\*

\* @author akil

\*/

public class FinalProject\_4 {

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic here

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Horiz1");

job.setJarByClass(FinalProject\_4.class);

job.setMapperClass(DelayMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(DelaysPojo.class);

job.setReducerClass(DelayReducer.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);

}

public static class DelayMapper extends Mapper<Object, Text, Text, DelaysPojo> {

private Text origin = new Text();

static int count = 0;

private DelaysPojo custompojo = new DelaysPojo();

@Override

protected void map(Object key, Text value, Mapper.Context context) throws IOException, InterruptedException {

String input = new String(value.toString());

String[] token = input.split(",");

count++;

if (!token[0].trim().contains("Year")) {

origin.set(token[16]);

if (!(token[24].trim().contains("NA"))) {

custompojo.setCarrierDelay(Integer.parseInt(token[24]));

}

if (!(token[25].trim().contains("NA"))) {

custompojo.setWeatherDelay(Integer.parseInt(token[25]));

}

if (!(token[26].trim().contains("NA"))) {

custompojo.setNasDelay(Integer.parseInt(token[26]));

}

if (!(token[27].trim().contains("NA"))) {

custompojo.setSecurityDelay(Integer.parseInt(token[27]));

}

if (!(token[28].trim().contains("NA"))) {

custompojo.setLateAircraftDelay(Integer.parseInt(token[28]));

}

context.write(origin, custompojo);

}

}

}

public static class DelayReducer extends Reducer<Text, DelaysPojo, Text, DoubleWritable> {

private DoubleWritable result = new DoubleWritable();

@Override

protected void reduce(Text key, Iterable<DelaysPojo> values, Context context) throws IOException, InterruptedException {

int carrierDelayTotal = 0;

int weatherDelayTotal = 0;

int nasDelayTotal = 0;

int securityDelayTotal = 0;

int lateAircraftDelayTotal = 0;

int totalDelay = 0;

int count = 0;

for (DelaysPojo val : values) {

//carrierDelayTotal = val.getCarrierDelay() + carrierDelayTotal;

if (val.getCarrierDelay() > 0) {

carrierDelayTotal++;

}

if (val.getWeatherDelay() > 0) {

weatherDelayTotal++;

}

//weatherDelayTotal = val.getWeatherDelay() + weatherDelayTotal;

if (val.getNasDelay() > 0) {

nasDelayTotal++;

//nasDelayTotal = val.getNasDelay() + nasDelayTotal;

}

if (val.getSecurityDelay() > 0) {

securityDelayTotal++;

}

//securityDelayTotal = val.getSecurityDelay() + securityDelayTotal;

if (val.getLateAircraftDelay() > 0) {

lateAircraftDelayTotal++;

//lateAircraftDelayTotal = val.getLateAircraftDelay() + lateAircraftDelayTotal;

}

count++;

}

totalDelay = totalDelay + carrierDelayTotal + weatherDelayTotal + nasDelayTotal + securityDelayTotal + lateAircraftDelayTotal;

double frequency = (double)totalDelay / (count );

System.out.println("key "+key);

System.out.println("count "+count);

System.out.println("totalDelay "+totalDelay);

result.set(frequency);

context.write(key, result);

}

}

}

5)

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalproject\_5;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import java.net.URI;

import java.net.URISyntaxException;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.filecache.DistributedCache;

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

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

/\*\*

\*

\* @author akil

\*/

public class FinalProject\_5 {

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic here

Configuration conf1 = new Configuration();

Job job1 = Job.getInstance(conf1, "distinct chaning");

job1.setJarByClass(FinalProject\_5.class);

job1.setMapperClass(DistinctMapper.class);

job1.setMapOutputKeyClass(Text.class);

job1.setMapOutputValueClass(DestinationDistPojo.class);

DistributedCache.addCacheFile(new URI(args[2]), job1.getConfiguration());

job1.setReducerClass(DistinctReducer.class);

job1.setOutputKeyClass(Text.class);

job1.setOutputValueClass(DestinationDistPojo.class);

FileInputFormat.addInputPath(job1, new Path(args[0]));

FileOutputFormat.setOutputPath(job1, new Path(args[1]));

System.exit(job1.waitForCompletion(true) ? 0 : 1);

}

public static class DistinctMapper extends Mapper<Object, Text, Text, DestinationDistPojo> {

private Text source = new Text();

private DestinationDistPojo destinationPojo = new DestinationDistPojo();

@Override

protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {

// super.map(key, value, context); //To change body of generated methods, choose Tools | Templates.

String input = new String(value.toString());

String[] token = input.split(",");

if (!token[0].trim().contains("Year")) {

source.set(token[16]);

destinationPojo.setDestination(token[17]);

destinationPojo.setDistance(Integer.parseInt(token[18]));

// System.out.println("source"+token[16]);

//System.out.println("row"+token[7]+" "+token[8] + " "+ token[9] +" "+ token[10]+ " "+token[17]);

context.write(source, destinationPojo);

}

}

}

public static class DistinctReducer extends Reducer<Text, DestinationDistPojo, Text, DestinationDistPojo> {

private DestinationDistPojo result = new DestinationDistPojo();

private HashMap<String, Integer> destMap = new HashMap();

private LookupPojoParcer metadata;

String fileName;

String airportName;

String destinationAirportName;

protected void setup(Context context) throws IOException {

try {

Path[] localFiles = DistributedCache.getLocalCacheFiles(context.getConfiguration());

for (Path eachPath : localFiles) {

// fileName = eachPath.getName().toString().trim();

//if (fileName.equals("airports.csv")) {

File myFile = new File(eachPath.toUri());

// BufferedReader bufferedReader = new BufferedReader(new FileReader(myFile.toString()));

metadata = new LookupPojoParcer();

metadata.initialize(myFile);

break;

//}

}

System.out.println("File : " + localFiles[0].toString());

} catch (NullPointerException e) {

System.out.println("Exception : " + e);

}

System.out.println("Cache : " + context.getConfiguration().get("mapred.cache.files"));

}

@Override

protected void reduce(Text key, Iterable<DestinationDistPojo> values, Context context) throws IOException, InterruptedException {

// super.reduce(key, values, context); //To change body of generated methods, choose Tools | Templates.

Integer distance;

String destination;

String airportName = metadata.getStationName(key.toString());

System.out.println("key"+key);

System.out.println("airportName"+airportName);

for (DestinationDistPojo d : values) {

distance = d.getDistance();

destination = d.getDestination();

destMap.put(destination, distance);

//result.setDestination(destination);

//result.setDistance(distance);

//System.out.println("source"+key);

//System.out.println("destination "+destination);

//context.write(key, result);

}

for (Map.Entry<String, Integer> entry : destMap.entrySet()) {

destinationAirportName = metadata.getStationName(entry.getKey());

result.setDestination(destinationAirportName);

// result.setDestination(entry.getKey());

result.setDistance(entry.getValue());

context.write(new Text(airportName), result);

}

}

}

}

6) /\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalproject\_6;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import org.apache.hadoop.conf.Configurable;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

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

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

/\*\*

\*

\* @author akil

\*/

public class FinalProject\_6 {

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic here

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Horiz1");

job.setJarByClass(FinalProject\_6.class);

job.setMapperClass(DelayMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(DelaysPojo.class);

job.setReducerClass(DelayReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

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

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

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

boolean success = job.waitForCompletion(true);

// return success ? 0 : 2;

Configuration conf2 = new Configuration();

Job job2 = Job.getInstance(conf2, "Chaining");

if (success) {

job2.setJarByClass(FinalProject\_6.class);

job2.setMapperClass(partitionerMapper.class);

job2.setMapOutputKeyClass(Text.class);

job2.setMapOutputValueClass(Text.class);

job2.setReducerClass(partionReducer.class);

job2.setOutputKeyClass(Text.class);

job2.setOutputValueClass(NullWritable.class);

job2.setPartitionerClass(partionerDelay.class);

job2.setNumReduceTasks(5);

FileInputFormat.addInputPath(job2, new Path(args[1]));

FileOutputFormat.setOutputPath(job2, new Path(args[2]));

System.exit(job2.waitForCompletion(true) ? 0 : 2);

}

}

public static class DelayMapper extends Mapper<Object, Text, Text, DelaysPojo> {

private Text origin = new Text();

static int count = 0;

private DelaysPojo custompojo = new DelaysPojo();

@Override

protected void map(Object key, Text value, Mapper.Context context) throws IOException, InterruptedException {

String input = new String(value.toString());

String[] token = input.split(",");

count++;

if (!token[0].trim().contains("Year")) {

origin.set(token[16]);

if (!(token[24].trim().contains("NA"))) {

custompojo.setCarrierDelay(Integer.parseInt(token[24]));

}

if (!(token[25].trim().contains("NA"))) {

custompojo.setWeatherDelay(Integer.parseInt(token[25]));

}

if (!(token[26].trim().contains("NA"))) {

custompojo.setNasDelay(Integer.parseInt(token[26]));

}

if (!(token[27].trim().contains("NA"))) {

custompojo.setSecurityDelay(Integer.parseInt(token[27]));

}

if (!(token[28].trim().contains("NA"))) {

custompojo.setLateAircraftDelay(Integer.parseInt(token[28]));

}

context.write(origin, custompojo);

}

}

}

public static class DelayReducer extends Reducer<Text, DelaysPojo, Text, Text> {

private Text result = new Text();

// private Set<String> dist = new HashSet<String>();

private List<String> dist = new ArrayList<String>();

@Override

protected void reduce(Text key, Iterable<DelaysPojo> values, Context context) throws IOException, InterruptedException {

int carrierDelayTotal = 0;

int weatherDelayTotal = 0;

int nasDelayTotal = 0;

int securityDelayTotal = 0;

int lateAircraftDelayTotal = 0;

for (DelaysPojo val : values) {

carrierDelayTotal = val.getCarrierDelay() + carrierDelayTotal;

weatherDelayTotal = val.getWeatherDelay() + weatherDelayTotal;

nasDelayTotal = val.getNasDelay() + nasDelayTotal;

securityDelayTotal = val.getSecurityDelay() + securityDelayTotal;

lateAircraftDelayTotal = val.getLateAircraftDelay() + lateAircraftDelayTotal;

}

dist.add(0, String.valueOf(carrierDelayTotal));

dist.add(1, String.valueOf(weatherDelayTotal));

dist.add(2, String.valueOf(nasDelayTotal));

dist.add(3, String.valueOf(securityDelayTotal));

dist.add(4, String.valueOf(lateAircraftDelayTotal));

int max = dist.indexOf(Collections.max(dist));

String delay = "";

if (max == 0) {

delay = "CarrierDelay";

}

if (max == 1) {

delay = "WeatherDelay";

}

if (max == 2) {

delay = "NASDelay";

}

if (max == 3) {

delay = "SecurityDelay";

}

if (max == 4) {

delay = "LateAircraftDelay";

}

result.set(delay);

context.write(key, result);

dist.clear();

}

}

public static class partitionerMapper extends Mapper<LongWritable, Text, Text, Text> {

@Override

protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

// super.map(key, value, context); //To change body of generated methods, choose Tools | Templates.

String input = new String(value.toString());

String[] token = input.split("\\t");

context.write(new Text(token[1]), new Text(token[0]));

}

}

public static class partionerDelay extends Partitioner<Text, Text> {

@Override

public int getPartition(Text key, Text value, int NumOfReduceTask) {

String input = new String(value.toString());

String[] token = input.split("\\t");

if (key.equals(new Text("CarrierDelay"))) {

return 0;

}

if (key.equals(new Text("WeatherDelay"))) {

return 1;

}

if (key.equals(new Text("NASDelay"))) {

return 2;

}

if (key.equals(new Text("SecurityDelay"))) {

return 3;

}

// if(key.equals(new Text("LateAircraftDelay"))){

return 4;

}

}

public static class partionReducer extends Reducer<Text, Text, Text, NullWritable> {

@Override

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

for (Text t : values) {

context.write(t, NullWritable.get());

}

}

}

}

7) /\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalproject\_7;

import java.io.IOException;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

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;

/\*\*

\*

\* @author akil

\*/

public class FinalProject\_7 {

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic here

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "top 10");

job.setJarByClass(FinalProject\_7.class);

job.setMapperClass(DelayMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(DelaysPojo.class);

job.setReducerClass(DelayReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

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

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

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

boolean success = job.waitForCompletion(true);

// return success ? 0 : 2;

Configuration conf2 = new Configuration();

Job job2 = Job.getInstance(conf2, "Chaining");

if (success) {

job2.setJarByClass(FinalProject\_7.class);

job2.setMapperClass(topTenMapper.class);

job2.setMapOutputKeyClass(NullWritable.class);

job2.setMapOutputValueClass(Text.class);

job2.setReducerClass(topTenReducer.class);

job2.setOutputKeyClass(NullWritable.class);

job2.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job2, new Path(args[1]));

FileOutputFormat.setOutputPath(job2, new Path(args[2]));

System.exit(job2.waitForCompletion(true) ? 0 : 1);

}

return success ? 0 : 2;

}

public static class DelayMapper extends Mapper<Object, Text, Text, DelaysPojo> {

private Text origin = new Text();

static int count = 0;

private DelaysPojo custompojo = new DelaysPojo();

@Override

protected void map(Object key, Text value, Mapper.Context context) throws IOException, InterruptedException {

String input = new String(value.toString());

String[] token = input.split(",");

count++;

if (!token[0].trim().contains("Year")) {

origin.set(token[16]);

if (!(token[24].trim().contains("NA"))) {

custompojo.setCarrierDelay(Integer.parseInt(token[24]));

}

if (!(token[25].trim().contains("NA"))) {

custompojo.setWeatherDelay(Integer.parseInt(token[25]));

}

if (!(token[26].trim().contains("NA"))) {

custompojo.setNasDelay(Integer.parseInt(token[26]));

}

if (!(token[27].trim().contains("NA"))) {

custompojo.setSecurityDelay(Integer.parseInt(token[27]));

}

if (!(token[28].trim().contains("NA"))) {

custompojo.setLateAircraftDelay(Integer.parseInt(token[28]));

}

context.write(origin, custompojo);

}

}

}

public static class DelayReducer extends Reducer<Text, DelaysPojo, Text, IntWritable> {

private DoubleWritable result = new DoubleWritable();

@Override

protected void reduce(Text key, Iterable<DelaysPojo> values, Context context) throws IOException, InterruptedException {

int carrierDelayTotal = 0;

int weatherDelayTotal = 0;

int nasDelayTotal = 0;

int securityDelayTotal = 0;

int lateAircraftDelayTotal = 0;

int totalDelay = 0;

int count = 0;

for (DelaysPojo val : values) {

//carrierDelayTotal = val.getCarrierDelay() + carrierDelayTotal;

if (val.getCarrierDelay() > 0) {

carrierDelayTotal++;

}

if (val.getWeatherDelay() > 0) {

weatherDelayTotal++;

}

//weatherDelayTotal = val.getWeatherDelay() + weatherDelayTotal;

if (val.getNasDelay() > 0) {

nasDelayTotal++;

//nasDelayTotal = val.getNasDelay() + nasDelayTotal;

}

if (val.getSecurityDelay() > 0) {

securityDelayTotal++;

}

//securityDelayTotal = val.getSecurityDelay() + securityDelayTotal;

if (val.getLateAircraftDelay() > 0) {

lateAircraftDelayTotal++;

//lateAircraftDelayTotal = val.getLateAircraftDelay() + lateAircraftDelayTotal;

}

count++;

}

totalDelay = totalDelay + carrierDelayTotal + weatherDelayTotal + nasDelayTotal + securityDelayTotal + lateAircraftDelayTotal;

double frequency = (double) totalDelay / (count);

// System.out.println("key " + key);

// System.out.println("count " + count);

// System.out.println("totalDelay " + totalDelay);

// result.set(frequency);

context.write(key, new IntWritable(totalDelay));

}

}

public static class topTenMapper extends Mapper<LongWritable, Text, NullWritable, Text> {

private TreeMap<Integer, Text> delaySourceMap = new TreeMap<Integer, Text>();

@Override

protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

// super.map(key, value, context); //To change body of generated methods, choose Tools | Templates.

String input = new String(value.toString());

String[] token = input.split(" ");

delaySourceMap.put(Integer.parseInt(token[1]), new Text(input));

if (delaySourceMap.size() > 10) {

delaySourceMap.remove(delaySourceMap.firstKey());

}

}

@Override

protected void cleanup(Context context) throws IOException, InterruptedException {

// super.cleanup(context); //To change body of generated methods, choose Tools | Templates.

for (Text t : delaySourceMap.values()) {

context.write(NullWritable.get(), t);

}

}

}

public static class topTenReducer extends Reducer<NullWritable, Text, NullWritable, Text> {

private TreeMap<Integer, Text> delaySourceMapRed = new TreeMap<Integer, Text>();

@Override

protected void reduce(NullWritable key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

//super.reduce(key, values, context); //To change body of generated methods, choose Tools | Templates.

for (Text value : values) {

String input = new String(value.toString());

String[] token = input.split(" ");

delaySourceMapRed.put(Integer.parseInt(token[1]), new Text(input));

if (delaySourceMapRed.size() > 10) {

delaySourceMapRed.remove(delaySourceMapRed.firstKey());

}

}

}

@Override

protected void cleanup(Context context) throws IOException, InterruptedException {

// super.cleanup(context); //To change body of generated methods, choose Tools | Templates.

for (Text t : delaySourceMapRed.descendingMap().values()) {

context.write(NullWritable.get(), t);

}

}

}

}

8) /\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalproject\_8;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Collections;

import java.util.HashMap;

import java.util.Map;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.NullWritable;

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.input.MultipleInputs;

import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

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

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

/\*\*

\*

\* @author akil

\*/

public class FinalProject\_8 {

/\*\*

\* @param args the command line arguments

\*/

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "join pattern");

job.setJarByClass(FinalProject\_8.class);

job.setMapperClass(sourceMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(CustomPojo.class);

job.setReducerClass(sourceReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(CustomPojo.class);

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

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

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

boolean success = job.waitForCompletion(true);

// return success ? 0 : 2;

Configuration conf2 = new Configuration();

Job job2 = Job.getInstance(conf2, "Chaining");

if (success) {

job2.setJarByClass(FinalProject\_8.class);

MultipleInputs.addInputPath(job2, new Path(args[1]), TextInputFormat.class, JoinMapper1.class);

MultipleInputs.addInputPath(job2, new Path(args[2]), TextInputFormat.class, JoinMapper2.class);

job2.setReducerClass(JoinReducer.class);

job2.setOutputFormatClass(TextOutputFormat.class);

TextOutputFormat.setOutputPath(job2, new Path(args[3]));

job2.setOutputKeyClass(Text.class);

job2.setOutputValueClass(Text.class);

System.exit(job2.waitForCompletion(true) ? 0 : 1);

}

return success ? 0 : 2;

}

public static class sourceMapper extends Mapper<Object, Text, Text, CustomPojo> {

private Text source = new Text();

private CustomPojo destinationPojo = new CustomPojo();

@Override

protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {

//super.map(key, value, context); //To change body of generated methods, choose Tools | Templates.

String input = new String(value.toString());

String[] token = input.split(",");

if (!token[0].trim().contains("Year")) {

source.set(token[16]);

destinationPojo.setDestination(token[17]);

destinationPojo.setFlightNo(token[9]);

context.write(source, destinationPojo);

}

}

}

public static class sourceReducer extends Reducer<Text, CustomPojo, Text, CustomPojo> {

private CustomPojo result = new CustomPojo();

private HashMap<String, String> destMap = new HashMap();

@Override

protected void reduce(Text key, Iterable<CustomPojo> values, Context context) throws IOException, InterruptedException {

// super.reduce(key, values, context); //To change body of generated methods, choose Tools | Templates.

String flightNo;

String destination;

String prev\_destination = "";

int count = 0;

System.out.println("key" + key);

// System.out.println("airportName"+airportName);

for (CustomPojo d : values) {

// distance = d.getDistance();

destination = d.getDestination();

flightNo = d.getFlightNo();

destMap.put(flightNo, destination);

//Collection val=destMap.values();

}

for (Map.Entry<String, String> entry : destMap.entrySet()) {

result.setDestination(entry.getValue());

result.setFlightNo(entry.getKey());

context.write(key, result);

}

}

}

public static class JoinMapper1 extends Mapper<Object, Text, Text, Text> {

private Text outKey = new Text();

private Text outValue = new Text();

public void map(Object key, Text value, Mapper.Context context) {

try {

String[] seperatedInput = value.toString().split(" ");

String source = seperatedInput[0].trim();

if (source == null) {

return;

}

outKey.set(source);

outValue.set("S" + value);

context.write(outKey, outValue);

} catch (IOException | InterruptedException ex) {

Logger.getLogger(FinalProject\_8.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

public static class JoinMapper2 extends Mapper<Object, Text, Text, Text> {

private Text outKey = new Text();

private Text outValue = new Text();

public void map(Object key, Text value, Mapper.Context context) {

try {

String input = new String(value.toString());

String[] token = input.split(",");

if (!token[0].trim().contains("iata")) {

String code = token[0];

String name = token[1];

// if (code != " " && name != " ") {

//hm.put(key, value);

outKey.set(code);

outValue.set("B" + value);

context.write(outKey, outValue);

}

} catch (IOException | InterruptedException ex) {

Logger.getLogger(FinalProject\_8.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

public static class JoinReducer extends Reducer<Text, Text, Text, Text> {

private static final Text EMPTY\_TEXT = new Text();

private Text tmp = new Text();

private ArrayList<Text> listA = new ArrayList<Text>();

private ArrayList<Text> listB = new ArrayList<Text>();

@Override

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

//super.reduce(key, values, context); //To change body of generated methods, choose Tools | Templates.

// super.reduce(key, values, context); //To change body of generated methods, choose Tools | Templates.

listA.clear();

listB.clear();

while (values.iterator().hasNext()) {

tmp = values.iterator().next();

if (tmp.charAt(0) == 'S') {

String[] seperatedInput = tmp.toString().split(" ");

listA.add(new Text(seperatedInput[1] +" "+ seperatedInput[2]));

} else if (tmp.charAt(0) == 'B') {

String[] seperatedInput = tmp.toString().split(",");

listB.add(new Text(seperatedInput[1]));

// listB.add(new Text(seperatedInput[1]));

}

}

executeJoinLogic(context);

}

private void executeJoinLogic(Context context) {

for (Text A : listA) {

if (!listB.isEmpty()) {

for (Text B : listB) {

try {

context.write(A, B);

} catch (IOException | InterruptedException ex) {

Logger.getLogger(FinalProject\_8.class.getName()).log(Level.SEVERE, null, ex);

}

}

} else {

try {

context.write(A, EMPTY\_TEXT);

} catch (IOException | InterruptedException ex) {

Logger.getLogger(FinalProject\_8.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

}

}

}

9 ) /\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalproject\_9;

import java.io.File;

import java.io.IOException;

import java.net.URI;

import java.net.URISyntaxException;

import java.util.HashMap;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.filecache.DistributedCache;

import org.apache.hadoop.fs.Path;

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;

import com.google.common.base.Charsets;

import com.google.common.hash.BloomFilter;

import com.google.common.hash.Funnel;

import com.google.common.hash.Sink;

import java.io.BufferedReader;

import java.io.FileReader;

import org.apache.hadoop.filecache.DistributedCache;

import org.apache.hadoop.io.NullWritable;

/\*\*

\*

\* @author akil

\*/

public class FinalProject\_9 {

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic here

Configuration conf1 = new Configuration();

Job job1 = Job.getInstance(conf1, "distinct chaning");

job1.setJarByClass(FinalProject\_9.class);

job1.setMapperClass(delayMapper.class);

job1.setMapOutputKeyClass(Text.class);

job1.setMapOutputValueClass(DestinationDelayPojo.class);

DistributedCache.addCacheFile(new URI(args[2]), job1.getConfiguration());

job1.setReducerClass(delayReducer.class);

job1.setOutputKeyClass(Text.class);

job1.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job1, new Path(args[0]));

FileOutputFormat.setOutputPath(job1, new Path(args[1]));

// System.exit(job1.waitForCompletion(true) ? 0 : 1);

boolean success = job1.waitForCompletion(true);

// return success ? 0 : 2;

Configuration conf2 = new Configuration();

Job job2 = Job.getInstance(conf2, "Filter");

if (success) {

job2.setJarByClass(FinalProject\_9.class);

job2.setMapperClass(BloomFilterMapper.class);

job2.setMapOutputKeyClass(Text.class);

job2.setMapOutputValueClass(NullWritable.class);

//adding the file in the cache having the Person class records

//job.addCacheFile(new Path("localhost:9000/bhavesh/LabAssignment/CacheInput/cache.txt").toUri());

DistributedCache.addCacheFile(new URI(args[4]), job2.getConfiguration());

job2.setNumReduceTasks(0);

FileInputFormat.addInputPath(job2, new Path(args[1]));

FileOutputFormat.setOutputPath(job2, new Path(args[3]));

job2.waitForCompletion(true);

}

}

public static class delayMapper extends Mapper<Object, Text, Text, DestinationDelayPojo> {

private Text source = new Text();

private DestinationDelayPojo delaypojo = new DestinationDelayPojo();

@Override

protected void map(Object key, Text value, Mapper.Context context) throws IOException, InterruptedException {

// super.map(key, value, context); //To change body of generated methods, choose Tools | Templates.

String input = new String(value.toString());

String[] token = input.split(",");

if (!token[0].trim().contains("Year")) {

source.set(token[16]);

if (token[15].isEmpty()) {

token[15] = "0";

}

if (token[14].isEmpty()) {

token[14] = "0";

}

delaypojo.setDepartureDelay(Double.parseDouble(token[15]));

delaypojo.setArrivalDelay(Double.parseDouble(token[14]));

// System.out.println("source"+token[16]);

//System.out.println("row"+token[7]+" "+token[8] + " "+ token[9] +" "+ token[10]+ " "+token[17]);

context.write(source, delaypojo);

}

}

}

public static class delayReducer extends Reducer<Text, DestinationDelayPojo, Text, Text> {

private Text result = new Text();

private HashMap<String, Integer> destMap = new HashMap();

private LookupPojoParcer metadata;

String fileName;

String airportName;

String destinationAirportName;

protected void setup(Context context) throws IOException {

try {

Path[] localFiles = DistributedCache.getLocalCacheFiles(context.getConfiguration());

for (Path eachPath : localFiles) {

// fileName = eachPath.getName().toString().trim();

//if (fileName.equals("airports.csv")) {

File myFile = new File(eachPath.toUri());

// BufferedReader bufferedReader = new BufferedReader(new FileReader(myFile.toString()));

metadata = new LookupPojoParcer();

metadata.initialize(myFile);

break;

//}

}

System.out.println("File : " + localFiles[0].toString());

} catch (NullPointerException e) {

System.out.println("Exception : " + e);

}

System.out.println("Cache : " + context.getConfiguration().get("mapred.cache.files"));

}

@Override

protected void reduce(Text key, Iterable<DestinationDelayPojo> values, Context context) throws IOException, InterruptedException {

// super.reduce(key, values, context); //To change body of generated methods, choose Tools | Templates.

String airportName = metadata.getStationName(key.toString());

int arrivalDelayCount = 0;

int departDelayCount = 0;

Double arrivalDelay = 0.0;

Double departDelay = 0.0;

for (DestinationDelayPojo d : values) {

//arrivalDelay = d.arrivalDelay;

//departDelay = d.departureDelay;

int arrret = d.arrivalDelay.compareTo(0.0);

if (arrret <= 0) {

arrivalDelayCount++;

}

int delret = d.departureDelay.compareTo(0.0);

if (delret <= 0) {

departDelayCount++;

}

}

String s = arrivalDelayCount + " " + departDelayCount;

context.write(new Text(airportName), new Text(s));

}

}

public static class BloomFilterMapper extends Mapper<Object, Text, Text, NullWritable> {

Funnel<FilterPojo> p = new Funnel<FilterPojo>() {

@Override

public void funnel(FilterPojo t, Sink sink) {

// throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

sink.putString(t.source).putInt(t.arrivalDelay)

.putInt(t.departDelay); }

};

private BloomFilter<FilterPojo> best = BloomFilter.create(p, 500, 0.01);

@Override

protected void setup(Context context) throws IOException, InterruptedException {

// super.setup(context); //To change body of generated methods, choose Tools | Templates.

String source;

int arrivalDelay;

int departDelay;

try {

Path[] files = DistributedCache.getLocalCacheFiles(context.getConfiguration());

if (files != null && files.length > 0) {

for (Path file : files) {

try {

File myFile = new File(file.toUri());

BufferedReader bufferedReader = new BufferedReader(new FileReader(myFile.toString()));

String line = null;

while ((line = bufferedReader.readLine()) != null) {

String[] split = line.split(",");

source = String.valueOf(split[0]);

arrivalDelay = Integer.parseInt(split[1]);

departDelay = Integer.parseInt(split[2]);

FilterPojo ft = new FilterPojo(source, arrivalDelay, departDelay);

best.put(ft);

}

} catch (IOException ex) {

System.err.println("Exception while reading file: " + ex.getMessage());

}

}

}

} catch (IOException ex) {

System.err.println("Exception in mapper setup: " + ex.getMessage());

}

}

@Override

protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {

// super.map(key, value, context); //To change body of generated methods, choose Tools | Templates.

String values[] = value.toString().split(" ");

FilterPojo ft = new FilterPojo(values[0].toString(), Integer.parseInt(values[1]), Integer.parseInt(values[2]));

//Person p = new Person(Integer.parseInt(values[0]), values[1], values[2], Integer.parseInt(values[3]));

if (best.mightContain(ft)) {

context.write(value, NullWritable.get());

}

}

}

}