|  |  |
| --- | --- |
| ../Downloads/flight-01.jpg  Flight Delay Analysis  Final Project | Advanced Database Management Systems    Pratik Singh  NUID: 001625041 |

**Summary**

The increase in delays in the National Airspace System (NAS) has been the subject of several studies in recent years. These reports contain delay statistics over the NAS data, along with some data specific to individual airports.

Different analysis has been performed with Hadoop Map reduce (Java) to analyze different areas for identifying the factors influencing the delay.

Along with the analysis, prediction is also performed to get the approximate delay for the flight by known factors including the flight number, day and month of travel.

**Data set**

The data is taken from Data Expo 2009 challenge which consists of flight arrival and departure details for all commercial flights within the USA, from January **2004 to April 2008**. This is a large dataset: 4 gigabytes when uncompressed.

      The U.S. Department of Transportation's (DOT) Bureau of Transportation Statistics (BTS) tracks the on-time performance of domestic flights operated by large air carriers. Summary information on the number of on-time, delayed, canceled and diverted flights appears in DOT's monthly Air Travel Consumer Report, published about 30 days after the month's end, as well as in summary tables posted on this website. Summary statistics and raw data are made available to the public at the time the Air Travel Consumer Report is released.

Links to Data Set:

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

<https://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp>

Table of Contents

[1. Top 10 Busiest Airport 6](#_Toc480944237)

[**1.1****Analysis** 6](#_Toc480944238)

[**1.2****Approach** 7](#_Toc480944239)

[**1.3****Design Patterns and Technologies:** 7](#_Toc480944240)

[2. Average Arrival delay with total flight count for each state (2008) 8](#_Toc480944241)

[**2.1****Analysis** 8](#_Toc480944242)

[**2.2****Approach** 9](#_Toc480944243)

[**2.3****Design Pattern and Technology** 9](#_Toc480944244)

[3. Average Departure Delay for each Month by Year 10](#_Toc480944245)

[**3.1****Analysis** 10](#_Toc480944246)

[**3.2****Approach** 11](#_Toc480944247)

[**3.3****Design Pattern and Technologies** 11](#_Toc480944248)

[4. Distribution of delay by type 12](#_Toc480944249)

[**4.1****Analysis** 12](#_Toc480944250)

[**4.2****Approach** 13](#_Toc480944251)

[**4.3****Design Pattern and Technologies:** 13](#_Toc480944252)

[5. Finding the Busiest route and identifying the airport connections 14](#_Toc480944253)

[**5.1****Analysis** 14](#_Toc480944254)

[**5.2****Approach** 15](#_Toc480944255)

[**5.3****Design patterns and Technologies** 15](#_Toc480944256)

[6. Different options (Direct Flight) to fly from Illinois to California 16](#_Toc480944257)

[**6.1****Analysis** 16](#_Toc480944258)

[**6.2****Approach** 16](#_Toc480944259)

[**6.3****Design Pattern and Technologies** 16](#_Toc480944260)

[7. Are older Aircraft prone to more delays then the Newer Aircraft? 17](#_Toc480944261)

[**7.1****Analysis** 17](#_Toc480944262)

[**7.2****Approach:** 18](#_Toc480944263)

[**7.3****Design Pattern and Technologies:** 18](#_Toc480944264)

[8. Carrier Popularity 19](#_Toc480944265)

[**8.1****Analysis** 19](#_Toc480944266)

[**8.2****Approach** 20](#_Toc480944267)

[**8.3****Design Pattern and Technologies** 20](#_Toc480944268)

[9. Finding the approximate delay for a flight 20](#_Toc480944269)

[9.1**Prediction** 21](#_Toc480944270)

[**9.2****Approach** 21](#_Toc480944271)

[**9.3****Design Pattern:** 22](#_Toc480944272)

[10. References 22](#_Toc480944273)

[11. Code Base 22](#_Toc480944274)

**List of Analysis**

* Top 10 Busiest Airport
* Average Arrival delay with total flight count for each state (2008)
* Average Departure Delay for each Month by Year
* Distribution of delay by type
* Finding the Busiest route and identifying the airport connections
* Different options (Direct Flight) to fly from Illinois to California
* Are older Aircraft prone to more delays than the Newer Aircraft?
* Carrier Popularity
* Finding the approximate delay for a particular flight

# **Top 10 Busiest Airport**

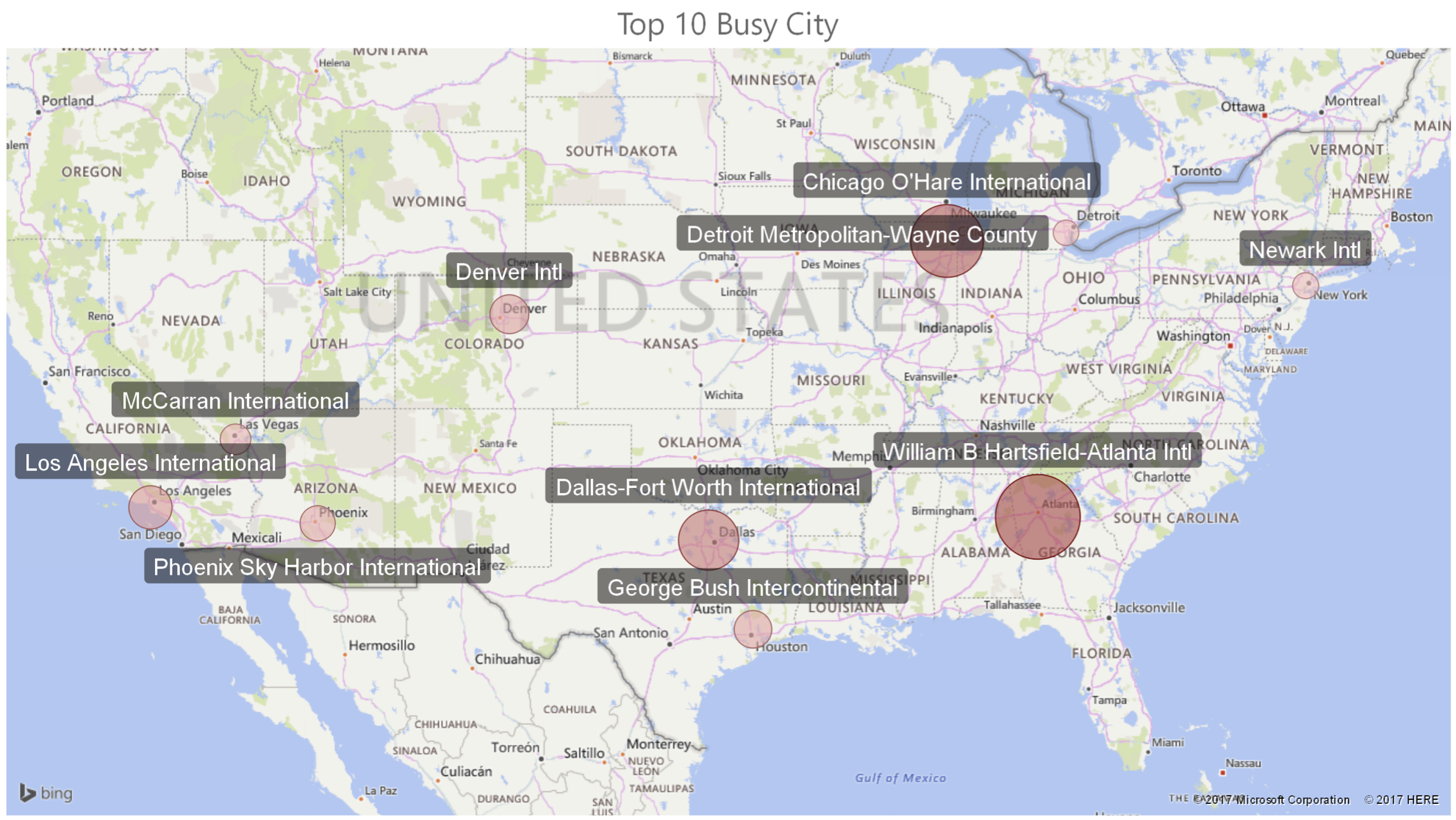
## **Analysis**

What is the count of total flights for each airport (Top 10) in last 5 years?



*Conclusion:* The bar graph shows the Top 10 busiest Airport based on along with their flight volumes for 5 years (2004-2008). This analysis can be used to compare the top 10 busiest airports in terms of Flight Count  
Maximum: William BHeartsfield, Atlanta  
Count:4.17M flights  
Minimum: Detroit Metropoliton - wayne county  
count:1.50M flights

Which airport is the most active airport?



*Conclusion*: The map shows the top 10 busiest airports with the actual locations. The weight for the activities is represented by bubble pointing the location.  
Maximum: William BHeartsfield, Atlanta  
Minimum: Detroit Metropoliton - wayne county

## **Approach**

Used Job Chaining to implement this Analysis. First job reads the input files for all years and Counts the total inbound flights. Reducer reads airports details files from the distributed cache and performs the replicated join. Second Job Calculates the total number of out bound flights for each airport. Final job reads the output from the previous two jobs (two Files using MultipleInputPath ) and computes the total number of flights. Used Pig to get the Top 10 Busy City

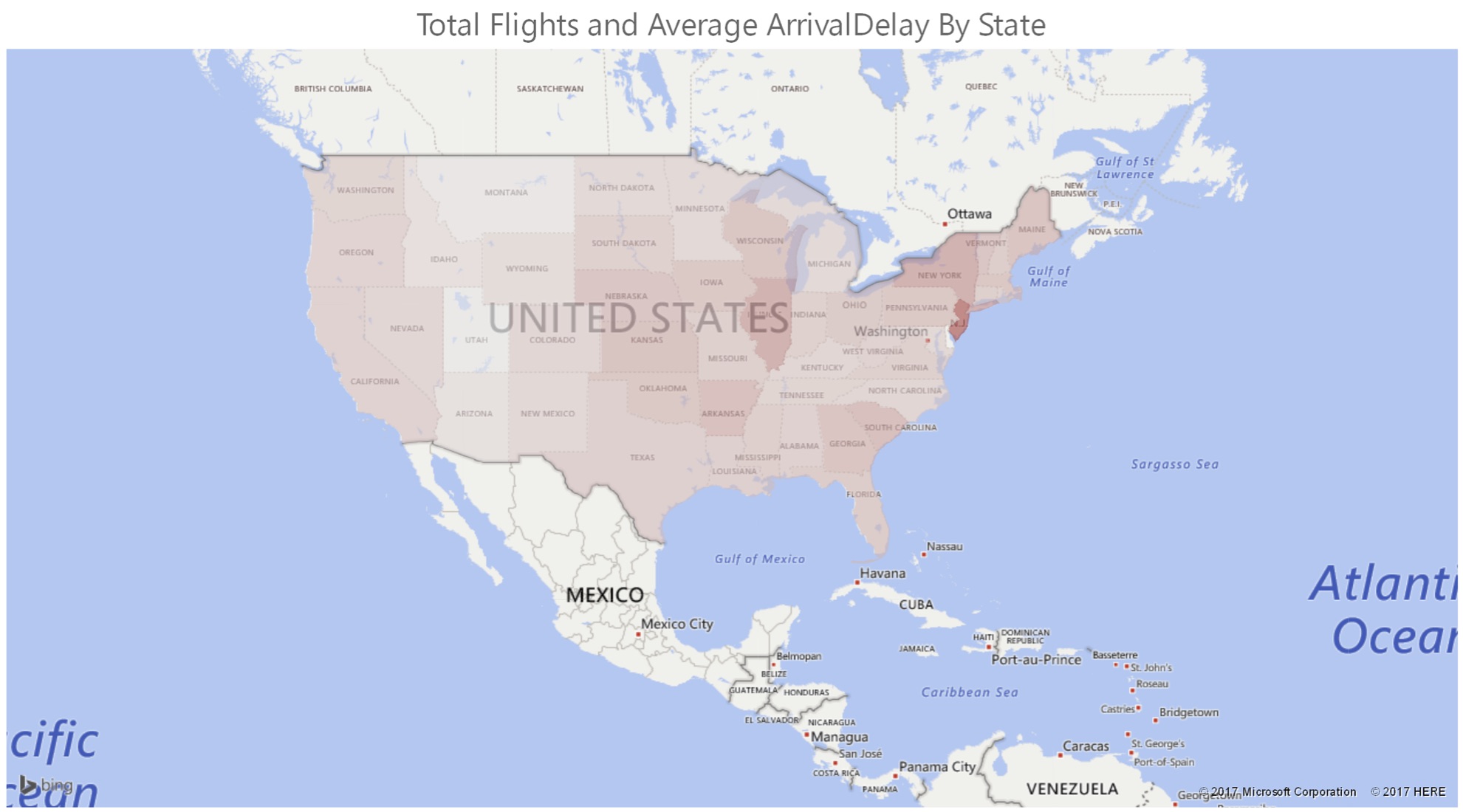
## **Design Patterns and Technologies:**

* Replicated Join, Numerical Summarization Counting, Filtering Pattern Top 10
* Map reduce, Distributed Cache, Apache Pig, Power BI

# **Average Arrival delay with total flight count for each state (2008)**

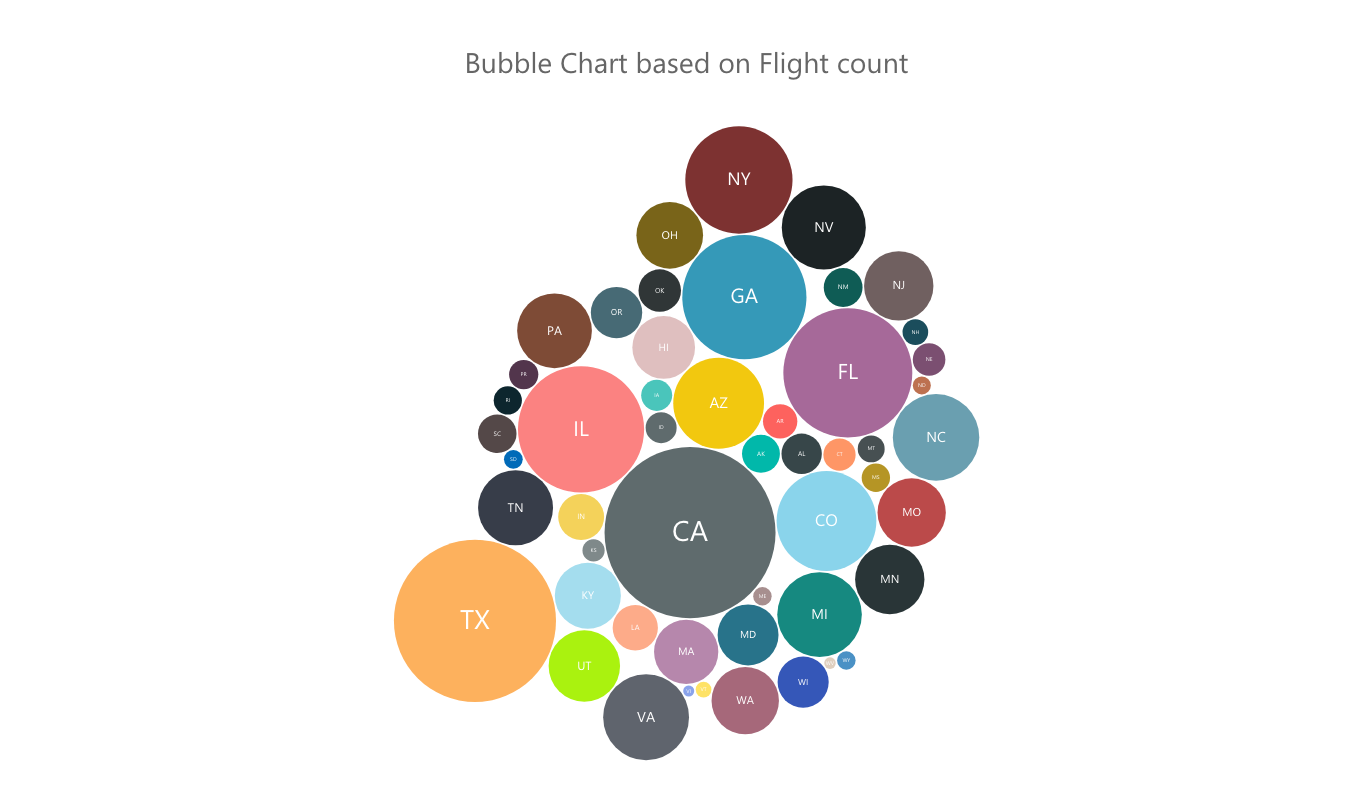
## **Analysis**

For which state the average arrival delay is high?



*Conclusion:* The map shows average arrival delay for each state based on the Inbound Traffic. The darker shade shows the state with high arrival delays.  
Maximum: New Jersey (darkest shade) with an average arrival delay of approximately 20 minutes.

Which state has more number of inbound flights?



*Conclusion:* The bubble shows the states with the state initials in it. The size of the bubbles depends on the flight traffic (inbound) in that state. Bigger bubbles show more flight traffic  
Maximum: CA (California) with grey bubble with maximum size   
Minimum: VI(Virgin Island) next to MA(blue) with least size  
Year:2008

## **Approach**

Computed the Average Arrival delay for each state for the year 2008 and also the count for storing the total inbound flights. Used Distributed Cache to get the State details by the origin airport code.

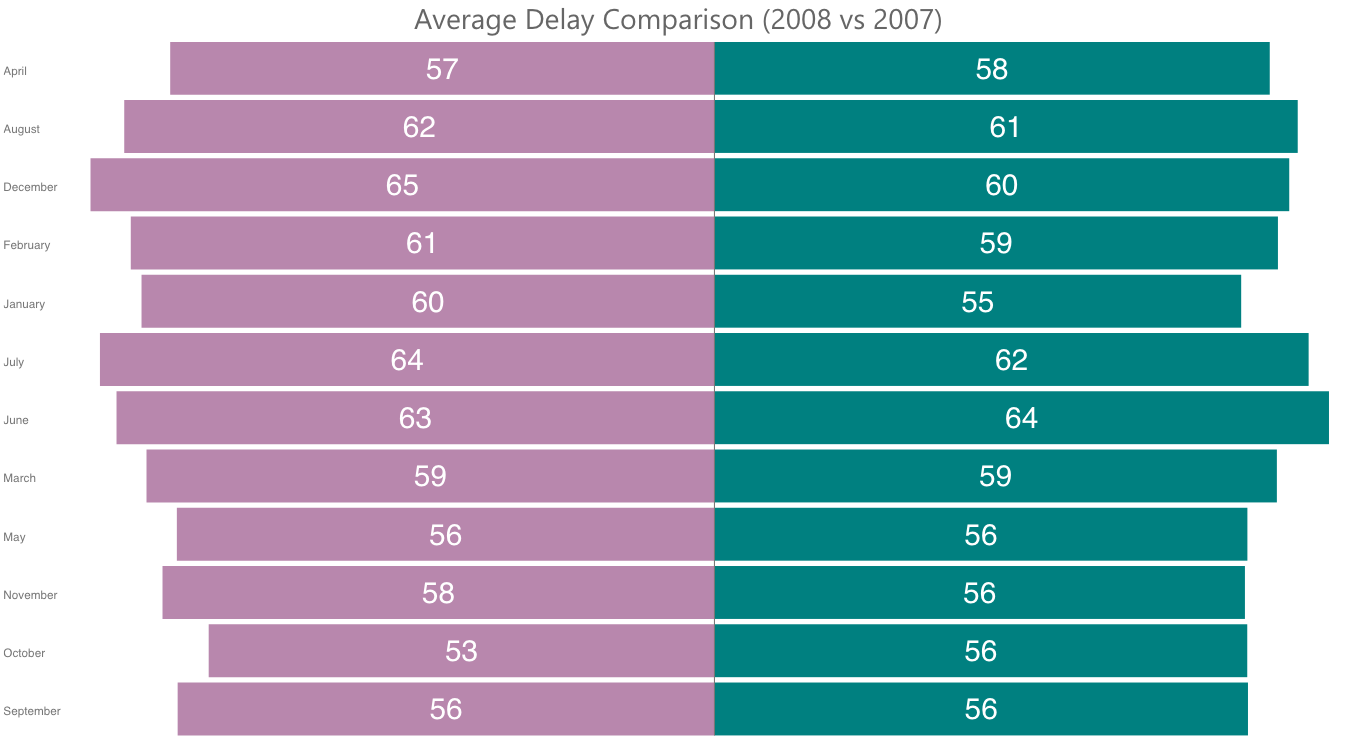
## **Design Pattern and Technology**

* Replicated Join, Numerical Summarization: Average
* Map Reduce, Distributed Cache, Power Bi

# **Average Departure Delay for each Month by Year**

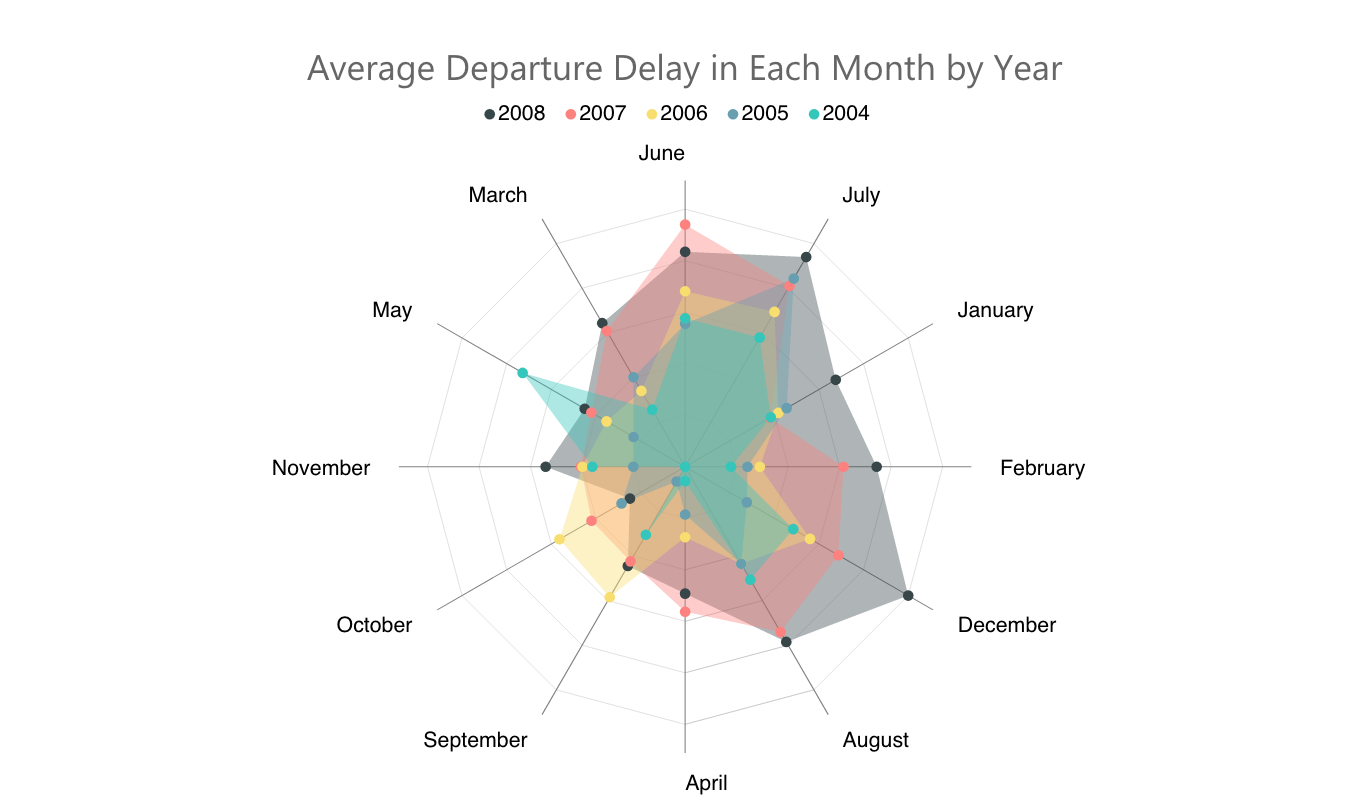
## **Analysis**

Which month has been the worse in terms of flight delay for the current year?



Conclusion: This chart shows the comparison of airline departure delay for each month in 2008 and 2007. December is the month with maximum average delay followed by June, July and August.   
For 2007, June has been the worst with maximum average departure delay.  
Probably, worst weather in December could have impacted the maximum flight departures

Is there a common pattern in the flight Departure Delay in all the years?



*Conclusion*: A radar chart is a graphical method of displaying Average departure delay in the form of a two-dimensional chart with average delay being quantitative variables represented on axes starting from the same point  
Analysis: August, June, July and December shows increase in flight departure delay.   
Summer vacation and winter weather could be reasons impacting the flight delays

## **Approach**

Used Job chaining to calculate the Average Departure delay for each month.First job calculates the Average departure delay for each month by computing the Average delay. The year and the month is taken as a primary key. Second job (Only mapper) uses the Binning pattern to separate the files in 5 different years with each file have all the months and their average for that year (bins).

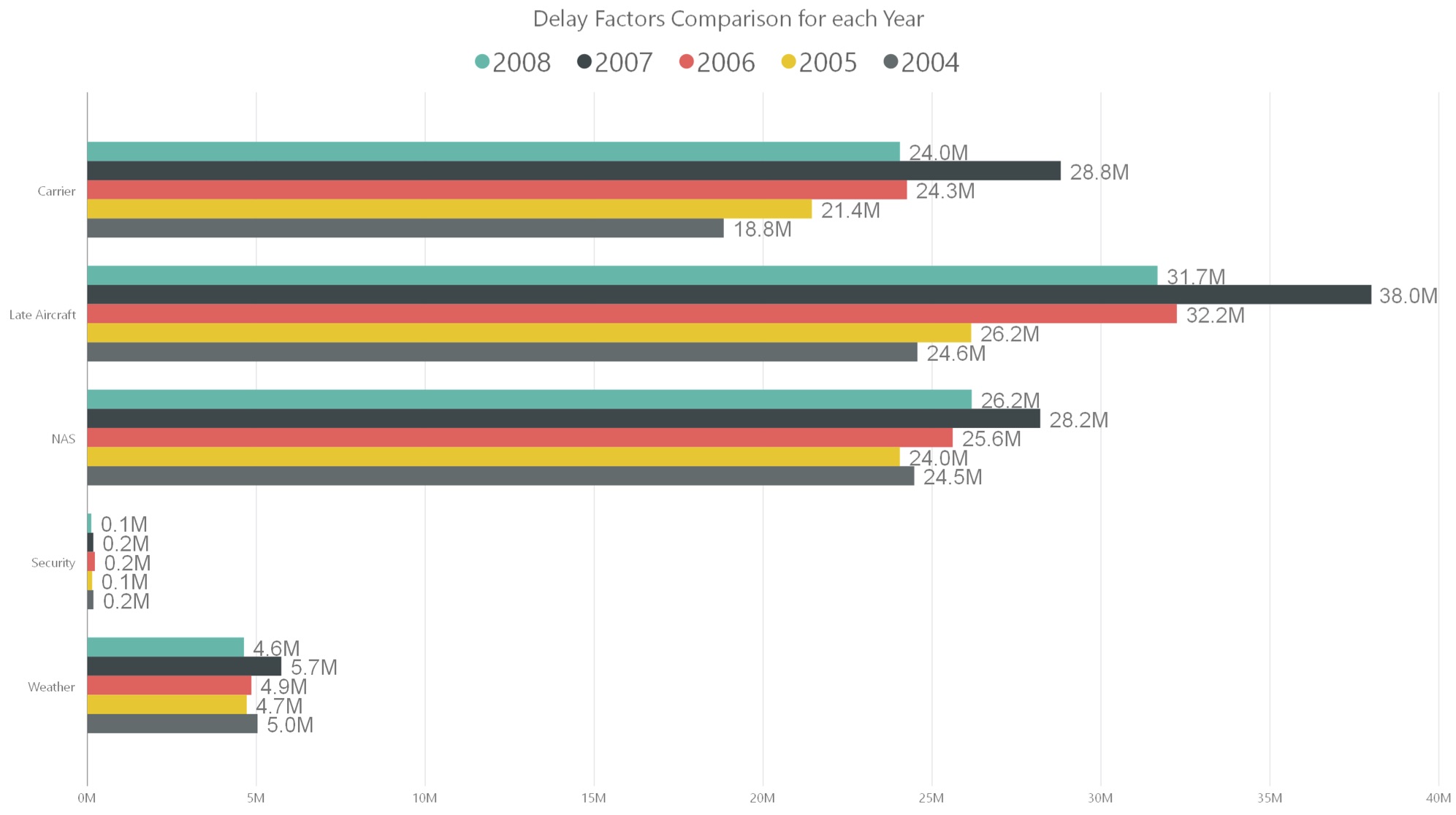
## **Design Pattern and Technologies**

* Data Organization Pattern Binning Pattern, Numerical Summarization Average, Secondary Sorting
* Map Reduce, Power BI

# **Distribution of delay by type**

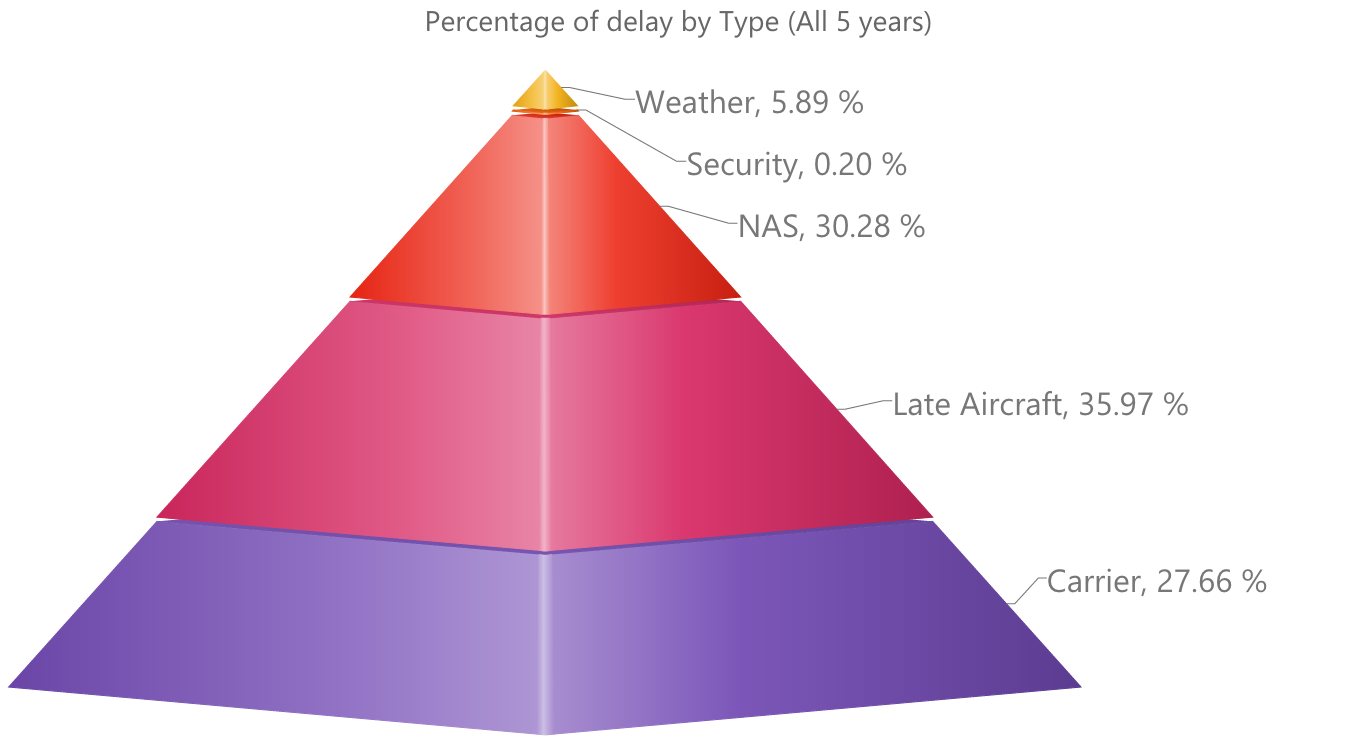
## **Analysis**

Which Year has been the worst causing maximum airlines to be delayed due to Weather?



Conclusion: From the graph, we can deduce that 2007 had highest number of flights delayed because of weather as compared to other years. We can also see that 2007 also had maximum Late Aircrafts. Weather could be one of the contributing factors towards late aircraft delay in year 2007

What are the different types of delays impacting the flight's scheduled departure time?



Conclusion: The pyramid shows that delay distribution by delay type. The height of each layer shows that average delay by particular type. The base represents the number of times the aircraft has been delayed because of that particular factor (delay type)  
Analysis: Late Aircraft has maximum average delay in last five years hence shown with maximum height. It contributes about 35.97% of the total delay   
The carrier delay has the maximum base which means that maximum number of flights were impacted with this type of delay

## **Approach**

Calculated the total delay for each type and stored the output with key as year and value as writable class. Also, calculated the total delay for each year to compute the percentage

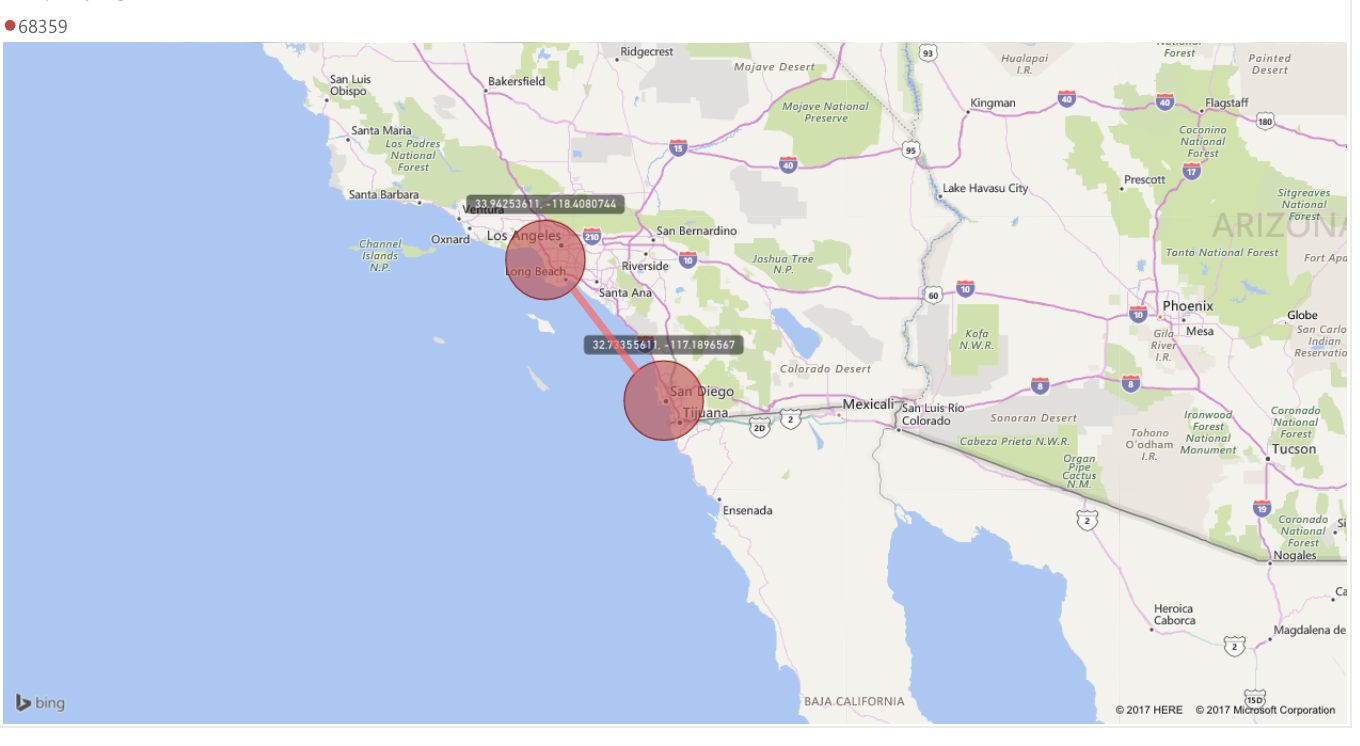
## **Design Pattern and Technologies:**

* Numerical Summarization Count, Composite Value
* Map Reduce, Power BI

# **Finding the Busiest route and identifying the airport connections**

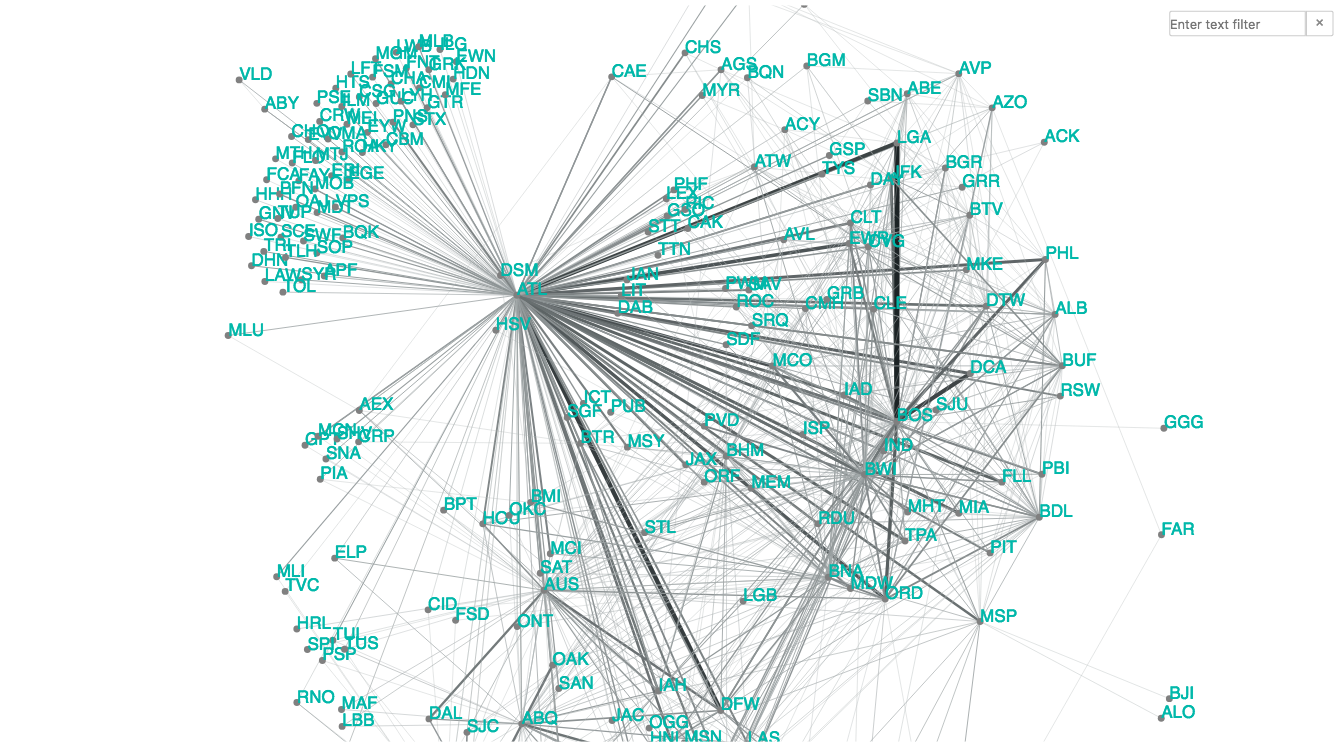
## **Analysis**

Which is Busiest Route?



Conclusion: The locations on the map shows the busiest route by flight volume. This analysis can be used to understand the air traffic and congestion on this route  
Locations:  
Los Angeles International Airport  
San Diego International-Lindberg  
Flight Count:  
68359

What are the possible options of flying from one destination to another?



Conclusion: The network navigator shows the connections of airports. Nodes represent the airports (by iata code) and edges represents the routes(connections).  
This analysis can be used to identify the connection between the airports   
Thickness of the route is depends on number of flights on that route

## **Approach**

Two different programs are used.  
1. Store Airport records: This is used to store the airport records in the Hbase.  
2. Busy Routes  
Used job chaining to find the busy routes. First Program reads the records from the file for 2008 and uses the count operation to calculate the count on each route. The key is taken as the the source and the destination airport codes. The second job reads the input records and finds the Most busiest route based on number of flights on that route. Also , the Mapper reads the records from HBase for getting the Source and destination details (Latitude Longitude etc. ).

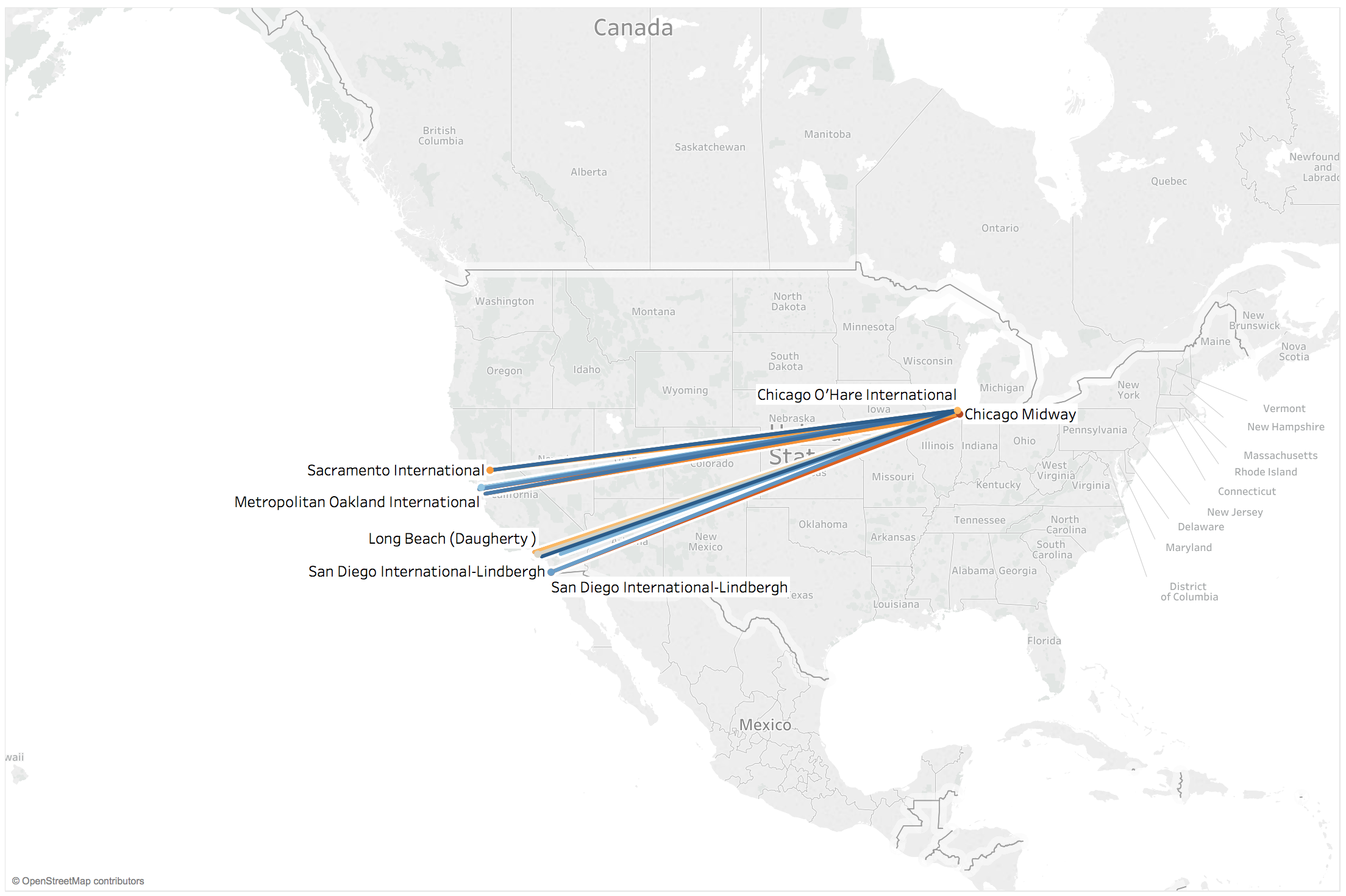
## **Design patterns and Technologies**

* Filtering Pattern: Top record, Numerical Summarization Count, Random Read from HBase
* Map reduce, Hbase, Power BI

# **Different options (Direct Flight) to fly from Illinois to California**

## **Analysis**

What are the different options (direct flight) to fly from one state to another?



Conclusion: The above map shows different possible options on a single day of the month for flying from one state to another from different airport. One can choose the convenient flight by choosing to travel to nearby airport (within Illinois ) to fly to specific destination (California).There could be multiple reasons for choosing different airport to fly:  
1.Cost  
2.Time  
3.Convinience  
Above graph shows all possible flights scheduled from nearby airports within Illinois to California for one specific day of the month

## **Approach**

Used two Programs for this Operation.  
First program produces the Filtered Data for the 1 day and stores it in hdfs. Then it uses second job for joining the Airport details file and the Actual file to get the Airport details.  
The second program reads the input from the previous code and finds all flights between California and Illinois. It reads all the values for getting the State detail from the distributed Cache.

## **Design Pattern and Technologies**

* Summarization Pattern Count, joining Pattern Reduce Side Join
* Map reduce, Distributed Cache, Tableau

# **Are older Aircraft prone to more delays then the Newer Aircraft?**

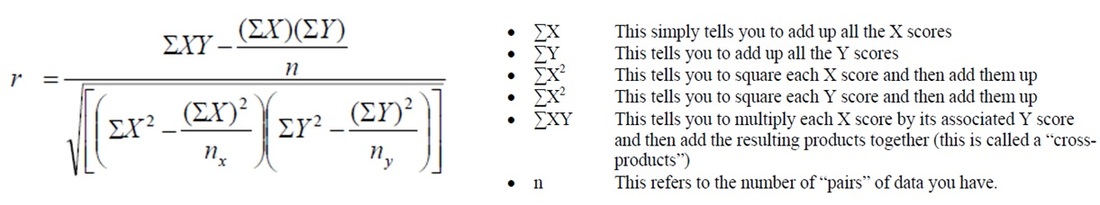
## **Analysis**

Are older Aircraft prone to more delays then the Newer Aircraft?

Pearson Correlation:

Following analysis uses Pearson Correlation to identify the relation between the aircraft's manufacturing year and the flight delays.

Pearson Correlation coefficient is the measure of linear correlation between two variables X and Y where X is the Year of manufacture and Y is the Average delay for all the aircrafts manufactured in that Year.   
First Map Reduce job reads each line of the record and uses the distributed cache to get the year of manufacture for the flights by Tail Number and calculates the average of delays for all the flights manufactured in that year. Second job takes the input from the first job and Calculates the Pearson Correlation coefficient.



The relationship between two factors is identified by following:  
0.1 < |r| < 0.3 => small correlation  
0.3 < |r| < 0.5 => Moderate correlation  
|r| > 0.5 => strong correlation

*Conclusion:* The result obtained for finding the relation between the Year of manufacture for a particular flight and the Average delay for all the Aircraft manufactured in that year is 0.2896.   
With this result, we can deduce that the Older or Newer aircrafts do not have much relation with Flight delays.

## **Approach:**

It uses Job chain for implementing this approach.  
First Map Reduce job reads each line of the record and uses the distributed cache to get the year of manufacture for the flights by Tail Number and calculates the average of delays for all the flights manufactured in that year. Second job takes the input from the first job and Calculates the Pearson Correlation coefficient.

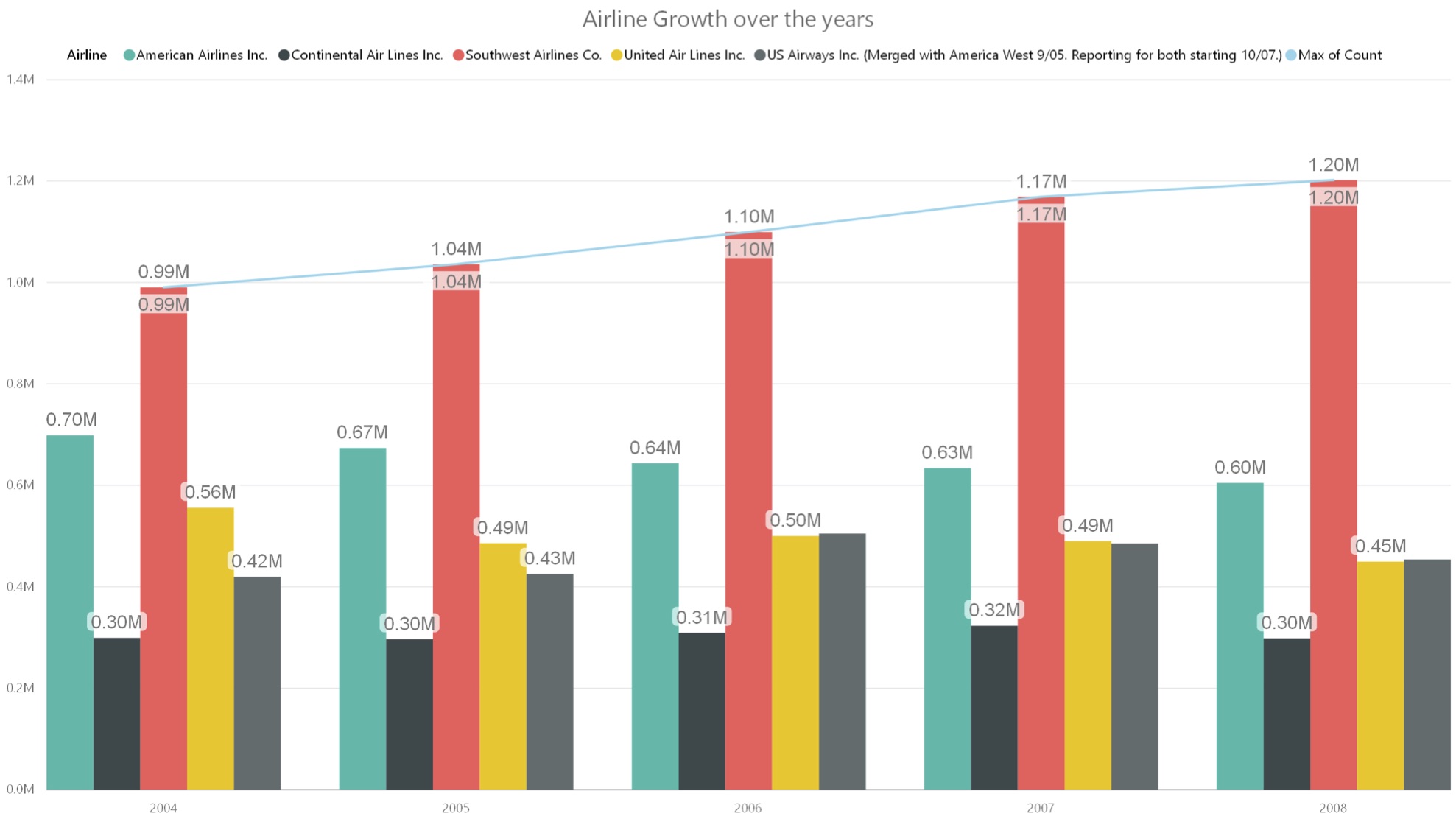
## **Design Pattern and Technologies:**

* Pearson Correlation, Joining Pattern Replicated Join
* Map reduce

# **Carrier Popularity**

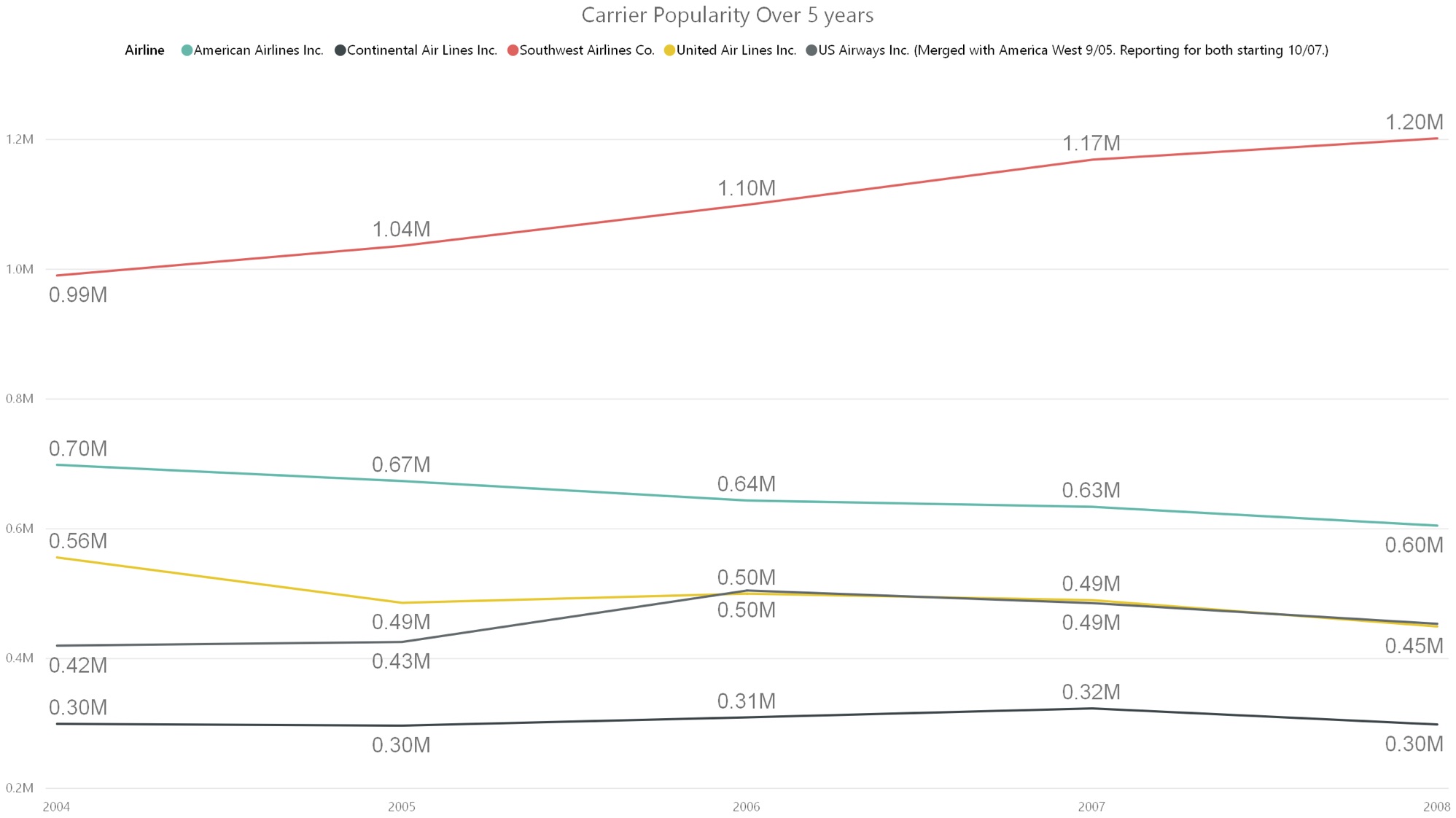
## **Analysis**

Which is the most preferred airline?



Conclusion: The line graph shows the comparison of 5 different airlines in each year. With the help of this graph, we can deduce that Southwest is the most preferred airline in all 5 years (2004-2008). In 2008, the number of scheduled flights has gone upto 1.20 M which is quite high when compared to other four airlines.  
It clearly states that southwest has been leading the airlines industry since 2004. It also shows that south west has shown continuos increase throught out

Which carrier is most popular in each year?



Conclusion: From the line graph, we can see that southwest is the most popular airline in all five year.   
American airline shows a continuous decrease in its scheduled flights over the years

## **Approach**

Cleaned data using map reduce to remove the remove the rows which has missing information.

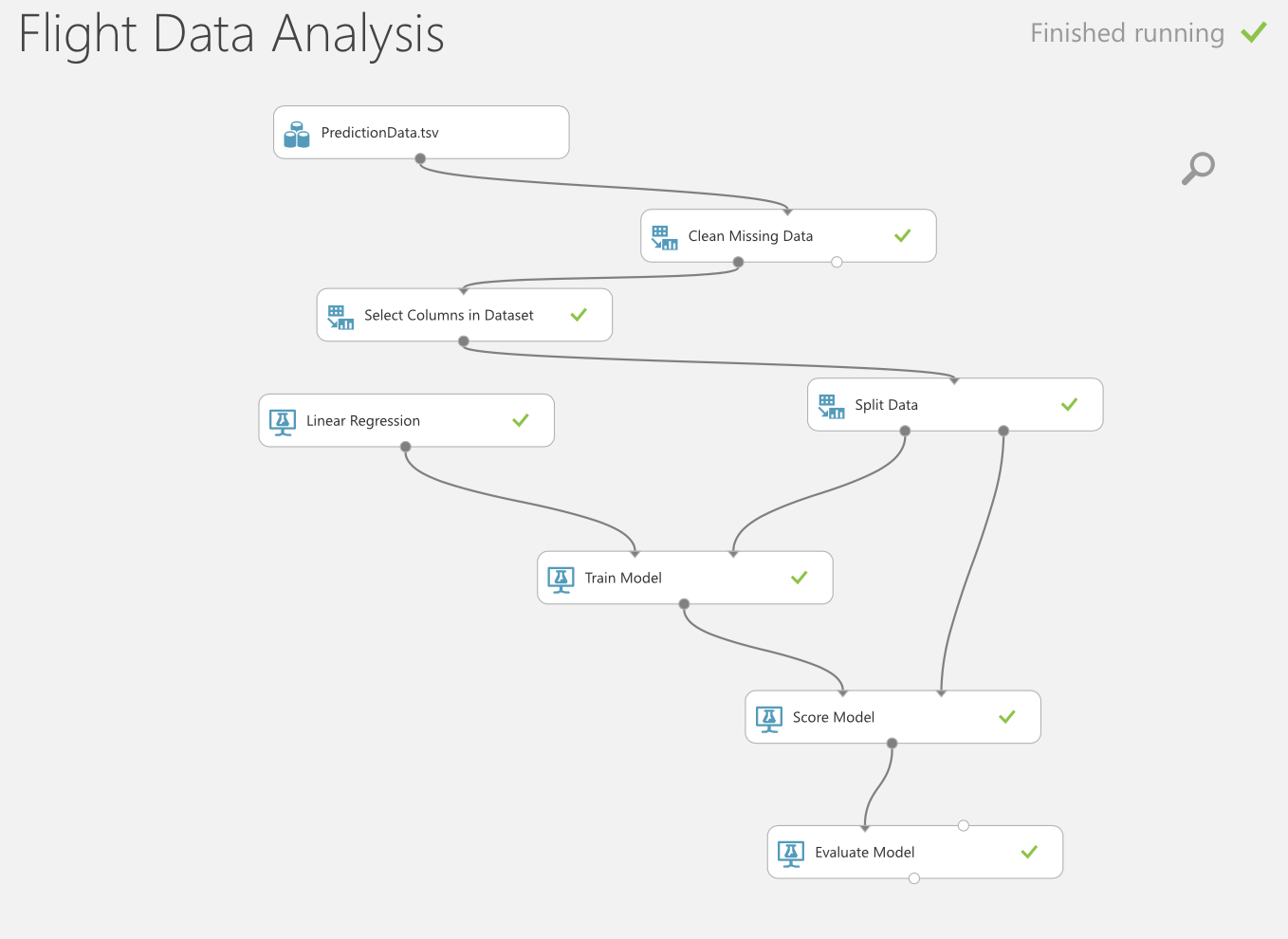
Only one map reduce job runs on for this analysis. The Mapper uses the Composite Key to store the year and Carrier.  
The reducer calculates the total of all and finds the total flight count for each carrier in each year. The carrier details are retrieved from the distributed cache.

## **Design Pattern and Technologies**

* Joining Pattern replicated Join
* Map reduce, Distributed Cache, Power BI

# **Finding the approximate delay for a flight**

* 1. **Prediction**  
     With the help of this prediction, we can determine the approximate delay for a particular flight. The prediction is performed by using the data set which has been produced by a map reduce job. Following factors have been taken into consideration for prediction:  
     1. Flight Number  
     2. Day of the month  
     3. Month  
     4. Delay for that flight on that particular day



Model: Training Data: 75% of whole  
Testing Data: 25% of whole  
Regression Technique: Linear Regression  
  
Linear Regression is an approach for modeling the relationship between a scalar dependent variable Y which is Average Delay and one or more explanatory variables (Flight Number, Day of the Month, Month) denoted X.

* 1. **Approach**

Used Map reduce code to clean the data. This is done by using a Composite key and Composite value. Identity reducer is used to write the data to the output file in the expected format. This Output is fed to the prediction model in Microsoft Azure

* 1. **Design Pattern:**
* Secondary Sorting

# **References**

* h[ttps://mindtrove.info/exploration-of-airline-on-time-performance/](https://mindtrove.info/exploration-of-airline-on-time-performance/)
* <https://www.slideshare.net/mingxuanli587/final-37296711>
* <https://rpubs.com/joshuapoirier/aircraftontime>
* <http://stat-computing.org/dataexpo/2009/posters/>
* <https://hbase.apache.org>
* <https://hadoop.apache.org>
* <https://pig.apache.org>
* http://barbie.uta.edu/~jli/Resources/MapReduce&Hadoop/MapReduce%20Design%20Patterns.pdf

# **Code Base**

1. **Top10BusyCity**

Code:

/\*

\* 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 top10busycity;

import java.io.IOException;

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.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;

/\*\*

\*

\* @author pratik

\*/

public class Top10BusyCity {

/\*\*

\* @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, "Total Number of inbound flights");

job.setJarByClass(Top10BusyCity.class);

job.setMapperClass(Top10\_Mapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

job.setReducerClass(Top10\_Reducer.class);

// job.setNumReduceTasks(1);

job.addCacheFile(new Path(args[0]).toUri());

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

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

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

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

if(job.waitForCompletion(true)){

Job job2 = Job.getInstance(conf, "Total Number of outbound flights");

job2.setJarByClass(Top10BusyCity.class);

job2.setMapperClass(Top10\_Mapper\_Outbound.class);

job2.setMapOutputKeyClass(Text.class);

job2.setMapOutputValueClass(IntWritable.class);

job2.setReducerClass(Top10\_Reducer.class);

job2.addCacheFile(new Path(args[0]).toUri());

// job2.setCombinerClass(Top10\_Reducer.class);

// job2.setNumReduceTasks(1);

job2.setOutputKeyClass(Text.class);

job2.setOutputValueClass(IntWritable.class);

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

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

if(job2.waitForCompletion(true)){

Job job3 = Job.getInstance(conf, "Total Number of flights");

job3.setJarByClass(Top10BusyCity.class);

MultipleInputs.addInputPath(job3, new Path(args[2]),TextInputFormat.class, Top10\_Mapper\_Combine.class);

MultipleInputs.addInputPath(job3, new Path(args[3]),TextInputFormat.class,Top10\_Mapper\_Combine.class);

job3.setMapOutputKeyClass(Text.class);

job3.setMapOutputValueClass(IntWritable.class);

job3.setReducerClass(Top10\_Reducer\_Combine.class);

//job3.setCombinerClass(Top10\_Reducer.class);

job3.setNumReduceTasks(1);

job3.setOutputKeyClass(Text.class);

job3.setOutputValueClass(IntWritable.class);

FileOutputFormat.setOutputPath(job3, new Path(args[4]));

System.exit(job3.waitForCompletion(true)?1:0);

// if(job3.waitForCompletion(true)){

//

// Job job4 = Job.getInstance(conf, "Top 10 busy city");

// job4.setJarByClass(Top10BusyCity.class);

// job4.setMapperClass(Top10\_Mapper\_Calculate.class);

// job4.setMapOutputKeyClass(Top10CompositeKeyWritable.class);

// job4.setMapOutputValueClass(NullWritable.class);

// job4.setReducerClass(Top10\_Reducer\_Calculate.class);

// //job3.setCombinerClass(Top10\_Reducer.class);

// // job4.setNumReduceTasks(1);

// job4.addCacheFile(new Path(args[5]).toUri());

// job4.setOutputKeyClass(Top10CompositeKeyWritable.class);

// job4.setOutputValueClass(NullWritable.class);

// FileInputFormat.addInputPath(job4,new Path(args[3]));

// FileOutputFormat.setOutputPath(job4, new Path(args[4]));

// System.exit(job4.waitForCompletion(true)? 0:1);

// }

}

}

}

}

/\*

\* 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 top10busycity;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

/\*\*

\*

\* @author pratik

\*/

public class Top10CompositeKeyWritable implements Writable,WritableComparable<Top10CompositeKeyWritable>{

private String city;

private Integer totalNumberOfFlights;

public String getCity() {

return city;

}

public void setCity(String city) {

this.city = city;

}

public Integer getTotalNumberOfFlights() {

return totalNumberOfFlights;

}

public void setTotalNumberOfFlights(Integer totalNumberOfFlights) {

this.totalNumberOfFlights = totalNumberOfFlights;

}

@Override

public void write(DataOutput d) throws IOException {

d.writeUTF(city);

d.writeInt(totalNumberOfFlights);

}

@Override

public void readFields(DataInput di) throws IOException {

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

city = di.readUTF();

totalNumberOfFlights = di.readInt();

}

@Override

public int compareTo(Top10CompositeKeyWritable o) {

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

int result =(-1)\*totalNumberOfFlights.compareTo(o.totalNumberOfFlights);

// if(result==0){

// result = lastName.compareTo(o.lastName);

// }

return result;

}

@Override

public String toString() {

return city + " " + totalNumberOfFlights ;

}

}

/\*

\* 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 top10busycity;

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;

/\*\*

\*

\* @author pratik

\*/

public class Top10\_Mapper extends Mapper<LongWritable,Text,Text,IntWritable>{

private Text outputKey = new Text();

private IntWritable outputValue = new IntWritable();

@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 inputArray[] = value.toString().split(",");

if(inputArray.length<29){

System.out.print("Some columns are missing for the record");

}else if (inputArray[0].equals("Year")){

System.out.print("Reading the Column name record");

}else{

outputKey.set(inputArray[16]);

outputValue.set(1);

context.write(outputKey,outputValue);

}

}

}

/\*

\* 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 top10busycity;

import java.io.IOException;

import java.util.TreeMap;

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.Mapper;

/\*\*

\*

\* @author pratik

\*/

public class Top10\_Mapper\_Calculate extends Mapper<LongWritable,Text,Top10CompositeKeyWritable,NullWritable>{

private Top10CompositeKeyWritable outputKey = new Top10CompositeKeyWritable();

@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 inputArray[] = value.toString().split("\\t");

outputKey.setCity(inputArray[0]);

outputKey.setTotalNumberOfFlights(Integer.parseInt(inputArray[1]));

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

}

}

/\*

\* 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 top10busycity;

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;

/\*\*

\*

\* @author pratik

\*/

public class Top10\_Mapper\_Combine extends Mapper<LongWritable,Text,Text,IntWritable>{

private Text outputKey = new Text();

private IntWritable outputValue = new IntWritable();

@Override

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

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

outputKey.set(inputArray[0]);

outputValue.set(Integer.parseInt(inputArray[1]));

context.write(outputKey, outputValue);

}

}

/\*

\* 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 top10busycity;

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;

/\*\*

\*

\* @author pratik

\*/

public class Top10\_Mapper\_Outbound extends Mapper<LongWritable,Text,Text,IntWritable>{

private Text outputKey = new Text();

private IntWritable outputValue = new IntWritable();

@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 inputArray[] = value.toString().split(",");

if(inputArray.length<29){

System.out.print("Some columns are missing for the record");

}else if (inputArray[0].equals("Year")){

System.out.print("Reading the Column name record");

}else{

outputKey.set(inputArray[17]);

outputValue.set(1);

context.write(outputKey,outputValue);

}

}

}

/\*

\* 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 top10busycity;

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.HashMap;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author pratik

\*/

public class Top10\_Reducer extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new IntWritable();

//private boolean flag = false;

private Text outputkey = new Text();

private static HashMap<String, String> airports = new HashMap<String, String>();

private BufferedReader brReader;

@Override

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

Path[] cacheFilesLocal = context.getLocalCacheFiles();

System.out.print("The lentgh is "+cacheFilesLocal.length);

for (Path eachPath : cacheFilesLocal) {

if (eachPath.getName().toString().trim().equals("airports.csv")) {

// context.getCounter(MYCOUNTER.FILE\_EXISTS).increment(1);

loadAirportsHashMap(eachPath, context);

}

}

}

private void loadAirportsHashMap(Path filePath, Context context) throws IOException {

String strLineRead = "";

try {

brReader = new BufferedReader(new FileReader(filePath.toString()));

// Read each line, split and load to HashMap

while ((strLineRead = brReader.readLine()) != null) {

System.out.println("Inside buffer");

String deptFieldArray[] = strLineRead.replace("\"", "").split(",");

System.out.println("The value is"+deptFieldArray[0]);

if(!deptFieldArray[0].equals("iata")){

String inValue=deptFieldArray[1].trim()+"\t"+deptFieldArray[5].trim()+"\t"+deptFieldArray[6].trim();

airports.put(deptFieldArray[0].trim(),inValue);

}

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}finally {

if (brReader != null) {

brReader.close();

}

}

}

@Override

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

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

int sum=0;

for(IntWritable val:values){

sum =sum+1;

}

if(airports.containsKey(key.toString())){

outputkey.set(airports.get(key.toString()));

result.set(sum);

context.write(outputkey, result);

}

}

}

/\*

\* 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 top10busycity;

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.TreeMap;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author pratik

\*/

public class Top10\_Reducer\_Calculate extends Reducer<Top10CompositeKeyWritable,NullWritable,Top10CompositeKeyWritable,NullWritable>{

private Top10CompositeKeyWritable result = new Top10CompositeKeyWritable();

private int count=0;

private static HashMap<String, String> airports = new HashMap<String, String>();

private BufferedReader brReader;

@Override

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

Path[] cacheFilesLocal = context.getLocalCacheFiles();

System.out.print("The lentgh is "+cacheFilesLocal.length);

for (Path eachPath : cacheFilesLocal) {

if (eachPath.getName().toString().trim().equals("airports.csv")) {

// context.getCounter(MYCOUNTER.FILE\_EXISTS).increment(1);

loadAirportsHashMap(eachPath, context);

}

}

}

private void loadAirportsHashMap(Path filePath, Context context) throws IOException {

String strLineRead = "";

try {

brReader = new BufferedReader(new FileReader(filePath.toString()));

// Read each line, split and load to HashMap

while ((strLineRead = brReader.readLine()) != null) {

System.out.println("Inside buffer");

String deptFieldArray[] = strLineRead.replace("\"", "").split(",");

System.out.println("The value is"+deptFieldArray[0]);

if(!deptFieldArray[0].equals("iata")){

airports.put(deptFieldArray[0].trim(),deptFieldArray[1].trim());

}

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}finally {

if (brReader != null) {

brReader.close();

}

}

}

@Override

protected void reduce(Top10CompositeKeyWritable key, Iterable<NullWritable> values, Context context) throws IOException, InterruptedException {

count++;

if(count<11) {

result.setCity(airports.get(key.getCity()));

result.setTotalNumberOfFlights(key.getTotalNumberOfFlights());

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

}

}

}

/\*

\* 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 top10busycity;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author pratik

\*/

public class Top10\_Reducer\_Combine extends Reducer<Text,IntWritable,Text,IntWritable>{

private IntWritable result = new IntWritable();

@Override

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

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

int sum=0;

for(IntWritable val:values){

sum = sum + val.get();

}

result.set(sum);

context.write(key, result);

}

}

Pig Code:

runt> records = LOAD '/Users/pratik/Downloads/testingPig' USING PigStorage('\t') as (airport:chararray,counting:int);

grunt> order\_count = order records by counting DESC;

grunt> top10 = limit order\_count 10;

grunt> dump top10;

1. **TotalFlightAndAverage**

Code:

/\*

\* 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 totalflightandaveragebystate;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

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;

/\*\*

\*

\* @author pratik

\*/

public class TotalFlightAndAverageByState {

/\*\*

\* @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, "Average By State");

job.setJarByClass(TotalFlightAndAverageByState.class);

job.setMapperClass(Mapper\_State.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(CompositeValue\_State.class);

job.addCacheFile(new Path(args[0]).toUri());

job.setReducerClass(Reducer\_State.class);

// job.setNumReduceTasks(3);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(CompositeValue\_State.class);

// job.setPartitionerClass(AverageDelay\_Partitioner.class);

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

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

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

}

}

/\*

\* 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 totalflightandaveragebystate;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

/\*\*

\*

\* @author pratik

\*/

public class CompositeValue\_State implements Writable{

private Integer count;

private Double arrivalDelay;

public Integer getCount() {

return count;

}

public void setCount(Integer count) {

this.count = count;

}

public Double getArrivalDelay() {

return arrivalDelay;

}

public void setArrivalDelay(Double arrivalDelay) {

this.arrivalDelay = arrivalDelay;

}

@Override

public void write(DataOutput d) throws IOException {

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

d.writeInt(count);

d.writeDouble(arrivalDelay);

}

@Override

public void readFields(DataInput di) throws IOException {

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

count = di.readInt();

arrivalDelay = di.readDouble();

}

@Override

public String toString() {

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

return (new StringBuilder().append(arrivalDelay).append("\t").append(count).toString());

}

}

/\*

\* 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 totalflightandaveragebystate;

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.HashMap;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author pratik

\*/

public class Mapper\_State extends Mapper<LongWritable,Text,Text,CompositeValue\_State>{

private Text outKey = new Text();

private CompositeValue\_State outValue= new CompositeValue\_State();

private static HashMap<String, String> airports = new HashMap<String, String>();

private BufferedReader brReader;

@Override

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

Path[] cacheFilesLocal = context.getLocalCacheFiles();

System.out.print("The lentgh is "+cacheFilesLocal.length);

for (Path eachPath : cacheFilesLocal) {

if (eachPath.getName().toString().trim().equals("airports.csv")) {

// context.getCounter(MYCOUNTER.FILE\_EXISTS).increment(1);

loadAirportsHashMap(eachPath, context);

}

}

}

private void loadAirportsHashMap(Path filePath, Context context) throws IOException {

String strLineRead = "";

try {

brReader = new BufferedReader(new FileReader(filePath.toString()));

// Read each line, split and load to HashMap

while ((strLineRead = brReader.readLine()) != null) {

System.out.println("Inside buffer");

String deptFieldArray[] = strLineRead.replace("\"", "").split(",");

if(!deptFieldArray[0].equals("iata")){

//String inValue=deptFieldArray[1].trim()+"\t"+deptFieldArray[5].trim()+"\t"+deptFieldArray[6].trim();

System.out.println("The key value is"+deptFieldArray[0]);

System.out.println("The out value is"+deptFieldArray[3]);

airports.put(deptFieldArray[0].trim(),deptFieldArray[3].trim());

}

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}finally {

if (brReader != null) {

brReader.close();

}

}

}

@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 inputString[] = value.toString().split(",");

if(inputString.length<29||inputString[14].equals("NA")){

System.out.println("Reading Columns");

}else{

if(airports.containsKey(inputString[17])){

String state = airports.get(inputString[17]) ;

outKey.set(state);

outValue.setArrivalDelay(Double.parseDouble(inputString[14]));

outValue.setCount(1);

context.write(outKey, outValue);

}

}

}

}

/\*

\* 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 totalflightandaveragebystate;

import java.io.IOException;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author pratik

\*/

public class Reducer\_State extends Reducer<Text,CompositeValue\_State,Text,CompositeValue\_State>{

private CompositeValue\_State result = new CompositeValue\_State();

@Override

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

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

Double sum =0.0;

Integer count =0;

for(CompositeValue\_State val :values){

sum = sum +val.getArrivalDelay();

count= count+val.getCount();

}

result.setArrivalDelay(sum/count);

result.setCount(count);

context.write(key, result);

}

}

1. **Average Departure Delay for each Month by Year**

Code:

/\*

\* 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 averagedelaybymonthbyyear;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.NullWritable;

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

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

/\*\*

\*

\* @author pratik

\*/

public class AverageDelayByMonthByYear {

/\*\*

\* @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, "Average Delay in Each Month by Year");

job.setJarByClass(AverageDelayByMonthByYear.class);

job.setMapperClass(Mapper\_AverageDelay.class);

job.setMapOutputKeyClass(CompositeKey\_AverageDelay.class);

job.setMapOutputValueClass(CompositeValue\_AverageDelay.class);

// job.setGroupingComparatorClass(AverageDelay\_GroupComparator.class);

// job.setCombinerClass(AverageDelay\_Reducer.class);

job.setReducerClass(Reducer\_AverageDelay.class);

// job.setNumReduceTasks(3);

job.setOutputKeyClass(CompositeKey\_AverageDelay.class);

job.setOutputValueClass(CompositeValue\_AverageDelay.class);

// job.setPartitionerClass(AverageDelay\_Partitioner.class);

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

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

boolean jobComplete =job.waitForCompletion(true);

if(jobComplete){

Job job2 = Job.getInstance(conf, "Distribution of files by Year");

job2.setJarByClass(AverageDelayByMonthByYear.class);

job2.setMapperClass(Mapper\_Bins.class);

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

MultipleOutputs.setCountersEnabled(job2, true);

job2.setNumReduceTasks(0);

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

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

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

}

}

}

/\*

\* 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 averagedelaybymonthbyyear;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

/\*\*

\*

\* @author pratik

\*/

public class CompositeKey\_AverageDelay implements Writable,WritableComparable<CompositeKey\_AverageDelay>{

private Integer year;

private String month;

public Integer getYear() {

return year;

}

public void setYear(Integer year) {

this.year = year;

}

public String getMonth() {

return month;

}

public void setMonth(String month) {

this.month = month;

}

@Override

public void write(DataOutput d) throws IOException {

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

d.writeInt(year);

d.writeUTF(month);

}

@Override

public void readFields(DataInput di) throws IOException {

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

year = di.readInt();

month = di.readUTF();

}

@Override

public int compareTo(CompositeKey\_AverageDelay o) {

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

int result = -(1)\*year.compareTo(o.year);

if(result ==0){

result = month.compareTo(o.month);

}

return result;

}

@Override

public String toString() {

// return "CompositeKey\_AverageDelay{" + "year=" + year + ", month=" + month + '}';

return (new StringBuilder().append(year).append("\t").append(month).toString());

}

}

/\*

\* 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 averagedelaybymonthbyyear;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

/\*\*

\*

\* @author pratik

\*/

public class CompositeValue\_AverageDelay implements Writable{

private Integer count;

private Double averageDelay;

public Integer getCount() {

return count;

}

public void setCount(Integer count) {

this.count = count;

}

public Double getAverageDelay() {

return averageDelay;

}

public void setAverageDelay(Double averageDelay) {

this.averageDelay = averageDelay;

}

@Override

public void write(DataOutput d) throws IOException {

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

d.writeInt(count);

d.writeDouble(averageDelay);

}

@Override

public void readFields(DataInput di) throws IOException {

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

count = di.readInt();

averageDelay = di.readDouble();

}

@Override

public String toString() {

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

return (new StringBuilder().append(averageDelay).toString());

}

}

/\*

\* 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 averagedelaybymonthbyyear;

import java.io.IOException;

import java.util.HashMap;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author pratik

\*/

public class Mapper\_AverageDelay extends Mapper<LongWritable,Text,CompositeKey\_AverageDelay,CompositeValue\_AverageDelay>{

private CompositeKey\_AverageDelay outKey = new CompositeKey\_AverageDelay();

private CompositeValue\_AverageDelay outValue = new CompositeValue\_AverageDelay();

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

@Override

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

months.put(1, "January");

months.put(2, "February");

months.put(3, "March");

months.put(4, "April");

months.put(5, "May");

months.put(6, "June");

months.put(7, "July");

months.put(8, "August");

months.put(9, "September");

months.put(10,"October");

months.put(11,"November");

months.put(12,"December");

}

@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 inputArray[] = value.toString().split(",");

if(inputArray.length<29||inputArray[0].equals("Year")){

System.out.println("Reading columns");

}else if(inputArray[0].equals("")||inputArray[1].equals("")||inputArray[1].equals("NA")){

System.out.println("Invalid columns");

}else if((!inputArray[21].equals("1"))&&(!inputArray[15].equals("NA"))){

String outMonth = months.get(Integer.parseInt(inputArray[1]));

if(Double.parseDouble(inputArray[15])>15.0){

//setting the put value with average and count

outValue.setAverageDelay(Double.parseDouble(inputArray[15]));

outValue.setCount(1);

outKey.setMonth(outMonth);

outKey.setYear(Integer.parseInt(inputArray[0]));

context.write(outKey, outValue);

}

}

}

}

/\*

\* 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 averagedelaybymonthbyyear;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

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

/\*\*

\*

\* @author pratik

\*/

public class Mapper\_Bins extends Mapper<LongWritable,Text,Text,NullWritable>{

private MultipleOutputs<Text, NullWritable> mos = null;

private Text inputHour = new Text();

@Override

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

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

mos = new MultipleOutputs(context);

}

@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 inputArray[]= value.toString().split("\\t");

if(inputArray[0].equals("2008")){

mos.write("bins", value, NullWritable.get(),"2008");

}else if(inputArray[0].equals("2007")){

mos.write("bins", value, NullWritable.get(),"2007");

}else if(inputArray[0].equals("2006")){

mos.write("bins", value, NullWritable.get(),"2006");

}else if(inputArray[0].equals("2005")){

mos.write("bins", value, NullWritable.get(),"2005");

}else if(inputArray[0].equals("2004")){

mos.write("bins", value, NullWritable.get(),"2004");

}

}

@Override

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

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

mos.close();

}

}

/\*

\* 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 averagedelaybymonthbyyear;

import java.io.IOException;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author pratik

\*/

public class Reducer\_AverageDelay extends Reducer<CompositeKey\_AverageDelay,CompositeValue\_AverageDelay,CompositeKey\_AverageDelay,CompositeValue\_AverageDelay>{

private CompositeKey\_AverageDelay resultKey = new CompositeKey\_AverageDelay();

private CompositeValue\_AverageDelay resultValue = new CompositeValue\_AverageDelay();

@Override

protected void reduce(CompositeKey\_AverageDelay key, Iterable<CompositeValue\_AverageDelay> values, Context context) throws IOException, InterruptedException {

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

Double sum =0.0;

int count =0;

for(CompositeValue\_AverageDelay val:values){

sum = sum +val.getAverageDelay();

count = count+val.getCount();

}

resultValue.setAverageDelay(sum/count);

resultValue.setCount(count);

context.write(key, resultValue);

}

}

1. **Distribution of delay by type**

Code:

/\*

\* 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 delayreasonbypercentage;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.mapreduce.Job;

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

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

/\*\*

\*

\* @author pratik

\*/

public class DelayReasonByPercentage {

/\*\*

\* @param args the command line arguments

\*/

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

Configuration conf = new Configuration();

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

job.setJarByClass(DelayPercentage\_CompositeKey.class);

job.setMapperClass(Mapper\_DelayPercentage.class);

job.setMapOutputKeyClass(IntWritable.class);

job.setMapOutputValueClass(DelayPercentage\_CompositeKey.class);

// job.setGroupingComparatorClass(AverageDelay\_GroupComparator.class);

// job.setCombinerClass(AverageDelay\_Reducer.class);

job.setReducerClass(Reducer\_DelayByPercentage.class);

// job.setNumReduceTasks(3);

job.setOutputKeyClass(IntWritable.class);

job.setOutputValueClass(DelayPercentage\_CompositeKey.class);

// job.setPartitionerClass(AverageDelay\_Partitioner.class);

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

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

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

}

}

/\*

\* 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 delayreasonbypercentage;

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;

/\*\*

\*

\* @author pratik

\*/

public class Mapper\_DelayPercentage extends Mapper<LongWritable,Text,IntWritable,DelayPercentage\_CompositeKey>{

IntWritable outKey = new IntWritable();

DelayPercentage\_CompositeKey outValue = new DelayPercentage\_CompositeKey();

@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 inputArray[] = value.toString().split(",");

if(inputArray.length<29){

System.out.println("Invalid Data");

}else if(inputArray[0].equals("Year")||inputArray[0].equals("")){

System.out.println("Reading column headers");

}else if((!inputArray[24].equals("NA"))&&(!inputArray[25].equals("NA"))&&(!inputArray[26].equals("NA"))&&(!inputArray[27].equals("NA"))&&(!inputArray[28].equals("NA"))){

outKey.set(Integer.parseInt(inputArray[0]));

outValue.setCarrierDelay(Integer.parseInt(inputArray[24]));

outValue.setWeatherDelay(Integer.parseInt(inputArray[25]));

outValue.setNasDelay(Integer.parseInt(inputArray[26]));

outValue.setSecurityDelay(Integer.parseInt(inputArray[27]));

outValue.setLateAirDelay(Integer.parseInt(inputArray[28]));

outValue.setTotalDelay(0);

context.write(outKey, outValue);

}

}

}

/\*

\* 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 delayreasonbypercentage;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author pratik

\*/

public class Reducer\_DelayByPercentage extends Reducer<IntWritable,DelayPercentage\_CompositeKey,IntWritable,DelayPercentage\_CompositeKey>{

private DelayPercentage\_CompositeKey result = new DelayPercentage\_CompositeKey();

@Override

protected void reduce(IntWritable key, Iterable<DelayPercentage\_CompositeKey> values, Context context) throws IOException, InterruptedException {

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

result.setCarrierDelay(0);

result.setWeatherDelay(0);

result.setNasDelay(0);

result.setSecurityDelay(0);

result.setLateAirDelay(0);

result.setTotalDelay(0);

for(DelayPercentage\_CompositeKey val:values){

//

// if(result.getCarrierDelay()==null){

// result.setCarrierDelay(val.getCarrierDelay());

// result.setWeatherDelay(val.getWeatherDelay());

// result.setNasDelay(val.getNasDelay());

// result.setSecurityDelay(val.getSecurityDelay());

// result.setLateAirDelay(val.getLateAirDelay());

// result.setTotalDelay(val.getTotalDelay());

// }else{

result.setCarrierDelay(result.getCarrierDelay()+val.getCarrierDelay());

result.setWeatherDelay(result.getWeatherDelay()+val.getWeatherDelay());

result.setNasDelay(result.getNasDelay()+val.getNasDelay());

result.setSecurityDelay(result.getSecurityDelay()+val.getSecurityDelay());

result.setLateAirDelay(result.getLateAirDelay()+val.getLateAirDelay());

result.setTotalDelay(result.getCarrierDelay()+result.getWeatherDelay()+result.getLateAirDelay()+result.getNasDelay()+result.getSecurityDelay());

// }

}

context.write(key, result);

}

}

/\*

\* 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 delayreasonbypercentage;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

/\*\*

\*

\* @author pratik

\*/

public class DelayPercentage\_CompositeKey implements Writable,WritableComparable<DelayPercentage\_CompositeKey>{

private Integer carrierDelay;

private Integer weatherDelay;

private Integer nasDelay;

private Integer securityDelay;

private Integer lateAirDelay;

private Integer totalDelay;

public Integer getTotalDelay() {

return totalDelay;

}

public void setTotalDelay(Integer totalDelay) {

this.totalDelay = totalDelay;

}

public Integer getCarrierDelay() {

return carrierDelay;

}

public void setCarrierDelay(Integer carrierDelay) {

this.carrierDelay = carrierDelay;

}

public Integer getWeatherDelay() {

return weatherDelay;

}

public void setWeatherDelay(Integer weatherDelay) {

this.weatherDelay = weatherDelay;

}

public Integer getNasDelay() {

return nasDelay;

}

public void setNasDelay(Integer nasDelay) {

this.nasDelay = nasDelay;

}

public Integer getSecurityDelay() {

return securityDelay;

}

public void setSecurityDelay(Integer securityDelay) {

this.securityDelay = securityDelay;

}

public Integer getLateAirDelay() {

return lateAirDelay;

}

public void setLateAirDelay(Integer lateAirDelay) {

this.lateAirDelay = lateAirDelay;

}

@Override

public void write(DataOutput d) throws IOException {

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

d.writeInt(carrierDelay);

d.writeInt(weatherDelay);

d.writeInt(nasDelay);

d.writeInt(securityDelay);

d.writeInt(lateAirDelay);

d.writeInt(totalDelay);

}

@Override

public void readFields(DataInput di) throws IOException {

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

carrierDelay = di.readInt();

weatherDelay = di.readInt();

nasDelay = di.readInt();

securityDelay = di.readInt();

lateAirDelay = di.readInt();

totalDelay = di.readInt();

}

@Override

public int compareTo(DelayPercentage\_CompositeKey o) {

int result =0;

return result;

}

@Override

public String toString() {

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

return (new StringBuilder().append(carrierDelay).append("\t").append(weatherDelay).append("\t").append(nasDelay).append("\t").append(securityDelay).append("\t").append(lateAirDelay).append("\t").append(totalDelay).toString());

}

}

1. **Finding the Busiest route and identifying the airport connections**

Code:

/\*

\* 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 storeairportrecordinpig;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.hbase.HBaseConfiguration;

import org.apache.hadoop.hbase.client.Scan;

import org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

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

/\*\*

\*

\* @author pratik

\*/

public class StoreAirportRecordInPig {

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic here

Configuration config = HBaseConfiguration.create();

Job job = new Job(config,"Example");

job.setJarByClass(StoreAirportRecordInPig.class);

Scan scan = new Scan();

scan.setCaching(500);

job.setMapperClass(Mapper\_StoreInPig.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

TableMapReduceUtil.initTableReducerJob("AirportDetails",Reducer\_StoreInPig.class, job);

job.setNumReduceTasks(1);

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

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

}

}

/\*

\* 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 storeairportrecordinpig;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author pratik

\*/

public class Mapper\_StoreInPig extends Mapper<LongWritable,Text,Text,Text>{

private Text outKey = new Text();

private Text outValue = new 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 inputArray[] = value.toString().replace("\"", "").split(",");

if(inputArray.length<7||inputArray[0].replace("\"", "").equals("iata")){

System.out.println("Reading Columns");

}else if(inputArray[0].equals("")||inputArray[1].equals("")){

System.out.println("Reading empty value");

}else{

String valueToStore=inputArray[1].replace("\"", "")+" "+inputArray[5].replace("\"", "")+" "+inputArray[6].replace("\"", "");

outKey.set(inputArray[0].replace("\"", ""));

outValue.set(valueToStore);

context.write(outKey, outValue);

}

}

}

/\*

\* 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 storeairportrecordinpig;

import java.io.IOException;

import org.apache.hadoop.hbase.client.Put;

import org.apache.hadoop.hbase.io.ImmutableBytesWritable;

import org.apache.hadoop.hbase.mapreduce.TableReducer;

import org.apache.hadoop.hbase.util.Bytes;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

/\*\*

\*

\* @author pratik

\*/

public class Reducer\_StoreInPig extends TableReducer<Text,Text,ImmutableBytesWritable>{

@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.

for(Text val:values){

Put put = new Put(Bytes.toBytes(key.toString()));

put.add(Bytes.toBytes("Airports"),Bytes.toBytes(key.toString()),Bytes.toBytes(val.toString()));

context.write(null, put);

}

}

}

/\*

\* 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 busyroutes;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.hbase.HBaseConfiguration;

import org.apache.hadoop.hbase.client.Get;

import org.apache.hadoop.hbase.client.HTable;

import org.apache.hadoop.hbase.client.Result;

import org.apache.hadoop.hbase.client.Scan;

import org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;

import org.apache.hadoop.hbase.util.Bytes;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.mapreduce.Job;

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

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

/\*\*

\*

\* @author pratik

\*/

public class BusyRoutes {

/\*\*

\* @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, "Total Number of flights per route");

job.setJarByClass(BusyRoutes.class);

job.setMapperClass(BusyRoute\_Mapper.class);

job.setMapOutputKeyClass(BusyRoute\_CompositeKey.class);

job.setCombinerClass(BusyRoute\_Reducer.class);

job.setMapOutputValueClass(IntWritable.class);

job.setReducerClass(BusyRoute\_Reducer.class);

job.setOutputKeyClass(BusyRoute\_CompositeKey.class);

job.setOutputValueClass(IntWritable.class);

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

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

boolean jobCompleted = job.waitForCompletion(true);

if(jobCompleted){

Configuration conf2 = HBaseConfiguration.create();

// HTable table = new HTable(conf2, "AirportDetails");

Job job2 = Job.getInstance(conf2, "Maximum busiest route");

//setting the htable

//ening htable

job2.setJarByClass(BusyRoutes.class);

job2.setMapperClass(BusyRoute\_Final\_Mapper.class);

job2.setMapOutputKeyClass(BusyRoute\_CompositeKey.class);

//job2.setCombinerClass(BusyRoute\_Reducer.class);

job2.setMapOutputValueClass(IntWritable.class);

job2.setReducerClass(BusyRoute\_Final\_Reducer.class);

job2.setOutputKeyClass(BusyRoute\_CompositeKey.class);

job2.setOutputValueClass(IntWritable.class);

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

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

boolean jobComplete = job2.waitForCompletion(true);

if(jobComplete){

Configuration config = HBaseConfiguration.create();

Job job3 = new Job(config,"Read from Hbase and write");

job3.setJarByClass(BusyRoute\_Mapper\_getDetails.class); // class that contains mapper

Scan scan = new Scan();

scan.setCaching(500); // 1 is the default in Scan, which will be bad for MapReduce jobs

scan.setCacheBlocks(false); // don't set to true for MR jobs

// set other scan attrs

job3.addCacheFile(new Path(args[2]+"/part-r-00000").toUri());

TableMapReduceUtil.initTableMapperJob(

"AirportDetails", // input table

scan, // Scan instance to control CF and attribute selection

BusyRoute\_Mapper\_getDetails.class, // mapper class

null, // mapper output key

null, // mapper output value

job3);

FileOutputFormat.setOutputPath(job3, new Path(args[3]));

job3.setNumReduceTasks(0);

System.exit(job3.waitForCompletion(true)?0:1);

}

}

}

}

/\*

\* 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 busyroutes;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author pratik

\*/

public class BusyRoute\_Reducer extends Reducer<BusyRoute\_CompositeKey,IntWritable,BusyRoute\_CompositeKey,IntWritable>{

// private BusyRoute\_CompositeKey result = new BusyRoute\_CompositeKey();

private IntWritable count = new IntWritable();

@Override

protected void reduce(BusyRoute\_CompositeKey key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {

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

int sum =0;

for(IntWritable val:values){

sum = sum +val.get();

}

count.set(sum);

context.write(key, count);

}

}

/\*

\* 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 busyroutes;

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.HashMap;

import javax.annotation.concurrent.Immutable;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.hbase.client.Result;

import org.apache.hadoop.hbase.io.ImmutableBytesWritable;

import org.apache.hadoop.hbase.mapreduce.TableMapper;

import org.apache.hadoop.hbase.util.Bytes;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

/\*\*

\*

\* @author pratik

\*/

public class BusyRoute\_Mapper\_getDetails extends TableMapper<Text,IntWritable>{

private final IntWritable count = new IntWritable(1);

private Text text = new Text();

private String source;

private String destination;

private Integer totalCount;

private String sourceName="";

private String destinationName="";

//############### reading from Distributed Cache

// private static HashMap<String, String> airports = new HashMap<String, String>();

private BufferedReader brReader;

@Override

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

Path[] cacheFilesLocal = context.getLocalCacheFiles();

System.out.print("The lentgh is "+cacheFilesLocal.length);

for (Path eachPath : cacheFilesLocal) {

if (eachPath.getName().toString().trim().equals("part-r-00000")) {

// context.getCounter(MYCOUNTER.FILE\_EXISTS).increment(1);

loadAirportsHashMap(eachPath, context);

}

}

}

private void loadAirportsHashMap(Path filePath, Context context) throws IOException {

String strLineRead = "";

try {

brReader = new BufferedReader(new FileReader(filePath.toString()));

// Read each line, split and load to HashMap

while ((strLineRead = brReader.readLine()) != null) {

System.out.println("Inside buffer");

String deptFieldArray[] = strLineRead.replace("\"", "").split("\\t");

System.out.println("The value is"+deptFieldArray[0]);

//if(!deptFieldArray[0].equals("iata")){

source = deptFieldArray[0].trim();

destination = deptFieldArray[1].trim();

totalCount = Integer.parseInt(deptFieldArray[2].trim());

// }

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}finally {

if (brReader != null) {

brReader.close();

}

}

}

//############### reading from Distributed Cache

public void map(ImmutableBytesWritable row, Result value, Context context) throws IOException, InterruptedException {

if(Bytes.toString(row.get()).equals(source)){

System.out.println("Inside source Row Hbase");

sourceName = new String(value.getValue(Bytes.toBytes("Airports"), Bytes.toBytes(source)));

}else if(Bytes.toString(row.get()).equals(destination)){

System.out.println("Inside Destination Row Hbase");

destinationName = new String(value.getValue(Bytes.toBytes("Airports"), Bytes.toBytes(destination)));

}

}

@Override

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

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

if(!sourceName.equals("")&&!destinationName.equals("")){

text.set(sourceName+" "+destinationName);

count.set(totalCount);

context.write(text, count);

}

}

}

/\*

\* 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 busyroutes;

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;

/\*\*

\*

\* @author pratik

\*/

public class BusyRoute\_Mapper extends Mapper<LongWritable,Text,BusyRoute\_CompositeKey,IntWritable>{

private BusyRoute\_CompositeKey outKey = new BusyRoute\_CompositeKey();

private IntWritable outValue = new IntWritable();

@Override

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

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

if(inputArray[0].equals("Year")){

System.out.println("reading the Column Name");

}else{

outKey.setOrigin(inputArray[16]);

outKey.setDestination(inputArray[17]);

outValue.set(1);

context.write(outKey, outValue);

}

}

}

/\*

\* 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 busyroutes;

import java.io.IOException;

import org.apache.hadoop.hbase.TableName;

import org.apache.hadoop.hbase.client.Connection;

import org.apache.hadoop.hbase.client.ConnectionFactory;

import org.apache.hadoop.hbase.client.HTable;

import org.apache.hadoop.hbase.client.Table;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author pratik

\*/

public class BusyRoute\_Final\_Reducer extends Reducer<BusyRoute\_CompositeKey, IntWritable, BusyRoute\_CompositeKey, IntWritable> {

private IntWritable maxCount = new IntWritable(0);

private BusyRoute\_CompositeKey result = new BusyRoute\_CompositeKey();

String name;

// HTable table;

@Override

protected void reduce(BusyRoute\_CompositeKey key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {

int sum =0;

for(IntWritable val:values){

sum = sum+val.get();

}

//finding out the max value and setting the results

if(maxCount.get()==0||maxCount.get()<sum){

maxCount.set(sum);

result.setOrigin(key.getOrigin());

result.setDestination(key.getDestination());

}

}

@Override

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

// String origin=getFromHbase(result.getOrigin(),context);

context.write(result, maxCount);

}

}

/\*

\* 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 busyroutes;

import java.io.IOException;

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.Mapper;

/\*\*

\*

\* @author pratik

\*/

public class BusyRoute\_Final\_Mapper extends Mapper<LongWritable,Text,BusyRoute\_CompositeKey,IntWritable>{

private IntWritable outValue = new IntWritable();

private BusyRoute\_CompositeKey outKey = new BusyRoute\_CompositeKey();

@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 inputArray[] = value.toString().split("\\t");

outKey.setOrigin(inputArray[0]);

outKey.setDestination(inputArray[1]);

outValue.set(Integer.parseInt(inputArray[2]));

context.write(outKey, outValue);

}

}

/\*

\* 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 busyroutes;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

/\*\*

\*

\* @author pratik

\*/

public class BusyRoute\_CompositeKey implements Writable,WritableComparable<BusyRoute\_CompositeKey>{

private String origin;

private String destination;

public String getOrigin() {

return origin;

}

public void setOrigin(String origin) {

this.origin = origin;

}

public String getDestination() {

return destination;

}

public void setDestination(String destination) {

this.destination = destination;

}

@Override

public void write(DataOutput d) throws IOException {

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

d.writeUTF(origin);

d.writeUTF(destination);

}

@Override

public void readFields(DataInput di) throws IOException {

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

origin = di.readUTF();

destination = di.readUTF();

}

@Override

public int compareTo(BusyRoute\_CompositeKey o) {

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

int result = origin.compareTo(o.origin);

if(result==0){

result = destination.compareTo(o.destination);

}

return result;

}

@Override

public String toString() {

return (new StringBuilder().append(origin).append("\t").append(destination).toString());

//return origin+"\\t"+destination ;//To change body of generated methods, choose Tools | Templates.

}

}

1. **Different options (Direct Flight) to fly from Illinois to California**

Code:

/\*

\* 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 joinflights;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Reducer;

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

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

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

/\*\*

\*

\* @author pratik

\*/

public class FindAllFlights {

/\*\*

\* @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, "Inner join");

job.setJarByClass(FindAllFlights.class);

MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class,Mapper\_JoinSource.class);

MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class,Mapper\_AirportFile.class);

// job.setNumReduceTasks(0);

job.setReducerClass(Reducer\_Join.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

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

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

boolean jobCompleted = job.waitForCompletion(true);

if(jobCompleted){

Job job2 = Job.getInstance(conf, "Inner join");

job2.setJarByClass(FindAllFlights.class);

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

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

// job.setNumReduceTasks(0);

job2.setReducerClass(Reducer\_Join.class);

job2.setOutputKeyClass(Text.class);

job2.setOutputValueClass(Text.class);

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

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

}

}

}

/\*

\* 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 joinflights;

import java.io.IOException;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author pratik

\*/

public class Mapper\_AirportFile extends Mapper<Object,Text,Text,Text>{

private Text outValue = new Text();

private Text outKey = new Text();

@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 inputArray[] = value.toString().split(",");

if(inputArray.length<6){

System.out.print("Some columns are missing for the record");

}else if (inputArray[0].replace("\"","").equals("iata")){

System.out.print("Reading the Column name record");

}else{

String outputValue = inputArray[1].replace("\"","")+"\t"+inputArray[3].replace("\"","")+"\t"+inputArray[5].replace("\"","")+"\t"+inputArray[6].replace("\"","");

outKey.set(inputArray[0].replace("\"",""));

outValue.set("B"+outputValue);

context.write(outKey,outValue);

}

}

}

/\*

\* 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 joinflights;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author pratik

\*/

public class Mapper\_Destination extends Mapper<LongWritable,Text,Text,Text>{

private Text outKey = new Text();

private Text outValue = new Text();

@Override

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

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

outKey.set(inputArray[1]);

outValue.set("A"+value);

context.write(outKey, outValue);

}

}

/\*

\* 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 joinflights;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author pratik

\*/

public class Mapper\_JoinSource extends Mapper<LongWritable,Text,Text,Text>{

private Text outValue = new Text();

private Text outKey = new 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.

// System.out.print(value);

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

System.out.println(inputArrayString[0]);

if(inputArrayString.length<29){

System.out.print("Some columns are missing for the record");

}else if (inputArrayString[0].equals("Year")){

System.out.print("Reading the Column name record");

}else if((Integer.parseInt(inputArrayString[1])==1)&&Integer.parseInt(inputArrayString[2])==1){

String outputValue = inputArrayString[16]+"\t"+inputArrayString[17];

outValue.set("A"+outputValue);

outKey.set(inputArrayString[16]);

context.write(outKey, outValue);

}

}

}

/\*

\* 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 joinflights;

import java.io.IOException;

import java.util.ArrayList;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author pratik

\*/

public class Reducer\_Join 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<>();

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

@Override

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

listA.clear();

listB.clear();

for(Text val:values) {

if (Character.toString((char) val.charAt(0)).equals("A")) {

System.out.print("Inside A");

listA.add(new Text(val.toString().substring(1)));

} else if (Character.toString((char) val.charAt(0)).equals("B")) {

System.out.print("Inside B");

listB.add(new Text(val.toString().substring(1)));

}

}

System.out.println("The length is A is ##### "+listA.size());

System.out.println("The length is B #### "+listB.size());

executeJoinLogic(context);

}

public void executeJoinLogic(Context context) throws IOException, InterruptedException{

if (!listA.isEmpty() && !listB.isEmpty()) {

for (Text A : listA) {

for (Text B : listB) {

context.write(A, B);

}

}

}

}

}

/\*

\* 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 joinflights;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

/\*\*

\*

\* @author pratik

\*/

public class Route\_Compositekey implements Writable,WritableComparable<Route\_Compositekey>{

String origin;

String destination;

String originCity;

String originState;

String destCity;

String destState;

String originLat;

String originLong;

String destLat;

String destLong;

@Override

public void write(DataOutput d) throws IOException {

d.writeUTF(origin);

d.writeUTF(destination);

d.writeUTF(originCity);

d.writeUTF(originState);

d.writeUTF(destCity);

d.writeUTF(destState);

d.writeUTF(originLat);

d.writeUTF(originLong);

d.writeUTF(destLat);

d.writeUTF(destLong);

}

@Override

public void readFields(DataInput di) throws IOException {

origin = di.readUTF();

destination = di.readUTF();

originCity = di.readUTF();

originState = di.readUTF();

destCity = di.readUTF();

destState = di.readUTF();

originLat= di.readUTF();

originLong = di.readUTF();

destLat= di.readUTF();

destLong = di.readUTF();

}

@Override

public int compareTo(Route\_Compositekey o) {

int result = origin.compareTo(o.origin);

return result;

}

}

/\*

\* 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 allflightswithinstates;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.NullWritable;

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;

/\*\*

\* @author pratik

\* Description: Uses the Joined File saved in the Hadoop File system for Flight Details

\*/

public class AllFlightsWithinStates {

/\*\*

\* @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, "Flights from IL to CA");

job.setJarByClass(AllFlightsWithinStates.class);

job.setMapperClass(Mapper\_AllFlights.class);

job.setMapOutputKeyClass(Text.class);

// job.setCombinerClass(BusyRoute\_Reducer.class);

job.setMapOutputValueClass(NullWritable.class);

job.setReducerClass(Reducer\_AllFlights.class);

// job.setNumReduceTasks(1);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(NullWritable.class);

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

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

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

}

}

/\*

\* 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 allflightswithinstates;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author pratik

\*/

public class Mapper\_AllFlights extends Mapper<LongWritable,Text,Text,NullWritable>{

private Text outKey = new 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 inputString[] = value.toString().split("\\t");

if(inputString[3].equals("IL")&&inputString[7].equals("CA")){

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

}

}

}

/\*

\* 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 allflightswithinstates;

import java.io.IOException;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author pratik

\*/

public class Reducer\_AllFlights extends Reducer<Text,NullWritable,Text,NullWritable>{

@Override

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

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

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

}

}

1. **Are older Aircraft prone to more delays then the Newer Aircraft?**

Code:

/\*

\* 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 aircraftdelaybasedonyearofmanufacture;

import java.io.IOException;

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.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 pratik

\*/

public class AircraftDelayBasedOnYearOfManufacture {

/\*\*

\* @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, "Finding Average for pearson correlation");

job.setJarByClass(AircraftDelayBasedOnYearOfManufacture.class);

job.setMapperClass(Mapper\_AverageDelay.class);

job.setMapOutputKeyClass(IntWritable.class);

job.setMapOutputValueClass(CompositeValue\_Average.class);

job.addCacheFile(new Path(args[0]).toUri());//input path to caches file

job.setReducerClass(Reducer\_AverageDelay.class);

job.setOutputKeyClass(IntWritable.class);

job.setOutputValueClass(CompositeValue\_Average.class);

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

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

boolean jobCompleted = job.waitForCompletion(true);

if(jobCompleted){

// Configuration c = new Configuration();

Job job2 = Job.getInstance(conf, "Finding the pearson Coeeficient");

job2.setJarByClass(AircraftDelayBasedOnYearOfManufacture.class);

job2.setMapperClass(Pearson\_Mapper.class);

job2.setReducerClass(Pearson\_Reducer.class);

job2.setMapOutputKeyClass(Pearson\_KeyWritable.class);

job2.setMapOutputValueClass(Pearson\_ValueWritable.class);

job2.setOutputKeyClass(Text.class);

job2.setOutputValueClass(DoubleWritable.class);

FileInputFormat.addInputPath(job2, new Path(args[2]));

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

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

}

}

}

/\*

\* 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 aircraftdelaybasedonyearofmanufacture;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

/\*\*

\*

\* @author pratik

\*/

public class CompositeValue\_Average implements Writable,WritableComparable<CompositeValue\_Average>{

Integer count;

Double averageDelay;

public Integer getCount() {

return count;

}

public void setCount(Integer count) {

this.count = count;

}

public Double getAverageDelay() {

return averageDelay;

}

public void setAverageDelay(Double averageDelay) {

this.averageDelay = averageDelay;

}

@Override

public void write(DataOutput d) throws IOException {

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

d.writeInt(count);

d.writeDouble(averageDelay);

}

@Override

public void readFields(DataInput di) throws IOException {

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

count = di.readInt();

averageDelay = di.readDouble();

}

@Override

public int compareTo(CompositeValue\_Average o) {

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

return 0;

}

@Override

public String toString() {

return (new StringBuilder().append(averageDelay).toString());

}

}

/\*

\* 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 aircraftdelaybasedonyearofmanufacture;

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.HashMap;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author pratik

\*/

public class Mapper\_AverageDelay extends Mapper<LongWritable,Text,IntWritable,CompositeValue\_Average>{

private static HashMap<String, String> airCraft = new HashMap<String, String>();

private BufferedReader brReader;

// private String strDeptName ="";

private IntWritable outKey = new IntWritable();

private CompositeValue\_Average outValue = new CompositeValue\_Average();

@Override

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

Path[] cacheFilesLocal = context.getLocalCacheFiles();

System.out.print("The lentgh is "+cacheFilesLocal.length);

for (Path eachPath : cacheFilesLocal) {

if (eachPath.getName().toString().trim().equals("part-r-00000")) {

// context.getCounter(MYCOUNTER.FILE\_EXISTS).increment(1);

System.out.println("Inside part-r-0000 present");

loadAircraftHashMap(eachPath, context);

}

}

}

private void loadAircraftHashMap(Path filePath, Context context) throws IOException {

String strLineRead = "";

try {

brReader = new BufferedReader(new FileReader(filePath.toString()));

// Read each line, split and load to HashMap

while ((strLineRead = brReader.readLine()) != null) {

String deptFieldArray[] = strLineRead.replace("\"", "").split("\t");

airCraft.put(deptFieldArray[1].trim(),deptFieldArray[0].trim());

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}finally {

if (brReader != null) {

brReader.close();

}

}

}

@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 inputString[] = value.toString().split(",");

if(inputString[0].equals("Year")||inputString.length<29||inputString[10].equals("")||inputString[15].equals("NA")){

System.out.println("Reading Columns");

}else{

if(airCraft.containsKey(inputString[10])){

System.out.println("The key is "+inputString[10]);

String year = airCraft.get(inputString[10]);

outKey.set(Integer.parseInt(year));

outValue.setAverageDelay(Double.parseDouble(inputString[15]));

outValue.setCount(1);

context.write(outKey, outValue);

}

}

}

}

/\*

\* 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 aircraftdelaybasedonyearofmanufacture;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

/\*\*

\*

\* @author pratik

\*/

public class Pearson\_KeyWritable implements Writable,WritableComparable<Pearson\_KeyWritable>{

private Integer rowIndex;

private Integer columnindex;

public Integer getRowIndex() {

return rowIndex;

}

public void setRowIndex(Integer rowIndex) {

this.rowIndex = rowIndex;

}

public Integer getColumnindex() {

return columnindex;

}

public void setColumnindex(Integer columnindex) {

this.columnindex = columnindex;

}

@Override

public void write(DataOutput d) throws IOException {

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

d.writeInt(rowIndex);

d.writeInt(columnindex);

}

@Override

public void readFields(DataInput di) throws IOException {

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

rowIndex = di.readInt();

columnindex = di.readInt();

}

@Override

public int compareTo(Pearson\_KeyWritable o) {

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

int result = rowIndex.compareTo(o.rowIndex);

if(result == 0){

result = columnindex.compareTo(o.columnindex);

}

return result;

}

@Override

public String toString() {

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

return (new StringBuilder().append(rowIndex).append("\t").append(columnindex).toString());

}

}

/\*

\* 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 aircraftdelaybasedonyearofmanufacture;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author pratik

\*/

public class Pearson\_Mapper extends Mapper<LongWritable,Text,Pearson\_KeyWritable,Pearson\_ValueWritable>{

private Pearson\_KeyWritable outKey = new Pearson\_KeyWritable();

private Pearson\_ValueWritable outValue = new Pearson\_ValueWritable();

@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 inputArray[] = value.toString().split("\\t");

int i=0;

int j=0;

while(i<inputArray.length){

j=i+1;

while(j<inputArray.length){

outKey.setRowIndex(i);

outKey.setColumnindex(j);

outValue.setElement1(Double.parseDouble(inputArray[i]));

outValue.setElement2(Double.parseDouble(inputArray[j]));

context.write(outKey, outValue);

j++;

}

i++;

}

}

}

/\*

\* 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 aircraftdelaybasedonyearofmanufacture;

import java.io.IOException;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author pratik

\*/

public class Pearson\_Reducer extends Reducer<Pearson\_KeyWritable,Pearson\_ValueWritable,Text,DoubleWritable>{

private Text resultKey = new Text();

@Override

protected void reduce(Pearson\_KeyWritable key, Iterable<Pearson\_ValueWritable> values, Context context) throws IOException, InterruptedException {

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

double x = 0.0d;

double y = 0.0d;

double xx = 0.0d;

double yy = 0.0d;

double xy = 0.0d;

double n = 0.0d;

for(Pearson\_ValueWritable val : values){

x += val.getElement1();

y += val.getElement2();

xx += Math.pow(val.getElement1(), 2.0);

yy += Math.pow(val.getElement2(), 2.0);

xy += val.getElement1() \* val.getElement2();

n += 1.0d;

}

double correlationFactor = getPearsonCorrelationFactor(x,y,xx,yy,xy,n);

resultKey.set("The pearson correlation factor is : ");

context.write(resultKey, new DoubleWritable(correlationFactor));

}

public double getPearsonCorrelationFactor(double x, double y, double xx, double yy, double xy, double n){

double numerator = xy - ((x \* y) / n);

double denominator1 = xx - (Math.pow(x, 2.0) / n);

double denominator2 = yy - (Math.pow(y, 2.0) / n);

double denominator = Math.sqrt(denominator1 \* denominator2);

double correlationFactor = numerator / denominator;

return correlationFactor;

}

}

/\*

\* 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 aircraftdelaybasedonyearofmanufacture;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

/\*\*

\*

\* @author pratik

\*/

public class Pearson\_ValueWritable implements Writable,WritableComparable<Pearson\_ValueWritable>{

private Double element1;

private Double element2;

public Double getElement1() {

return element1;

}

public void setElement1(Double element1) {

this.element1 = element1;

}

public Double getElement2() {

return element2;

}

public void setElement2(Double element2) {

this.element2 = element2;

}

@Override

public void write(DataOutput d) throws IOException {

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

d.writeDouble(element1);

d.writeDouble(element2);

}

@Override

public void readFields(DataInput di) throws IOException {

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

element1 = di.readDouble();

element2 = di.readDouble();

}

@Override

public int compareTo(Pearson\_ValueWritable o) {

int result = element1.compareTo(o.element1);

if(result == 0){

result = element2.compareTo(o.element2);

}

return result;

}

@Override

public String toString() {

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

return (new StringBuilder().append(element1).append("\t").append(element2).toString());

}

}

/\*

\* 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 aircraftdelaybasedonyearofmanufacture;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author pratik

\*/

public class Reducer\_AverageDelay extends Reducer<IntWritable,CompositeValue\_Average,IntWritable,CompositeValue\_Average>{

private CompositeValue\_Average result = new CompositeValue\_Average();

@Override

protected void reduce(IntWritable key, Iterable<CompositeValue\_Average> values, Context context) throws IOException, InterruptedException {

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

Double sum =0.0;

int count=0;

for(CompositeValue\_Average val:values){

sum = sum+val.getAverageDelay();

count = count+val.getCount();

}

result.setAverageDelay(sum/count);

result.setCount(count);

context.write(key, result);

}

}

1. **Carrier Popularity**

Code:

/\*

\* 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 carrierpopularity;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

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

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

/\*\*

\*

\* @author pratik

\*/

public class CarrierPopularity {

/\*\*

\* @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, "Total Number of inbound flights");

job.setJarByClass(CarrierPopularity.class);

job.setMapperClass(Mapper\_Carrier.class);

job.setMapOutputKeyClass(Carrier\_CompositeKey.class);

job.setMapOutputValueClass(IntWritable.class);

job.setReducerClass(Carrier\_Reducer.class);

// job.setNumReduceTasks(1);

job.addCacheFile(new Path(args[0]).toUri());

job.setOutputKeyClass(Carrier\_CompositeKey.class);

job.setOutputValueClass(IntWritable.class);

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

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

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

}

}

/\*

\* 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 carrierpopularity;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

/\*\*

\*

\* @author pratik

\*/

public class Carrier\_CompositeKey implements Writable,WritableComparable<Carrier\_CompositeKey>{

Integer year;

String carrierName;

public Integer getYear() {

return year;

}

public void setYear(Integer year) {

this.year = year;

}

public String getCarrierName() {

return carrierName;

}

public void setCarrierName(String carrierName) {

this.carrierName = carrierName;

}

@Override

public void write(DataOutput d) throws IOException {

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

d.writeUTF(carrierName);

d.writeInt(year);

}

@Override

public void readFields(DataInput di) throws IOException {

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

carrierName = di.readUTF();

year = di.readInt();

}

@Override

public int compareTo(Carrier\_CompositeKey o) {

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

int result = year.compareTo(o.year);

if(result==0){

result = carrierName.compareTo(o.carrierName);

}

return result;

}

@Override

public String toString() {

return (new StringBuilder().append(year).append("\t").append(carrierName).toString());

}

}

/\*

\* 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 carrierpopularity;

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.HashMap;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author pratik

\*/

public class Carrier\_Reducer extends Reducer<Carrier\_CompositeKey,IntWritable,Carrier\_CompositeKey,IntWritable>{

private Carrier\_CompositeKey resultKey = new Carrier\_CompositeKey();

private IntWritable resultValue = new IntWritable();

private static HashMap<String, String> carrier = new HashMap<String, String>();

private BufferedReader brReader;

@Override

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

Path[] cacheFilesLocal = context.getLocalCacheFiles();

System.out.print("The lentgh is "+cacheFilesLocal.length);

for (Path eachPath : cacheFilesLocal) {

if (eachPath.getName().toString().trim().equals("carriers.csv")) {

// context.getCounter(MYCOUNTER.FILE\_EXISTS).increment(1);

loadAirportsHashMap(eachPath, context);

}

}

}

private void loadAirportsHashMap(Path filePath, Context context) throws IOException {

String strLineRead = "";

try {

brReader = new BufferedReader(new FileReader(filePath.toString()));

// Read each line, split and load to HashMap

while ((strLineRead = brReader.readLine()) != null) {

System.out.println("Inside buffer");

String deptFieldArray[] = strLineRead.replace("\"", "").split(",");

System.out.println("The value is"+deptFieldArray[0]);

if(!deptFieldArray[0].equals("Code")){

carrier.put(deptFieldArray[0].trim(),deptFieldArray[1].trim());

}

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}finally {

if (brReader != null) {

brReader.close();

}

}

}

@Override

protected void reduce(Carrier\_CompositeKey key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {

int sum =0;

for(IntWritable val:values){

sum =sum+1;

}

resultKey.setCarrierName(carrier.get(key.getCarrierName()));

resultKey.setYear(key.getYear());

resultValue.set(sum);

context.write(resultKey, resultValue);

}

}

/\*

\* 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 carrierpopularity;

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;

/\*\*

\*

\* @author pratik

\*/

public class Mapper\_Carrier extends Mapper<LongWritable,Text,Carrier\_CompositeKey,IntWritable>{

private Carrier\_CompositeKey outKey = new Carrier\_CompositeKey();

private IntWritable outValue = new IntWritable();

@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 inputArray[] = value.toString().split(",");

if(inputArray[0].equals("Year")||inputArray.length<29||inputArray[8].equals("")){

}else if(inputArray[8].equals("US")||inputArray[8].equals("WN")||inputArray[8].equals("UA")||inputArray[8].equals("CO")||inputArray[8].equals("AA")){

outKey.setCarrierName(inputArray[8]);

outKey.setYear(Integer.parseInt(inputArray[0]));

outValue.set(1);

context.write(outKey, outValue);

}

}

}

1. Finding the approximate delay for a particular flight

Code:

/\*

\* 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 predictaveragedelay;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.mapreduce.Job;

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

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

/\*\*

\*

\* @author pratik

\*/

public class PredictAverageDelay {

/\*\*

\* @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, "Predictions");

job.setJarByClass(PredictAverageDelay.class);

job.setMapperClass(Predict\_Mapper.class);

job.setMapOutputKeyClass(Predict\_CompositeKey.class);

job.setMapOutputValueClass(Predict\_CompositeValue.class);

job.setReducerClass(Pridict\_Reducer.class);

job.setOutputKeyClass(Predict\_CompositeKey.class);

job.setOutputValueClass(Predict\_CompositeValue.class);

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

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

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

}

}

/\*

\* 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 predictaveragedelay;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

/\*\*

\*

\* @author pratik

\*/

public class Predict\_CompositeKey implements Writable,WritableComparable<Predict\_CompositeKey>{

private Integer flightNumber;

private Integer dayOfMonth;

private Integer month;

public Integer getFlightNumber() {

return flightNumber;

}

public void setFlightNumber(Integer flightNumber) {

this.flightNumber = flightNumber;

}

public Integer getDayOfMonth() {

return dayOfMonth;

}

public void setDayOfMonth(Integer dayOfMonth) {

this.dayOfMonth = dayOfMonth;

}

public Integer getMonth() {

return month;

}

public void setMonth(Integer month) {

this.month = month;

}

@Override

public void write(DataOutput d) throws IOException {

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

d.writeInt(flightNumber);

d.writeInt(dayOfMonth);

d.writeInt(month);

}

@Override

public void readFields(DataInput di) throws IOException {

flightNumber = di.readInt();

dayOfMonth = di.readInt();

month = di.readInt();

}

@Override

public int compareTo(Predict\_CompositeKey o) {

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

int result = flightNumber.compareTo(o.flightNumber);

if(result==0){

result = (-1)\*month.compareTo(o.month);

if(result==0){

result = (-1)\*dayOfMonth.compareTo(o.dayOfMonth);

}

}

return result;

}

@Override

public String toString() {

// return "Predict\_CompositeKey{" + "flightNumber=" + flightNumber + ", dayOfMonth=" + dayOfMonth + ", month=" + month + '}';

return (new StringBuilder().append(flightNumber).append("\t").append(month).append("\t").append(dayOfMonth).toString());

}

}

/\*

\* 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 predictaveragedelay;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

/\*\*

\*

\* @author pratik

\*/

public class Predict\_CompositeValue implements Writable,WritableComparable<Predict\_CompositeValue>{

private Integer count ;

private Double Avereage;

public Integer getCount() {

return count;

}

public void setCount(Integer count) {

this.count = count;

}

public Double getAvereage() {

return Avereage;

}

public void setAvereage(Double Avereage) {

this.Avereage = Avereage;

}

@Override

public void write(DataOutput d) throws IOException {

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

d.writeInt(count);

d.writeDouble(Avereage);

}

@Override

public void readFields(DataInput di) throws IOException {

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

count = di.readInt();

Avereage = di.readDouble();

}

@Override

public int compareTo(Predict\_CompositeValue o) {

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

return 0;

}

@Override

public String toString() {

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

return (new StringBuilder().append(Avereage).toString());

}

}

/\*

\* 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 predictaveragedelay;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author pratik

\*/

public class Predict\_Mapper extends Mapper<LongWritable,Text,Predict\_CompositeKey,Predict\_CompositeValue>{

private Predict\_CompositeKey outKey = new Predict\_CompositeKey();

private Predict\_CompositeValue outValue = new Predict\_CompositeValue();

@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 inputValue[] = value.toString().split(",");

if(inputValue.length<29){

System.out.println("Invalid column");

}

else if(inputValue[0].equals("Year")){

System.out.println("Reading the column names");

}else if(!inputValue[15].equals("NA")){

outKey.setFlightNumber(Integer.parseInt(inputValue[9]));

outKey.setDayOfMonth(Integer.parseInt(inputValue[2]));

outKey.setMonth(Integer.parseInt(inputValue[1]));

outValue.setCount(1);

outValue.setAvereage(Double.parseDouble(inputValue[15]));

context.write(outKey, outValue);

}

}

}

/\*

\* 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 predictaveragedelay;

import java.io.IOException;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author pratik

\*/

public class Pridict\_Reducer extends Reducer<Predict\_CompositeKey,Predict\_CompositeValue,Predict\_CompositeKey,Predict\_CompositeValue>{

private Predict\_CompositeValue result = new Predict\_CompositeValue();

@Override

protected void reduce(Predict\_CompositeKey key, Iterable<Predict\_CompositeValue> values, Context context) throws IOException, InterruptedException {

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

// Double sum = 0.0;

// Integer count =0;

//

for(Predict\_CompositeValue val:values){

// sum = sum+val.getAvereage();

// count = count+val.getCount();

context.write(key, val);

}

// result.setAvereage(sum/count);

// result.setCount(count);

}

}

1. Clean Carrier Data

Code:

/\*

\* 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 cleancarrierdata;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.mapreduce.Job;

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

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

/\*\*

\*

\* @author pratik

\*/

public class CleanCarrierData {

/\*\*

\* @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, "Clean aircraft details data");

job.setJarByClass(CleanCarrierData.class);

job.setMapperClass(Mapper\_CleanAircraft.class);

job.setMapOutputKeyClass(CompositeKey\_AircraftDetails.class);

// job.setCombinerClass(BusyRoute\_Reducer.class);

job.setMapOutputValueClass(NullWritable.class);

//job.setReducerClass(Reducer\_AllFlights.class);

// job.setNumReduceTasks(1);

job.setOutputKeyClass(CompositeKey\_AircraftDetails.class);

job.setOutputValueClass(NullWritable.class);

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

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

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

}

}

/\*

\* 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 cleancarrierdata;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

/\*\*

\*

\* @author pratik

\*/

public class CompositeKey\_AircraftDetails implements Writable,WritableComparable<CompositeKey\_AircraftDetails>{

private String tailNumber;

private Integer year;

private String aircraftType;

public String getTailNumber() {

return tailNumber;

}

public void setTailNumber(String tailNumber) {

this.tailNumber = tailNumber;

}

public Integer getYear() {

return year;

}

public void setYear(Integer year) {

this.year = year;

}

public String getAircraftType() {

return aircraftType;

}

public void setAircraftType(String aircraftType) {

this.aircraftType = aircraftType;

}

@Override

public void write(DataOutput d) throws IOException {

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

d.writeUTF(tailNumber);

d.writeUTF(aircraftType);

d.writeInt(year);

}

@Override

public void readFields(DataInput di) throws IOException {

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

tailNumber = di.readUTF();

aircraftType = di.readUTF();

year = di.readInt();

}

@Override

public int compareTo(CompositeKey\_AircraftDetails o) {

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

int result =(-1)\*year.compareTo(year);

return result;

}

@Override

public String toString() {

//return "CompositeKey\_AircraftDetails{" + "tailNumber=" + tailNumber + ", year=" + year + ", aircraftType=" + aircraftType + '}';

return (new StringBuilder().append(year).append("\t").append(tailNumber).append("\t").append(aircraftType).toString());

}

}

/\*

\* 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 cleancarrierdata;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author pratik

\*/

public class Mapper\_CleanAircraft extends Mapper<LongWritable,Text,CompositeKey\_AircraftDetails,NullWritable>{

private CompositeKey\_AircraftDetails outKey = new CompositeKey\_AircraftDetails();

@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 inputArray[] = value.toString().replace("\"", "").split(",");

if(inputArray.length<9||inputArray[0].equals("tailnum")){

System.out.println("Invalid record");

}else if(inputArray[8].equals("None")||inputArray[8].equals("")||inputArray[0].equals("")||inputArray[4].equals("")){

System.out.println("Invalid Year");

}else{

outKey.setAircraftType(inputArray[4]);

outKey.setTailNumber(inputArray[0]);

outKey.setYear(Integer.parseInt(inputArray[8]));

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

}

}

}