Project 1

Airline Delay Analysis using MapReduce

Shraavanth Penugonda | 35764082 | Information Storage

Background:

Maintaining on-time schedules of airlines is an important factor in satisfying the current customers and attracting new ones. However, flight schedules are often subjected to irregularity. Even though life has been made easy with flights, the delay of flights can cause frustration and can also be very expensive to both the airlines and customers. Delays are extremely irritating for passengers especially when they have to attend any events or catch multiple flights and the delay of one can cause the passenger to miss the other flight. Delays can occur due to problems at the origin, at the destination or even during the journey. Some of the possible reasons for delays include maintenance, extreme weather, air traffic, late arrival of previous aircraft or delayed baggage loading.

Analysis of the dataset of timings of various flights over time can give various insights that can be useful for passengers while making choices for their flight journey. The available dataset contains the date, scheduled and actual arrival and departure timings of various flights over the year in US. With more than 87,000 flights crossing the US every day, the amount of data to work on becomes very large. Therefore, to handle such large amounts of data, the usage of Hadoop is the most efficient solution for this problem.

Motivation:

Owing to the large number of flights operating every day, the large amount of data available can be used for analysis to mine various statistics that can be beneficial for people in deciding the date, time or the flight to be taken. Due to the presence of large volumes of data, usage of big data tools such as Hadoop and MapReduce would be optimal to handle this problem. With the available data, various kinds of useful statistics can be extracted such as the best time of the day to take flights to have minimum delays, flight carriers that cause maximum delays, cities that cause the maximum delays and so on. Such information can be useful for passengers while planning journeys.

Proposed Methods:

In this project, I have planned to compute the statistics for couple of problems. Firstly, finding the best time of the day to fly so that minimum amount of delays can be expected and secondly, which flight carriers cause the most amount of delays. Using MapReduce and the available dataset, it is possible to compute the required statistics.

Implementation:

The available dataset consists of many columns depicting the various flight details such as origin, destination, distance, scheduled and actual arrival and departure times and several other details during various days of a year. For the first problem, mapper class can be designed to compute the departure time and the total delay (delay in departure+ delay in arrival) made by a flight as key and values respectively. The reducer class can be used to compute the total delay that has occurred in a particular range of time interval.

The code for mapper class is below

```
public class Map extends Mapper<Object,Text,Text, IntWritable> {
            public void map(Object key, Text value, Context context){
                  Text Date = new Text();
                  try{
                        String[] data = value.toString().split(",");
                        String time = data[7];
                        IntWritable totalDelay = new IntWritable();
                        int totDelay = getDelay(data);
                        Date.set(time);
                        totalDelay.set(totDelay);

                        context.write(Date, totalDelay);
                  } catch(Exception ex){
                        ex.printStackTrace();
                  }
            }
      }

      public static int getDelay(String[] fields){
            int sumDelay = 0;
            sumDelay =
Integer.parseInt(fields[14])+Integer.parseInt(fields[15]);
            return sumDelay;
      }
}
```

```
public static String computeTime(String time){
	DateFormat df = new SimpleDateFormat("HHmm");
	Calendar cal = Calendar.getInstance();
	int hour = Integer.parseInt(time.substring(0, 2));
	int minute = Integer.parseInt(time.substring(2, 4));
	minute = (minute<30)?0:30;
	cal.set(Calendar.HOUR_OF_DAY, hour);
	cal.set(Calendar.MINUTE, minute);
	return df.format(cal.getTime());
}
```

Firstly, the available data is split using commas as separator into a string array. The scheduled departure time is parsed to nearest 30 minute interval in the function computeTime. In the function, the departure time is given as input and it is changed to the nearest 30 minute interval. For example, the time 2315 which corresponds to 11:15PM is converted to 2300. All the times from 2300 to 2329 is mapped to the time 2300. With this method, all the timings of various flights on various days are computed. The corresponding delay is calculated as the sum of the arrival and departure delays. The map function returns this time and delay as key and value pairs respectively.

The code for the reduce function is given below

```
public class delay1Reduce extends
Reducer<Text,IntWritable,Text,IntWritable> {
		public void reduce(Text time, Iterable<IntWritable> values,
Context context){
			int totalCount = 0;
			try {
				for(IntWritable val:values){
					totalCount+=val.get();
				}
				context.write(time,new IntWritable(totalCount));
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
	}
```

The reducer function iterates over the available values to compute the total of the delay timings. The output is written to the file and the screenshot of the results are attached along with the report. In the output, the first part displays the time period (2300 corresponds to 11:00PM) and the second part displays the total amount of delay caused in minutes.

For the second problem, the mapper class can be used to split the data using comma as separator. In the given dataset, the ninth column consists of the flight carrier. The corresponding delays for the flight carrier can be computed using the function getDelay() from the first problem statement. The mapper

output is the key and value pair of name of flight carrier and its corresponding delay. The mapper code for the second problem statement is given below

```
public static class delay2Map extends Mapper<Object,Text,Text, IntWritable>
{
            public void map(Object key, Text value, Context context){
                  Text Carrier = new Text();
                  try{
                        String[] data = value.toString().split(",");
                        Carrier.set(data[8]);
                        IntWritable totalDelay = new IntWritable();
                        int totDelay = delay1.getDelay(data);
                        totalDelay.set(totDelay);
                        context.write(Carrier, totalDelay);

                  }catch(Exception e){
                        e.printStackTrace();
                  }
            }
      }
```

In the reduce function, the delays for the flight carriers are iterated over the carrier names. The displayed output screenshot consists of flights and their corresponding delays in descending order. In this output, the first part denoted the short name of flight carriers and the corresponding delays caused by it in minutes.

The reducer code for the second problem statement is given below

```
public class delay2Reduce extends
Reducer<Text,IntWritable,Text,IntWritable> {
            public void reduce(Text time, Iterable<IntWritable> values,
Context context){
                  int totalCount = 0;
                  try {
                        for(IntWritable val:values){
                              totalCount++;
                        }
                        context.write(time,new IntWritable(totalCount));
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
            }
      }
```

The screenshots of both the problem statements are attached with the report. In the first problem, we can infer the time period where the maximum amount of delays are occurring. For the second time period, the flight carrier having the maximum amount of delay can be found.

Future Work:

With the vast amount of attributes given in the dataset and the large data available, mapreduce can be used for finding various statistics that may be helpful in the field of airlines for both the passengers and the companies. The data set can be computed for statistics such as the cities that cause the maximum weather related delays, the airports that cause the maximum delays and so on.

Conclusion:

The statistics for the two problem statements have been computed and it has helped us to compute the time period having the maximum delays and also the flight carrier having the most delay. This information can be helpful for passengers in making a decision in choosing the timings and the airlines that they want to travel in. Similar kind of statistics can be computed using the vast amount of dataset which can be valuable.

References:

[1] Big Data Analysis by Classification Algorithm Using Flight Data Set, Ujjwala Urkude, University of RGPV , Bhopal.

[2] Analysis of Operational Flight Data in Hadoop using MapReduce and the MATLAB Distributed Computing Server, Lukas Höhndorf Javensius Sembiring, Robin Karpstein, and Florian Holzapfel.