**FLIGHT DATA ANALYSIS USING OOZIE WORKFLOW**

As a part of course CS644: Introduction to big data

Under guidance of Prof. Chase Wu

Done by:

Murali Siddappa Badiger (mb687)

Shishir Sumant (ss3688)

**Oozie Workflow Structure**

START

MapRed Probability Schedule

error

success

MapRed Taxi Time

error

success

KILL

MapRed Cancel Reason

error

success

END

**Algorithm**

There are three problems addressed in the project as follows:

1. Calculate the 3 airlines with the highest and lowest probability, respectively, for being on schedule:

* The CSV file provided as input contains the delay column. The mapper function checks the delay column for every unique carrier. For a delay more than 10 minutes, an entry is made to the context, the key being the unique carrier name and the value as DoubleWritable 1.0. For those cases which do not have a delay, an entry is made to the context with unique carrier name as the kay and DoubleWritable value of 0.0.
* The reducer class is then deployed, it is designed to contain the maximum and minimum three probabilities of flights being on schedule. Reduce class will hold the max 3 and min 3 values of probabilities. Comparisons are done for each unique Carrier acting as the key and the top 3 entries are taken.
* Proceeding the reducer, cleanup function is invoked and displays the Carriers with maximum and minimum probabilities of being on schedule.

1. Calculate 3 airports for having longest and shortest taxi time.

* In the CSV file provided, the mapper first checks Taxi In and Taxi Out time for the source airport and destination airport respectively. Non integer values are skipped. For a positive value, an entry is made to the context with Taxi In and Taxi Out time as values for source airport and destination airport as keys.
* In the reducer function, a total taxi time is calculated from which the average time is obtained. These values are inserted into a tree set inserting them in ascending order. This is done by using the compareTo method.The functions pollfirst() and polllast() provided by the interface is called to display shortest and longest taxi time respectively.
* This is proceeded by invoking cleanup function which displays longest and shortest 3 values of average Taxi In and Taxi Out Time

1. Find the most common reason for flight cancellations.

* In the CSV file provided, the mapper first checks isCancelled column and its corresponding cancellation reason. If the value of isCancelled is non Integer, it is skipped. If the value is 1, an entry is made into context with cancellation reason as key and IntWritable with value 1.
* Reducer class contains cancellation reason with the maximum value.
* Cleanup function then displays the most common reason for flight cancellations.

**Performance measures**

1. Based on number of VMs used against the execution time

Table below shows execution time for different number of nodes for the flight data of all 22 years.

|  |  |
| --- | --- |
| Number of VMs | Execution Time (min) |
| 2 | 98 |
| 3 | 74 |
| 4 | 54 |
| 5 | 40 |

The map-reduce solution is implemented on fully distributed mode using 2,3,4 and 5 nodes.

We noticed that the execution time was 98 minutes when only 2 nodes were used. The execution time however dropped significantly on increasing the number of nodes to 3 by 24 minutes. On increasing the number of nodes to 4 and 5, the execution time further reduced by 20 minutes and 14 minutes. The execution time reduced significantly when number of nodes were increased from 2 to 3. The reason for the decrease in execution time to be not so significant in the other two cases may be due to computation overheads.

The decrease in execution time vs increase in number of nodes can be visualized by the following graph:

1. Based on the size of input provided

The table below shows the variation in execution time with respect to size of the input data provided.

|  |  |
| --- | --- |
| Data files (number of years) | Execution Time (seconds) |
| 1 | 188 |
| 2 | 274 |
| 3 | 483 |
| 4 | 740 |
| 5 | 816 |
| 6 | 868 |
| 7 | 940 |
| 8 | 1012 |
| 9 | 1156 |
| 10 | 1216 |
| 11 | 1256 |
| 12 | 1348 |
| 13 | 1448 |
| 14 | 1540 |
| 15 | 1620 |
| 16 | 1712 |
| 17 | 1856 |
| 18 | 1956 |
| 19 | 2052 |
| 20 | 2212 |
| 21 | 2360 |
| 22 | 2415 |

The above results are when the Hadoop Oozie cluster is running on 5 VMs. The number of files provided are yearly flight data and increases by 1 year for every step. The first row is the data for year 1987, The second row for years 1987-1988 and the last row represents data for years 1987 to 2008 (totally 22 years). We can notice the increase in execution time as the years increase. The graph below represents the increase in computation time as the data increases.