CS 644-101
Introduction to Big Data

FLIGHT DATA ANALYSIS

Team:

Divya Reddy Regatte (dr499@njit.edu)
Pavan Ghuge (pbg7@njit.edu)

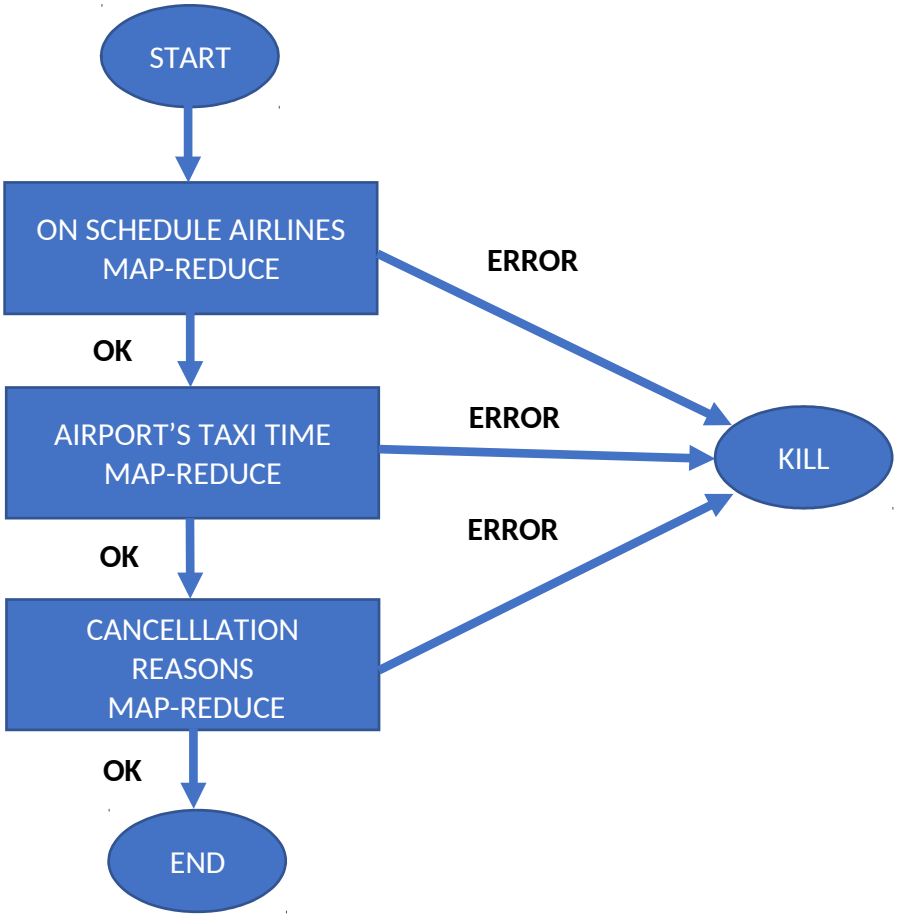
INTRODUCTION:

In this project, we have analyzed the Airline On-time Performance data set (flight data set) from the period of October 1987 to April 2008 from the following website https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/HG7NV7

We have designed and implemented an Oozie workflow for that to solve following 3 problems:

1. the 3 airlines with the highest and lowest probability, respectively, for being on schedule
2. the 3 airports with the longest and shortest average taxi time per flight (both in and out)
3. the most common reason for flight cancellations.

STRUCTURE OF OOZIE WORKFLOW:



The oozie workflow is implemented with three map-reduce jobs that run in fully distributed mode. As per the diagram, the jobs get killed when an error is encountered.

ALGORITHMS:

First Map-Reduce Job: On Schedule Airlines:

1. The Mapper reads the data by splitting the csv file and storing the data in arrays.
2. From the above stored arrays, we fetch three columns:
    a. UniqueCarrier – stores unique carrier code of airline
    b. ArrDelay – arrival delay in minutes
    c. DepDelay – departure delay in minutes
3. At mapper we set the key value of the UniqueCarrier to 1 if the flight is on-schedule and to 0 if the flight has been delayed. The delay time has been set for greater than 10minutes.
4. Mapper <key,value>:<UniqueCarrier,1 or 0>
5. The Reducer calculates the total no of flights and the total number of flights on schedule that is delay time is less than or equal to 10 mins.
6. Reducer <key,value>:<UniqueCarrier,probability>
7. To find the probability of flights on-schedule we count the flights on-schedule and divide it by total flights.
8. Probability = (# of 1) / (# of 1 and 0)
9. Reducer then uses the comparator function do the sorting. After sorting, output the 3 airlines with the highest and lowest probability. These are added to tree sets of a user-defined data type.

Second Map-Reduce Job: Airports Taxi Time:

1. The Mapper reads the data by splitting the csv file and storing the data in arrays.
2. From the above stored arrays, we fetch four columns:
    a. Origin airport code

b. Destination airport code
c. Taxi in – stores time in minutes for destination
d. Taxi out- stores time in minutes for origin

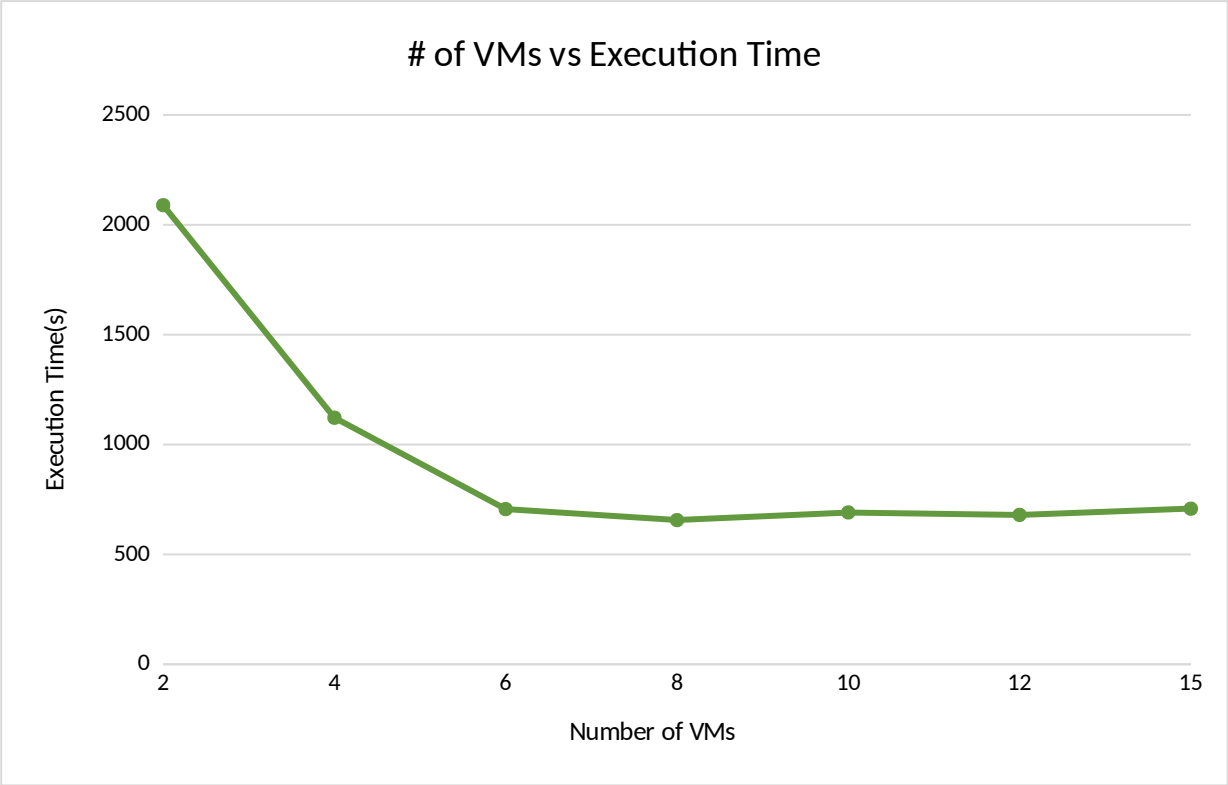
3. Mapper <key,value>: <IATA airport code, TaxiTime>: <Origin,TaxiOut> or <Dest,TaxiIn>
4. Reducer sums the value from the mapper of the same key (normal) and calculates the total number of times this key is found.
5. Average Taxi Time per flight = Total taxi in and out time of airport (taxi in for origin and taxi out for destination) / total number of airlines that flew or landed from that airport.
6. Reducer <key,value>: <IATA airport code, Average TaxiTime>
7. Reducer then uses the comparator function do the sorting. After sorting, output the 3 airports with the longest and shortest average taxi time. Two tree maps are used for flights with highest taxi time and one for lowest taxi time.

Third Map-Reduce Job: Flight Cancellation Reasons

1. The Mapper reads the data by splitting the csv file and storing the data in arrays.
2. From the above stored arrays, we fetch the column cancellationcode. The cancellation codes are : A- carrier, B- weather, C- NAS, D- security.
3. If the value CancellationCode is not NA, we set the key to 1 i.e., output:
   < CancellationCode, 1> .
4. Reducer sums the value from the mapper of the same key.
5. Reducer <key,value>: < CancellationCode, sum of the 1s>
6. The key and values form above step are inserted into tree map.
7. Reducer then uses the comparator function do the sorting. After sorting, output the most common reason for flight cancellations.

PERFORMANCE MEASUREMENT

a) A performance measurement plot that compares the workflow execution time in response to an increasing number of VMs used for processing the entire data set (22 years) and an in-depth discussion on the observed performance comparison results.



We can infer from the above figure that as the number of VM's increases, the execution time of OOZIE workflow will decrease. As the number of VMs increase, the processing ability of the Hadoop cluster will also increase, because then data can be processed in parallel on multiple data nodes. The execution time of map-reduce jobs will also be less than before. However, after a certain point we see that even by increasing the number of VMs there is no significant decrease in the execution time. It remains constant.

b) A performance measurement plot that compares the workflow execution time in response to an increasing data size (from 1 year to 22 years) and an in-depth discussion on the observed performance comparison results.

From the above figure we can infer that, as the size of the data increases, the execution time of the oozie workflow will also increase. The number of VMs is kept constant at 22. At the beginning, the increase in execution time with increase in data is slow, this is due to the fact that there is not much of an increase in the data size. However, after 1998 we see that the time-consuming increase is very fast that is execution time is very high. Hence, the slope becomes much steeper compared to the initial years. The reason for this is that the flight data between 1998 - 2008 has seen a sharp increase. We see that a greater number of people have chosen to travel by plane.