INTRODUCTION TO BIGDATA

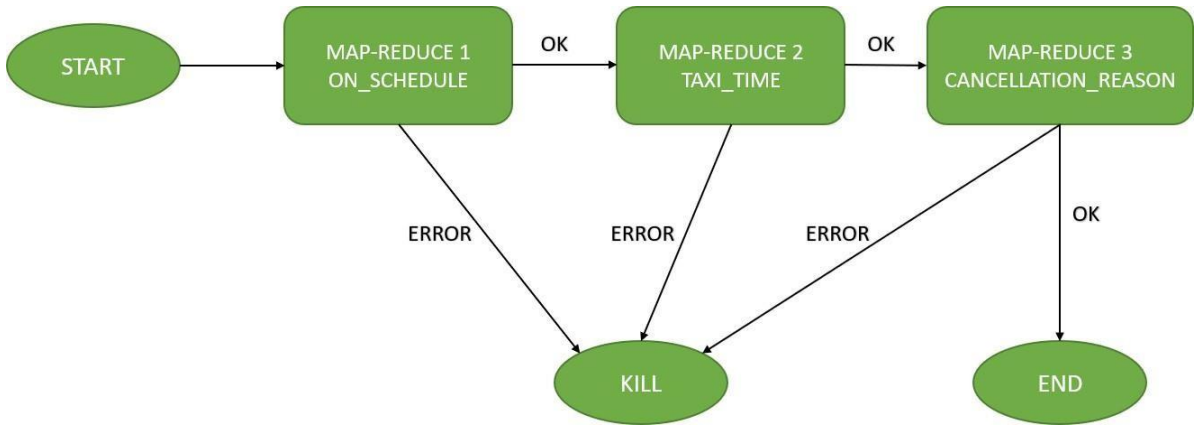
PROJECT REPORT

FLIGHT DATA ANALYSIS

Team:

CHASMINI CHERUKURI
BHOGESWARI SOMISETTI
SAI PRATHYUSHA DEVARAPALLI
MANOJ GANGADARA

OOZIE WORKFLOW



ALGORITHM

The main program is split into 3 Map-Reduce functions as it can be seen from the Oozie Workflow diagram.

Map-Reduce 1: On_Schedule (to find out the 3 airlines with the highest and lowest probability for being on schedule)

1. Define a variable called delayThreshold = 10
2. For each flight, the Mapper program computes the sum of arrivalDelay and departureDelay.
3. If the sum is less than delayThreshold, the flight is considered on Time. Out (UniqueCarrier, 1) to denote that this flight is on Time.
4. Else if the sum is less than delayThreshold, the flight is considered as delayed. Out (UniqueCarrier, 0) to denote that this flight is NOT on Time.
5. At Reducer, for each UniqueCarrier key, count the total number of values as tCount.
6. At Reducer, for each UniqueCarrier key, if value is 1, increment onTime count by one.
7. Compute the on-Time probability as onTime/tCount.
8. Add the (probability, UniqueCarrier) to an ArrayList(flightScheduleList).
9. Repeat steps 5-8 for each UniqueCarrier key received at the reducer.
10. At context cleanup, sort the arraylist(flightScheduleList) in decreasing order of probabilities.
11. Write the arraylist(flightScheduleList) values (UniqueCarrier, probability) to HDFS.

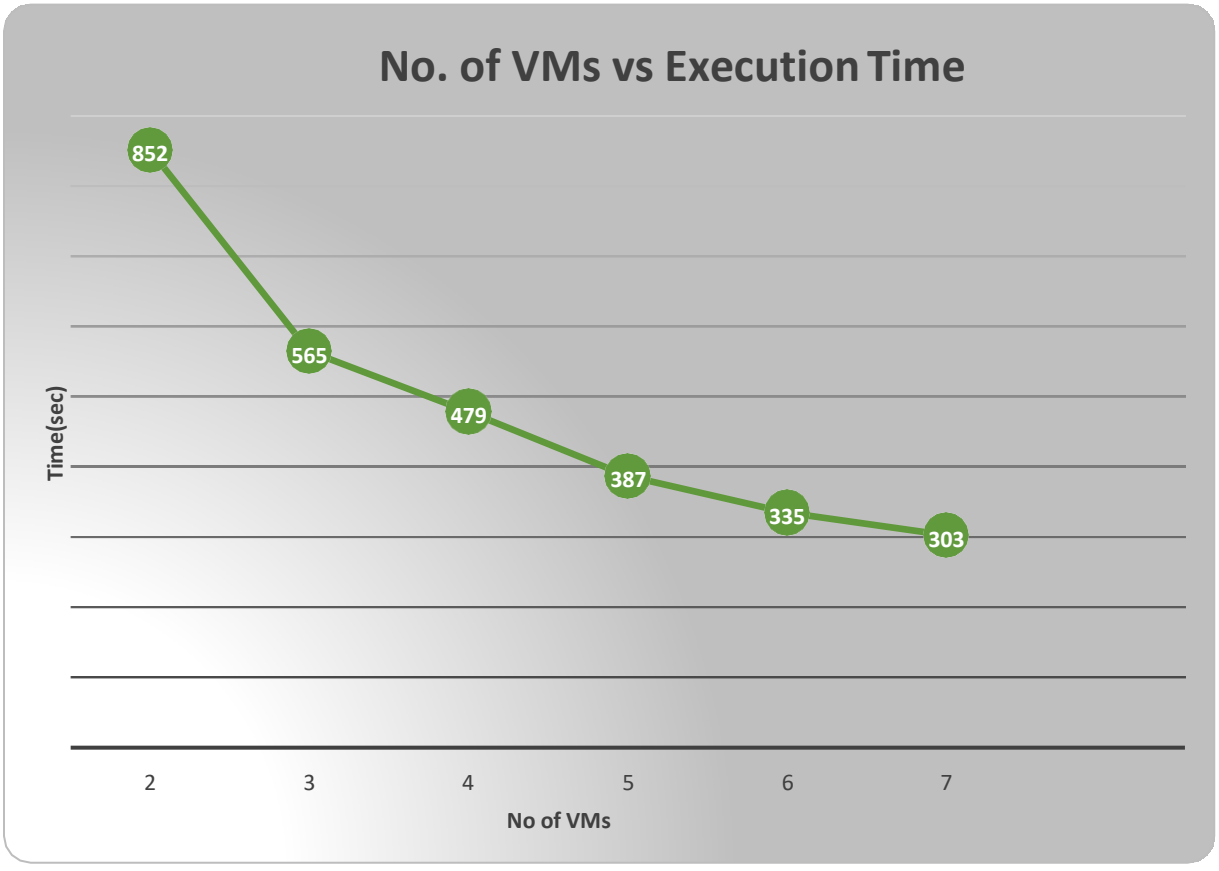
Map-Reduce 2: Taxi_Time (to find out the 3 airports with the longest and shortest average taxi time per flight)

1. For each flight, the Mapper Outs (Origin, TaxiOut) & (Destination, TaxiIn)
2. At Reducer, for each Airport(Origin/Destination) key, count the total number of values as tCount.
3. At Reducer, for each Airport key, compute the total taxi time by adding all the values (TaxiIn/TaxiOut).
4. Compute Average Taxi Time for that Airport by dividing the total taxi time obtained in above step by the tCount.
5. Add the (avgTaxiTime, airportName) to an ArrayList(taxiTimeAvg).
6. Repeat steps 3-5 for each Airport key received at the reducer.
7. At context cleanup, sort the arraylist(taxiTimeAvg) in decreasing order of average taxi times.
8. Write the arraylist(taxiTimeAvg) values (Airport, avgTaxiTime) to HDFS.

Map-Reduce 3: Cancellation_Reason (to find out the most common reason for flight cancellations)

1. For each flight, the Mapper outs (CancellationCode, 1) if the flight is cancelled (Cancelled =1)
2. At reducer, for each CancellationCode key, add all the values to get the tCount.
3. Write (CancellationCode, tCount) to HDFS.
4. Repeat steps 2-3 for each CancellationCode key received at the reducer.

PERFORMANCE MEASURE PLOT (NO OF VMs)



The program was experimented with an increasing number of VMs in each iteration to the Ooziee cluster (2-7). A linear relation can be observed between the number of VMs and the execution time. The execution time is high when a cluster with only 2 VMs are used (852 sec) and it reduces as we add more VMs to the clusters (303 sec).

Hence, we can conclude that, with the increase in number of nodes, there is an increase in the resources available which leads to a decrease in execution time.

PERFORMANCE MEASURE PLOT (YEARS)

For this experiment, a cluster of 7 VMs were used. Once again, a linear relation can be observed between the data size and execution time. The execution time is lower when data of only 1 year is computed (69 sec) and it increases when we gradually add more years to the computation (303 sec).

Hence, we can conclude that, when the computational resources are kept constant, the execution time increases with the increase in data size.