FLIGHT ANALYSIS REPORT

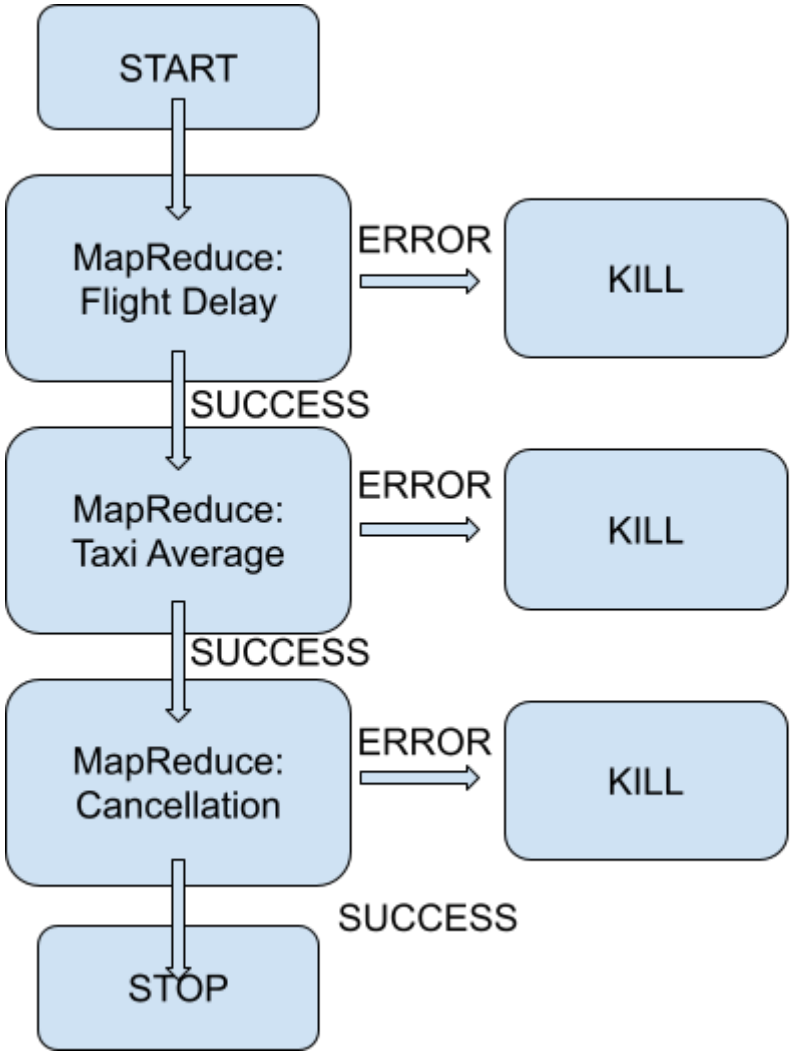
COURSE PROJECT

BY:

Lovkya Sushya Bajaru (lb352)

Pavan Venkat Reddy Bhumula (pb229)

A) STRUCTURE OF OOZIE WORKFLOW:



B) ALGORITHMS

The three airlines with the highest and lowest probability for being on schedule:

MAPPER PHASE:

1) Read input files line by line
2) As it is csv (comma separated file), we split it based on the the comma and store all fields in an array
3) Fetch the values for fields corresponding to airlines (8), arrival delay (14)

and departure delay (15)

4) Now, we find all the airlines and we also count the airlines on time, whose delay (both departure and delay) is less than 10 minutes
5) Write to context <airlines all, 1> and context <airlines ontime, 1>

REDUCER PHASE:

1) Check whether the current key suffix is all airlines and we count them. This gives the total count which includes both delayed and on time airlines. Also we check if this key is the current element and if not we set it to the current element.
2) Now when the airlines suffix is not "all". i. e we get airlines on time. We count them and we divide "airlines on time" with "airlines all" to get the probability of an airline being on schedule.
3) These probability values are inserted into a tree map, and in-order to sort the values based on the probability we use a customized comparator.
4) Two tree maps are utilized for highest probability and another for the lowest probability. We pull out the last and first values when the sorted output size is greater than 3.
5) Write the output to context.

<u>Calculation of 3 airports with longest and shortest average Taxi time (In and Out):</u>

MAPPER PHASE:

1) Read input files line by line.
2) Since it's a comma separated file (cs) we split it based on comma and store all the fields in an array
3) Fetch the values for fields corresponding to origin(16), destination(17), taxiln(19) and taxiOut(20)
4) We check if the taxiln and TaxiOut time is an integer and then we write to context<airportorigin, taxiln> and context<airportdestination, taxiOut>

REDUCER PHASE:

1) Read context. It will receive all the values corresponding to a key.
2) We Iterate over the context values which is a list of taxiln and taxiOut values for

the airport.

3) Calculate sum of all the values and total number of values.
4) Calculate average taxi = sum of all values / total number of values.
5) These average values are inserted into a tree map, and in-order to sort the values based on the probability we use a customized comparator.
6) Create a class OutPair with airportname and average taxi time as instance variables and implement interface Comparable
7) Sort based on average taxi time in the compareTo method
8) Two tree sets are utilized for airports with highest taxi time and another for airports with lowest time. We pull out the last and first values when the sorted output size is greater than 3.
9) Write the output to context.

Find most common reason for cancellation of flight:

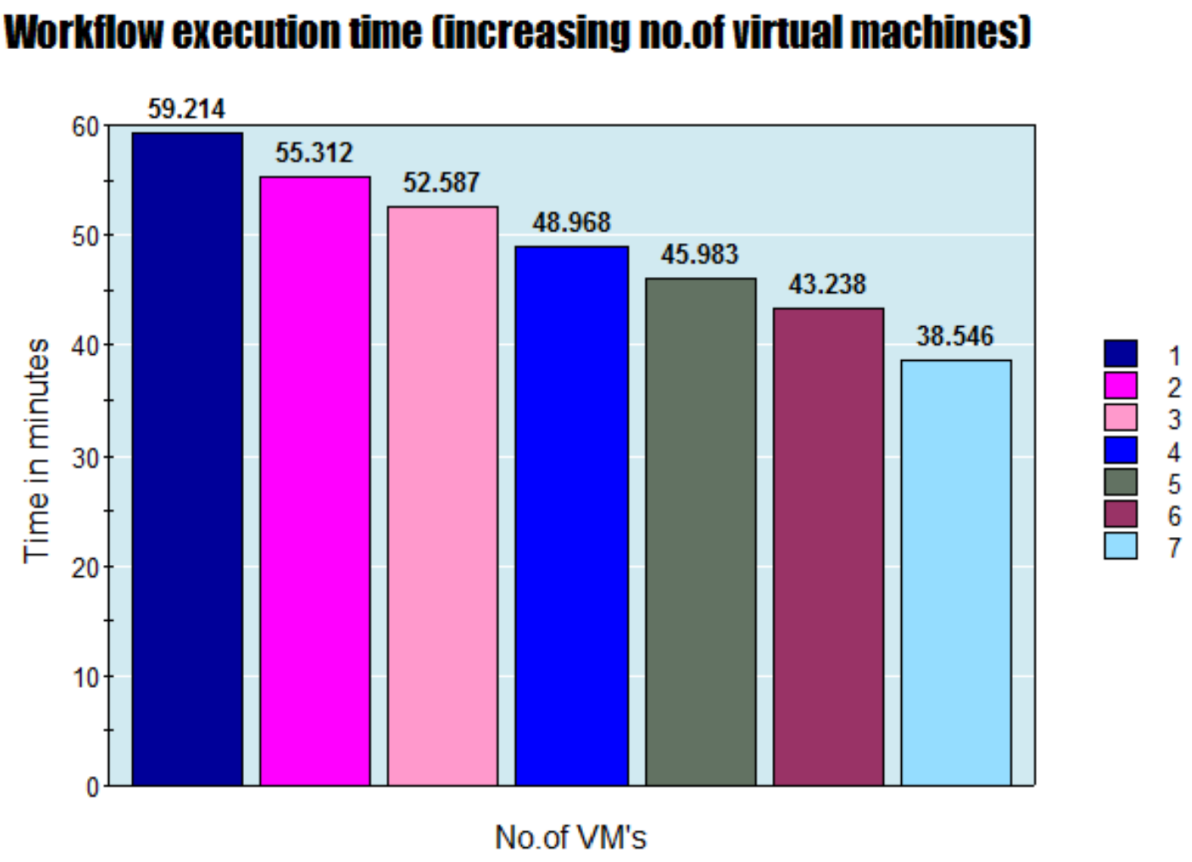
MAPPER PHASE:

1) Read input files line by line.
2) Since it's a comma separated file(cs) we split it based on comma and store all the fields in an array
3) Fetch the values for fields corresponding to cancellationCode column (22)
4) If cancellationCode doesn't correspond to " " and "CancellationCode" and "NA' we select such cancellation codes(i.e A, B,C, D)
5) Write to context<cancellationCode, 1>

REDUCER PHASE:

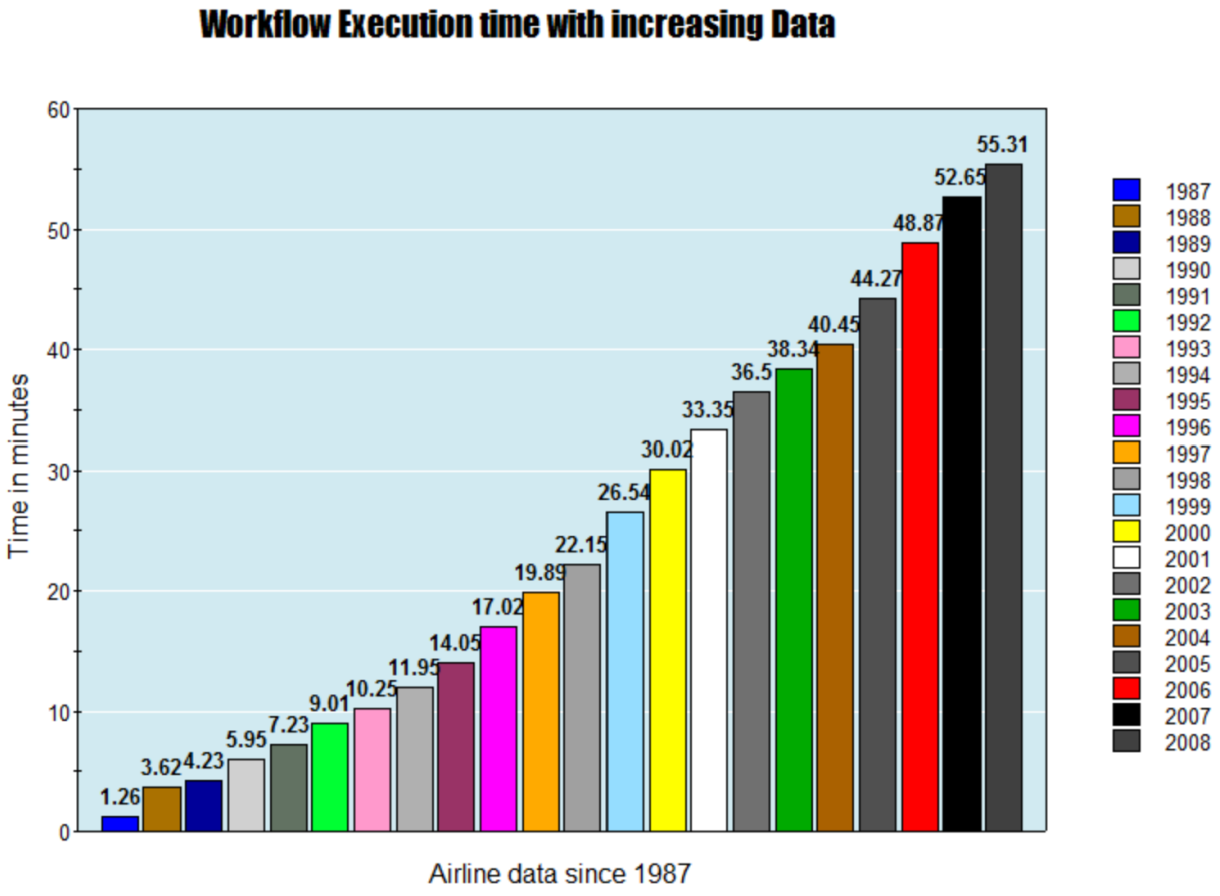
1) Read context. Each Combiner will receive all the values corresponding to a key.
2) Iterate over the context values and Calculate sum of all the values.
3) These values and the key are inserted into a tree map, and in-order to sort the values based on the highest no of values we use a customized comparator.
4) We pull out the last value when the sorted output size is greater than 1.
5) Write the output to context.

C) A Performance measurement plot which gives the workflow execution time vs no.of VMs used for processing the entire dataset (22 years)



- Here we are doing an experiment on performance of workflow having mapreduce jobs by varying the number of resources used.
- We are keeping data constant for all the runs i.e. flight data for all 2 years.
- We are starting by using Hadoop on 1 Virtual machine.
- The total execution time taken is 59.214 mins.
- Now we increase VM one at a time. We notice that as we increase the number of VMs there is a significant drop in time taken for execution.
- Hence, we conclude that performance and number of resources used for processing big data are directly proportional.

D) A performance measurement plot that which gives workflow execution time Vs increasing data size (from year 1 to year 22 i.e., 1987 to 2008)



- In this experiment, we want to find out performante with respect to varying input data.
- We are using 2 Virtual machines throughout this experiment. First, we execute workflow on only one data file (1987.cs).
- We see that execution completes in very negligible time.
- Now, we increase data by one year for every run and record the execution time. We observe that the execution time gradually increases as the input data increases.
- Hence, we can conclude that performance and input data size are inversely proportional.