FLIGHT DATA ANALYSIS

CS 644

TERM PROJECT

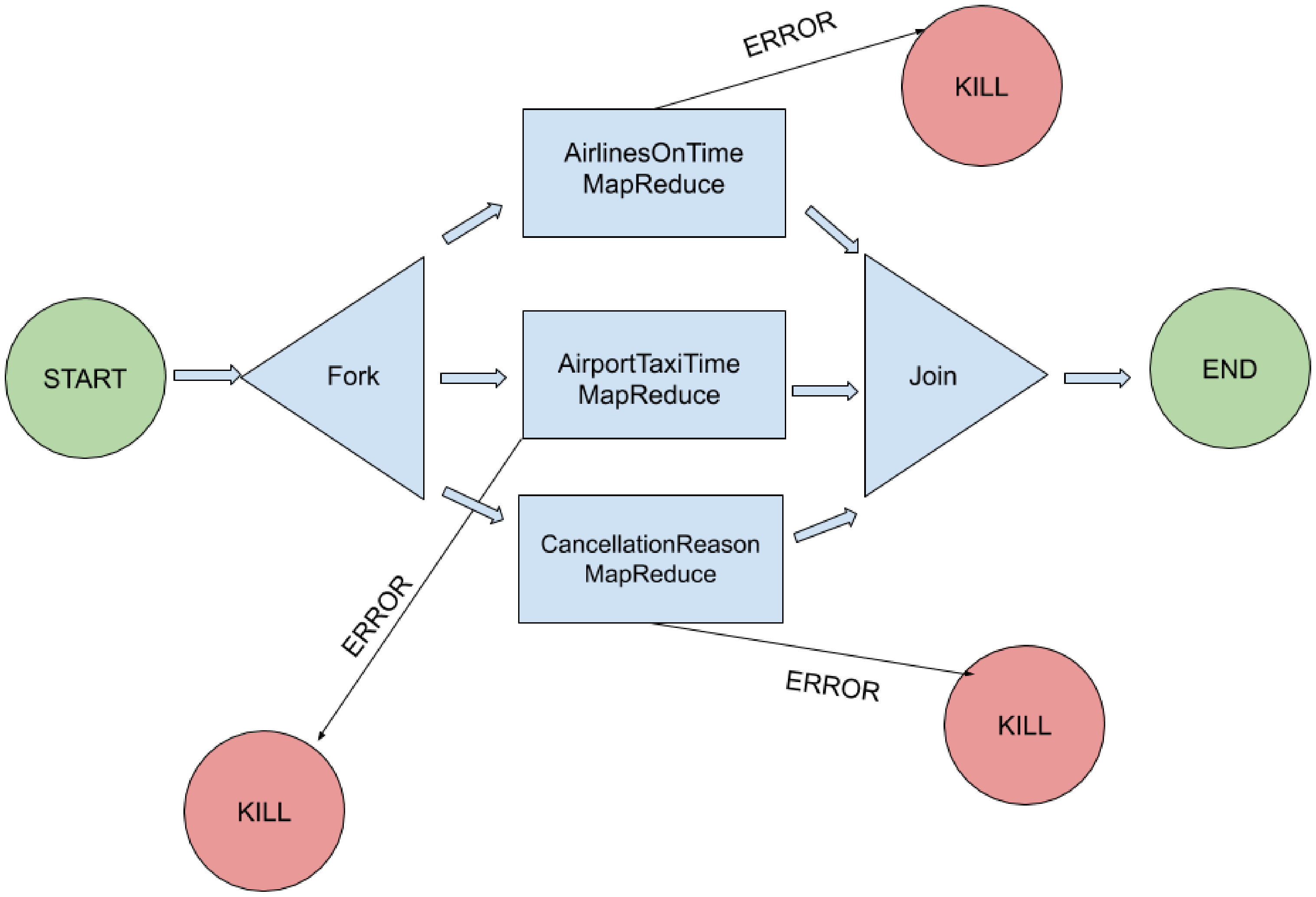
SPRING 2020

PREPARED BY

RAYON MYRIE

AMI PATEL

Oozie Workflow Diagram:



Algorithm Description:

AirlinesOnTime MapReduce

Uses mapreduce to find the three airlines with highest and lowest probability of being on time.

Mapper (Key - Unique Carrier Code; Value - 0 or 1)

1. Read each line of data and ignore any NA values.
2. Use values in column indices 14 (ArrDelay) and 15 (DepDelay). An 'On Time' flight is defined as a flight with a total delay of less than or equal to 10 minutes.

3. Write the key value pair with the key from column 8, the unique carrier, and value of 0 if the total is more than 10 minutes and 1 otherwise

Reducer (Key - Unique Carrier Code; Value - probability of being on time)

1. Use the mapper results and count the number of 1's as well as the total number of flights for each carrier code
2. Calculate the probability of the airline being on time:
$$P = \frac{\#\ on\ time}{total\ flights}$$
3. Using the comparator class, sort the probabilities in decreasing order
4. Output the highest and lowest three airline carrier codes and their probabilities

AirportTaxiTime MapReduce

Uses mapreduce to identify the airports with the three longest and shortest taxi times.

Mapper (Key - IATA airport code; Value - taxi in/out time)

1. Read each line of data and ignore any NA values in column indices 19 (TaxiIn) and 20 (TaxiOut)
2. For each of these columns, write the origin (column 16) and the taxi out time or the destination (column 17) and the taxi in time as the key value pair.

Reducer (Key - IATA airport code; Value - average taxi time)

1. Use the mapper results and sum the taxi times (both in and out together) and also count the number of occurrences of that airport.
2. Calculate the average taxi time for the airport:
$$Avg = \frac{sum\ tazi\ times}{total\ occurrences}$$
3. Using the comparator class, sort the average times in decreasing order
4. Output the airports with the longest and shortest taxi times with the taxi time

CancellationReason MapReduce

Uses mapreduce to find the most common reason for flight cancellations.

Mapper (Key - Cancellation Code; Value - 1)

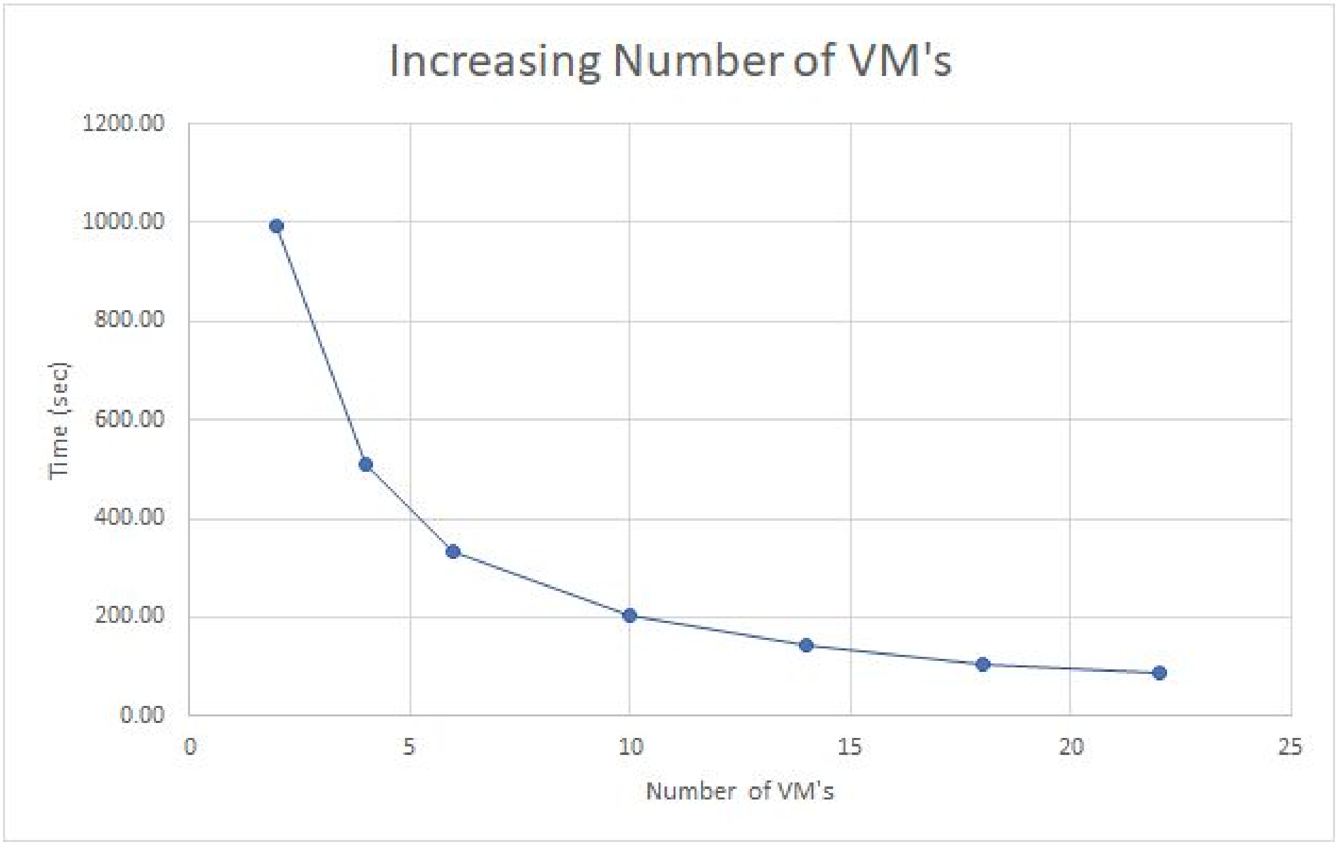
1. Read each line of data and check for proper format in column index 21 (Cancelled)
2. For each line with a "1" in the Cancelled column, indicating that the flight had indeed been cancelled, the reason in column index 22 (CancellationCode) is used as the key and the value is 1.

Reducer (Key - Cancellation Code; Value - total occurrences)

1. Use the mapper results to find the sum of occurrences for each cancellation cde
2. Find the maximum of the total values
3. Output the maximum as the most common reason for cancellation

Performance Measurements:

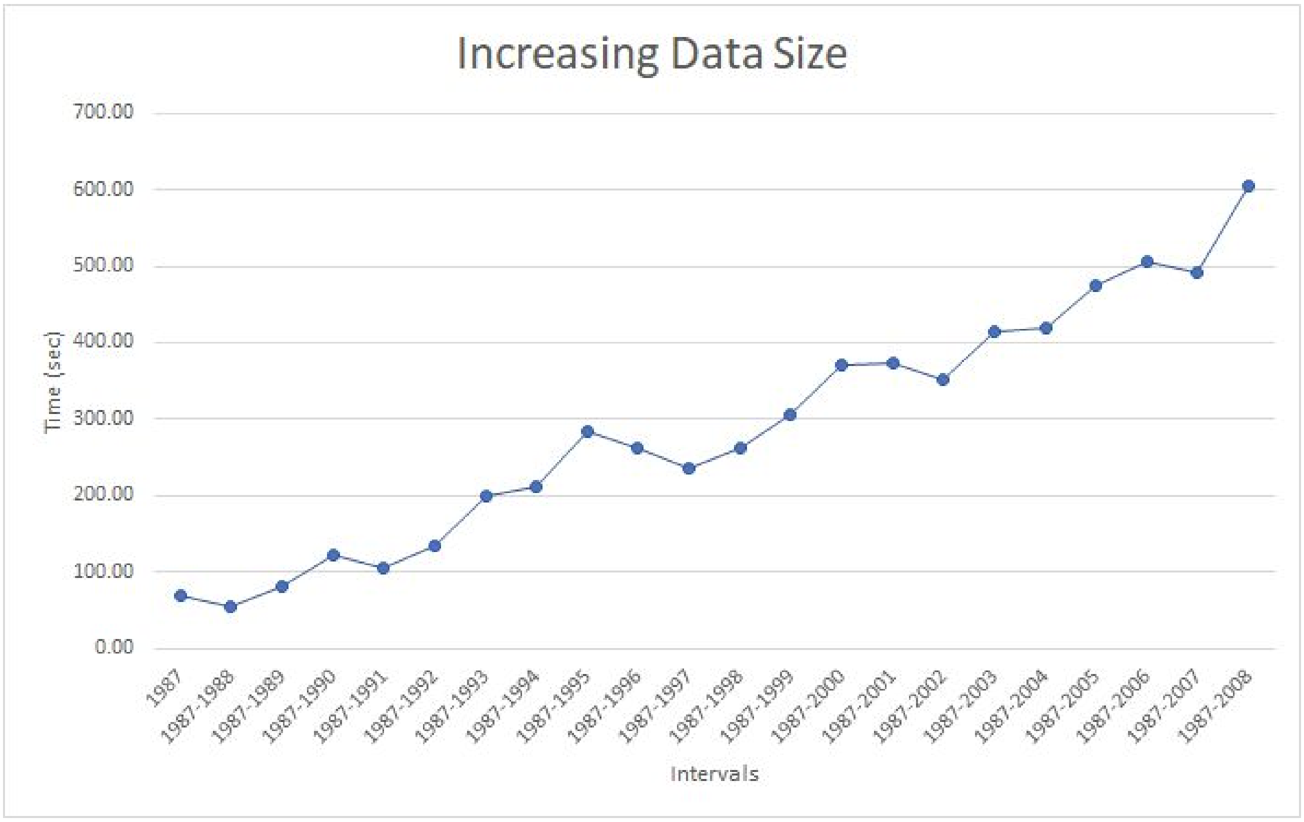
Execution Time vs. Number of VM’s



Discussion

To compare the execution times for different number of VM instances, all 22 years of data was used with VM’s varying from 2 to 23. It is clear from the plot that a higher number of VM’s results in a low execution time. This is due to the number of parallel processing running at a time. When there are more VM instances, there are more data nodes that can process the data and run mapreduce simultaneously, lowering the runtime. The overall time difference is significant between the minimum and maximum number of allowed instances. The largest drop in runtime occurs when the number of instances is increased from 2 to 4, indicated by the initially high slope of the curve. After this point, the runtime does decrease, but not as significantly. This can be attributed to the amount of communication occurring when there is a high number of VM instances. Even though data is still being processed in parallel, each VM has so many more other VM’s to communicate with, which limits the extent to which execution time can actually be minimized.

Execution Time vs. Data Size



<u>Discussion</u>
To test the effects of increasing data size on execution time, one year of airport data was added to each testing interval and the runtime was recorded. As can be seen in the figure above, the overall conclusion would be that a larger data size results in a higher execution time. This is because, since the amount of resources allotted to this job are fixed, as the size of the data increases, more processing time is required to get the output. The execution times do not increase at a steady rate between each interval and this can be due to the fact that all of the file sizes are not equal; some years' files may be larger or smaller than others, causing slight anomalies in the rate at which the execution time increases.