FINAL PROJECT REPORT

FLIGHT DATA ANALYSIS

COURSE: CS644

INTRODUCTION TO BIG DATA

Submitted By

YATISH CHANDRA EMMANI

(NJIT ID - 31572255)

and

SHARUKYA SMITESH MARNENI

(NJIT ID – 31570763)

ALGORITHM

Calculating the 3 airlines with the highest and lowest probability, respectively, for being on schedule.

Mapper Phase:

1. **Read through the input files line by line.**
2. **Separate lines by commas and put all fields in an array**
3. **Retrieve the values for the airline, arrival delay, and departure delay variables.**
4. **We're looking at two counts here. One for counting the airlines with arrival and departure delays of 10 minutes or less, and another for calculating the overall number of airlines. Later, the probability is calculated by dividing these two counts.**

Combiner Phase:

1. **Understand the context. Each Combiner will get all of the values associated with a key.**
2. **Compute the sum of all the airlines (excluding those that are delayed) and airlines total (all airlines)**

Reducer Phase:

1. **Determine if the current key is the airline's total.**
   a) **If true, return the total number of airlines and airlines total. Then divide these numbers to get the likelihood.**
   b) **If no, then associate the current key with the airlines and calculate the total number of airlines (filtered)**
2. **Once the probability is determined, the probability values are entered into a tree map, where a custom comparator is developed to sort the data based on probability.**
3. **We have two different tree maps. One for the highest likelihood and one for the lowest.**

4. **Write the output to context.**

The Determine the three airports with the longest and least average taxi time per flight.

Mapper Phase:

1. **Read through the input files line by line.**

2. **Separate lines by commas and put all fields in an array**

3. **Retrieve the values for the origin, destination, taxiIn, and taxiOut fields.**

4. **Write to context<origin, taxiOut> and context<destination, taxiIn>**

Reducer Phase:

1. **Read the context in the reducer phase. Each Combiner will get all of the values associated with a key.**

2. **Iterate over the context, the airport's taxiIn, and taxiOut values.**

3. **Determine the sum of all values and the total number of values.**

4. **Determine the average taxi: sum/number of values.**

5. **Create a class AirportTaxiTime with instance variables for airportname and average taxi time, and implement an interface comparable.**

6. **In the compareTo method, sort by average cab time.**

7. **Make two tree sets, one for airports with the highest taxi and for airports with the lowest taxi.**

8. **We only want the top three and bottom three results, so we delete the rest via Treeset's pollFirst and pollLast methods.**

9. **Save the output to the context.**

Calculate the most common reason for cancellation of flight

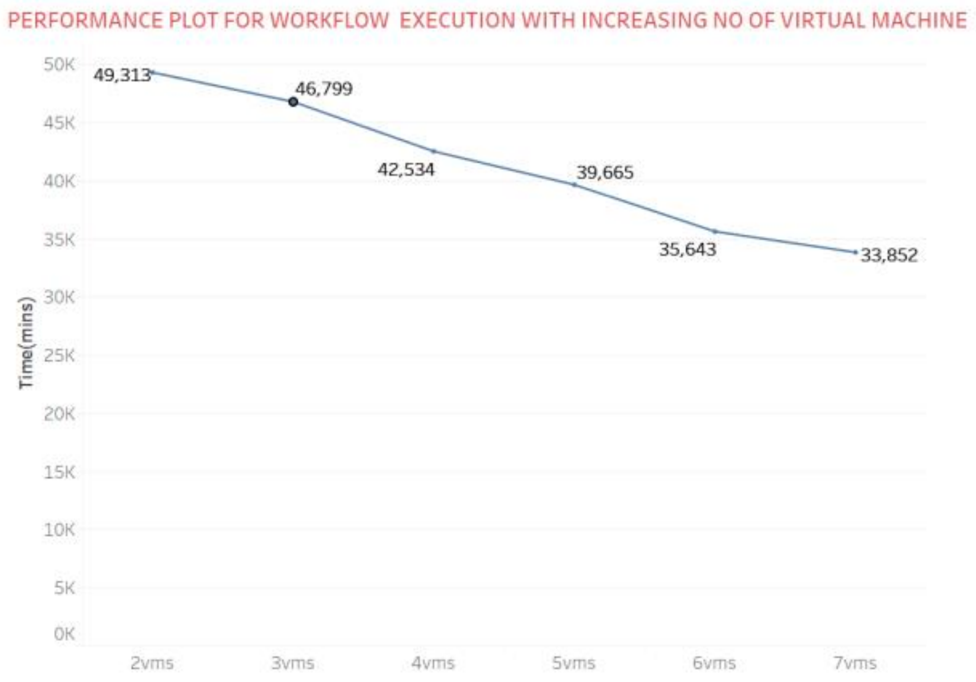
Mapper Phase:

1. **Read input files line by line.**
2. **Split line based on comma and store all fields in an array**
3. **Fetch the values for fields corresponding to cancellationCode column**
4. **If cancellationCode corresponds to either A, B, C or 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. **Create a class with cancelCode and count as instance variables and implement interface comparable.**
4. **Add the class objects to Treeset in order to maintain a sorted order.**
5. **Write the topmost reason to context**

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)



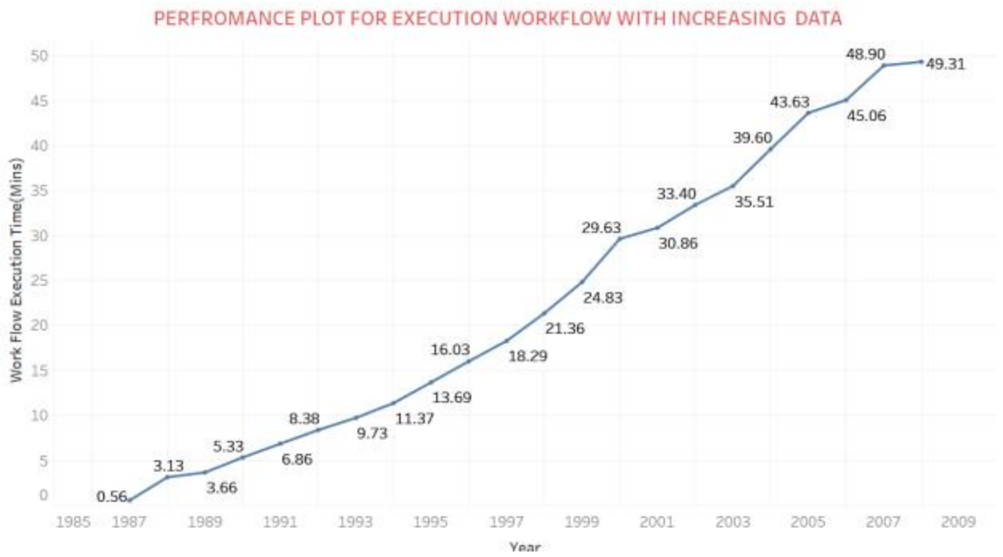
We are testing the performance of a workflow using MapReduce tasks by altering the number of resources needed. We maintain consistent data for all runs, i.e., flight data for all 22 years.

We will begin by running Hadoop on two virtual computers. The entire execution duration is 49.3 minutes. Following that, we increase VM one at a time. We note that the execution time decreases significantly when the number of VMs increases.

CONCLUSION:

As a result, we infer that performance is proportional to the number of virtual machines (resources) employed to analyze massive data.

A performance measurement plot that compares the workflow execution time in response to an increasing data size (from 1 year to 22 years)



We want to see how performance changes as the input data changes in this experiment. Throughout this experiment, we will be using two virtual machines.
First, we run the procedure on a single data file (1987.csv). We can observe that the execution takes very little time.
Then, we increase the data by one year for each run and report the execution time. The execution time continuously increases as the input data grows larger.

CONCLUSION:

As a result, we may deduce that our algorithm's performance for the three specified tasks is inversely proportional to input data.