Flight Data Analysis

Jaime Corea, Racheal Antonita, Soumya Sharma, Surbhi Gupta

Introduction

Our team has developed an Oozie workflow to process and analyze a significant volume of flight data for this project. We use three MapReduce programs and run them in fully distributed AWS EC2 clusters. We first execute the workflow on two VMs to evaluate the whole data set (spanning 22 years from 1987 to 2008), then gradually expand the system size to the maximum allowed number of VMs for at least 5 increment steps, measuring each workflow execution time. On the maximum allowed number of VMs, run the workflow to analyze the data in a progressive manner with a 1 year increment, i.e. the first year (1987), the first 2 years (1987-1988), the first 3 years (1987-1989),..., and the total 22 years (1987-2008), and measure each corresponding workflow execution time.

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

(1) MapperOne

We can use them together as the key because each combination of airline carrier and flight number is unique. For each flight, Mapper One adds the Arrival Delay and Departure Delay (each row in the table).
When this amount exceeds the threshold time (15 minutes), the flight is considered to be behind schedule. Then put context (total, 1) and context (total, 1). (count , 1). In addition, the number of delayed flights and total flights for each airline carrier will increase by one. Otherwise, if the sum of Arrival Delay and Departure Delay is less than the threshold, indicating that the aircraft is on time, we must use context write to increase the on time flight by one (count , 1). This will not affect airline carriers' delayed flights and will simply increase the total number of flights by one.

(2) ReducerOne

The total number of flights for each airline carrier, as well as the total number of flights that are not on schedule for this carrier, are required to calculate the on schedule probability. The number of total flights and not on schedule flights for each carrier are then counted. The delayed probability for each carrier can then be calculated by dividing the two count numbers, i.e. the number of not on scheduled flights by the total number

of flights. This reducer stores the delayed probabilities in a linked list. The airlines with the highest likelihood of being on time can be found by sorting this linked list in ascending order, while the airlines with the lowest probability of being on time can be found by sorting this linked list in descending order. Maintain two size 3-sorted lists, one for the highest percentage of on-time flights and the other for the lowest percentage of on-time flights.

2. Find the 3 airports with the longest and shortest average taxi time per flight (both in and out), respectively

(1) MapperTwo

MapperTwo uses the airport code as the key and the taxi time as the value. It generates the arrival and departure taxi times (Origin, taxiTime) as well as the taxi out time (Destination, taxiTime).

(2) ReducerTwo

ReducerTwo adds the taxiIn and taxiOut values to get the total taxi time (in and out time) for each airport. ReducerTwo also keeps track of the overall number of flights for each airport. The average taxi time can then be computed by dividing total taxi time by the number of flights. The value will then be stored in a linked list. The airports with the shortest average taxi time can be found by sorting this linked list in ascending order, while the airports with the longest average taxi time can be found by sorting this linked list in descending order. Maintain two size 3 sorted lists to contain the highest and lowest average taxi time at the airport.

3. Find the most common reason for flight cancellations

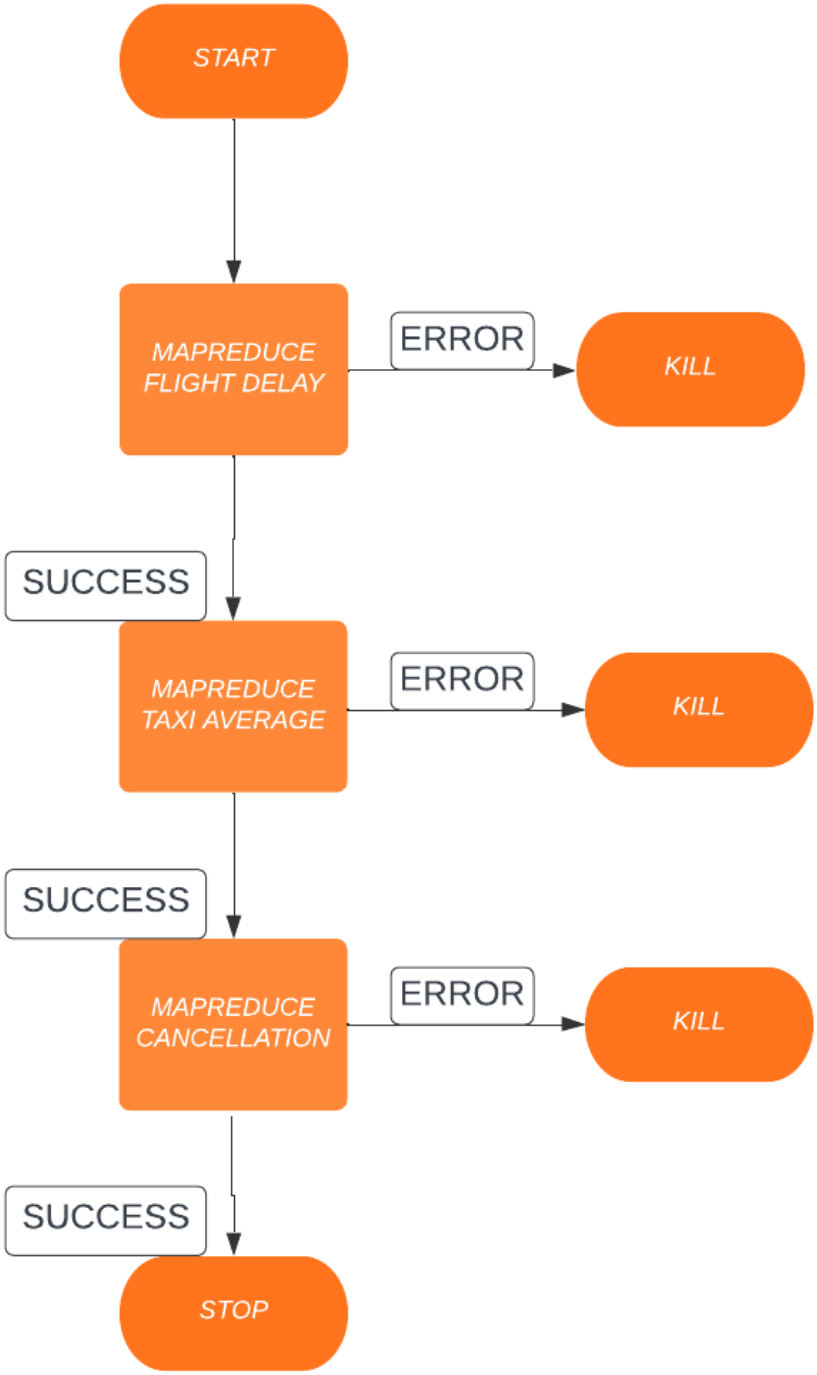
(1) MapperThree

When a flight is canceled, MapperThree counts this cancellation reason by context writing the cancellation code and one.

(2) ReducerThree

This Reducer is used to determine the most common cause of flight cancellations. All we have to do now is keep track of which cancellation code has the most cancelled flights. Count the number of flights that were cancelled for the same cause (same cancel code), and compare the number to the current code with the most cancelled flights. Replace the cancel code with this count number and make it the new max number if the count number is greater than the existing maximum number. The most prevalent reason for flight cancellations can be discovered this way.

A) Workflow Structure



B) Algorithms:

The three airlines having the highest and lowest chances of being on time, respectively.

Mapper Phase:
1. Go through the input files line by line.
2. Because it's a comma-separated file (csv), we split it into fields and put them all in an array.
3. Retrieve the values for the fields airlines(8), arrival delay(14), and departure delay(15) (15).
4. Now we find all the airlines (all airlines) and count the airlines (on time airlines) with a delay of less than 10 minutes (departure and arrival delay).
5. Write to context<airlines all, 1> and context<airlines ontime, 1>.

Reducer Phase:
1.Count the number of times the current key suffix is all (airlines all). This tells you the overall number of flights, which includes both delayed and on-time flights. We also check if this key is the current element, and if it isn't, we change it.
2.Now, when the suffix of the airline is not "all," we get airlines on time. To calculate the chance of an airline being on time, we count them and divide "airlines on time" by "airlines all."
3. These probability values are entered into a tree map, and we use a modified comparator to sort the data by likelihood.
4.For the highest likelihood and the lowest probability, two tree maps are used. When the sorted output size exceeds 3, we extract the last and first values.
5. Add the output to the context.

Calculation of the longest and shortest average taxi times for three airports (In and Out)

Mapper Phase:
1. Go through the input files line by line.
2. Because it's a comma separated file (csv), we split it into fields and put them all in an array.
3. Retrieve the values for the variables origin(16), destination(17), taxiIn(19), and taxiOut(20) (20)
4. If the taxiIn and taxiOut times are both integers, we write to contexts airportorigin, taxiIn> and airportdestination, taxiOut>.

Reducer Phase:
1. Understand the context. It will be given all of the values that correspond to a certain key.
2.We iterate over the context values, which are a list of airport taxiIn and taxiOut values.
3. Add together all of the values to get the total number of values.
4. Determine the average taxi by multiplying the sum of all values by the entire number of values.
5.These average values are entered into a tree map, and we utilize a modified comparator to sort the data by probability.

6. Create a class OutPair and implement an interface with airportname and average taxi time as instance variables. Comparable
7. In the compareTo method, sort by average cab time.
8. Two tree sets are used for airports with the longest taxi times and one for airports with the shortest taxi times. When the sorted output size exceeds 3, we extract the last and first values.
9. Write the output to context.

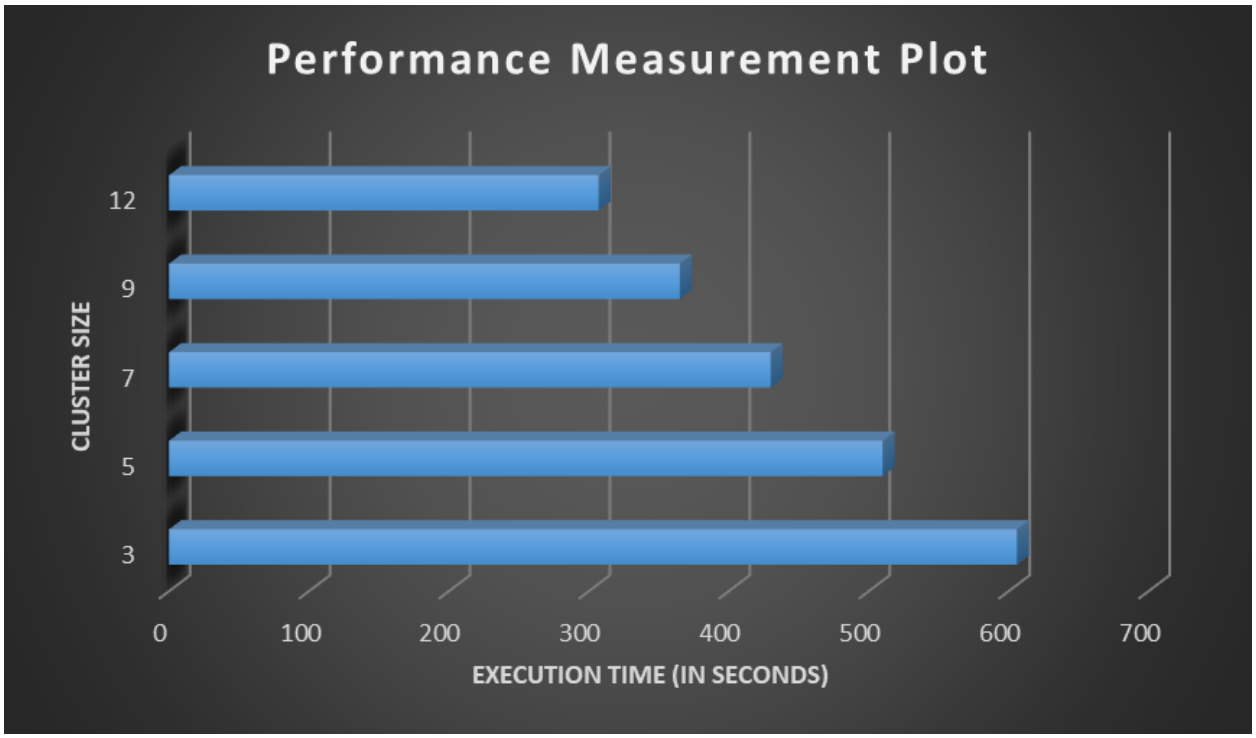
Discover the most common reasons for flight cancellation.

Mapper Phase:
1. Go through the input files line by line.
2. Because it's a comma separated file (csv), we split it into fields and put them all in an array.
3. Get the values for the fields that relate to cancellation.
Column for code (22).
4. If cancellationCode does not match " ", " CancellationCode," or "NA," we choose other cancellation codes (i.e A,B,C,D)
5. Add <cancelationCode, 1> to the context

Reducer Phase:
1. Each Combiner will receive all of the values associated with a certain key.
2. Calculate the sum of all the values by iterating through the context values.
3. These values and the key are entered into a tree map, and we use a customized comparator to sort the data depending on the highest number of values.
4. When the sorted output size is more than 1, we extract the last value.
5. Add the output to the context.

C) A performance metric is a numerical representation of an activity's labor and its outcomes. For 22 years, a performance assessment plot contrasts workflow execution time in response to an increasing number of virtual machines employed to analyze the whole data set. The cluster size rises from 3 to 12 nodes in this case.

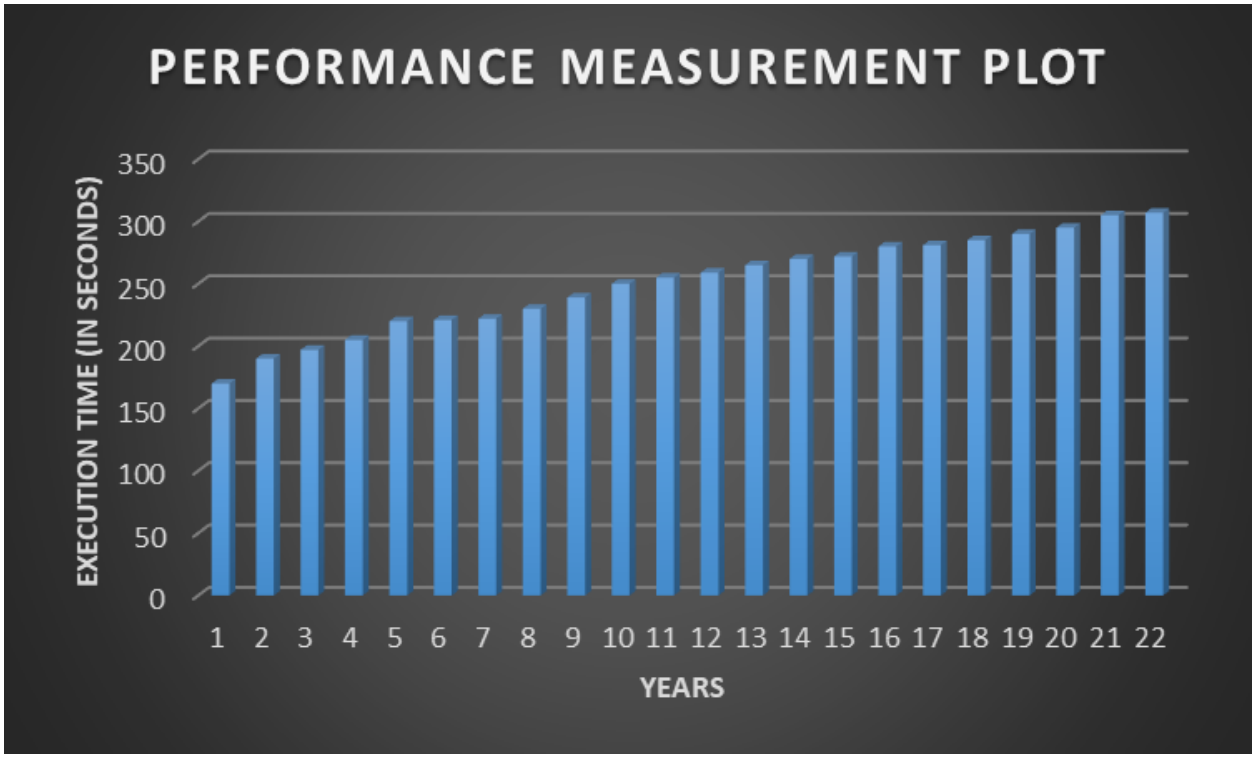
By altering the number of resources used, we are testing the performance of a workflow with MapReduce tasks. For all of the runs, we're using flight data from the last 22 years. We note that when the number of clusters increases, the execution time decreases significantly. As a result, we infer that large data processing performance and the number of resources consumed are closely related.



Cluster Size	3	5	7	9	12
Execution Time (In Seconds)	606	510	430	365	307

D) From the first year to the 22nd year, a performance measurement graphic compares process execution time as data size grows. With 12 nodes, the dataset is changed from 1 to 22 years. We want to see how performance changes as the input data changes in this experiment.

To begin, we run the procedure on only one data file1987, the first year. We can observe that the execution takes very little time. For each run, we now increase the data by one year and note the execution time. We see that as the input data is larger, the execution time increases. As a result, we may deduce that performance and the amount of the input data are inversely related.



Years	Execution Time (In Seconds)
1	170
2	190
3	197
4	205
5	220
6	221
7	222
8	230
9	239
10	250
11	255
12	259

13	265
14	270
15	272
16	280
17	281
18	285
19	290
20	295
21	305
22	307