Flight Data Analysis

Ming Wu, Zhang Hongyang

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

(1) MapperOne

As each combination of airline carrier and flight number is unique, thus we can make them together as the key here. MapperOne calculate the sum of ArrDelay and DepDelay for each flight (each row in the table). When this sum is greater than the threshold time (15 min), then this flight can be seen as not on schedule. Then context write (total <carrier>, 1) and context write (count <carrier>, 1). Also each airline carrier's delayed flights and total flights will increase by 1. Otherwise, when the sum of ArrDelay and DepDelay is not larger than the threshold, which means the flight is on schedule, then we need to increase the on time flight by 1 with context write (count <carrier>, 1). This will not change the delayed fights of airline carriers and will only increase its total flight number by 1.

(2) ReducerOne

To calculate the on_schedule probability, the total flights number of each airline carrier and the total number of flights which are not on_schedule of this carrier are needed. Then for each carrier, its total flights number as well as its not_on_schedule flight number are counted. Then here, by dividing the two count number, i.e. the number of not_on_schedule flights and the number of total flights, the delayed probability can be obtained for each carrier. In this reducer, this delayed probabilities are stored in linkedlist. Then by sorting this linkedlist in ascending order we can get the airlines with highest probability for being on schedule, by sorting this linkedlist in desending order we can get the airlines with lowest probability for being on schedule. Maintain two size 3 sorted lists to sort highest percentage of on schedule flights and lowest percentage of on schedule flights.

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

(1) MapperTwo

MapperTwo use airport code as key and corresponding taxi time as value. It produce the Origin airport and departure taxi time(Origin, taxiTime) and taxi out time (Dest, taxiTime).

(2) ReducerTwo

ReducerTwo calculate the total taxi time (in time and out time) for each airport by adding taxiIn and taxiOut values. Also ReducerTwo count the total number of flight for each airport. Then the avg taxi time can be calculated from the two values by dividing total taxi time by number of flights. Then this value will be saved in a linkedlist. Then by sorting this linkedlist in ascending order we can get the airports with shortest average taxi time, by sorting this linkedlist in desending order we can get the airports with longest average taxi time. Maintain two size 3 sorted lists to store highest average taxi time airport and lowest average taxi time airport.

3. Find the most common reason for flight cancellations

(1) MapperThree

MapperThree count this cancellation reason by context write the cancellation code and 1 when a flight is cancelled.

(2) ReducerThree

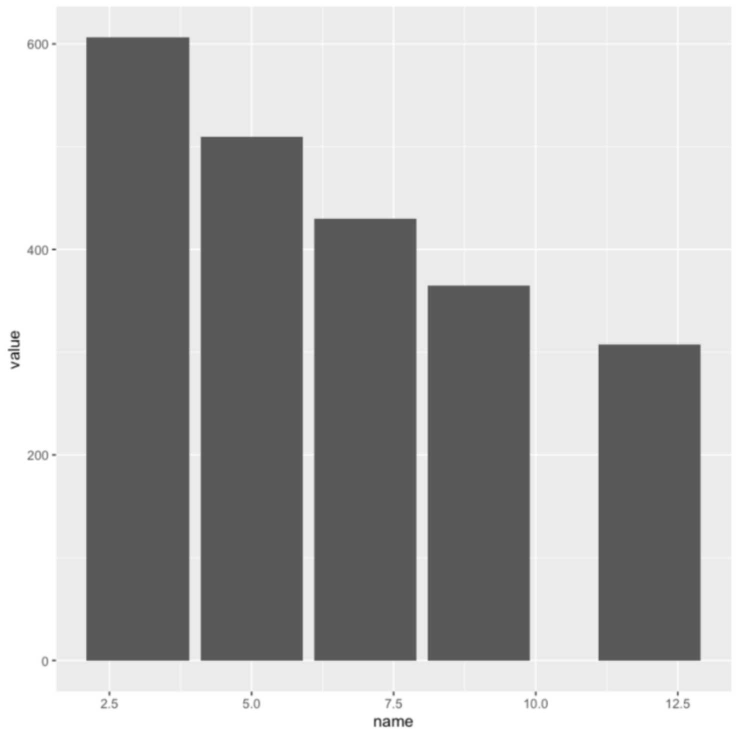
This Reducer is used to find the highest reason for flight cancellations. All we need to do is to maintain a cancellation code with the highest number of cancel flights. Here count the number of flights which cancelled for the same reason (same cancel code), and when the number increased it needs to compare with the current code with highest cancel flights. If the count number is larger than current highest number then replace the cancel code and make this count number as the new max number. By this way the most common reason for flight cancellations can be obtained.

4. Performance measurement

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)

Increase the cluster size from 3 to 12 nodes

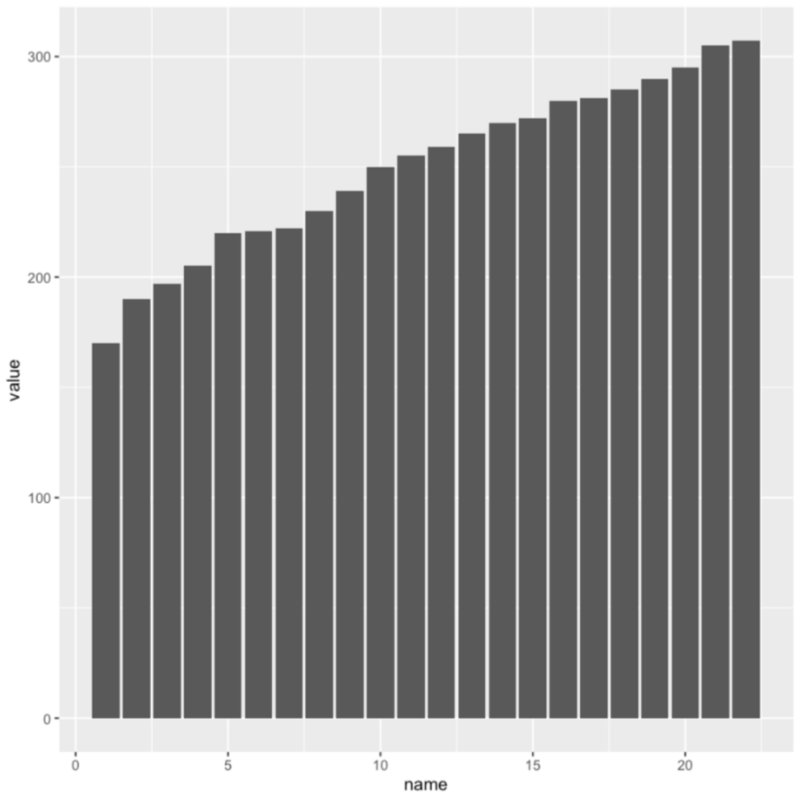
Cluster Size	**3**	**5**	**7**	**9**	**12**
Execution Time	10 min 6 s	8 min 30 s	7 min 10 s	6 min 5 s	5 min 7 s



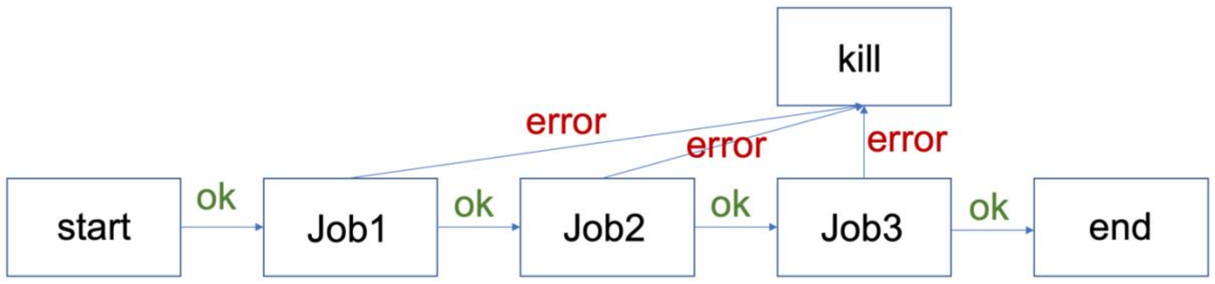
A performance measurement plot that compares the workflow execution time in response to an increasing data size (from 1 year to 22 years)

Changing dataset from 1 year to 22 years with 12 nodes

Years	Time
1	2 min 50 s
2	3 min 10 s
3	3 min 17 s
4	3 min 25 s
5	3 min 40 s
6	3 min 41 s
7	3 min 42 s
8	3 min 50 s
9	3 min 59 s
10	4 min 10 s
11	4 min 15 s
12	4 min 19 s
13	4 min 25 s
14	4 min 30 s
15	4 min 32 s
16	4 min 40 s
17	4 min 41 s
18	4 min 45 s
19	4 min 50 s
20	4 min 55 s
21	5 min 5 s
22	5 min 7 s



5. Workflow structure graph:



kill
error
error
error
start
ok
Job1
ok
Job2
ok
Job3
ok
end