Project of CS 644: Introduction to Big Data
Flight Data Analysis

Fall 2016

Submitted by : Nalinaksh Gaur

UCID: ng294

Contents

Introduction and Overview 3

Oozie Workflow 4

Algorithms Description 5

Performance Measurement Plot -1 6

Performance Measurement Plot -2 7

Introduction and Overview

In this project, an Oozie based workflow is developed which is used to execute 3 mapreduce programs to analyze the flight data. The flight data analyzed can be downloaded from the link below:

http://statcomputing.org/dataexpo/2009/the-data.html

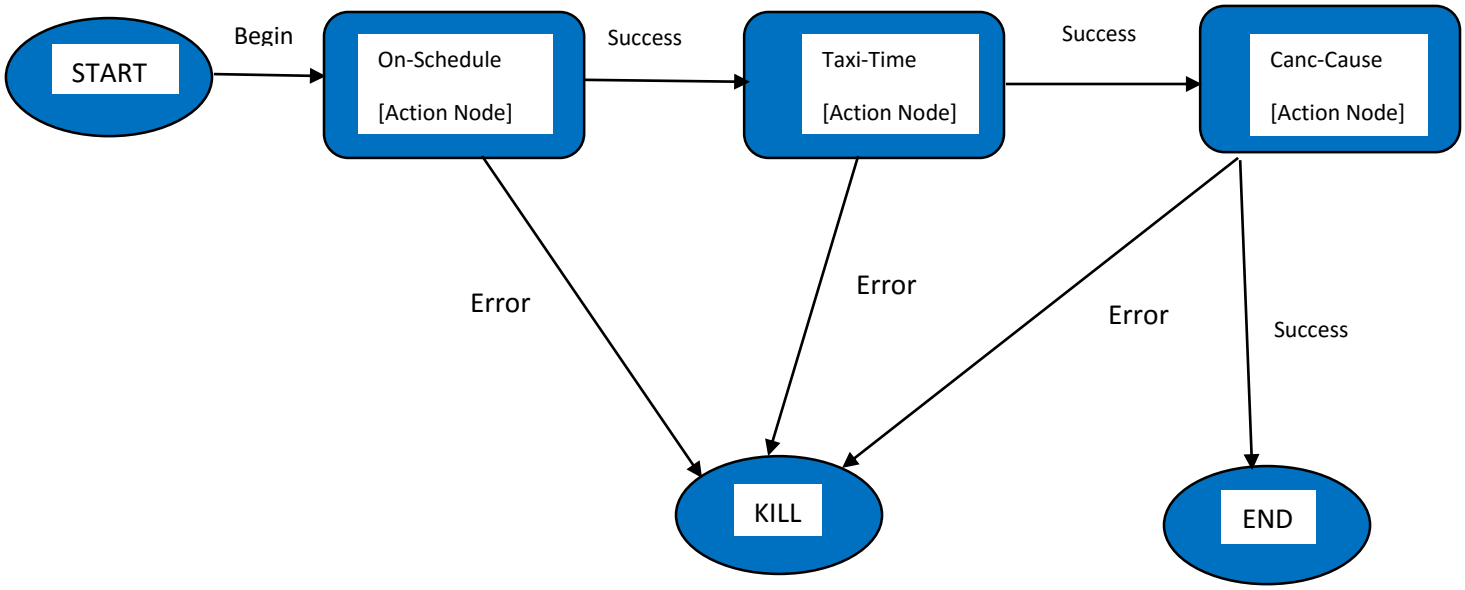
Each mapreduce program solves one of the following problems:

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

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

c. the most common reason for flight cancellations

Oozie Workflow



Algorithms Description

a. On Time Flights

1. Define a variable called *delayThreshold* = 10
2. For each flight, the Mapper program computes the sum of ArrDelay and DepDelay.
3. If the sum is less than *delayThreshold*, the flight is considered on schedule. Emit (UniqueCarrier, 1) to denote that this flight is on schedule.
4. Else if the sum is less than *delayThreshold*, the flight is considered as delayed. Emit (UniqueCarrier, 0) to denote that this flight is NOT on schedule.
5. At Reducer, for each UniqueCarrier key, count the total number of values as totalCount.
6. At Reducer, for each UniqueCarrier key, if value is 1, increment onSchedule count by one.
7. Compute the on-schedule probability as onSchedule/totalCount.
8. Add the (probability, UniqueCarrier) to an ArrayList.
9. Repeat steps 5-8 for each UniqueCarrier key received at the reducer.
10. At context cleanup, sort the arraylist in decreasing order of probabilities.
11. Write the arraylist values (UniqueCarrier, probability) to HDFS.

b. Average Taxi Time

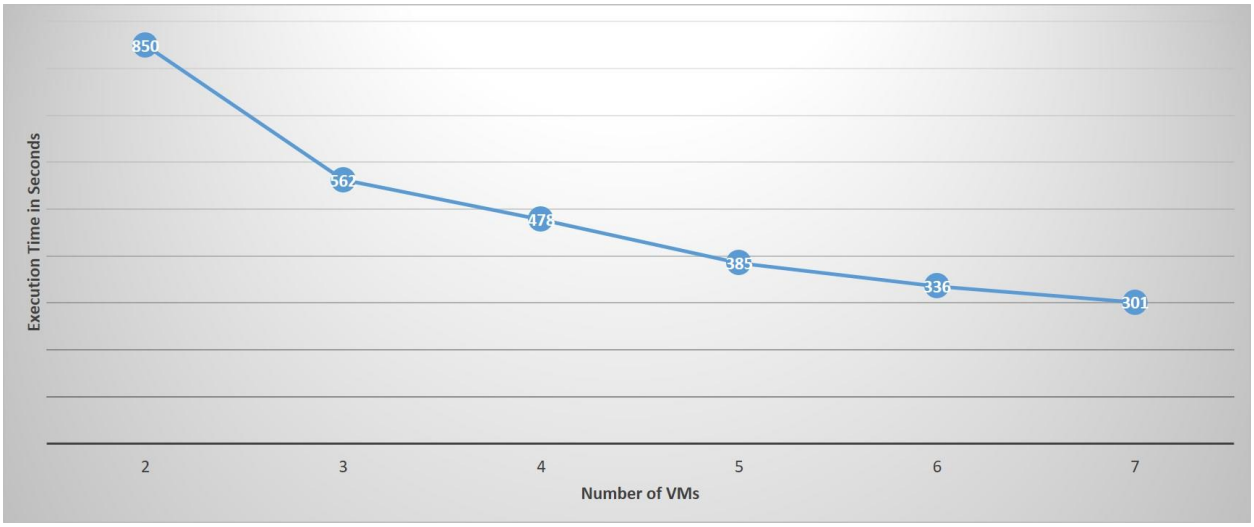
1. For each flight, the Mapper emits (Origin, TaxiOut) & (Dest, TaxiIn)
2. At Reducer, for each Airport(Origin/Dest) key, count the total number of values as totalCount.
3. At Reducer, for each Airport key, compute the total taxi time by adding all the values (TaxiIn/TaxiOut).
4. Compute Average Taxi Time for that Airport by dividing the total taxi time obtained in above step by the totalCount.
5. Add the (avgTaxiTime, Airport) to an ArrayList.
6. Repeat steps 3-5 for each Airport key received at the reducer.
7. At context cleanup, sort the arraylist in decreasing order of average taxi times.
8. Write the arraylist values (Airport, avgTaxiTime) to HDFS.

c. Most Common Cancellation Cause

1. For each flight, the Mapper emits (CancellationCode, 1) if the flight is cancelled (Cancelled =1)
2. At reducer, for each CancellationCode key, add all the values to get the totalCount.
3. Write (CancellationCode, totalCount) to HDFS.
4. Repeat steps 2-3 for each CancellationCode key received at the reducer.

Performance Measurement Plot -1

Compares the workflow execution time in response to an increasing number of VMs used for processing the entire data set (22 years)



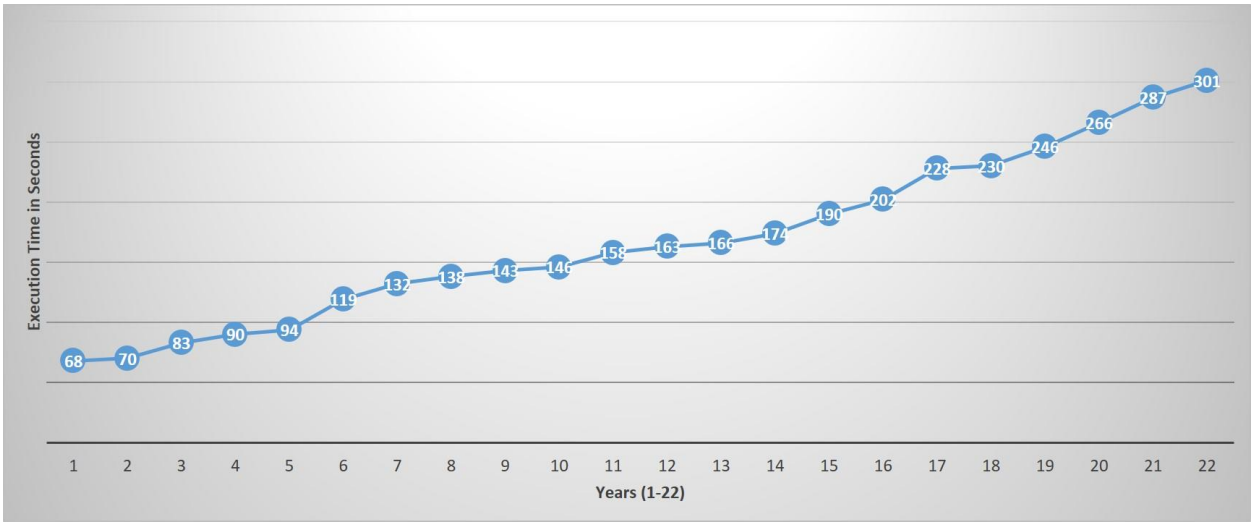
The number of VM's are increased from 2 to 7, 1 additional VM is added to the Hadoop/Oozie cluster at a time.

The execution time drops sharply as VM's are increased from 2 to 3. After this, addition of VMs leads to almost linear decrease in the workflow execution time. For a cluster with 7 VMs, the total time is 301 seconds, which is an improvement by a factor of around 2.83 in comparison to that of the initial cluster with only 2 VMs (850 seconds).

The above workflow execution times are in consistent with the theory. As more computational resources are available by increasing the number of VMs, the more parallel processing occurs, which leads to faster processing of the data.

Performance Measurement Plot -2

Compares the workflow execution time in response to an increasing data size (from 1 year to 22 years).



Maximum number of allowable VMs (7 in this case) are used for the above plot.

To start with, only the data for a single year (1987) is analyzed. At each step, additional data for the subsequent year is added to the Hadoop/Oozie cluster. In the end, the entire data from 1987-2008 is analyzed.

As expected, as the amount of data is increased, the processing time increases. The workflow execution time increases by a factor of approximately 4.4 (from 68 seconds to 301 seconds) in response to increase in data size of 1 year to 22 years.

This can be understood from the fact that since the computational resources are fixed in this case, increasing the data size will require more processing to compute the final output.