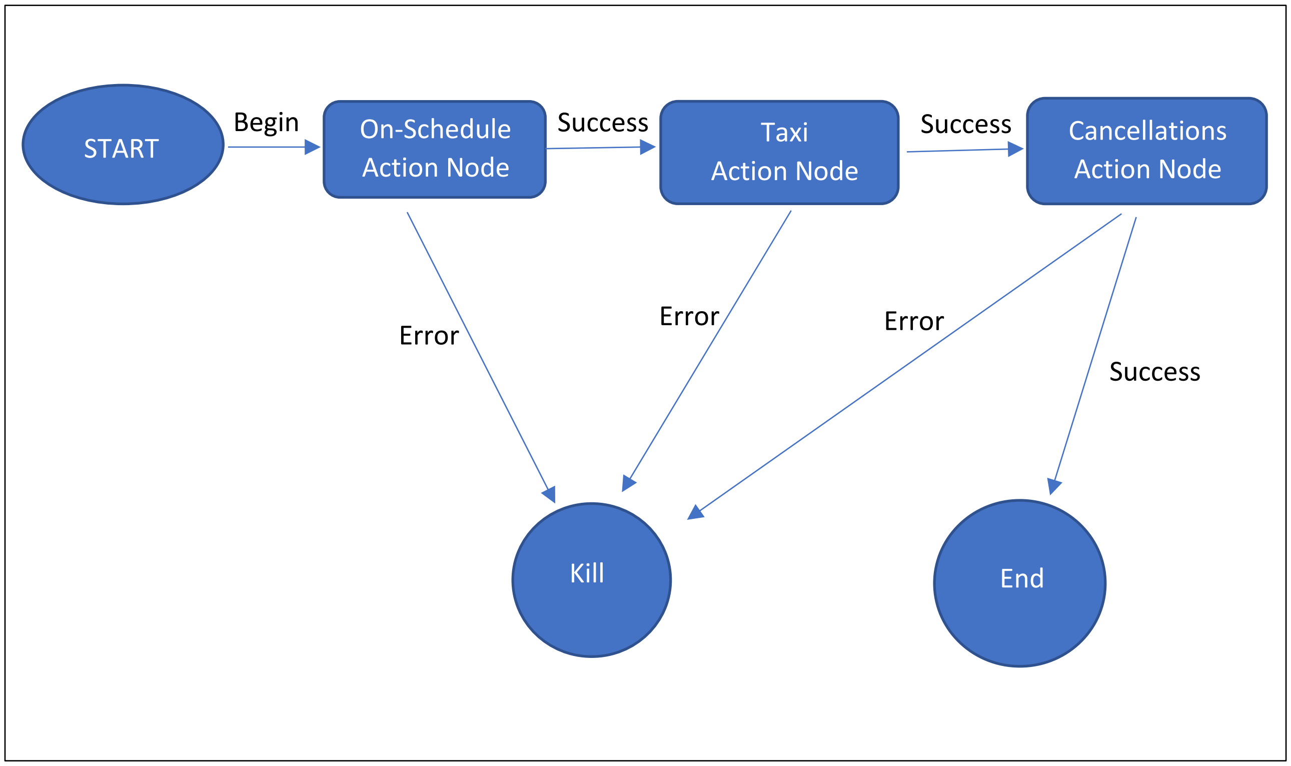
Flight Data Analysis Project

Andres Castellano and Shuang Dong

CS 644: Introduction to Big Data

Fall 2019 – Prof. Bader

Oozie Workflow Diagram



Algorithms

A) On Time Flights

1) A variable called *delayThreshold* =10 is defined
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) 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) 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) Cancellations

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 Measurements

A) Increasing Cluster Size

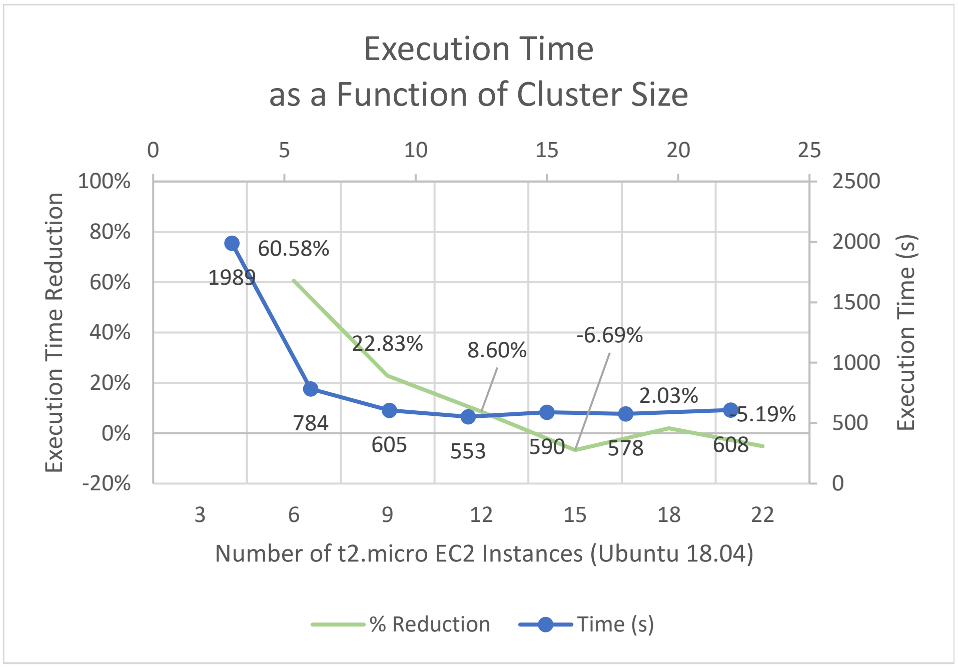


Figure 1

The chart above represents the data collected from analyzing 22 years of flight data on an increasing number of EC2 instances. Increasing the number of virtual machines (VMs) from 3 to 6 cuts the execution time by 60.58%. However, increasing the number of machines from 6 to 9 does not result in the same time reduction. In fact, the execution time only decreases by 22.83%, still significant but not as impressive. Continuing this process shows that there is an implicit law of diminishing returns, since adding more machines improves performance, but marginal performance gains are smaller every time a set of VMs is added. Analogous to the law of diminishing returns in classical Economics, the phenomenon can be explained by the overhead costs necessary to implement these workers. In a classical sense, as a factory adds more workers to its operation, it also has to add more tools, space, machines and managers to properly utilize those workers, at one point, adding to many workers is no longer economical because of added costs of tools, space, machines, etc. In the same way, adding more VMs to a cluster, also requires communication overhead which at first is small compared to the performance gains. However, at one point, the communication overhead of an increasing number on nodes, becomes inefficient compared to the performance gains of new nodes. So much so that in some cases, performance can actually DECREASE with an increasing number of VMs. Therefore, as in classical Economics, this is a problem of identifying economies of scale, that is the point on the chart with the best cost-performance tradeoff.

B) Increasing Data Size

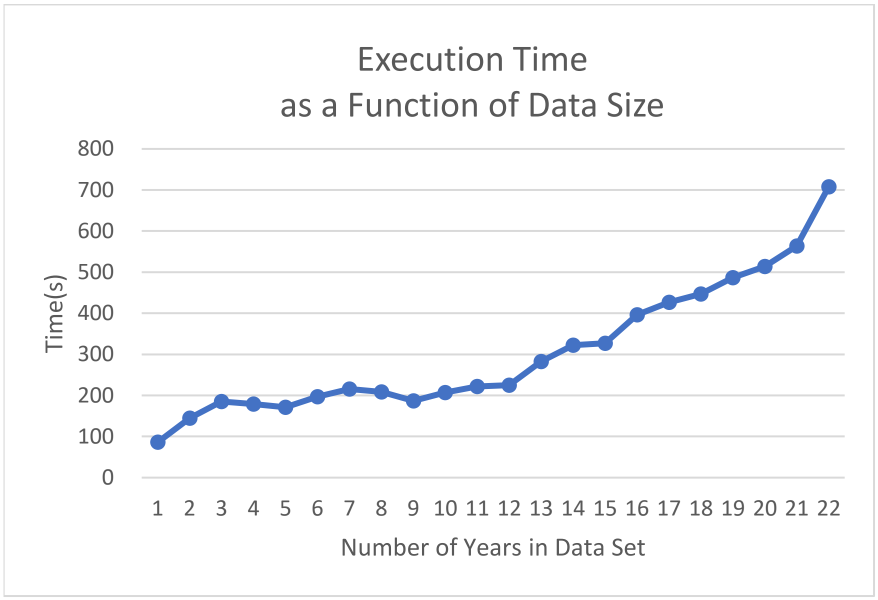


Figure 2

Figure 2 is a graphical representation of the data collected from running an increasing number of years in the dataset with a fixed number of resources, 22 t2.micro Ubuntu EC2 instances. As expected, the execution time increases mostly monotonically, the data does not increase uniformly and there is no reason to believe that the data from one year is particularly larger or smaller than the previous years. Therefore, The execution time increases but there is no reason to believe that it increases by the same factor with increasing number of years in the data set.