

**Submitted By:**

**Chitrali Rai: cr252**

**Kshitiz Arora: ka287**

**Final Term Project Report**

**Instructor: Prof Chase Q. Wu**

**FLIGHT DATA ANALYSIS**

**WITH OOZIE**

**CS-698 Big Data**

Table of Contents

Introduction2

Platforms and Tools2

Structure of Oozie workflow 2

Algorithms Used3

Performance Measurement Graphs4

Workflow Execution in response to increasing data size-----------------------------------------4

Workflow Execution in response to increasing VM’s----------------------------------------------5

**Introduction**

This Project aims at the Design and Development of Oozie workflow to process and analyze a large volume of flight data.

**Platforms and Tools**

1. Programming Language: Java
2. Operating System: Mac OS 10.10.5
3. Frameworks: Apache Hadoop 2.6.1, Apache Oozie 4.1.0, Apache maven 3.3.3
4. JAVA JDK 1.7 on Amazon EC2 VM instances

**Structure of Oozie Workflow**

Error

OK

Error

Find the most popular reason for flight cancellation.

Ok

OK

Find the three airports with least and most TAXI In and TAXI out time

Find the three least and most probability of being on schedule

Error

Ok

**Algorithm used**

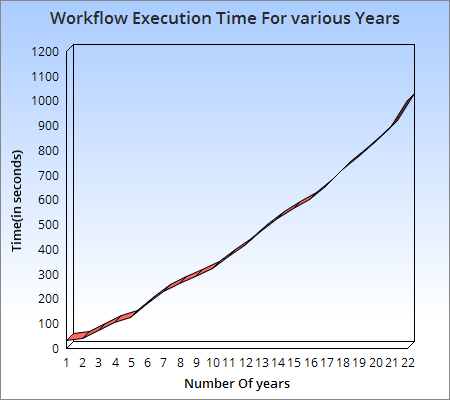
1. **Computation of airlines on schedule**
2. For each line passed in when the map function is called, we will split on “,” creating a String Array.
3. The next step is to store the airline, year, arrival delay and departure delay from the string array read from the input file into separate strings.
4. Next the total delay for each flight in each airline and each line is calculate as a sum of arrival delay and departure delay.
5. If the total delay is greater than 15 minutes as per the airline standards, it is considered otherwise the total delay is considered as zero.
6. Next the key to be passed in the reducer is airline:year and the value is the total delay.
7. In the Reducer part for each airline as key, we compute the total flights in an airline and the flights on time by checking all those flights that have a total delay of 0.
8. Next the probability of each airline to be on schedule is computed and inserted into a List .
9. This collection list is sorted and printed as output in the cleanup method.
10. **Computation of Taxi time of each airport per flight both in and out**
11. Read the Destination, Time In and Time Out from the input data files.
12. Make sure there is no NA value included.
13. We made use of Composite Writables to enable multiple values for our Map Reduce.
14. The composite Writable class contains the respective time in and time out values and individual count for each airport.
15. In the mapper : We use the input files to find the individual airports, the taxi in and taxi out times for each airport.
16. We also use 1 to specify the count each time an airport is displayed in the input.
17. In the reducer : We use the individual counts of each airport along with the total taxi In and taxi out times and divide the two with the total count

to find the average taxi In and taxi out and display them in the output.

1. **Determination of reasons for flight cancellations**
2. Read the CancellatonCode from the input data files.
3. Make sure there is no NA value included.
4. In the Mapper : Count individual occurrence of each code in the input data.
5. In the Reducer: Count the total occurrences by adding up all the individual counts of the codes.
6. Sort the total codes to find the code with maximum occurrence.

**Performance Measurement Graphs:**

**Workflow execution time in response to an increasing data size**

****

# In the above presented results, we can see that as we increase the input data of our application the execution time required for the end to end workflow execution increases.

# Initially when the data size is small Oozie is able to process the end-to-end workflow with minimum delay due to less shuffles required and the overall processing is quite fast.

**Workflow execution time in response to an increasing data size**

# 

# In the above presented graph results we can deduce that as we increase the number of VM’s the end-to-end workflow execution time reduces up to a certain number of VM’s as there are more resources that can be utilized for the overall processing of the submitted job. But as we go on increasing the number of VM’s the intercommunication overheads start to show their impact and the overall end-to-end time takes a hit and it increases instead of going down all the time.