Airline On-Time Streaming Data Analysis

Arwa EL-Hawwat
Rutgers University
Piscataway, NJ, USA
ame126@scarletmail.rutgers.edu

Jaini Patel
Rutgers University
Piscataway, NJ, USA
jp1891@scarletmail.rutgers.edu

Rahul Dev Ellezhuthil
Rutgers University
Piscataway, NJ, USA
re263@scarletmail.rutgers.edu

Supervised by Professor James Abello Mondero, Harshini Bonam, & Haoyang Zhang for CS543 Group 6, Fall 2021

I. ABSTRACT

Airline consumers today are overwhelmed by the sheer number of carriers and flight patterns offered to them when choosing flights. While they can easily compare costs and layovers at booking time, it can be much more difficult to predict delays on the day of departure. Utilizing past airline data, we can extract a model to formulate possible delays and when they are most likely to occur. In addition, we are able to determine the likelihood of certain carriers or trip patterns to experience difficulties.

II. PROJECT DESCRIPTION

A. *Overview of Project*

Our project will follow the format of database queries in a streaming fashion, and will therefore involve us maintaining a window of viewing the data while simultaneously communicating with our already existing analyses to draw conclusions based on real-world flight time and delay patterns in the United States. Through this, we are able to extract busiest flight times throughout the year as well as identify trends in delays as well as how often they occur and last to better predict such occurrences in the future. Our project uses Pyspark as the querying background for our data and models it using a Flask web application hosted on the Rutgers iLab machines for ease of access and sharing as well as improved processing power for queries and visualizations.

B. *Database*

Our data is sourced from the publicly available flight data from the 2009 data expo [1] (modified by Harvard's dataverse [2]) of commercial flights flown in the USA from October 1987 to April 2008 (approximately 120 million entries). The dataset contains several csv files, each representing the flight records of all flights that took place in a particular year all around the world. Each row in the csv files contains information of a specific flight such as flight number, origin, destination, delay in minutes, cause of delay etc. Other than yearly data, it contains 3 more files (airport.csv, carriers.csv and plane_data.csv) that contains additional information about airports, carriers and airplanes. The following ER diagram shows the relationship between this additional information and the yearly flight data. The yearly flight data is represented as flight_data **[Figure 1]**.

[Figure 1] ERD of Data Expo 2009 Ontime Airline Performance Data for domestic US flights.

C. *Objective*

Our main objective for this project is to be able to answer the following questions based on the available flight data: Which airline carrier is the most reliable in terms of punctuality? What were the worst months to fly historically? What are the busiest airports and paths in the United States? We aim to be able to organize and display our findings in a simple process and model.

D. *Challenges*

As the full dataset we are sourcing is incredibly large, one of the major challenges we faced is the issue of timing out in computations throughout our project (importing data, analyzing it, performing computations, etc.). Additionally, we were faced with the challenge of treating the data querying process in a streaming fashion which meant that we had to get creative when it came to dividing up our data in a meaningful way while still maintaining the integrity of its structure when amalgamating the results for ordering and visualization. One other challenge we overcame was the random nature in the way the bucketed months were arranged and we had to figure out how to sequentially arrange them for processing and modeling.

III. IMPLEMENTATION

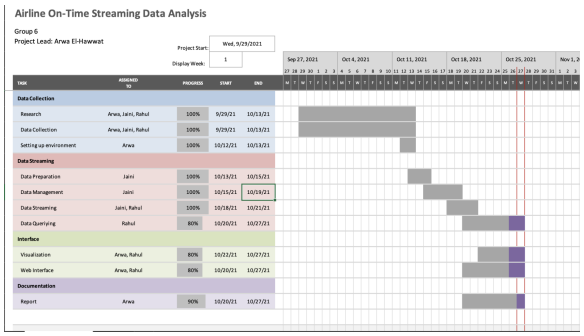
A. *Tools we used*

In order to process our raw data, we first migrated all csv files to the Rutgers iLab machines for ease of access and processing power. We used Python as the programming language background and Pyspark as the framework to connect to the iLab spark cluster. To simulate the streaming processing of the data, we decided to divide the original data into groups of monthly bucketed data partitioned by year which was then stored in a local Hive file system. The streaming and processing operations were written as functions and queries in Jupyter Notebook.

For the visualization aspect of the project, we opted to set up a web application using the Python Flask [3] framework where we passed Spark data in order to set up our visualization graph. When the queried data is first imported into Flask, we save it as a Pandas data frame of strings that is then fed into amCharts JS [4] to produce charts that update in real time with the processing of our data in the backend in Spark.

B. *Project Timeline & Division of Labor*

The three main steps of our project were the research stage, the data manipulation stage, and the visualization stage. All three of us cooperated on the research stage during the first three weeks where we located the database we wanted to work with as well as decide upon the best practices for processing and divvying the data in a manner that best suited our purposes. The following phases were all also worked on in a collaborative effort, but we had assigned one member to manage each stage's progress. Next was the data manipulation stage headed by Jaini. Afterwards came the querying and visualization stages headed by Rahul. Lastly we had other operations that were being carried out throughout these stages like setting up the shared project folders, UI testing, and report writing headed by Arwa.



[Figure 2] Working timeline from September 29 - October 27.

C. *Algorithm Description & Complexity*

The querying operations that we perform are all simple arithmetic computations of the individual records, like SUM or AVERAGE. As the bulk of the data processing we perform is handled by Apache Spark, we can look at the number of records to find the runtime complexity. If we assign N = 120 million for our input, then the runtime complexity of our operations comes out to $O(N)$ as the arithmetic operations performed in memory are a constant $O(1)$. It is important to note that these calculations are based on the assumption that Apache Spark queries these arithmetic operations in constant time as they are read in the streams. Additionally, the remainder of our querying is handled by the graph visualizations on the front end after all the data has been processed and saved in the system, which brings the space complexity to be $O(2N)$ reduced to $O(N)$ for the bucketed and post-processed records.

D. *Interface Mockup*

As stated previously, the interface we selected to display the results of our data processing is through a web application based in a Python Flask framework hosted from the Rutgers iLab machines. The purpose of the web application is to give users a simple manner to start the streaming operation in the Spark backend through a button click and then be able to view the processing and graph visualization in real time through a continually amending table and chart on the screen. The operation terminates on its own once all the plane data's yearly data has been processed (querying is inactive) and displays the final table and graph. In Part VI, **Figure 3** is an example of the table that is computed while **Figure 4** shows a snapshot of the graph that is displayed. In the web application itself, further functionality has been integrated allowing users to toggle on certain axes and lines in the graphs to hone in on certain flights or from an expanded perspective.

IV. PROJECT HIGHLIGHTS

At this point in our analyses, we have made several observations based on the data that we would like to highlight here:

- The total number of flight records increased steadily from 1987 to 2008, which is understandable as travel increased as it became cheaper and more accessible,
- In studying the percentage punctuality of different airline carriers, we noticed various fluctuations throughout the years for a majority of them ranging from around 10-20%. This could be attributed to certain events happening during those years to prompt more delays, like the 1996 Summer Olympics in Atlanta, Georgia.

The work we have produced will be of particular interest to domestic U.S. travellers as well as airline carriers to understand the nature of airline travel in the country as well as identify the different factors (spatial or temporal) associated with flight scheduling errors or delays.

V. FUTURE WORK

The purpose of our project was to answer questions based on the flight data we had which is solely domestic American flights between the years of 1987 and 2008. Work in the future could expand these observations to international flights to determine the punctuality of trips that take multitudes of times longer, as well as

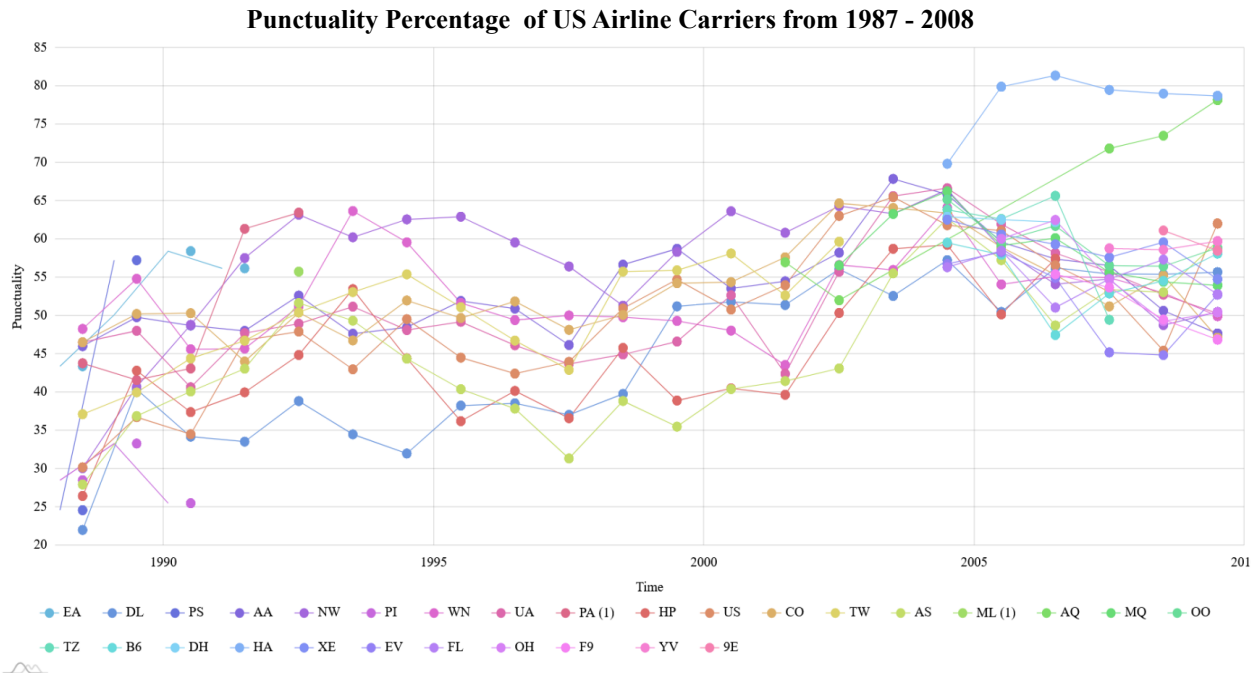
consider flight paths that may include differing stopping points to best predict the behavior of multi-layover trips.

VI. VISUALIZATION FIGURES

Punctuality Percentage of US Airline Carriers from January 1987

Year	Month	UniqueCarrier	punctuality
1989	1	TW	0.441093
1989	1	UA	0.434712
1989	1	DL	0.342901
1989	1	AA	0.467814
1989	1	EA	0.588467
1989	1	NW	0.516254
1989	1	CO	0.523406
1989	1	AS	0.379845
1989	1	PI	0.318617
1989	1	US	0.436432
1989	1	HP	0.377351
1989	1	WN	0.406445
1989	1	PA (1)	0.420453

[Figure 3] Tabular data of punctuality of some US flights from January 1989



[Figure 4] Visualization of different US airline carriers and their calculated punctuality as a percentage from 1987 - 2008 using amChart

VII. REFERENCES

[1] 2008, *Data Expo 2009 - Airline on-Time Performance*, American Statistical Association, https://community.amstat.org/jointscsg-section/dataexpo/dataexpo2009

[2] 2008, *Data Expo 2009: Airline on time data*, Harvard Dataverse, V1, https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/HG7NV7

[3] Python Flask, https://flask.palletsprojects.com/en/2.0.x/

[4] amCharts JS library, https://www.amcharts.com/