Michael Alexander, Mike Amodeo, Roseanna Hopper, Yifan Sun
W205 Data Storage and Retrieval
Final Project Architecture
April 24, 2017

Airline Data Dashboard Application Documentation

The airline data dashboard, hosted through R, examines trends in U.S. flight data and weather from 2007 through 2016. The application downloads historical flight and weather data, uploads the data into a Hadoop Distributed File System (HDFS), loads the data into tables in Hive, queries the tables according to several custom defined metrics related to airport and flight performance, and exports the output of these metrics to an R-based dashboard that graphically displays the metrics.

Data Sources

The Airline Data Dashboard application uses publicly available data from two federal government sources:

- Bureau of Transportation Statistics (BTS)
    - Aviation Data Library - database of every flight departing or arriving from a United States airport with schedule and performance data. updated quarterly https://www.transtats.bts.gov/
- National Oceanographic and Atmospheric Administration (NOAA)
    - Global Historical Climatology Network (GHCN) - daily weather data from land surface stations around the world with data as far back as 1763. updated daily https://www.ncdc.noaa.gov/data-access/land-based-station-data/land-based-datasets/global-historical-climatology-network-ghcn
- Weather Underground
    - 10 day forecast data accessed through a custom API https://www.wunderground.com/

Architecture

- **HDFS** – Used to store the raw data files; chosen because of large data volume which will only increase over time
- **Hive** – Used to create tables for easy SQL querying. Because of the large data volume, accessing the Hive tables in Spark-SQL allows for much faster execution of the final queries.
- **R –** Used to create an interactive dashboard, which can easily be updated with new data and expanded to include new tabs and visualizations.

Directory and File Structure

Below follows a list of all files and directories contained within the project repository.

- **run_all.sh –** Executes all downloads, loading, and queries necessary to replicate the project
- **load_flight_data -** Directory containing files related to loading of BTS data to HDFS
    - **load_2007_2016.sh** – BTS data are separated into monthly files; load_2007_2016.sh downloads all month files from 2007 through 2016
    - **load_flight_data.sh** – Creates all necessary directories in HDFS and calls load_2007_2016.sh to download files
- **transform_flight_data -** Directory containing files related to transformation and loading of BTS data to Hive
    - **load_2007_2016.sql** – Loads all of the monthly files from 2007-2016 into Hive
    - **months_to_years.sql –** Compiles all of the separate monthly files by year
    - **one_to_five_years.sql** – Combines the years into two five-year blocks (2007-2011, 2012-2016)
    - **all_years.sql** – Compiles all years into one table
    - **cleanup_months.sql** – Removes all of the individual month tables and the five-year tables
    - **flight_data_HDFS_cleanup.sh-** Removes the flight data from HDFS to free up space
- **airline_eda -** Directory containing files related to airport performance metrics
    - **tables_for_csv_export.sql** – Creates tables used in final queries, including the creation of a "Route" column and tables including only the top 100 airport origins by overall departure volume
    - **queries_for_csv_export.sql** – Queries used to create the final CSVs used in the dashboard
    - **final_csv_export.sh –** Runs the queries in queries_for_csv_export and exports CSVs to flight_app/final_csv
- **weather_data -** Directory containing files related to loading of GHCN data
    - **load_weather_data.sh -** Executable shell file that downloads GHCN climate data by year starting in 2007, GHCN weather stations data file, and airports from BTS
    - **stations_airports.py (do not execute)** - Python script that matches airports with nearest weather station. Output from this file is included as CSV and referenced directly by other files.
    - **airports_stations.csv** - Output from stations_airports.py script that includes airport ID and 3 letter code with the ID of nearest GHCN weather station
    - **hive_weather_data_import.sql** - Creates tables of annual weather data and airports with nearest weather station in Hive

    - **hive_weather_merge.sql -** Merges weather tables of years into a single table. Also creates a table of all precipitation that occurred at a station near an airport to be used for metric development
    - **airports.csv -** BTS file containing coordinates and codes for all airports internationally.
- **weather_investigation -** Directory containing queries of weather-related metrics
    - **weather_investigations.sh -** Executable shell file that runs all of the queries contained in this subfolder and exports output as tab separated value files to /flight_app/final_csv directory
    - **weather_queries.sql -** initial queries related to weather data linking flight data to weather data
    - **weather_delay_queries.sql -** Additional queries calculating likelihood of delays at individual airports and other items
    - **Q1_precip_days_ann.sql -** Query that creates first weather metric, a table of the 25 busiest airports with the average days of precipitation over study period
    - **Q2_likelihood_month.sql -** Query of the second weather metric, table of the likelihood of weather delay per airport per month
    - **Q3_likelihood_based_on_depth.sql -** Query of the third weather metric, table of the likelihood of a delay given a depth of precipitation
    - **weather_forecast.py -** Python script that connects to a Weather Underground API to access 10 day forecasts for the 25 busiest US airports. The depth of rainfall per day is then compared against the historical data and likelihood of delays generated for previous queries to estimate the number of delays at each airport for each of the next 10 days.
- **flight_app** – Final dashboard
    - **final_csv** – directory containing CSV exports of metrics for loading into R
    - **air_travel_dashboard.Rmd** – Flight and weather data dashboard, hosted in R

Execution Instructions

Prerequisites: In order to replicate the results for this project, clone this directory in an EC2 instance. Due to the data being analyzed, we recommend that the instance be of at least extra-large size, with attached EBS storage of **at least 200 GB**. The execution also requires the following being set up prior to execution:

- HDFS, Hive and Spark-SQL on EC2
- RStudio with markdown capabilities installed locally

Once an instance is set up and has both Hadoop and Postgres running, the application and all of its data can be replicated in either of two ways. First, it can be run by executing the run_all.sh shell file in the main directory of the repository. This will download all data, load to HDFS, and build and execute queries in Hive. Once complete, a user can open the file air_travel_dashboard.Rmd on their local machine with R installed, run the file, and view the output as an html file.

Alternatively, the user can run stepwise by completing the following steps:

1. Navigate to and execute /load_flight_data/load_flight_data.sh. This will call the shell file /load_flight_data/load_2007_2016.sh to download the flight data and store on the EC2 instance in a temporary directory. The file will also load the data to HDFS and clean up the temporary staging directory.
2. Navigate to the folder /transform_flight_data and execute the following SQL files:
    a. In Hive, run load_2007_2016.sql. This will load all of the files from 2007-2016 into Hive, separated by month.
    b. Run months_to_years.sql. This will compile all of the separate monthly files by year.
    c. Run one_to_five_years.sql, which will separate the full ten years into five-year blocks.
    d. Run all_years.sql. This will compile all of the years into one table.
    e. Run cleanup_months.sql in order to remove all of the individual month tables.
3. Navigate to the folder /airline_eda and run tables_for_csv_export.sql in Hive or Spark-SQL before running final_csv_export.sh.
4. Navigate to the folder /weather_data and perform the following steps:
    a. Execute the file load_weather_data.sh to download all weather data and load into HDFS.
    b. Execute hive_weather_data_import.sql in Hive to load the yearly weather data in Hive.
    c. Execute hive_weather_merge.sql in Hive to merge all weather data.
5. Navigate to the folder /weather_investigation and execute the file weather_investigations.sh. This file will perform the weather metrics queries on the tables in Hive and export tsv files to be used in the dashboard.
    a. To run the last weather metric (10 day forecasted delays), navigate back up to the main directory of the repository and run the python script /weather_investigation/weather_forecast.py
6. Open R Studio on a local machine, and execute the file /flight_app/air_travel_dashboard.Rmd. This file will create an html file of the dashboard displaying all metrics. It references output files that are pre-loaded into the repo on Github. To view any replicated files, the files in the folder /flight_app/final_csv will need to be migrated by the user from the EC2 instance to the user's local machine.