Project Summary and Nest Steps

I. Project Scalability

Data Storage

Our main data storage and retrieval process happens in PostgreSQL, which is a very powerful database engine. Like other main stream databases, such as Oracle, MS SQL Server, it is designed to handle large structured tabular data set in a very efficient way for data read and write. Its Query Optimizer can be very helpful when dealing with large data volume and enable the tables to scale well to a certain limit.

Our major data set in the on-time performance data which has about 1.8M rows per month. We believe we have enough space to grow and extra capabilities to continue handling this data set for at least 10 more years.

Besides, given the on-time performance data is partitioned, for computing intensive queries, we can easily split up the time filter and query data in chunks, month by month, if we experience bottlenecks for query performance.

IPython Notebook

We have already started to experience some of the limitations in IPython Notebook during the project, the notebook can easily stuck or fill up server memory when dealing with large dataset. If we want to build model on larger dataset than we currently handle, it requires us to pursue a better server with higher CPU and memory that the notebook would sit on.

Tableau/Web Application

For the serving layer, we would envision it to have the capabilities to serve concurrent users. This requires us building in the future machine clusters that allow parallel processing.

II. Project Evolution

Twitter Dataset

For the next steps, we hope to incorporate real-time twitter data through twitter API, try to collect public sentiment on flight delays and even predict flight delays based on the traffic for different topics in twitter.

Automation

The data acquisition process is still very manual at this stage of the project and it requires us clicking around in the on-time performance website, downloading the dataset and transferring it onto EC2 via FileZilla.

We hope to fix that and make a fully self-sufficient process by:

- Reaching out to “United States Department of Transportation” and asking if they provide any APIs that would allow us easily download the files onto the EC2 server by running bash command.
- Leveraging Data Pipeline APIs offered by AWS, to schedule feed loading in an automated fashion, for example, scheduling real-time twitter feed collection, nightly weather data crawling, and monthly on-time performance data download and ML model retrain.