Comments.
1. Overall this is a well-planned project.
2. You may have potentially many more customers other than just the air traffic controllers.
3. You may want to think about what kind of frontend you want to present to the user. Is it a cell phone app or web-based app.
4. You may want to think about how to prevent fault tolerance.
5. You may want to think about what CAP theorem you want to focus on.
6. To spread the newest ML models to the edges, you may want to use multicast.

CS 5412 Project Plan

Harshwardhan Jain (hj364), Sanjana Kaundinya (ssk228), Rohit Khanwani (rk632)
February 28th, 2018

1 Overview

We will create an application for airports and airlines that uses flight schedules and real time feed from airborne aircrafts to give routes that minimize traffic, drag, and flight costs using machine learning models trained on historical data. Our clients will be the air traffic controllers in the air traffic watch towers monitoring flights and talking to the pilots about flight take-off and landing.

2 Data

We visualize our system, on a higher level, dealing with two kinds of data:

• Static Data: Data that doesn't change much, for example flight numbers, airports, possible routes and long-term flight schedules.

• Dynamic Data: Data that flows in through the sensors in real-time, for example aircraft altitude, location and weather updates.

We plan on researching free APIs that could provide us with the data we need to test our application. However, we realize that the variety of data required is large, and might not be covered by free APIs. In that case, we plan on constructing data that somewhat represents real-world scenarios for the categories/quantities we are unable to obtain through free APIs. One place that we hope to source a lot of our data for our project will be from https://openflights.org/data.html This website has tons of data ranging

from a vast airport database, flight route database, and and a an airline database. The data from the routes will give us the ability to use this data, apply our machine learning models on it, and output results that can be beneficial for our clients. This data from the routes will essentially mimic real time data coming in from sensors on the ground.

3 Interface

The interface we plan to have has essentially three different components that the client will be able to interact with. The first component will be monitoring ongoing arrivals and departures of aircrafts. Our system will do computations to figure out the most optimal sequence in which the aircrafts should take out and land such that it can reduce drag, thereby overall fuel usage. These computations will take into account static data such as aircraft size and weight as well as real-time data of incoming flights to land and outgoing flights to take off. Taking these constraints into mind, we will come up with an algorithm that will most optimally plan landing and taking off for the aircrafts and reduce drag as much as possible.

The next component will be to reduce overall flight traffic. This will also provide data to our clients working in the air traffic control tower and make recommendations to also time landings and take-offs to reduce traffic as much as possible on the ground, and in the air. Reducing flight traffic will help reduce delay times and allow for passengers to board and get off their plane faster, instead of having to wait on the plane while it taxis and tries to find an open gate; or wait on the runway trying to leave the airport. This will probably be based on mostly static data (aka the flight schedule for the day) and optimally order the flights

1

in the best way to reduce traffic.

The final component, and probably the most important component, will be to significantly reduce delays and missed flights for passengers. This will be based on active real time delays due to weather, aircraft maintenance, and other miscellaneous reasons. Our system will apply a machine learning model that will look at passengers' connecting flights and try to delay these flights so that a large number of passengers won't miss their flight. Essentially it will optimize for a large group of people with the same connecting flight, effectively causing a delay for these passengers such that they won't miss their connection and can get to their destination on time. These calculations will have to be done in real time, and will have to be

fast and accurate, so the recommendation that the system makes to the air traffic controller will be the best one at the time.

4 Implementation/Architecture

Our system will have three main tiers through which the data from sensors, radars, and controllers will pass through. They are the following:

Figure 1: Architectural Diagram for our System

• Sensors: The aircrafts will act as sensors, sending real time feed like temperature, position, direction, traffic, drag, wind speed, etc. This information will be used to update routes for existing flights/flights departing soon based on the output of machine learning models from the first tier.

• Tier 1: The sensors will send real time feed to tier 1 of the system. The first tier will run local machine learning models (updated periodically by the central machine learning model) and communicate any

2

changes/information with the client over the network. The client would then get in touch with pilots to communicate appropriate changes. The first tier will be the only point of contact between the sensors and the client and all communication will be done over a secure network with encrypted messages. We imagine our clients to be the air traffic controllers that are in constant touch with pilots. Additionally, the first tier will also filter and send relevant data to tier 2 (discussed below) for storage and for updating the central machine learning algorithm.

• Tier 2: This will be the storage unit for the application. All data (historical, new data sent from tier 1, schedules, etc.) reside on this part of the stack. This tier is a central data unit for the machine learning model to train on. Every unit in the first tier sends data to this tier.

• Tier 3 - ML: This is the last layer of the application. This layer contains the central machine learning model(s) that is locally cached (with some local changes) at tier 1. It gets new data from tier 2 and updates the machine learning model. It pushes the updated models to the first tier periodically.

We plan to use one of the many available cloud hosting services (most likely will be using Google Cloud) to host our Tier 2 and Tier 3 services. Google Cloud has a service called Unified Object Storage to specifically store live data that will essentially be taking in the filtered important data from the Tier 1. Tier 3 will also be housed in Google Cloud as well, modeled as a different service for ML and backend computation and will communicate directly with the Tier 1 system. The Tier 1 system in a real world scenario would be the edge in this situation, that could be modeled as multiple points of presence in the real world. For our project, it will be a service that runs on a laptop that will constantly filter in the real time data coming in from the sensors (which we will model as active data being fed in from the open source flight data we find). This Tier 1 would likely be implemented in some sort of server side language (like C++ or PHP). In addition, this Tier is the one that interfaces with the client so it will have some sort of GUI that will make recommendations based on the three components that we outlined previously in our Interface section.

5 Evaluation

To simulate a real world environment, we will start off by running n (with values of n from 1 to 10_2) instances of the application on different virtual machines on our desktop and then we will deploy it on the cloud to run more instances. For around 7-8 sensors, we will have 1 process that acts as the tier 1 for those sensors and communicates with them. We will vary the number of processes for each tier to account for different traffic situations and depict the busier areas like Atlanta, etc. We will also have some tests for the machine learning model to analyze its correctness and soundness. We will log the latency for different scenarios to make sure acceptable standards of performance are met.

To check if the machine learning model is updated periodically, we will train the model on datasets drastically different from the live feed and check if the machine learning classifier is updated as planned. This will ensure that information flows accurately among different tiers of the application. Another aspect we aim on testing is whether or not a sensor/flight is sending data to the correct ATC airport, as per proximity to departure and arrival airports.

In addition to the data that has to be retained in tier 2, we will add additional useless data that has to be filtered by tier 1 to simulate real-world situations. This would allow us to test if our first tier is only retaining necessary data accurately.

We believe that evaluating the application using these methods would allow us to exhaustively test if our application is scalable and high-performing in terms of both number of clients and number of sensors/flights supported. The operating costs of the application can be assessed by analyzing bandwidth. We will be adding more evaluation methods as needed while we continue to implement the application.

6

.

..

....

..

..

Milestones and Group Assignments

March 7th, Sanjana and Harsh: Research appropriate server side languages and start laying foundation of detailed implementation for the tiered system

March 7th, Harsh and Rohit: Research ML models we can potentially use for our system

March 21st, Sanjana, Harsh and Rohit: Start coding up server side and look into Google Cloud storage and ML services

March 21st Harsh and Rohit Finalize on an ML model and being implementation

April 4th, Sanjana and Rohit: Hook up server that's been coded up and Google Cloud services

April 4th, Rohit and Harsh: Integrate ML model into Google Cloud services

April 18th, Sanjana and Rohit: Test server side code and its filtering properties with data from openflight

April 18th, Harsh: Train ML model on data

May 2nd, Sanjana and Rohit: Start creating a testing suit to rigorously test the Tier 2 and Tier 3 services

May 2nd, Harsh: Create testing suite and try to further optimize ML model

May 9th, everyone: Work together to create proper real time data use cases and rigorously test the system for fault tolerance, race conditions, latency, etc.

4