Jeremy Moore
COSC 480 Cloud Computing
Fall 2019

Drone Project: Part 2

WHAT WE GOT TO WORK

Problem 1.

- Created fly.py script that takes user information as command line arguments, concatenates it with flight data (e.g. x, y, and z) and prints it to the console. Only the first parameter (name) is required.
- Created test_flight.py that runs the drone without command line arguments.
- Both scripts provide video.

Additional Work:

- Developed a simple web page running on a local web server to accept user information and start/stop the flight session. The script fly.py can accept user information from either source (command line or web site).
- Installed a local instance of DSE in a Docker container and setup the required keyspace and table.
- Connected to the Cassandra cluster from our Python script and wrote flight data to the local database.
- Setup a three-node Cassandra cluster on AWS. We were able to connect to the cluster using the command line but have not yet connected via Python.

Repository:

https://github.com/xfgoovaerts/DroneProjectPartOne

WHAT WE TRIED HARD BUT DIDN'T GET TO WORK

- Have not been able to connect to the AWS DSE cluster from our Python script.
- The AWS cluster should be moved to EBS so that data is not lost when the nodes are stopped.

Jeremy Moore
COSC 480 Cloud Computing
Fall 2019

WIKI

1. GETTING FLIGHT DATA

In your python code, after connecting to the drone, subscribe to log events by adding the line...
drone.subscribe(drone.EVENT_LOG_DATA, log_data_handler)
...where the second parameter (log_data_handler) is a function that accepts the log data as an argument.

The received log data has an object name ‘mvo’ and that object contains the properties: pos_x, pos_y, pos_z which contains the x, y, and z coordinates respectively.

It is possible to turn log data down to include only error messages, but doing so will eliminate the flight data we are looking to obtain.

The x, y and z coordinates will often drop to near zero for no apparent reason. My best guess at the moment is some of the log data is not position related and we are seeing invalid data. More exploration is required to filter non-flight data messages from the log data received and see if this corrects the problem.

2. WEB INTERFACE

Install Python Flask using these instructions:
https://linuxize.com/post/how-to-install-flask-on-ubuntu-18-04/

Flask is a web micro-framework for Python which comes with a simple web server. The web server runs in a virtual environment to separate it from the main OS. Because of this all TelloPy related libraries have to be reinstalled within the virtual environment.

To follow conventions static pages (CSS, JS) should be put in a folder named ‘static’ and html pages (or templates) should be put in a folder named ‘template’.

We created a separate Python file named app.py to handle the web page. This file will serve the website for entering user data and call the fly.py application once user data is returned to from the website.

Data from the website is returned via a JS fetch call (similar to an ajax call) from the front end to the back end when the ‘Start’ and ‘Stop’ buttons are clicked.

Jeremy Moore
COSC 480 Cloud Computing
Fall 2019

3. INSTALL DSE ON A LOCAL DOCKER CONTAINER

Follow these instructions to install Docker on Ubuntu:
https://docs.docker.com/install/linux/docker-ce/ubuntu/

Pull down the DSE server image by running the command:
docker pull datastax/dse-server

Create a container from the image with the docker run command:
docker run -e DS_LICENSE=accept --name my-dse -d datastax/dse-server

And then you an open a cql command prompt within the docker container with the command:
sudo docker exec -it my-dse cqlsh

You may not need to run this as a sudo command, I had to due to some peculiarity of my docker install.

Once you have a cql prompt within the Docker container, you can run these commands to create the keyspace and setup the required table:

CREATE KEYSPACE competition WITH replication = {'class':'SimpleStrategy', 'replication_factor':1};

USE competition;

CREATE TABLE positional (flight_id TIMEUUID, ts TIMESTAMP, x DOUBLE, y DOUBLE, z DOUBLE, latest_ts TIMESTAMP STATIC, station_id UUID STATIC, num_crashes INT STATIC, name TEXT STATIC, group TEXT STATIC, org_college TEXT STATIC, major TEXT STATIC, valid BOOLEAN STATIC, PRIMARY KEY (flight_id,ts)) WITH CLUSTERING ORDER BY (ts DESC);

INSERT INTO positional (flight_id, ts, x, y, z, latest_ts, station_id, num_crashes, name, group, org_college, major, valid) VALUES (50554d6e-29bb-11e5-b345-feff819cdc9f, toTimeStamp(now()), 0.01, 0.02, 0.03, toTimeStamp(now()), 274a8e05-cb67-5eb5-8592-6145a580450c, 0, 'John Doe', 'student', 'cas', 'Computer Science', true);

Note: If you copy the insert command from Canvas, you'll need to add the parenthesis around the column names and replace the backticks with single quotes.

Jeremy Moore
COSC 480 Cloud Computing
Fall 2019

4. CONNECT PYTHON SCRIPT TO LOCAL CLUSTER

To connect to the Cassandra cluster from within Python, you'll need to the following commands:

```
from cassandra.cluster import Cluster
cluster = Cluster(['<your ip address>])
session = cluster.connect('competition')
```

You'll need to provide your ip address to the Cluster command and the keyspace you want to connection to must be included in the connect command.

To execute a cql statement use: session.execute(<statement>) where <statement> is a string containing the query you'd like to execute.

(over)

Jeremy Moore
COSC 480 Cloud Computing
Fall 2019

5. SETUP DSE CLUSTER ON AWS

Follow this tutorial to setup a three-node Cassandra cluster on AWS EC2 instances:
https://academy.datastax.com/content/hands-tutorial-install-dse-aws-cloud-3-easy-steps

When creating the KEYSPACE we ran into a little trouble with the supplied command:
CREATE KEYSPACE competition WITH replication = {'class':'NetworkTopologyStrategy', 'replication_factor':3};

We received a message stating that ‘replication_factor’ is only a valid argument for SimpleStrategy and not for NetwokTopologyStrategy.

Here is the command we ran to create the KEYSPACE, where DC1 is the name of our data center.
CREATE KEYSPACE competition WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'DC1' : 3 };

It seems that the NetworkTopologyStrategy may only be required if multiple datacenters are required so I believe we can use

CREATE KEYSPACE competition WITH replication = {'class':'SimpleStrategy', 'replication_factor':3};

… for our purposes.

We were able to log onto this cluster through the command line and execute cql commands, but have not yet been able to access it via our Python script.

The nodes talk to each other over their private IP addresses. We attempted adding the public IP addresses to the rpc_address in the yaml file but so far it has not worked.