Data-challenge-Final-Report

April 17, 2017

0.1 # Summary

The challenge given was to find nearest airport for each user.

There were aprroximately 1 million users and 6889 airports.

I tried many techniques to solve this problem and learned new approaches as well. Below are some of them:

1. Wrote a simple python script that calculates Haversine distance by iterating both the lists. Needless to say it was not time efficient at all.
2. Then I tried various python geo libraries that calculates geo distance like geopandas, geopy, geog. Though they were giving correct results but they were low in performance.

   I also parallelized the process using **Multiprocessing** library in Python. Though it decreased the execution time but I was not satisfied with the performance.
3. Then I decided to opt for external in-memory database called **Redis**. This is the approach I have implemented below. I havent worked with Redis before so I learned it and decided to use it's geo module for distance calculation. As redis implements geodis with a time complexity of O(log(N)) I experienced that it was definitely faster. But it was not fast enough. I also ran a Redis cluster on AWS EC2 instance but it did not had any significant time improvements.
4. If given more time I would like to implement a **MapReduce** solution to it using Spark as after trying the above techniques, I came to the conclusion that its better to treat it as a big data problem for best performance.

0.1.1 Final Thoughts

The final solution that I am submitting might not have the best performance but I enjoyed every bit of these five days of challenge as I learned alot of new tricks and techniques. I had this zeal of researching and coming up with an optimized solution the whole time. I definitely know that the best solution is through Spark but as I am new to Spark, its taking a little time to implement it. Anyways, I will not rest until I implement it even after the submission deadline. I would love to hear your feedback on if I was on the right track or not.

---

1 Code begins below

1.1 Import required libraries

```
In [2]: import pandas as pd
        import redis
        import csv
        import numpy as np
        import time
```

1.2 Import files into dataframe

```
In [3]: user_data = pd.read_csv('../../tavel_audience_challenge/data/sample_data.cs
        airport_data = pd.read_csv('../../tavel_audience_challenge/data/optd-sample
```

1.3 Check data

Select top 5 rows

```
In [4]: user_data.head()
```

```
Out[4]:                                    uuid  geoip_latitude  geoip_longitude
        0  DDEFEBEA-98ED-49EB-A4E7-9D7BFDB7AA0B      -37.833302       145.050003
        1  DAEF2221-14BE-467B-894A-F101CDCC38E4       52.516701         4.666700
        2  31971B3E-2F80-4F8D-86BA-1F2077DF36A2       35.685001       139.751404
        3  1A29A45C-D560-43D8-ADAB-C2F0AD068FFE       44.840401        -0.580500
        4  A6EC281B-B8EC-465A-8933-F127472DB0A3       51.963299         4.499700
```

```
In [13]: user_data['uuid'].nunique()
```

```
Out[13]: 996980
```

There are some duplicate uuid in the data. Total records are 1 million but unique number of uuid is 996980.

```
In [10]: airport_data.head()
```

```
Out[10]:   iata_code   latitude   longitude
         0       AAA -17.352606 -145.509956
         1       AAB -26.693170  141.047800
         2       AAC  31.073330   33.835830
         3       AAE  36.822225    7.809167
         4       AAF  29.729380  -85.028800
```

1.4 Convert pandas dataframe to matrix

```
In [5]: user_data_np = user_data.as_matrix()
        airport_data_np = airport_data.as_matrix()
```

1.5 Connect to Redis Server

```
In [8]: r = redis.StrictRedis(host='localhost', port=6379, db=0)
```

1.6 Add user and airport data to redis geo set

```
In [9]: # add user data to geo set
        for row in user_data_np:
            r.geoadd("data", row[2], row[1], row[0])
```

```
In [7]: # read airport data into redis geo set
        for row1 in airport_data_np:
            r.geoadd("data", row1[2], row1[1], row1[0])
```

1.7 Create two lists having airport names and uuid.

```
In [11]: airp = []
         for i in r.zrange("data", 0, -1):
             if len(i)==3:
                 airp.append(i)
```

```
In [12]: uuser = []
         for i in r.zrange("data", 0, -1):
             if len(i) > 3:
                 uuser.append(i)
```

1.8 Find the nearest airport.

Below code will output a dictionary that will have uuid and nearest airport. I am leveraging Redis geodist() that is efficient in calculating geodistance in a set.

```
In [ ]: start = time.time()

        distt = {}
        for i in uuser: # iterate over each user
            dis_a = {}
            for j in airp: # iterate over each airport
                dis_a[j] = r.geodist("data", i, j) # calculate geo distance for eac
            distt[i] = min(dis_a, key=dis_a.get) # find the minimum and save it in

        end = time.time()
        print((end - start)/60) # print time taken to execute the program.
```

1.9 Write the output file to csv.

```
In [ ]: with open('final_output.csv', 'w') as f:
            w = csv.DictWriter(f, distt.keys())
            w.writeheader()
            w.writerow(distt)
```