CSE 427S Final Project: Geo-Location Clustering Using K-Means Algorithm in Spark

Jiahui Li, Bing Zhang, Ying Zhu, Ruojing Jia

December 13, 2019

1 Motivation

In this project, we are dealing with geo-location clustering problems. Clustering is an unsupervised learning task to separate a great amount of datasets or populations into a number of groups so that the data points are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

The algorithm applied to solve geo-location clustering is K-Means algorithm, which is a distance-based method that iteratively updates the location of k cluster centroids until convergence. The tool we used to implement K-Means algorithm is Apache Spark, which is a powerful tool in cloud computation and overcomes many limitations of Hadoop MapReduce. We will firstly execute the algorithm to cluster device location data, synthetic location data and DBpedia location data locally with different distance methods and different values of parameter k. We visualized the clustering result and analyzed the runtime of different experiment setups. Then we will apply it to a large-scale Detroit criminal offenses dataset on a real cluster hosted in Amazon EMR.

On one hand, K-Means geo-location clustering provides great importance to real-world circumstances such as marketing, supply chain logistics and criminal detection, we can make use of the results to build a more efficient and effective world. On the other hand, in the context of Big Data Analysis and Cloud Computing, it also gives us a chance to study further for this project with AWS implementation, which is a powerful engine, to learn the parallel computation.

2 Data Preparation

2.1 Prepare device status data

Before implementing the K-Means algorithm, we have to firstly pre-process the data to obtain a clean and standard format. The following steps describes the procedure of pre-processing device location data:

- Use `textFile()` to load the device status dataset.
- Use `map()` transformation to split each line by the character at position 19 so that each record is split by either commas or pipes.
- Use `filter()` transformation to filter out records which do not have exactly 14 values. Those records will not be parsed correctly.
- Use `map()` transformation to extract the attributes including longitude and latitude and store them as the first two fields followed by date, model and device ID. For each record, the model field is then separated by manufacturer name and model name by space.
- Use `filter()` to filter out the records which have longitude or latitude with values equal to 0.
- For each record, use `map()` to transform each record to one string with attributes separated by comma.
- Use `saveAsTextFile()` to save the extracted data to comma delimited text file in /loudacre/devicestatus_etl directory on HDFS.

The following screenshot shows the part of the pre-processed device location data:

```
33.6894754264,-117.543308253,2014-03-15:10:10:20,manufacturer Sorrento,model F41L,8cc3b47e-bd01-4482-b500-28f2342679af
37.4321088904,-121.485029632,2014-03-15:10:10:20,manufacturer MeeToo,model 1.0,ef8c7564-0a1a-4650-a655-c8bbd5f8f943
39.4378908349,-120.938978486,2014-03-15:10:10:20,manufacturer MeeToo,model 1.0,23eba027-b95a-4729-9a4b-a3cca51c5548
39.3635186767,-119.400334708,2014-03-15:10:10:20,manufacturer Sorrento,model F41L,707daba1-5640-4d60-a6d9-1d6fa0645be0
33.1913581092,-116.448242643,2014-03-15:10:10:20,manufacturer Ronin,model Novelty,db66fe81-aa55-43b4-9418-fc6e7a00f891
33.8343543748,-117.330000857,2014-03-15:10:10:20,manufacturer Sorrento,model F41L,ffa18088-69a0-433e-84b8-006b2b9cc1d0
37.3803954321,-121.840756755,2014-03-15:10:10:20,manufacturer Sorrento,model F33L,66d678e6-9c87-48d2-a415-8d5035e54a23
34.1841062345,-117.9435329,2014-03-15:10:10:20,manufacturer MeeToo,model 4.1,673f7e4b-d52b-44fc-8826-aea460c3481a
32.2850556785,-111.819583734,2014-03-15:10:10:20,manufacturer Ronin,model Novelty,a678ccc3-b0d2-452d-bf89-85bd095e28ee
45.2400522984,-122.377467861,2014-03-15:10:10:20,manufacturer Sorrento,model F41L,86bef6ae-2f1c-42ec-aa67-6acecd7b0675
37.9248961741,-122.206868167,2014-03-15:10:10:20,manufacturer iFruit,model 3,27178d24-3a61-42f7-a784-e3263f25cc6f
38.1653163975,-122.151608378,2014-03-15:10:10:20,manufacturer Titanic,model 2400,b4a15931-9a69-469f-9823-a45974472c51
33.323126641,-116.472234745,2014-03-15:10:10:20,manufacturer Ronin,model S1,e75dc777-b531-4dbd-80d5-39c772666e6a
33.1774985363,-116.889226299,2014-03-15:10:10:20,manufacturer Ronin,model Novelty,d4ebd9ae-4dad-4fb4-ba1d-d2a9610a458d
32.2083493316,-111.434102713,2014-03-15:10:10:20,manufacturer Ronin,model Novelty,b954db08-1f97-4311-8d42-1a7ba39d8dcf
34.0487620041,-111.928871717,2014-03-15:10:10:20,manufacturer MeeToo,model 1.0,16085fbf-cda5-4489-84b9-0fad888f9e7a
37.9031053656,-121.561451342,2014-03-15:10:10:20,manufacturer iFruit,model 1,6474caf1-7bbf-4594-a526-9ba8ea82e151
36.032967794,-118.970108886,2014-03-15:10:10:20,manufacturer MeeToo,model 5.0,668e6f06-a8aa-4be5-8609-899c45d3caa8
45.0400810371,-117.858004521,2014-03-15:10:10:20,manufacturer Ronin,model Novelty,6d195272-8dba-42d6-9b1f-fe61edf7f2a2
35.2338863976,-114.3057523,2014-03-15:10:10:20,manufacturer Sorrento,model F10L,d228cdab-8b35-4733-9f2e-e4760dfacb30
```

Figure 1: Pre-processed device location data

The following figure shows the visualization of the device location data:

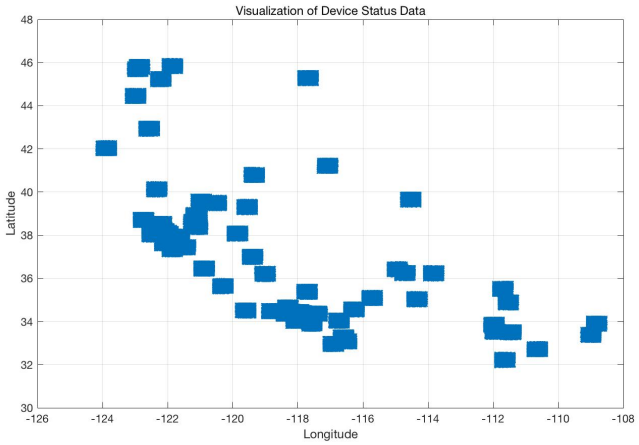

Figure 2: Visualization of pre-processed device location data

2.2 Get and Visualize synthetic location data

For synthetic location data, we downloaded the data and visualized the (latitude, longitude) pairs and the following figure shows the visualization of the synthetic location data:

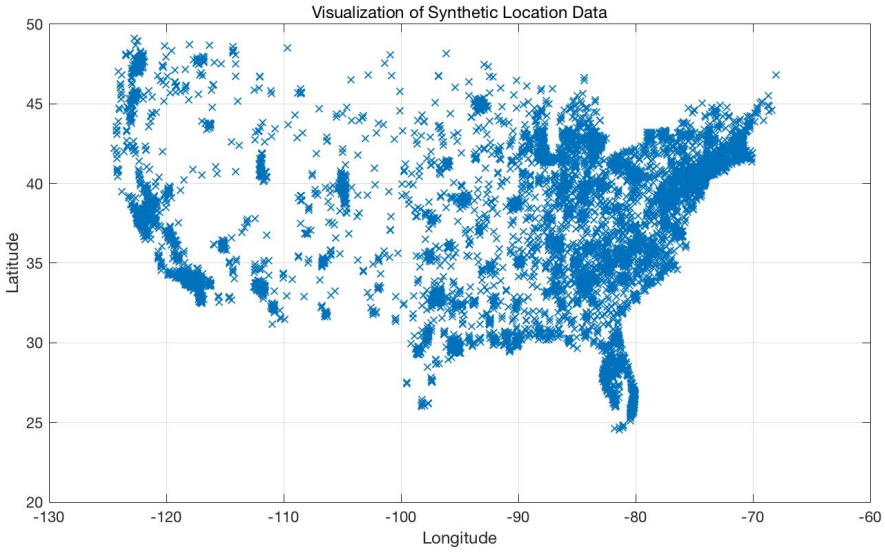

Figure 3: Visualization of synthetic location data

2.3 Get and Pre-process the DBpedia location data

We downloaded the large-scale clustering data of (latitude, longitude) pairs extracted from DB-pedia. The following figure shows the visualization of the DBpedia location data:

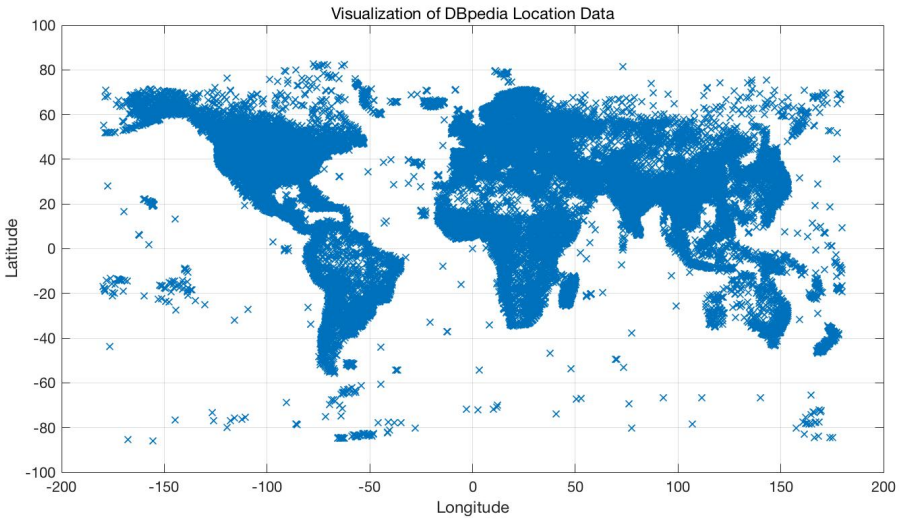

Figure 4: Visualization of DBpedia location data

Since the DBpedia data is a global geo-location data, we can put a bounding box around the US and filter only those records inside the bounding box to get a smaller sample of this dataset. We set the longitude ranging between -66 and -125 and the latitude ranging between 25 and 47. The following figure shows the visualization of the small sample of DBpedia location data:

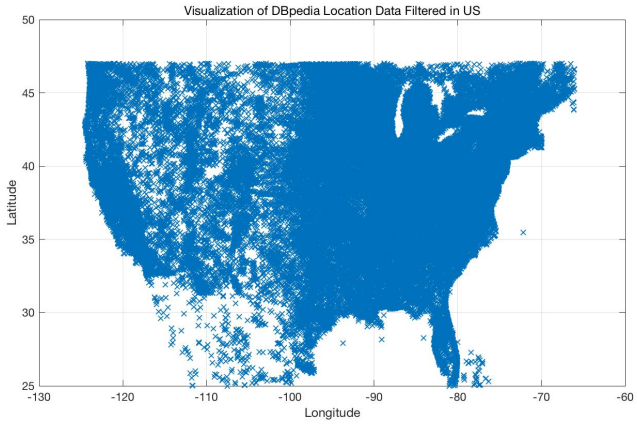

Figure 5: Visualization of small sample of DBpedia location data

3 Clustering Approach - K-Means Algorithm

In this section, we will introduce the approach of our clustering task which is K-Means algorithm and the implementation of K-Means algorithm in Spark.
The K-Means algorithm we applied is an iterative algorithm that tries to partition the dataset into k pre-defined distinct non-overlapping clusters where each data point belongs to only one group. It tries to make the inter-cluster data points as similar as possible while also keeping the clusters as far as possible by assigning data points to a cluster such that the sum of the squared distance between the data points and the clusters centroid is at the minimum.
Basically, K-Means algorithm has three phases:

- Initialization:
  Firstly, we randomly sample k initial data points from the dataset as the initialized k centroids.

- Data assignment:
  Secondly, every centroids defines the corresponding clusters. For each data point, calculate the distance for every centroids and assign it to the nearest centroid.

- Centroid update:
  After all data points are assigned to one cluster, the centroid of each cluster are recomputed. We use the mean of all data points assigned to that centroid's cluster as the new centroid.

In this project, we have two methods to calculate the distance between two data points which are represented in (latitude, longitude) pairs: Euclidean distance and Great Circle distance. Euclidean distance is the straight-line distance between two points in Euclidean space. Given two geo-location points (x, y), the formula of Euclidean distance is

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2},$$

where x_1, y_1 represents latitude and x_2, y_2 represents longitude. The Great Circle distance is the shortest distance between two points on the surface of a sphere, measured along the surface of the sphere. Given two geo-location points (x, y), the formula of Euclidean distance is

$$\Delta\sigma = \arctan \frac{\sqrt{(\cos\phi_2 \sin\Delta\lambda)^2 + (\cos\phi_1 \sin\phi_2 - \sin\phi_1 \cos\phi_2 \cos\Delta\lambda)^2}}{\sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos\Delta\lambda},$$

where $\lambda_1, \phi_1, \lambda_2, \phi_2$ are the geographical longitude and latitude in radians of two points (x, y). The algorithm iterates between steps two and three until a stopping criteria is met. The original method use the sum of the distances is minimized as stopping criteria. Considering that we are dealing with thousands, even millions of data points in a big data set, calculating the sum of distance is a relatively huge workload. We use the criteria that the sum of Euclidean distance between previous center points and newly assigned center points, instead of total sum of distances, should be minimized to improve the efficiency and effectiveness of this approach. The function is as follows:

$$\sum_{i=1}^{k} d(x_i, y_i) = \sum_{i=1}^{k} \sqrt{(x_{i1} - y_{i1})^2 + (x_{i2} - y_{i2})^2}$$

We set the converging criteria at 10^{-2} for the total distance change between two iterations.

4 Small Dataset on Pseudo Cluster

In this section, we executed out K-Means algorithm on three different datasets which were introduced in previous section: device location data, synthetic location data, DBpedia location data. By performing our experiments on a pseudo cluster, we aimed to get the clustering result of each dataset with different values of k and different distance methods, and analyze the runtime of experiments with different setups.

4.1 Clustering Result

We executed K-Means algorithm for three datasets using both Euclidean distance and Great Circle distance and compare the result of clustering for different distance methods and different value of k. The goal is to observe how the choice of distance methods and values of k influences the clustering results.
Firstly, we calculated the K-Means clusters for the device location data using $k = 5$. The following two figures correspond to visualized clusters of the device location data with Euclidean and Great Circle distance respectively.

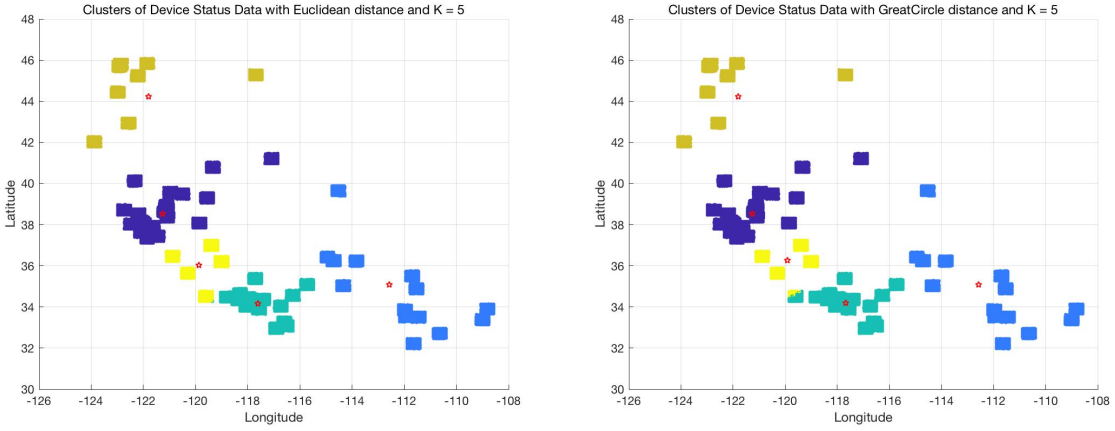

Figure 6: Clustering of device location data with $k = 5$

From the figures above, we can see that there is nearly no difference between the clusters using Euclidean distance and the clusters using Great Circle distance with $k = 5$. We know that the device location data is generated from the spots located on the west cost of U.S. so the range of latitude and longitude is narrow. So we can view the the spherical location data as two-dimensional location data, thus there is almost no difference between the calculation of Euclidean and Great Circle distance methods. What's more, we can see that there already exists some groups in the device location data so the clustering results of K-Means algorithm are similar.

Next, we calculated the K-Means clusters for the synthetic location data using $k = 2$ and $k = 4$. The following two figures correspond to the visualized clusters of the synthetic location data using Euclidean and Great Circle distance with $k = 2$.

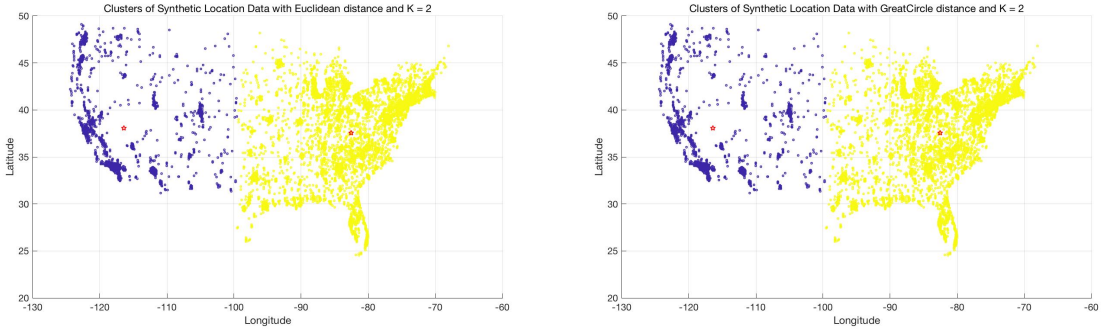

Figure 7: Clustering of synthetic location data with $k = 2$

From the above two figures, we can see that the two K-Means clusters roughly separates the synthetic location data to left and right groups. The amount of data points in the left cluster is much smaller than the right. Still, there is no obvious difference between the the two clustering results using Euclidean distance and Great Circle distance because of the narrow range of latitude and longitude.
The following two figures correspond to the visualized clusters of the synthetic location data using Euclidean and Great Circle distance with $k = 4$.

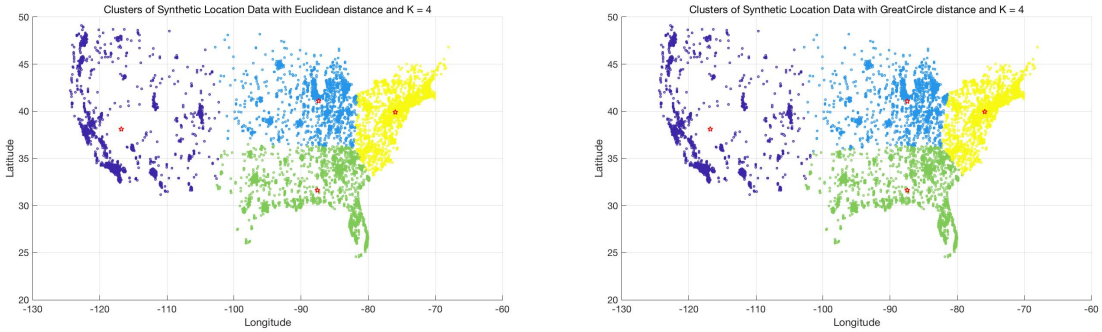

Figure 8: Clustering of synthetic location data with $k = 4$

From the figures above, we can see that there is little difference at the boundary between the clusters using Euclidean and Great Circle distances when $k = 4$. What's more, we noticed that when k increased from 2 to 4, the centroid in the left side barely changed while the new two centroids appeared in the right side and separated the data points on the right side into three groups. Consider the distribution of the data points, we can say that the centroids are more likely to appear within a dense cluster of points than sparse one.

Finally, we calculated the K-Means clusters for the large-scale DBpedia location data. As we can see, the DBpedia location data is a fairly large dataset with over 400 thousands amount of data points that locates over the world. We use $k = 6$ for a start and then increases the value of k to find the optimized solution. The following two figures shows the visualized clusters of the DBpedia location data using Euclidean and Great Circle distance with $k = 6$.

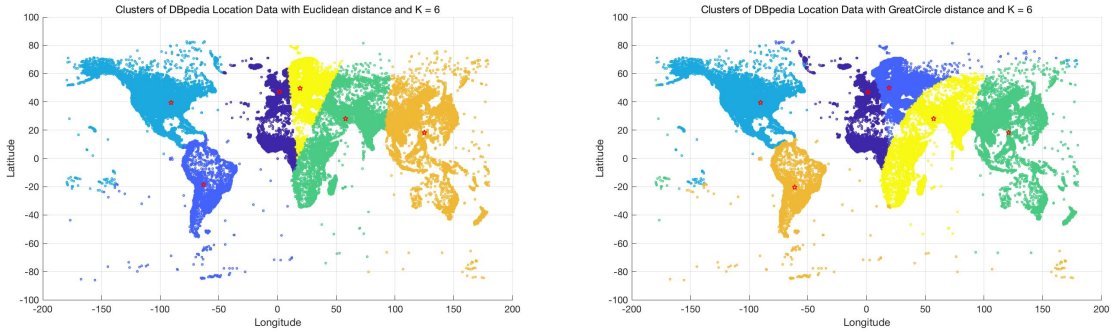

Figure 9: Clustering of DBpedia location data with $k = 6$

From the figures above, we can see that there is obviously much more difference between the clusters using Euclidean and Great Circle than the previous two datasets. This is because the range of latitude and longitude is much larger of DBpedia data so the difference between Euclidean and Great Circle distances when calculating two data points is greater. The difference of the two distance methods lead to the different cluster results. What's more, we can see that the boundary of clusters using Great Circle distance is arc-shaped, which denotes the sphericity of the data points.
Then we increase the value of k as $k = 7, 8$. The following four figures shows the visualized clusters of the DBpedia location data using Euclidean and Great Circle distance with $k = 7, 8$.

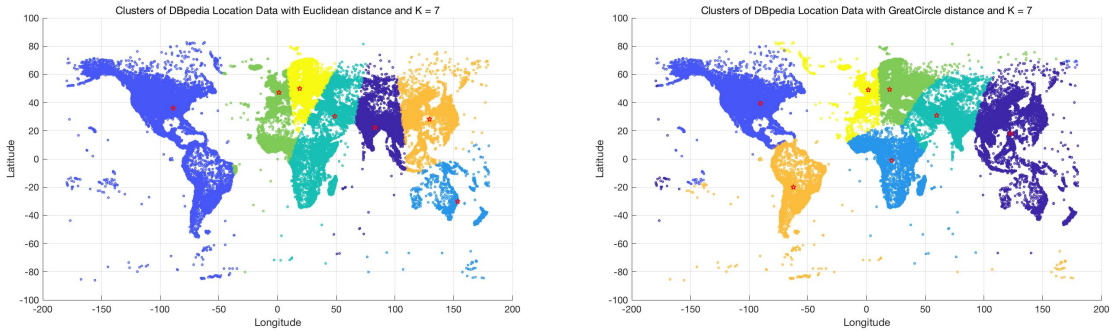

Figure 10: Clustering of DBpedia location data with $k = 7$

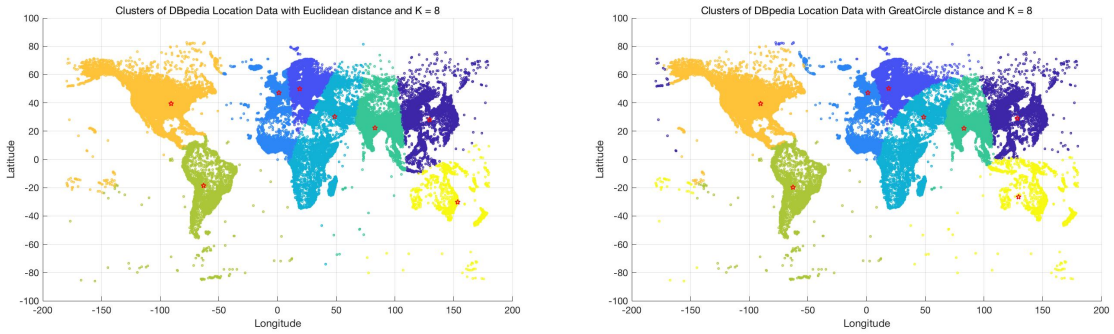

Figure 11: Clustering of DBpedia location data with $k = 8$

When $k = 7$ and 8 it makes more sense, because the clusters could represent the continents in reality. For Euclidean distance, when $k = 7, 8$, Australia can be separated from Asia. For Great Circle distance, when $k = 7$, Africa can be separated from Eurasia. When $k = 8$, Australia can be separated from Asia.

From all the figures above, we can see that there is obvious difference between the clusters and the cluster centers using Euclidean distance and Great Circle distance with $k = 6, 7, 8$. When $k = 6$, the figure using Euclidean distance and the figure using Great Circle distance have different boundary between clusters, especially the boundary lies in west Asia. When $k = 7$, the graph using Euclidean distance can separate Australia from Asia, while the Great Circle distance can not. And the cluster centers are different. When $k = 8$, the cluster centers are different, especially the cluster center in Australia. In reality, longitude and latitude are angular measurements. When the dataset has broad latitude and longitude, the difference between clustering using Euclidean distance and clustering using Great Circle distance might be obvious.

What's more, we randomly sampled 30% data points from the DBpedia dataset and calculated the K-Means clusters with $k = 6$ on both distance methods. The result is as the following figures.

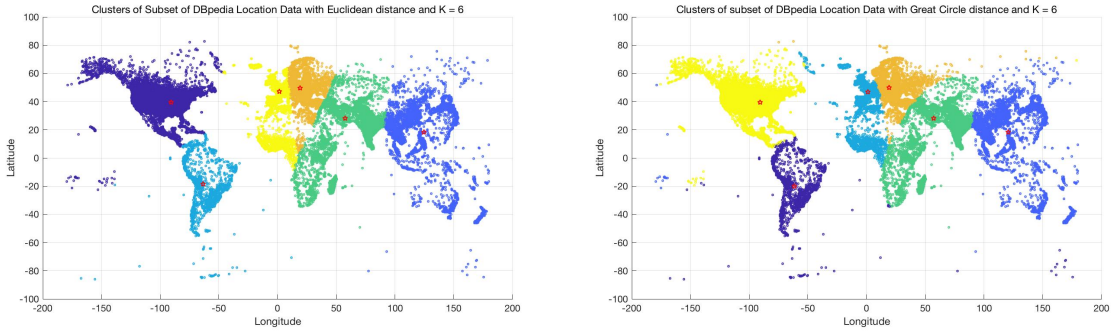

Figure 12: Clustering of 30% sample DBpedia location data with $k = 6$

From the above figures, we noticed that the two distance methods produced different clusters like the clustering result of the whole dataset. So we have the conclusion that the size of the datasets actually does not influence the result of two distance methods. Instead, the distribution of geo-location datasets decides the clustering result of Euclidean and Great Circle distance. When the dataset is sparsely and largely distributed around the world, the difference between the results produced by K-Means algorithm using Euclidean distance and Great Circle distance is much more obvious.

4.2 Runtime Analysis

After comparing the cluster results for the three datasets using different value of k and distance methods, we want to compare the difference of runtime for different setups of our experiments. We aim to find out how the size of a dataset, distance methods, value of k and the number of threads could affect the Spark runtime performance.
Firstly, we want to compare the runtime between different values of k. We ran the K-Means implementation using $k = 2, 4, 6$ for the three data sets using local mode with 2 and 4 threads. The following graphs show the runtime result of each experiment.

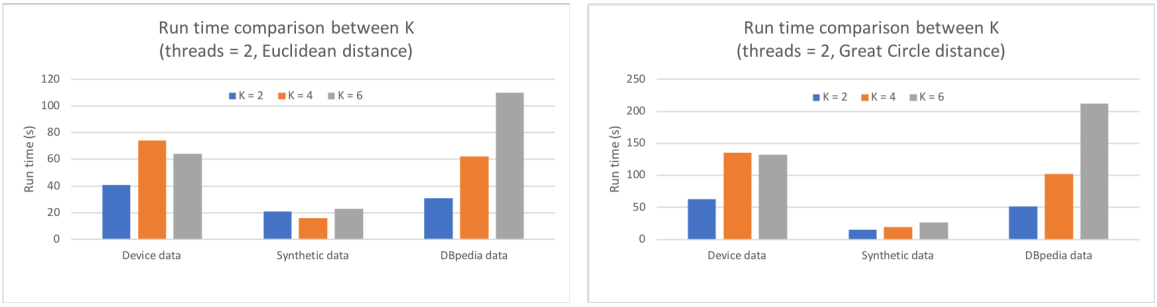

Figure 13: Runtime using local mode with 2 threads

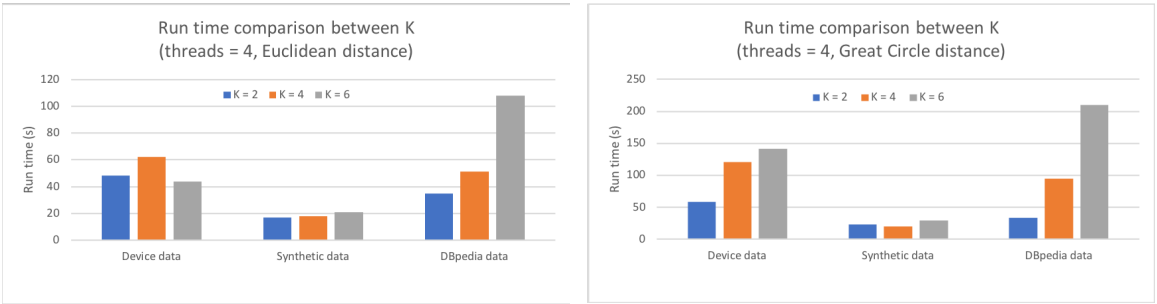

Figure 14: Runtime using local mode with 4 threads

We can see from the graphs that when the value of k increases, the total runtime increases in most of the experiments. Since when the number of clusters increases, number of computation of assigning each data point to its closest centroids in each iteration increases. So the total computation and runtime should also increase. On the other side, the increasement of clusters may also lead to a early stop of the iterations since it is easier to groups the data points with

more clusters. In this way, we can say that the change of the value of k affects the runtime in two ways: amount of computation and number of iterations.
Next, we want to compare how the number of threads affect the runtime. We ran the K-Means implementation using $k = 2, 4, 6$ for the three data sets using local mode with single thread, 2 and 4 threads. The following graphs show the runtime result of each experiment.

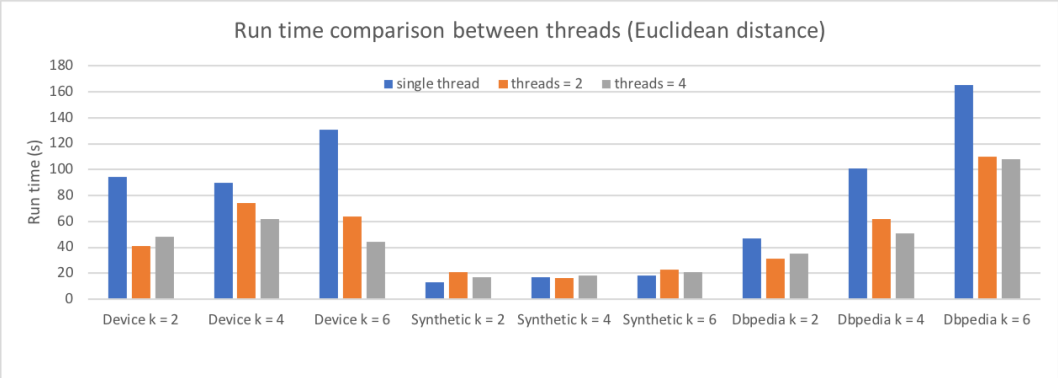

Figure 15: Runtime using Euclidean distance method

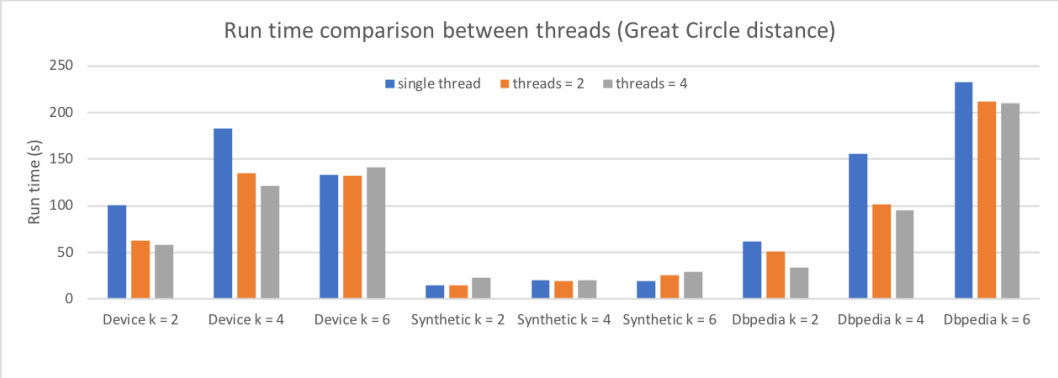

Figure 16: Runtime using Great Circle distance method

We can see from the graphs that when the number of threads increases, the runtime decreases in most of the experiments. But the degree of decreasement of runtime largely depends on the size of the dataset. For a small amount of dataset like synthetic location data, the runtime in single thread, 2 and 4 threads are almost equally small. For some larger datasets like device location data and DBpedia data, the decreasement of runtime is much more obvious. What's more, we also noticed that when number of threads increases from 2, the runtime does not improve much for any of the dataset.
Finally, we want to compare how persistent RDDs will affect runtime performance. We ran the K-Means implementation with persistent and without persistent separately using $k = 2, 4, 6$ for the three data sets with 2 threads. The following graphs show the runtime result of each experiment.

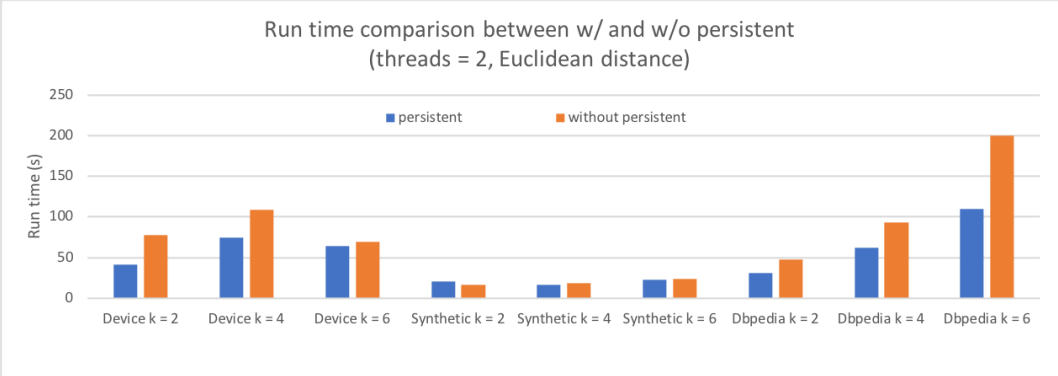

Figure 17: Runtime using Euclidean distance method

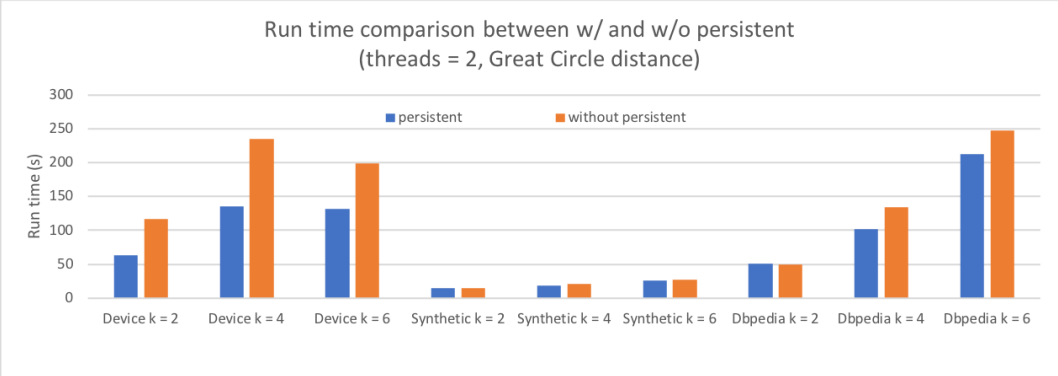

Figure 18: Runtime using Great Circle distance method

We can see from the graphs that when we do not use persist RDDs in our Spark implementation, the runtime increases. After loading and parsing the input data points into a persistent RDD, the RDD is saved in memory and do not have to re-executes the lineage transformation for each iteration. If the RDD is not persistent, each action during the iterations will cause re-execution and reload the data which costs redundant computation and longer runtime. What's more, the change of runtime also depends on the size of dataset. For a small amount of data like synthetic location data, the decreasement of runtime using persistent RDD is not obvious. For larger dataset like device location data and DBpedia data, using persistent RDD surely decreases the runtime.

5 Big Data and Cloud Execution

In this section, we found our own large-scale dataset and clustered this data by executing the Spark program on Amazon EMR.

5.1 Introduction of Dataset

This data is from Detroit Police Department's CrisNet/NetRMS records management system in data.world website (https://data.world/detroit/dpd-crime-incidents-2009-2016). The size of the data is 134.3 MB, with $1,048,576$ rows (records). The data reflects the reported criminal offenses that have occurred in the City of Detroit from January 1, 2009 to June 26, 2016. The following figure shows some of the original data records.

```
1,1099487,1321797,910020373.1,MISCELLANEOUS,99009,1/1/09,0,1107,11,City Council District 3,CONANT GARDENS,5070,-83.0649,42.4261
2,1117507,1344185,911060289.1,MISCELLANEOUS,99009,1/1/09,0,,,,,9999999,999999.0001,999999
3,985415,1181882,902190512.1,MISCELLANEOUS,99009,1/1/09,0,1005,10,City Council District 5,PECK,5313,-83.1058,42.3821
4,986019,1182632,902200294.1,MISCELLANEOUS,99009,1/1/09,0,414,4,City Council District 6,HUBBARD-RICHARD,5211,-83.083,42.3145
5,996883,1195867,903170149.1,LARCENY,23003,1/1/09,0,908,9,City Council District 3,BURBANK,5052,-83.008,42.4134
6,967855,1160270,901080218.1,LARCENY,23003,1/1/09,0,312,3,City Council District 6,NECKLACE DISTRICT,5172,-83.0494,42.3358
7,1015981,1219182,904240362.1,LARCENY,23007,1/1/09,0,411,4,City Council District 6,BOYNTON,5247,-83.1481,42.282
8,1011804,1213977,904160245.1,FRAUD,26001,1/1/09,0,601,6,City Council District 1,ELIZA HOWELL,5441,-83.2627,42.3906
9,1038456,1246721,906090137.1,ASSAULT,13003,1/1/09,0,908,9,City Council District 4,LASALLE COLLEGE PARK,5052,-83.0048,42.4185
10,964664,1156290,901010120.1,DAMAGE TO PROPERTY,29000,1/1/09,0,201,2,City Council District 1,CERVENY,5375,-83.1969,42.4055
```

Figure 19: Some records of CrisNet dataset

In order to protect the privacy of crime victims, the geographic coordinates (longitude and latitude) have been randomly adjusted to provide accuracy only to the street block and do not identify precise crime locations. Some incidents involve the commission of multiple offenses, such as a domestic assault where property was also vandalized. Accordingly, the data describe all offenses associated with all reported incidents, excluding sexual assaults.
After preprocessing, we will use K-Means method clustering this data by executing Spark program on Amazon EMR. By calculating the K-Means clusters of the CrisNet location data, we can group the whole Detroit area into several clusters by the location of reported criminal offenses. The cluster centroids can be viewed as the central area with high frequency of criminal offenses. In practical term, the clustering result of CrisNet data can reflect the distribution of criminal offenses in Detroit and can be view as a safety guide of living in Detroit.

5.2 Data Pre-processing

Firstly, in order to get the splitted features, we delimited the data by comma. Then, we deleted the irrelevant features(only the latitude and longitude data were kept), as what we wanted to do was geo-location clustering. Finally, we deleted the data that was apparently unreasonable or outside Detroit. To be more specific, we only kept the data with latitude between 42.25 and 42.45, longitude between -83.3 and -82.9, considering the location of Detroit.
After pre-processing, it remains 1,043,976 rows of geo-location data. The size of the data reduces to 17.5MB. The following figure show some records of preprocessed dataset.

```
42.3821,-83.1058
42.3145,-83.083
42.4134,-83.008
42.3358,-83.0494
42.282,-83.1481
42.3906,-83.2627
42.4185,-83.0048
42.4055,-83.1969
42.3766,-83.0318
```

Figure 20: Some records of pre-processed CrisNet dataset

We visualized the pre-processed CrisNet data shown in the following figure. We can see that the visualization of the data set is exactly the Detroit area. We also notice that some area are extremely dense while some area are sparse. We want to use K-Means clustering to reveal the pattern of the data points distribution.

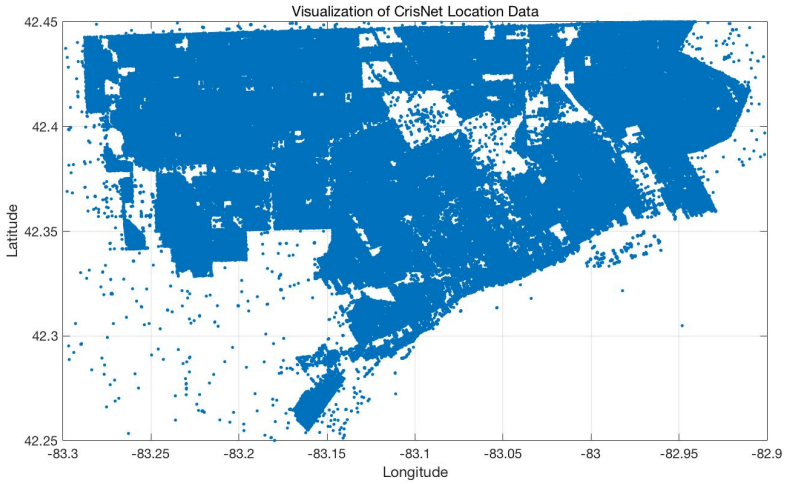

Figure 21: Some records of CrisNet dataset

5.3 Cloud Execution Approach

We used AWS EMR as our real cluster to run K-Means cloud execution on the large-scale CrisNet dataset. Firstly, we launched an Amazon EC2 instance (Amazon Linux AMI) as a virtual machine on the cloud. Next, we setup out S3 Bucket to store the program and data files. We uploaded the implemented K-Means code and preprocessed CrisNet data to the S3 bucket inside the EMR directory in order to process the big dataset on Amazon EMR. Then, we created the cluster and add a step by specifying arguments including the spark script, input and output directories, k (the number of clusters we would like to use), and distance measure (either Euclidean distance or Great Circle distance). The following figures show organization of the Amazon EC2 instance, S3 bucket and the cluster jobs.

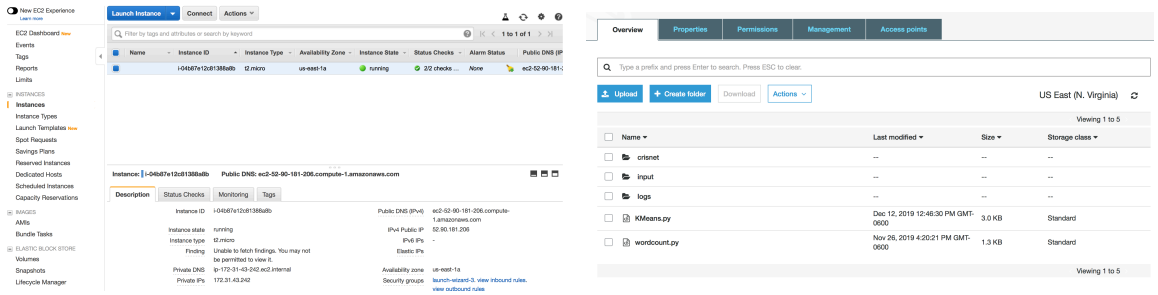

Figure 22: Amazon EC2 and B3 bucket

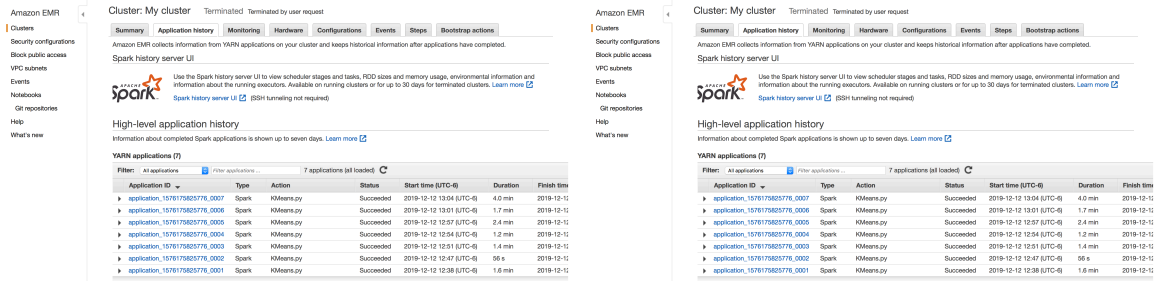

Figure 23: Cluster steps status and job history

5.4 Clustering Result

We calculated the K-Means clusters of CrisNet dataset using Euclidean and Great Circle distances and $k = 4, 5, 6, 8$. The following eight figures correspond to visualized clusters of the criminal offenses data in Detroit using Euclidean and Great Circle distance, with $k = 4, 5, 6, 8$ respectively.

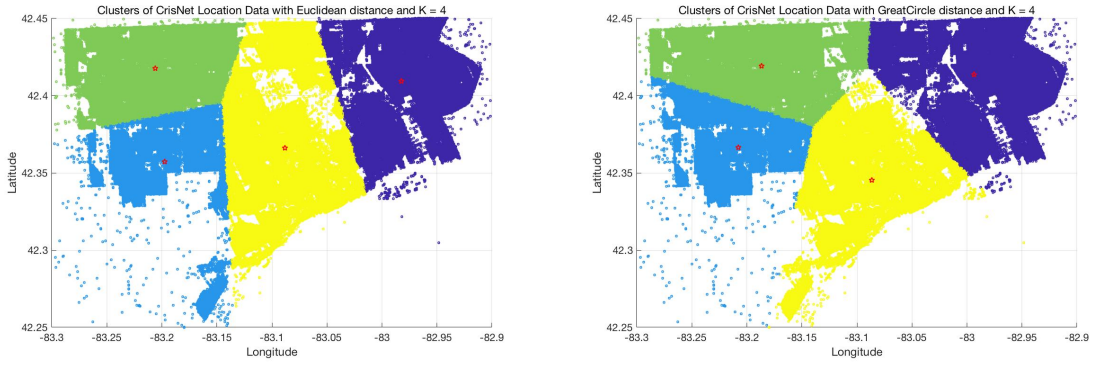

Figure 24: Clustering result using $k = 4$

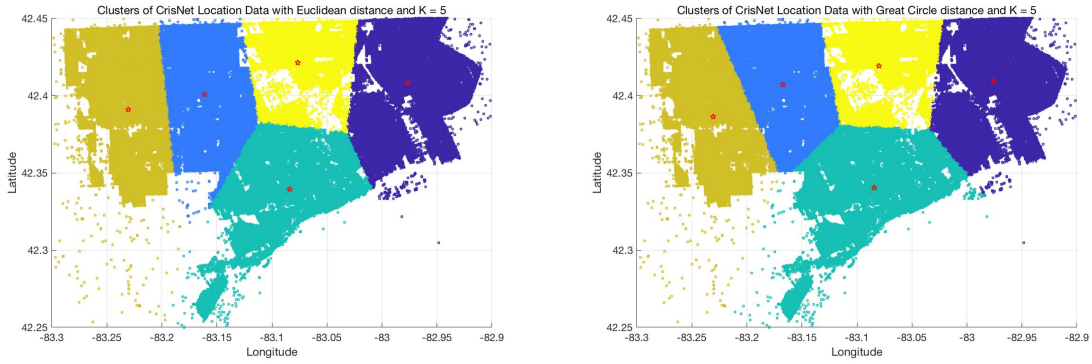

Figure 25: Clustering result using $k = 5$

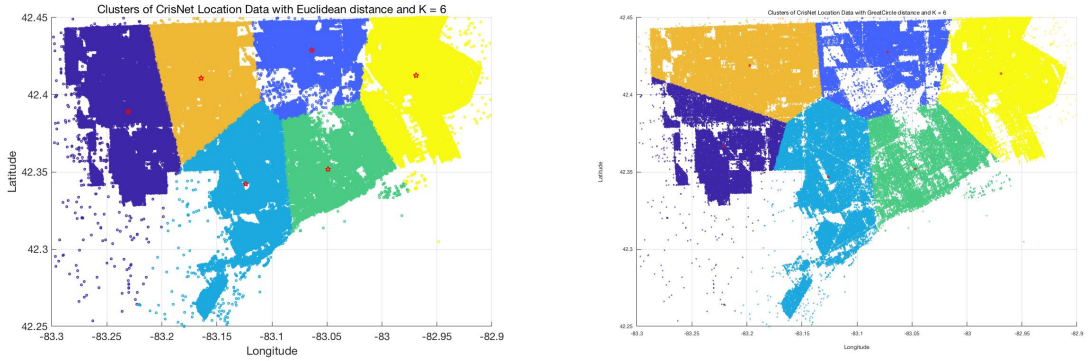

Figure 26: Clustering result using $k = 6$

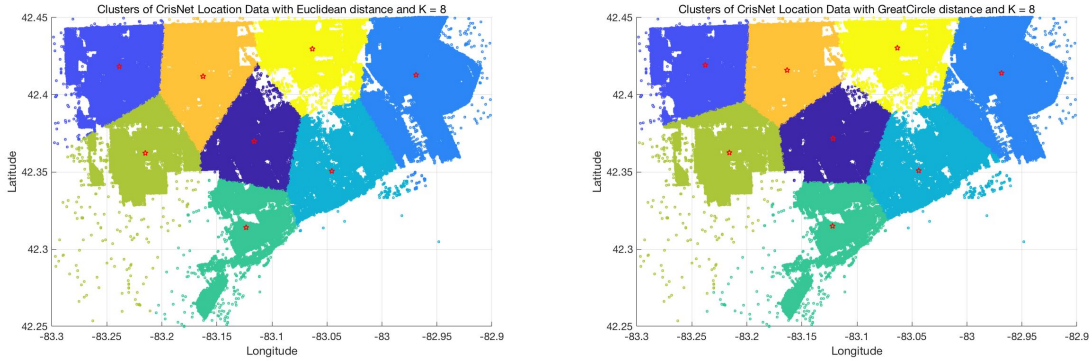

Figure 27: Clustering result using $k = 8$

From the clusters above, we can see that there is nearly no difference between the clusters using Euclidean distance and the clusters using Great Circle distance with $k = 4, 5, 6, 8$. This might because the range of the latitude and the range of the longitude is narrow (the latitude and the longitude in this dataset only lies in Detroit).
What's more, we noticed that there are already some boundaries lie in the original CrisNet dataset, which is shown as blank area and lines in the visualization of the dataset. From visual inspection, we get relatively good results when $k = 6$ and $k = 8$ since the clusters almost separate the Detroit area in the same way the blank boundaries do. This means that our K-Means clustering results corresponds with the real districts and the clusters could represent the districts in Detroit in reality.

5.5 Runtime Analysis

We have compared the runtime for cloud execution using two distance measures including Euclidean distance and Great Circle distance. We explored the differences in runtime among different number of clusters when $k = 4, 5, 6, 8$. The following graph shows the runtime for all experiments.

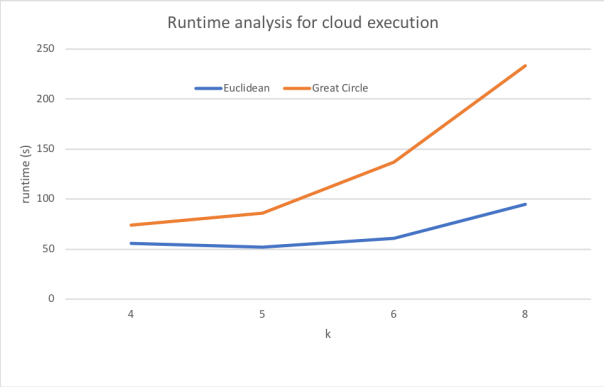

Figure 28: Some records of CrisNet dataset

From the line graph above, we can see that the runtime for Great Circle distance is much longer than Euclidean distance when same number of clusters are used. When k = 4, the runtime for Euclidean distance is a little bit over 50 seconds, comparing to the runtime for Great Circle distance which takes up to around 75 seconds. It is because that the mathematical computations of Great Circle distance is much more complicated than Euclidean distance. In addition, as k, the number of clusters increases, the runtime also increases due to the increased amount of computations needed for calculating the distance of each datapoint with respect to the cluster center.

6 Conclusion and Lesson Learned

K-Means algorithm is a useful application which is easily transferable if we would like to explore any other datasets with features including longitude and latitude.
We have implemented the K-means algorithm and tested on total of 4 datasets, including three small datasets run on local machine and one large dataset run on cloud. These datasets are all based on features of geolations-longitude and latitude.The k-means algorithm allowed us to visually inspect the similarities of data points which are grouped together within a specific range.
As for exploring the number of clusters should be used in application, it is extremely important to consider the size of the dataset, as well as the expected accuracy and runtime. In order to determine which k is the best, we have done some experiments and found out that as k increases, the runtime also increases, no matter if the application is executed on local machine or cloud. As more values of k are tested, the average distance to the closest cluster center will change little when k increases, which means that the increasing k does not help improving the clustering accuracy too much with respect to the computational cost. Therefore, the best k cannot be easily determined in general, since the best k varies by which datasets we use and what conclusion we expect to reach. As for the dataset (Detroit Police Department's CrisNet/NetRMS records management system) we have used in cloud execution, we believed that a relatively good k is 6 or 8 by visual inspection. The clusters being grouped by the k-means algorithm using k = 6 and k = 8 is similar to what we have expected out of the real boundaries of Detroits geological locations.
Furthermore, regarding the use of Euclidean distance and Great Circle distance, we believed that there was not much big difference when the range of both longitude and latitude is small. However, if we would like to apply the algorithm on datasets on a broader longitude and latitude scale, we should better use the Great Circle distance since it is more representative of the reality that the earth is round-shaped. However, there will be a runtime and accuracy tradeoff due to the increasing time needed by using Great Circle distance measure instead of Euclidean distance measure.
In addition, by using multiple threads instead of a single thread on Spark can effectively reduce the runtime of analysis. However, there was not much significant improvement in runtime as we increased 2-thread execution to 4-thread execution.

7 Future Work

In this project, we mainly have utilized visual inspection technique in determining the good k. However, visual inspection can only serve as a general guidance because it is too subjective and requires a lot of human judgments. Hence, if we would like to determine the breakpoint at which the accuracy and runtime tradeoff is the most balanced, we should do some technical analysis with respect to k. For the same dataset, more experiments on k should be done while other variables like distance measure and number of parallel execution processes being same. Then the average distance to the closest cluster center should be calculated with respect to different ks. While the average distance to the closest cluster center does not decrease too much as k increases, we can reach the conclusion about which k should be chosen as the optimal k.