Sixth International Conference on Information Science and Technology
Dalian, China; May 6-8, 2016

A Quadtree-based Hierarchical Clustering Method for Visualizing Large Point Dataset

Qing CAI and Yimin ZHOU
School of Optical-Electrical and Computer Engineering
University of Shanghai for Science and Technology
Shanghai 200093
Email: tsing.tsai@icloud.com, zhouyimin_58@163.com

***Abstract*—Many digital map applications have the need to present large quantities of precise point data on the map. Such data can be PM2.5 value, weather information, population in towns, etc. With the development of Internet of Things, we expect such data will grow at a rapid pace. How to visualize such magnitude of data on mobile devices and web broswers becomes a problem. This paper describes a method for interactively visualizing such data, using a combination of grid-based clustering and hierarachical clustering, along with quadtree spatial indexing.**

Keywords*—Geovisualization, hierarchical clustering, quadtree spatial index, big data

I. Introduction

With the development of sensor technology and Internet of Things, a great number of geospatial data source are emerging every year. Such data range from temperature, wind speed, geo-taged photos and traffic information from highway cameras, etc. The number of these data can easily jump to billion level as time goes by. They are important for policy-makers and business decisions, and there's growing needs to visualize these data on mobile devices or on computers for end users.

To present such data, choropleth maps, which use color-shadings to represent quantities, are often used in statistical presentation. Another common type is dot maps, which show a dot for each geo-referenced data. Choropleth map is good for visualizing approximate data that gives you instant sense about the values. To present precise geo-referenced data, dot map is better. Users can move the mouse over a dot to see more information about it. Moreover, a dot map can be combined with choropleth map to create an even more interactive and informative map.

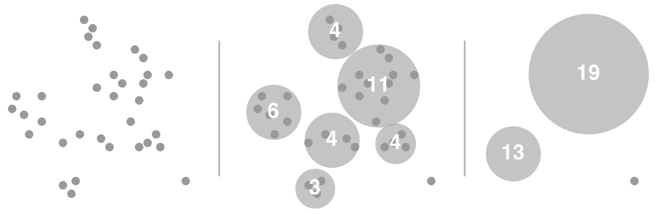

Fig. 1: From left to right, figures are original point set, clustered points at level 1, clustered pointes at level 2, respectively

Fig. 1 gives a general idea of our approach. When users zoom in to the most-detailed level, all points are drawn directly on the map. Zooming out a bit gives view of larger area but points are clustered together. Each cluster has a caption, usually the total number of raw data inside, or any other meaningful representation for the cluster. This is a case of LOD(Level of detail) technology.

Currently, popluar point clustering methods use grid-based approach only, such as *leaflet.markercluster*, which are fast but the clusters usually are not representative, and tends to align the points vertically and horizontally as if in a grid [1]. We choose a combination of grid-based clustering and hierarchical clustering to improve the quality of clustering.

The effects of largeness, according to Antony Unwin and Martin Theus, are storage, data quality, complexity, speed, and display [2]. In our context, two issues have to be addressed: how to effectively cluster points and how to design a hierarchical storage method. A good clustering method will produce high quality clusters with high intra-class similarity and low inter-class similarity. Major clustering approaches include partioning, hierarchical, density-based, grid-based etc [3], some of which can be used together. Quadtree is usually used for in-memory spatial indexing [4], but we can also exploit its idea for on-disk storage. This paper proposes a grid-based, partitioning, hierachical clustering method on quadtree file system storage.

II. Nearby Points Clustering

Hierarchical clustering is an important, well-established technique in unsupervised machine learning. The common hierarchical, agglomerative clustering methods share the same algorithmic definition but differ in the way in which inter-cluster distances are updated after each clustering step. There are seven common clustering schemes, including Ward, centroid and median, etc [5]. Grid-based methods have fast processing time that is related to the size of the grid, rather than the size of dataset. These methods use a single uniform grid mesh to divide the entire plane into cells and choose one point to represent the whole cell.

The space complexity for the basic agglomerative hierarchical clustering algorithms is O(m^2), and time complexity is O(m^3), which severely limits its usage, especially in mobile devices. One possible solution is to use grid-based clustering for preprocessing. This step greatly reduces the amount of total points. Agglomerative hierarchical clustering is, then, performed on the grid cells, instead of the original dataset. The number of the grids is usually much smaller than that of original dataset, hence the processing efficiency can be significantly improved.

Basic agglomerative hierarchical clustering algorithm starts with individual points as clusters, successively merges the two closest clusters until some conditions are met. Typical end conditions can be,

- Certain cluster number is reached, similar to K-Means
- No more clusters are close enough to be merged
- A cluster contains maximum nodes

Unlike some other clustering methods, such as K-Means, hierarchical clustering doesn't have an objective function. To control the final clusters, the aforementioned end conditions should be tested and used.

III. Quadtree-like Nodes Storage on File System

A quadtree is a tree data structure in which each internal node has exactly four children. Each cell has a maximum capacity. When maximum capacity is reached, the bucket splits. We use a variation of quadtree, which has fixed depth and no maximum capacity, allowing O(1) access time for any nodes given a region and resolution. This works especially well if the height of tree is carefully chosen according to the distinctive features of dataset. Quadtree is originally designed for in-memory spatial indexing. Large datasets might not fit in the memory. We can apply quadtree to the tree structure of file system, in the following way:

1) A cell is splited into four equally sized subcells, with the name “a”, “b”, “c”, “d” in clockwise order. A folder is created for each subcell.
2) A file named “node” is created for each node to save node data.

There are two types of hierarchies that we need to take into account:

- Point clustering hierarchy
- File system hierarchy

These two hierarchies can be matched together for unified access. Resolution of map defines the depth of nodes, and view region defines the access path. The query algorithm is listed in the following section.

A. Height of Quadtree

As mentioned before, the height of quadtree should be carefully selected, to keep the number of points in leaf node below certain threshold. A proper height should satisfy following conditions:

1) In the leaf nodes, no points meet the condition to be clustered.
2) At least one parent node of leaves is a clustered node.

Using WGS-84 Mercator Projection, the size of objects appears larger as the latitude increases from the Equator to the poles. Assuming same density of points, in the preprocessing phase, if grids near Equator meet the threshold condition, all other region on the earth will as well.

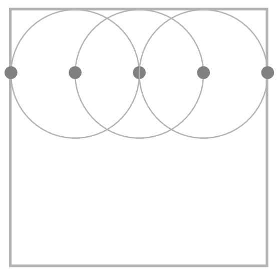

Fig. 2: The best arrangement of 5 points with minimum distance in a square

Near Equator, the distance of 1 degree of longitude is 111.320 km, and for latitude is 110.574 km. Let the maximum number of points in the leaf node be n. Assume the minimum distance between points is dis_min km. The best arrangement of n points with minimum distance in a square area is for them to be aligned in a vertical or horizontal line, as in Fig 2. The width of such grid is $(n-1) * dis\_min$. Converting to degree of latitude and longitude, the width is $(n-1) * dis\_min/110$ degree. Let the height of the tree be h. h is the smallest interger that satifies the following equation:

$$2^{h-1} * (n-1) * dis\_min/110 > 360$$

Take air quality stations as example. Minimum distance between two air quality stations is normally 2 km. Let each grid contain a maximum of 20 stations. Thus a grid occupies a $2 * 39/110 = 0.71$ degree of latitude and longitude. By the previous formula, the minimum height is approximately 11.

B. Node Operations

1) Initialization of Quadtree: First of all, quadtree should be intialized with the dataset. The initialization is a bottom-up process. All points are added directly to the corresponding leaf nodes without grouping. Grid clustering then takes place. Hierarchical clustering is then applied on the representatives of the grids. Nodes in higher levels are generated by nodes in the adjacent lower level.

Assume the depth of leaf node is 0 and the root node is $height\_of\_tree - 1$. The initialization algorithm is listed below.

Algorithm 1 Initialization of Quadtree

```
1: for all points do
2:     Add point directly to the leaves of quadtree without grouping.
3: end for
4: → Grid-based clustering
5: for all leaf nodes do
```

We use centroid method (using Euclidean distance) to calculate distance between two clusters. The stop condition is solely the minimum distance between two clusters.

```
6:      Calculate centroid of points in each node, then save
        the centroid point and number of containing data point
7:  end for
8:  → Following is hierarchical clustering
9:  for i=1 to height of tree1 do
10:     repeat
11:         Using clusters from level $i$
12:         Merge two closest clusters into one cluster
13:         Update distance matrix to reflect the distance between the new cluster and other clusters
14:     until The distance for each two clusters in level $i$ is greater than $min\_distance$
15:     Add cluster points to the nodes in level $i+1$ using quadtree path
16: end for
```

2) Insertion, Deletion and Update: Insertion, deletion and update share similar operation on the quadtree, as no merge or split is required. For insertion of new points, add the new point to the corresponding leaf node, then add 1 to the node count in each level on the traversal path [6]. This is illustrated in Fig.3

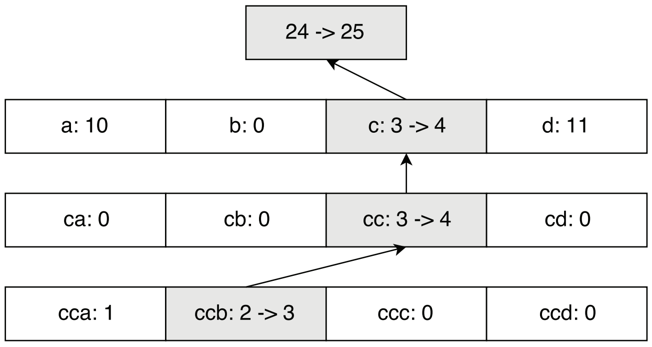

Fig. 3: Insertion of point

The deletion and update are quite similar: update leaf node and the counting of each level all the way up to the root node. To maintain a good cluster, the quadtree should be rebuilt from time to time.

3) Spatial Query: Spatial query is very efficent. Based on the current resolution of the map, only access to corresponding nodes on certain level is needed. To define the current resolution, we use the following equation:

$$ratio = width/longitude$$

$width$ is the visible map width in pixel and $longitude$ is the span of longitude in degree. $ratio$ can be converted to h discret levels. Each level corresponds to one depth of the quadtree. The spatial query algorithm is listed below.

The nodes returned by this function can be easily displayed on map. Since the returned region is larger than the visible area, we don't have to initiate another query if the visible area shifts but still within the returned region.

Algorithm 2 Spatial Query

```
1: $l$ ← corresponding depth of quadtree according to map resolution.
2: $topright$ ← node of the top right corner of the visible map region at level $l$.
3: $bottomleft$ ← node of the bottom left corner of the visible map region at level $l$.
4: $lca$ ← the lowest common ancester of $topright$ and $bottomleft$
       return all the descendants of $lca$ at level $l$.
```

C. Node Definition

To facilitate the aforementioned operations, we can define node in the following way.

```
typedef struct {
        double sum_latitude;
        double sum_longitude;
        double latitude, longitude;
        int count;
        string title;
} Node;
```

In this definition, $sum\_latitude$ and $sum\_longitude$ are the sum of latitude and longitude of all the comprising points. $latitude$ and $longitude$ is the coordinate of the representative and $count$ can be also used for the point title. These data are necessary for calculating the centroid of a cluster.

IV. Experiment Results

Weather data from OpenWeatherMap.org is used as the data source in our test. This dataset contains 209,580 records of temperature samples around the world. It's a good dataset to represent city-based database, such as wind data, air quality data, and population data, etc.

We use R [7] to process the dataset. However, the built-in $hclust$ [8] function cannot satisfy our need so we write a modified version on our own. Performance is compared against tranditional relational SQLite3 database, which contains the same data and latitude and longitude are indexed. Test program for SQLite3 is written in Python.

A. Quadtree Initialization

We divide the Mercator-projected world map into a 72 rows and 144 columns grid. The size of each square grid 2.5 degree.

As we said earlier, the time complexity for hierarchical clustering is O(n^3). The most time consuming operation is computing the distance matrix. On Macbook Air, computing the distance matrix of 2596 points costs about 2 seconds. After each iteration one node is removed. Fig. 4 shows the original dataset plotted on world map. Fig. 5 is the grided data on the original dataset for United States. Fig. 6 is the clustered plot using minimum distance 5, while Fig. 7 is using minimum distance 8.

Test was conducted on a 1.8 GHz Intel Core i5 MacBook Air. Initialization using our proposed method took about 2

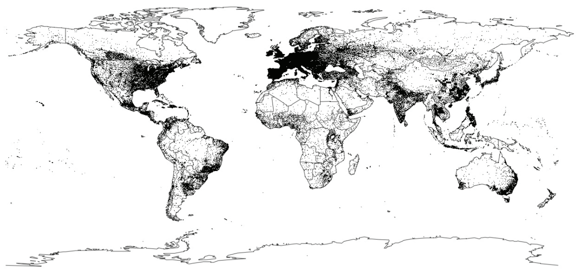
Fig. 4: Raw Point Data

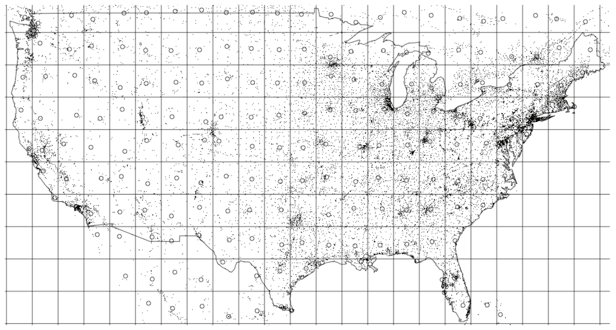
Fig. 5: Hybrid of Original and Grided Data for United States

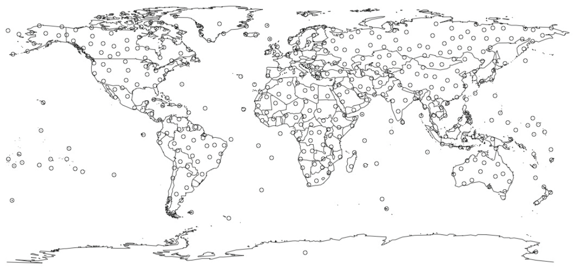
Fig. 6: Cluster distance = 5

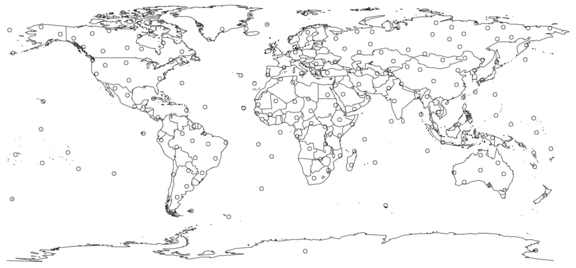
Fig. 7: Cluster distance = 8

minutes, which is long but we think there is space to optimize. In fact we can simply ship the software with initialized data to skip this process. Since there is no clustering for SQLite solution, it took just 13 seconds to insert all the records into the database.

B. Spatial Query

Spatial query is blazing fast using this paradigm, because of the significantly shrinked dataset. Comparison between this method(M1) and relational database(M2) is done on different map resolution and region. The result is as following.

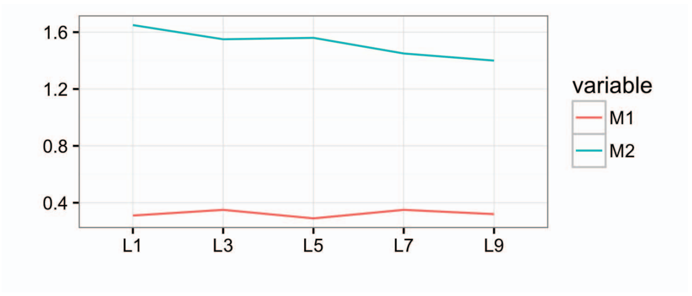

Fig. 8: Comparison of Spatial Query

X-axis is the different granularity level, while Y-axis is the accumulation time of 10 consecutive spatial queries. As we can see, the proposed method has near constant query time.

V. Conclusion

In this paper, we presented a method for interactively visualizing large point dataset using hierarchical clustering and quadtree spatial indexing. Precise data appears only when users zoom in to the interested area. We investigated on different cluster methods and choose agglomerative hierarchical clustering method for its flexibility and hierarchical property. Finally we tested this method on a typical dataset and it works well except for the long initialization. Further research can be done on combining choropleth map and dot map to create more interactive maps [9].

References

[1] S. Burigat and L. Chittaro, Decluttering of Icons Based on Aggregation in Mobile Maps. *Map-based Mobile Services*, Springer Berlin Heidelberg, pages 13-32, 2008.

[2] A. Unwin, M. Theus, and H. Hofmann, *Graphics of Large Datasets: Visualizing a Million*. Springer, 2006.

[3] M. Steinbach, V. Kumar, and P. Tan, *Introduction to Data Mining*. Pearson, 2006.

[4] H. Samet, "Quadtree and Related Hierarchical Data Structures," *ACM Computing Surveys (CSUR) Volume 16 Issue 2*, pages 187-260, 1984

[5] D. Mullner, "Modern hierarchical, agglomerative clustering algorithms," *arXiv preprint arXiv*:1109.2378. 2011 Sep 12.

[6] Sarah F. Frisken and Ronald N. Perry, "Simple and efficient traversal methods for quadtrees and octrees," *Journal of Graphics Tools*, pages 1-11, 2002.

[7] N. Venables, N. William, and David M. Smith, *An introduction to R*. Network Theory Ltd., 2009.

[8] D. Mullner, "fastcluster: Fast hierarchical, agglomerative clustering routines for R and Python," *Journal of Statistical Software*, pages 1-18, 2013.

[9] Andrienko, L. Gennady, and V. Natalia, "Interactive maps for visual data exploration," *International Journal of Geographical Information Science*, pages 355-374, 1999.