Error indexing of vertices in spatial database -- A case study of OpenStreetMap data

Xinlin Qian, Kunwang Tao and Liang Wang
Chinese Academy of Surveying and Mapping
28 Lianhuachixi Road, Haidian district, Beijing China 100830
{qianxl,taokw,wangl}@casm.ac.cn

Abstract

To make online data retrieve with no geographic extent limit possible for geospatial database such as OSM database, a framework for dynamic simplification of data with error-bounded or size-bounded for specific operation is presented. A PostgreSQL extension which implements the framework using PL/PGSQL is explained. With the extension, an OSM database actually builds an index of vertices' error for data simplification. Using this error index, error-bounded or size-bounded data query can be performed efficiently. Experiments against real-world OSM dataset show the effectiveness and efficiency of the framework and its implementation.

Keywords: spatial database, data simplification, error bounded query, size bounded query, OpenStreetMap data, spatial data structure

I. Introduction

Real-time visualization of data from large scale database is inevitable for interactive visualization, data analysis and etc. Query results are not always ready for analysis or visualization because of its unpredictable size. In other words, query results which have a voluminous size and are not simplified are prohibitive from analysis or visualization. To resolve this problem, a framework for building a database which supports error-bounded or size-bounded data query is presented in this paper. The framework is motivated by the requirement of online data retrieve and visualization from voluminous spatial database such as OpenStreetMap(OSM) database. As a case study, we have implemented the framework on PostgreSQL-based OSM database to support real-time linestring simplifications which make OSM data can be retrieved and transferred to client for rendering regardless the geographical extent of the request.

The framework comprises five components whose acronym is DOPAS(data, operation, pick, assessment, structure), namely data to be simplified, operations on data, algorithm for data element picking, function for error assessment and data structure for intermediate results storage.

For the OSM case, the framework is implemented by integrating several existing techniques. With the extension, an OSM database actually builds an index of vertices' error for data simplification. Using this error index, error-bounded or size-bounded data query can be performed efficiently. For example, Linestring simplification is converted to a query with specified constraints on result size or error. For example, SQL statements "select * from way_nodes where id =123 order by error limit 200" can select 200 vertices from a linestring whose id is 123 to create a simplified version of that linestring. SQL statement "select * from way_nodes where error > 1 group by way_id order by sequence_id" can select vertices whose error great than a specified value from linestrings to create simplified version of those linestrings. In the above example, "error" is a column built and maintained by the extension. In this paper, the framework is explained and

the OSM database with real-time simplification functionality is outlined and some experiments are discussed.

II. Related work

Visualization and analysis of voluminous data set, aka big data, is a hot topic in computer science and related fields. The latest research include Scalar[Battle et al. 2013], MapD[Mostak 2013], BigVIS[Wickham 2013], Spatial Sampling[Das Sarma et al. 2012], PiSQL[Armbrust et al. 2011], iMMens[Liu et al. 2013], Nanocubes[Lins et al. 2013], BlinkDB[Agarwal et al. 2013]. The general approach from these researches is binning, aggregating, sampling and filtering the data before it can be visualized or analyzed. Depending on whether it stores and utilizes the pre-computations of the data, these researches fall into two catalogs. one is for Scalar, MapD, BigVis, Spatial Sampling and PiSQL which generate the simplified data on-the-fly each time a data query is issued. They don't take advantages of pre-processing results. While ImMens, Nanocube and BlinkDB do much work in preprocessing and storing intermediate results in auxiliary data structure. This process resembles the stage of map tile generating in geospatial data visualization such as Google maps. Like map tiles, the preprocessed data become "data tiles' which are used in subsequent analyses. The research of this paper falls into the latter catalog.

Compared to the existing researches, the contributions of this paper are: first, a general framework and its implementation for error-bounded or size-bounded data simplification WRT any specific operation are presented. Second, experiments on OSM database show effectiveness and efficiency of the framework for an update-frequent, OLTP styled spatial database.

III. Framework

The framework for a database supporting error-bounded or size-bounded data simplification comprises five components:

1. Data.

Data has a specific type and associates with specific operations. For a query result of the data, the simplification means creating a subset of the results to replace the original set to make any operations on it more efficient.

2. Operation

To speed up operations on data is the aim of simplification. To assess the quality of data simplification, there are two criteria. One is how much the simplification speed up data operation. The other is how much the difference between the operation outcomes from simplified data and original data. Because usually time cost of data operation positively correlates with the size of the data set, the former criterion can be roughly quantified with the size of the simplified data. The latter is quantified by error assessment function which explained below.

It is worth noting that data simplification is operation dependent which means different operations need different data simplifications, and no one data simplification can server all operations well. In this paper, data simplification actually means data simplification for specific operation.

3. Pick algorithm

Usually data simplification process begins with a fully resolution data set and ends up to a simplified one by iteratively eliminating some of the data elements, but this paper do data simplification in an opposite direction.

Starting from a minimized data set, data elements are iteratively added to the dataset until a fully resolution version achieved. During this process, a pick algorithm determines which elements to be added to the dataset and errors for specific operations are computed at each turn. Any of the snapshots in this whole process can be seen as a data simplification result.

4. Assessment function for error

In the running of pick algorithm, the error for specific operation is computed using assessment function at each turn. The error is stored in data structure and will be used in error-bounded or size-bounded data query.

In the framework, error assessment function required to be monotonic which means in the running of pick algorithm error will keep decreasing. This requirement is consistent with human intuition that more data will create more accurate results. Monotonic error make error-bound or size-bound query can be processed by setting a error threshold for data element.

5. Data structure for intermediate results.

The intermediate results from the running of pick algorithm and error assessment function are stored in some data structure to make the pre-computation can be used in future query processing.

To answer the error-bounded query, data element whose error is great than the value specified in query condition are selected.

To answer the size-bounded query, the top m data elements from a list of data elements which sorted by its error are selected.

IV. Framework implementation

For OSM database, the five components of the framework are implemented as follows:

1. Data that need to be simplified is set of linestrings. OSM data comprise nodes, ways and relations. Linestring include ways, relations whose members are ways and tags are ‘route’, ‘rail’, etc.

2. Operation is the rendering of the linestrings which means a rendering engine convert the coordinate data of a linestring into an imagery whose single element is pixel.

3. Pick algorithm is the Douglas-Peucker algorithm which recursively select the nodes which are furthest from the baseline.

4. Error assessment function measures the deviation between the rendered image of original linestring and simplified linestring in pixels.

5. Data structure to store the intermediate results and errors is DBLG-tree[Qian 2011], which is a variant of Cartesian tree.

The framework is used to compute error for every node in every linestring. The whole detailed process is: first running the Douglas-Peucker algorithm for every linestring, then build the DBLG-tree to store the data produced during the running of Douglas-Peucker algorithm, then in every DBLG-tree node associate with error, at last, DBLG -trees are stored alongside the OSM dataset.

OSM database are built on PostgreSQL database management system. We develop an extension for PostgreSQL to make it support error-bounded and size-

bounded data query. PL/PGSQL is used to program the Douglas-Peucker algorithm. Nested set method is used to store DBLG-tree into relational tables. Compared to adjacency list, nested set is more convenient in process tree traversal though more space is needed.

V. Experiments

An OSM database containing all features in United Kingdom is established. The numbers of nodes, ways and relations are 50 million, 12 million and 16 thousands. The database extension that support error-bounded or size-bounded data query is installed. It takes several hours on a commodity computer to do data pre-computation, namely the DBLG-trees building and storing. Once the DBLG-trees structures finished it building, the database is capable of answering error-bounded or size-bounded data query. The effectiveness and efficiency of the method are proved in the experiment.

VI. Conclusion

To make online data retrieve with no geographic extent limit possible for geospatial database such as OSM database, a framework for dynamic simplification of data with error-bounded or size-bounded for specific operation is presented. A PostgreSQL extension which implements the framework using PL/PGSQL is explained. Experiments against real-world OSM dataset show its effectiveness and efficiency. Implementing the framework on other large dataset with the requirement of online interactive visualization and responsive analysis, such as online visualization of point cloud or large-scale network analysis, will be the future work.

References

Agarwal, Sameer, et al. "BlinkDB: queries with bounded errors and bounded response times on very large data." Proceedings of the 8th ACM European Conference on Computer Systems. ACM, 2013.

Armbrust, Michael, et al. "PIQL: Success-tolerant query processing in the cloud." Proceedings of the VLDB Endowment 5.3 (2011): 181-192.

Battle, Leilani, Michael Stonebraker, and Remco Chang. "Dynamic reduction of query result sets for interactive visualizaton." Big Data, 2013 IEEE International Conference on. IEEE, 2013.

Das Sarma, Anish, et al. "Efficient spatial sampling of large geographical tables." Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. ACM, 2012.

Lins, Lauro, James T. Klosowski, and Carlos Scheidegger. "Nanocubes for Real-Time Exploration of Spatiotemporal Datasets." Visualization and Computer Graphics, IEEE Transactions on 19.12 (2013): 2456-2465.

Liu, Zhicheng, Biye Jiang, and Jeffrey Heer. "imMens: Real- time Visual Querying of Big Data." Computer Graphics Forum. Vol. 32. No. 3pt4. Blackwell Publishing Ltd, 2013.

Mostak,Todd. “An Overview of MapD (Massively Parallel Database).” Tech. Report of MIT. 2013.

Qian, Xinlin. “Research on the Representation and Management of Geospatial Data from Volunteered Geographic Information.” PhD thesis(in Chinese), Wuhan University, 2011.

Wickham, Hadley. Bin-summarise-smooth: a framework for visualising large data. Tech. rep., had. co. nz, 2013.