A Distributed Polygon Retrieval Algorithm using MapReduce

Q. Guo , B. Palanisamy, H. A. Karimi*

Geoinformatics Laboratory, School of Information Sciences, University of Pittsburgh
- (qiulei, bpalan, hkarimi)@pitt.edu

KEY WORDS: Hadoop, Polygon Retrieval, Distributed Algorithm, GIS

ABSTRACT:

The burst of large-scale spatial terrain data due to the proliferation of data acquisition devices like 3D laser scanners poses challenges to spatial data analysis and computation. Among many spatial analyses and computations, polygon retrieval is a fundamental operation which is often performed under real-time constraints. However, existing sequential algorithms fail to meet this demand for larger sizes of terrain data. Motivated by the MapReduce programming model, a well-adopted large-scale parallel data processing technique, we present a MapReduce-based polygon retrieval algorithm designed with the objective of reducing the IO and CPU loads of spatial data processing. By indexing the data based on a quad-tree approach, a significant amount of unneeded data is filtered in the filtering stage and it reduces the IO overhead. The indexed data also facilitates querying the relationship between the terrain data and query area in shorter time. The results of the experiments performed in our Hadoop cluster demonstrate that our algorithm performs significantly better than the existing distributed algorithms.

1. INTRODUCTION

Cloud computing is continually being improved for computational geometry, such as the operations commonly used in GIS. Of particular interest, and high demand, is the spatial analysis and computation that typically involves processing large volumes of spatial data. Some example applications include urban environment visualization, shadow analysis, visibility computation, and flood simulation. For these GIS applications, the polygon retrieval is a common operation where very large terrain data within a given polygon's boundary for further analysis is retrieved (Mark de Berg, 2008; Willard, 1982). Willard (Willard, 1982) proposed the polygon retrieval problem and devised an algorithm with O($N^{log_2^2}$) time complexity in the worst-case. To speed up this time complexity, several efficient algorithms have been proposed; (Mark de Berg, 2008; Paterson and Frances Yao, 1986; Sioutas et al., 2008; Tung and King, 2000) are among the most notable algorithms. However, with advanced large-scale spatial data acquisition techniques and devices like 3D laser and satellite, terrain datasets in tens or even hundreds of gigabytes are currently available. Efficient processing of such large terrain datasets is beyond the capability of current algorithms that run on single machines and therefore a distributed solution is highly desired.

Efficiently computing polygon retrieval is very crucial since it is a CPU-intensive operation, especially for very large spatial datasets. In this paper, we present a distributed polygon retrieval algorithm based on MapReduce. The challenges for processing polygon retrieval in a large terrain dataset include how to organize, partition and distribute very large spatial datasets across 10s or 100s of nodes in a cloud datacenter so that the applications can query and analyze the data very quickly and cost-effectively. To address these challenges, we first index the data based on a quad-tree, which is simpler compared with the R-tree index(Eldawy and Mokbel, 2013). This allows to efficiently filter the spatial data that are not relevant for the query, thereby improving the query performance and efficiency. We conduct two experiments on our cluster consisting of 20 nodes to validate the efficiency of our algorithm and the results show that our algorithm is efficient and reduces the job execution time significantly.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 describes the idea of our MapReduce-based polygon retrieval algorithm. The experimental results are showed in Section 4. The conclusion of our work is discussed in Section 5.

2. RELATED WORKS

Polygon retrieval is a common operation needed in a diverse number of GIS applications. Willard (Willard, 1982) was the first one who defined the polygon retrieval problem formally and proposed a polygon retrieval algorithm with $O(N^{log_2^2})$ time complexity. To speed up this performance, efficient algorithms have been proposed (Mark de Berg, 2008; Paterson and Frances Yao, 1986; Sioutas et al., 2008; Tung and King, 2000). These sequential algorithms work well under certain conditions, however, as the terrain datasets are increasingly becoming very large, these algorithms fail to meet the demand for real-time response. As cloud computing has emerged to be an effective and promising solution for both compute- and data-intensive geo-computation, the work in (Karimi et al., 2011) explored the feasibility of using Google App Engine, the cloud computing technology by Google, to process terrain data, usually in triangulated irregular network (TIN) form.

Considering Hadoop has become the defacto standard for distributed computation on a large scale, some recent works have developed several MapReduce-based algorithms for geo-

* Corresponding author

computation. Puri et al. (Puri et al., 2013) proposed and implemented a MapReduce algorithm for distributed polygon overlay computation in Hadoop. Ji et al. (Ji et al., 2012) presented MapReduce-based approaches that construct inverted grid index and process kNN query over large spatial datasets. Akdogan et al. (Akdogan et al., 2010) created a unique spatial index, Voronoi diagram, for given points in 2D space which enabled efficient processing of a wide range of geospatial queries such as RNN, MaxRNN and kNN, with the MapReduce programming model. Hadoop-GIS (Wang et al., 2011) and Spatial-Hadoop (Eldawy et al., 2013) are two scalable and high-performance spatial data processing systems for running large-scale spatial queries in Hadoop. These systems provide support for some fundamental spatial queries like minimal bounding box query, but they do not directly support polygon retrieval operation addressed in this work.

3. MAPREDUCE-BASED POLYGON RETRIEVAL ALGORITHM

In this section, we discuss our proposed MapReduce-based distributed polygon retrieval algorithm. Our algorithm is composed of two parts: (1) using a quad-tree to index the terrain data and (2) organizing the terrain datasets based on the quad-tree prefix to minimize the IO load.

To accelerate the processing of terrain data, we first divide the entire space based on a complete quad-tree. Compared with other spatial indexing techniques, quad-tree has several advantages for polygon retrieval. One such advantage is that we can directly partition the space into four sub spaces recursively. In addition, with the quad-tree indexing, the topological relation among the terrain data and the query area can be inferred from the indices' prefix directly. The key idea here is that if a grid cell is within a query area, then all its sub grids are also guaranteed to be within the query area. In other words, if the prefix of one spatial object's quad-tree index exists in the intersecting set, then that object is guaranteed to be within the query area. This property helps avoid the time-consuming point-in-polygon computation in the map phase enabling the MapReduce jobs to complete significantly faster.

To further increase query efficiency, we use a prefix tree to organize the prefix of all the grid entries that interact with the query area so that the query time is reduced to $O(k)$, where k is the length of the index prefix. A prefix tree, also called radix tree or trie, is an ordered tree data structure that is used to store a dynamic set or associative array where the keys are usually strings(Wikipedia). The idea behind a prefix tree is that all strings that share a common prefix inherit a common node. Thus, with our prefix tree optimization, testing a prefix of a quad-tree index in a given dataset can be accomplished in just $O(k)$ time.

For implementation, in the pre-processing stage, we first consider the coarse-grained grid cells and recursively test whether they overlap with the query area. Once a grid cell intersects the query area, we test the corresponding sub-grid cells unless we are at the deepest level of the quad-tree. If the grid cell is within the query area, we stop subdividing the grid cell and insert its index into the prefix tree. If the grid cell is outside the query area, we just ignore it. From the perspective of prefix tree, if the prefix of a quad-tree index (but not whole index) ends in a leaf node, it means that the corresponding spatial elements are within the query area.

After the prefix tree is created in the pre-processing stage, it is effectively used in the map function. When each mapper receives a spatial element record, the relation between the spatial record and the query area is inferred based on the prefix tree created in the pre-processing phase.

Finally, our quad-tree prefix-based spatial file filtering strategy tries to read in only the necessary spatial data rather than scanning the whole dataset stored in HDFS. Similar to the idea of using the prefix tree to organize the quad tree indices, we separate the spatial data files into fairly smaller files such that each file shares the same prefix. After we organize the terrain file in this manner, we use it in the file filtering stage which scans only the required records to filter those files that are outside the query area which results in the minimum amount of spatial data needed to be processed.

4. EXPERIMENT

In this section, we present the experimental evaluation of our distributed polygon retrieval algorithm. We first introduce the dataset and the computing environment used in the experiment. We then evaluate and compare the proposed approach with existing solutions.

4.1 Dataset and Experiment Environment

There are several data structures to represent the terrain surfaces, two common examples are digital elevation model (DEM) and TIN. The latter (TIN), which is based on vector model, is widely used in many applications. It consists of irregularly distributed nodes and lines arranged in a network of non-overlapping triangles. In our experiment we used TIN datasets. TIN requires considerably a large storage capacity as it can be used to represent surfaces with much higher resolution and detail.

For our experiments, we used the TIN data of Pittsburgh, which is originally divided into 5*5 equally sized grid cells and each grid cell represents a terrain of 10000 metes * 10000 meters. There are 3 million points and 6 million triangles in each grid cell and the size of each grid's TIN file is approximately 500 MB. We conducted our experiments on a cluster of 20 virtual machines created by OpenStack hosted on a 5-node experimental cluster. Each server in the cluster has an Intel Xeon 2.2GHz 4 Core with 16 GB RAM and 1 TB hard drive at 7200 rpm. Each virtual machine in our setup has 1 VCPU with 2 GB RAM and 20 GB hard drive with Ubuntu Server 12.04 (32 bit).

4.2 Algorithm Efficiency

To demonstrate the time performance of the polygon retrieval algorithm in relation to the query area size, we generated a polygon area for each query randomly. We compared our results with the Spatial-Hadoop(Eldawy et al., 2013) as the benchmark. Since Spatial-Hadoop does not provide support for polygon retrieval in the TIN data format directly, we have modified their interfaces and executed the polygon retrieval operation as suggested in the Spatial Hadoop tutorial(SpatialHadoop). Table 1 shows the relationship between the time performance of the algorithm and the polygon query area on our cluster. From the table, it can

be seen that as the query area becomes larger, the time performance generally increases. This is due to the increased amount of TIN data that needs to be processed in the map and reduce phases, but the trend is not based on an strict increasing function since the query shape is irregular, and the spatial data are processed by the predefined unit of grid cell. From the result, we also infer that our algorithm on an average runs 25% faster than the existing technique. This is partly due to the fact that our algorithm significantly avoids the geometry floating point computation in the map phase, especially when the query area is not very large. Therefore, when the query area becomes larger, the I/O time dominates the CPU time and hence the CPU time savings become less significant.

Query Area(m^2)	Time(ms) – Proposed Algorithm	Time(ms) - Spatial-Hadoop (Benchmark)
6.78e+5	14659	40996
3.45e+6	34127	44302
5.26e+6	37608	50487
9.88e+6	37995	51276
1.19e+7	38217	50569
2.16e+7	39773	53906
2.48e+7	37469	54612

Table 1. The query time vs. query area

4.3 Scalability

We next evaluate the effectiveness of our polygon retrieval algorithm by varying the size of the Hadoop cluster in terms of the number of VMs such as 5, 10, 20. For this experiment, we used the random query shapes generated previously and ran queries on different cluster sizes. The result is in Table 2. From Table 2 we can find that overall our proposed technique scales well and showed a significant reduction in job execution time as the number of nodes in the Hadoop cluster increase.

Query Area(m^2)	Time(ms) – VM Size 5	Time(ms) – VM Size 10	Time(ms) – VM Size 20
6.78e+5	19956	18552	14659
3.45e+6	39776	37893	34127
5.26e+6	44526	39248	37608
9.88e+6	43099	40543	37995
1.19e+7	44447	41854	38217
2.16e+7	59872	43893	39773
2.48e+7	58205	42098	37469

Table 2. The query time under different query area and cluster size

5. CONCLUSION

In this paper we presented a distributed polygon retrieval algorithm based on MapReduce. We apply two optimization strategies to reduce the CPU and IO loads of polygon retrieval by using a quad-tree to index the terrain data and organizing the terrain data into small files based on the quad-tree prefix. The experiment results show that our approach achieves high efficiency and outperforms existing solutions.

REFERENCES

Akdogan, A., Demiryurek, U., Banaei-Kashani, F., Shahabi, C., 2010. Voronoi-based geospatial query processing with mapreduce, Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. IEEE, pp. 9-16.

Eldawy, A., Li, Y., Mokbel, M.F., Janardan, R., 2013. CG_Hadoop: computational geometry in MapReduce, Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, pp. 284-293.

Eldawy, A., Mokbel, M.F., 2013. A demonstration of SpatialHadoop: an efficient mapreduce framework for spatial data. Proceedings of the VLDB Endowment 6, 1230-1233.

Ji, C., Dong, T., Li, Y., Shen, Y., Li, K., Qiu, W., Qu, W., Guo, M., 2012. Inverted grid-based knn query processing with mapreduce, ChinaGrid Annual Conference (ChinaGrid), 2012 Seventh. IEEE, pp. 25-32.

Karimi, H.A., Roongpiboonsopit, D., Wang, H., 2011. Exploring Real-Time Geoprocessing in Cloud Computing: Navigation Services Case Study. Transactions in GIS 15, 613-633.

Mark de Berg, O.C., Marc van Kreveld, Mark Overmars, 2008. Simplex Range Searching, Computational Geometry, 3 ed. Springer Berlin Heidelberg, pp. 335-353.

Paterson, M.S., Frances Yao, F., 1986. Point retrieval for polygons. Journal of Algorithms 7, 441-447.

Puri, S., Agarwal, D., He, X., Prasad, S.K., 2013. MapReduce algorithms for GIS polygonal overlay processing, Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International. IEEE, pp. 1009-1016.

Sioutas, S., Sofotassios, D., Tsichlas, K., Sotiropoulos, D., Vlamos, P., 2008. Canonical polygon queries on the plane: A new approach. arXiv preprint arXiv:0805.2681.

SpatialHadoop, SpatialHadoop, Extensible operations.

Tung, L.H., King, I., 2000. A two-stage framework for polygon retrieval. Multimedia Tools and Applications 11, 235-255.

Wang, F., Lee, R., Liu, Q., Aji, A., Zhang, X., Saltz, J., 2011. Hadoop-gis: A high performance query system for analytical medical imaging with mapreduce. Technical report, Emory University.

Wikipedia, Trie.

Willard, D.E., 1982. Polygon retrieval. SIAM Journal on Computing 11, 149-165.