A Distributed Polygon Retrieval Algorithm using MapReduce

Qiulei Guo Balaji Palanisamy Hassan A. Karimi

Geoinformatics Laboratory, School of Information Sciences, University of Pittsburgh
{qiulei, bpalan, hkarimi }@pitt.edu

***Abstract*—The proliferation of data acquisition devices like 3D laser scanners had led to the burst of large-scale spatial terrain data which imposes many challenges to spatial data analysis and computation. With the advent of several emerging collaborative cloud technologies, a natural and cost-effective approach to managing such large-scale data is to store and share such datasets in a publicly hosted cloud service and process the data within the cloud itself using modern distributed computing paradigms such as MapReduce. For several key spatial data analysis and computation problems, polygon retrieval is a fundamental operation which is often computed under real-time constraints. However, existing sequential algorithms fail to meet this demand effectively given that terrain data in recent years have witnessed an unprecedented growth in both volume and rate. In this work, we develop a MapReduce-based parallel polygon retrieval algorithm which aims at minimizing the IO and CPU loads of the map and reduce tasks during spatial data processing. The results of the preliminary experiments on a Hadoop cluster demonstrate that the proposed techniques are scalable and lead to more than 35% reduction in execution time of the polygon retrieval operation over existing distributed algorithms.**

I. Introduction

The proliferation of cost-effective data acquisition devices like 3D laser scanners has enabled the acquisition of massive amounts of terrain data at an ever-growing volume and rate. With the advent of several emerging collaborative cloud technologies, a natural and cost-effective approach to managing such large-scale data is to store and share such datasets in a publicly hosted cloud service and process the data within the cloud itself using modern distributed computing paradigms such as MapReduce. Examples of applications that process such terrain data include urban environment visualization, shadow analysis, visibility computation, and flood simulation. Many geo-spatial queries on such large datasets are intrinsically complex to solve and are often computed under real-time constraints, thus requiring fast response times for the queries. However, most existing sequential algorithms fail to meet this demand effectively given that terrain data in the recent years have witnessed an unprecedented growth in both volume and rate. Therefore, a common approach to speed up spatial query processing is parallelizing the individual operations on a cluster of commodity servers.

Polygon retrieval is a fundamental geospatial operation which is often computed under real-time constraints. Polygon retrieval involves retrieval of all terrain data within an area of interest [8][9] for further analysis. With the increasing proliferation of terrain data, real-time processing of such a large amount of data is not possible through sequential computations and a distributed parallel computation is needed to meet the fast response time requirements.

We argue that such large scale spatial datasets can effectively leverage the MapReduce programming model[1] to compute spatial operations in parallel. In this paper, we develop a MapReduce-based parallel algorithm for distributed processing of polygon retrieval operation in Hadoop [2]. Our proposed algorithm first hierarchically indexes the spatial terrain data using a quad-tree index, with the help of which, a significant amount of data is filtered out in the pre-processing stage based on the query object. Our preliminary experiment results on a 20 node Hadoop cluster show that the proposed algorithm is scalable and performs faster than existing distributed algorithms.

II. TIN Data representation and Polygon retrieval oepration

In this section, we provide the required background and preliminaries about the spatial data representation format namely TIN and discuss MapReduce based parallel processing for large-scale datasets.

A. TIN Data

TIN[3] is a commonly used model for representing spatial data and it consists of irregularly distributed nodes and lines arranged in a network of non-overlapping triangles. Traditionally, TINs are stored as a file, in ASCII or the ESRI TIN dataset file format. To improve the efficiency of processing large TIN datasets, [5] has proposed new TIN data structures and operations for spatial databases that allow storing, querying and reconstructing TINs more efficiently. However, we note that there are no standards on the data structures and operations for TIN [4]; Oracle has defined a proprietary data type and operations for managing large TINs in their own spatial database [6]. In our work, we adopt the data format from [4] which comprises of two types of data entities: TIN_Points and TIN Triangles, as shown in Figure 1. Both types have their unique IDs. The TIN_Points type has five properties and the TIN_Triangles entity has three properties. For the TIN_Point, the Adj_TriangleID[] array stores the IDs of its adjacent triangles. For the TIN_Triangle, the Point_ID array and Coordinate array contain the IDs and coordinates for the three vertices of each triangle.

B. Polygon Retrieval

In this subsection, we describe the polygon retrieval problem using data represented in TIN. Given the boundary of a simple polygon, the polygon retrieval operation retrieves all the terrain data, represented by TIN that intersects with the polygon. As there could be many possible situations of intersection, in this work, we consider an intersection when

COLLABORATECOM 2014, October 22-25, Miami, United States

DOI 10.4108/icst.collaboratecom.2014.257705

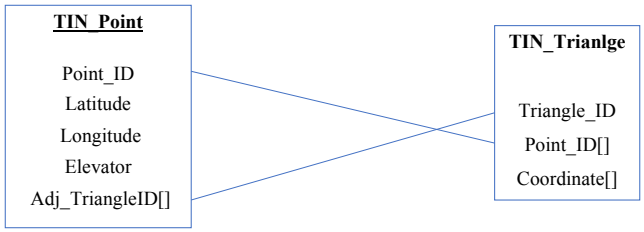

Fig. 1: TIN representation

at least one of its vertex of the TIN triangles intersects with the query area. We note that point-in-polygon algorithms can be used to determine whether a point is inside or outside the polygon. One such well-known algorithm is ray tracing algorithm which is usually referred to as crossing number algorithm or even-odd rule algorithm in the literature.

III. MapReduce-based Parallel Polygon Retrieval

In this work, we focus on MapReduce-based parallel processing of TIN for the polygon retrieval operation. We note that in addition to the programming model, MapReduce [1] also includes the system support for processing the MapReduce jobs in parallel in a large scale cluster. Apache Hadoop[2] is a popular open source implementation of the MapReduce framework. Hadoop is composed of two major parts: storage model, Hadoop Distributed File System (HDFS) and compute model (MapReduce). A key feature of the MapReduce framework is that it can distribute a large job into several independent map and reduce tasks over several nodes of a large data center and process them in parallel. MapReduce can effectively leverage data locality and processing on or near the storage nodes and result in faster execution of the jobs. The framework consists of one master node and a set of slave nodes. In the map phase, the master node schedules and distributes the individual map tasks to the worker nodes. A map task executing in a worker node processes the smaller chunk of the file stored in HDFS and passes the intermediate results to the appropriate reduce tasks executing in a set of worker nodes. The reduce tasks collect the intermediate results from the map tasks and combine/reduce them to form the final output.

We first develop a naive implementation of parallel polygon retrieval operation using MapReduce and then develop a series of optimization techniques that significantly improves this basic polygon retrieval algorithm. Our proposed algorithm employs a sequence of optimization techniques that aims at reducing the computation overhead of the map and reduce tasks. First, our proposed technique divides the whole dataset stored in HDFS into several chunks of files based on a quad-tree prefix. Then for each range query, we use a prefix tree to organize the set of quad-indices whose corresponding grids intersect the query area. Prior to processing a query, we employ these indices to filter the unnecessary TIN data as part of the data filtering stage so that unwanted data processing is minimized in the map phase. Finally, the proposed approach pre-tests the relationship between the TIN data and the query shape through the built prefix tree in the map function in order to minimize the computation.

IV. Experimental Evaluation

For our experimental evaluation, we use the LIDAR data of Pittsburgh city and convert it into TIN format with the help of the LASTool[22]. The data of Pittsburgh 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 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 had 1 VCPU with 2 GB RAM and 20 GB hard drive with Ubuntu Server 12.04(32 bit).

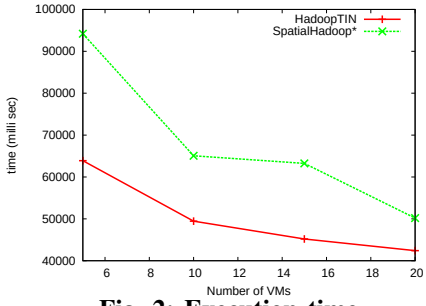

Fig. 2: Execution time

We evaluate the effectiveness of our polygon retrieval algorithm by varying the size of the Hadoop cluster in terms of the number of VMs. For comparing our results with existing distributed polygon retrieval techniques, we use SpatialHadoop[7] as the benchmark. Figure 2 shows the time cost on various cluster sizes when the query area is $2.7e + 7m^2$. We infer that the execution time decreases gradually as the cluster size becomes larger. Overall, the proposed technique scales well with the number of nodes in the Hadoop cluster showing a significant reduction in job execution time with increase in cluster size.

References

[1] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI*, 2004.
[2] Hadoop. http://hadoop.apache.org.
[3] Peucker, Thomas K., et al. "The triangulated irregular network". *Amer. Soc. Photogrammetry Proc. Digital Terrain Models Symposium. Vol. 516. 1978.*.
[4] Karimi, Hassan Ali, Duangduen Roongpiboonsopit, and Haopeng Wang. "Exploring RealTime Geoprocessing in Cloud Computing: Navigation Services Case Study". *Transactions in GIS 15.5 (2011): 613-633.*
[5] Al-Salami, A. "TIN support in an open source spatial database". *MS Thesis, International Institute for Geo-information Science and Earth Observation (ITC), Enschede, The Netherlands (2009).*
[6] Kothuri, Ravi, Albert Godfrind, and Euro Beinat. "Pro oracle spatial for oracle database 11g". *Berkeley: Apress, 2007.*
[7] SpatialHadoop: *http://spatialhadoop.cs.umn.edu/operations.html*
[8] Willard, D.E. "Polygon retrieval". *SIAM Journal on Computing 11.1 (1982): 149-165.*
[9] Mark de Berg, O.C., Marc van Kreveld, Mark Overmars. "Simplex Range Searching". *Computational Geometry*,2008