EFFICIENT LIDAR POINT CLOUD DATA MANAGING AND PROCESSING IN A HADOOP-BASED DISTRIBUTED FRAMEWORK

C. Wang[a], F. Hu[b], D. Sha[b], X. Han[a]*

[a] Hainan Geomatics Center, National Administration of Surveying, Mapping and Geoinformation of China, HaiKou, HaiNan, 570203, China – cx8989@163.com, 110260634@qq.com

[b] Department of Geography and GeoInformation Science and Center for Intelligent Spatial Computing, George Mason University, Fairfax, VA, 22030-4444, USA – fhu@gmu.edu, dsha@gmu.edu

KEY WORDS: Lidar Data, Point Cloud, Hadoop, PCL, HDFS, MapReduce, GIS, Distributed Computing

ABSTRACT:

Light Detection and Ranging (LiDAR) is one of the most promising technologies in surveying and mapping，city management, forestry, object recognition, computer vision engineer and others. However, it is challenging to efficiently storage, query and analyze the high-resolution 3D LiDAR data due to its volume and complexity. In order to improve the productivity of Lidar data processing, this study proposes a Hadoop-based framework to efficiently manage and process LiDAR data in a distributed and parallel manner, which takes advantage of Hadoop's storage and computing ability. At the same time, the Point Cloud Library (PCL), an open-source project for 2D/3D image and point cloud processing, is integrated with HDFS and MapReduce to conduct the Lidar data analysis algorithms provided by PCL in a parallel fashion. The experiment results show that the proposed framework can efficiently manage and process big LiDAR data.

1. INTRODUCTION

Developed in the 1960s and incorporated on airborne platforms in the1980s, light detection and ranging (LiDAR) technologies such as airborne, mobile and terrestrial laser scanning have been able to efficiency obtain high-resolution surface data and cover data (e.g. canopy, buildings) over large spatial extents. LiDAR data have been applied on and are playing an increasingly important role in many fields, such as mapping, environments, natural disaster (Yonglin et al., 2010) and engineering applications (Youn et al., 2014, Svanberg et al., 2015).

LiDAR data are acquired in the form of three-dimensional point cloud, and usually contain tens or hundreds of points per square meter (Guan et al., 2013). As a result, even for processing the Lidar data in a small area will still be quite computing- and data-intensive. However, the current LiDAR processing applications, such as LASTools, Terrosolid and Point Cloud Library(PCL), are based on a single computer, and at most, make use of the multi-core CPU. In order to process big Lidar data in parallel, this paper proposes a Hadoop-based framework to efficiently manage and process data in a distributed and parallel manner, which takes advantage of Hadoop's storage and computing ability. At the same time, the Point Cloud Library (PCL), an open-source project for 2D/3D image and point cloud processing, is integrated with HDFS and MapReduce to conduct the Lidar data analysis algorithms provided by PCL in a parallel fashion.

2. RELATED WORKS

A bunch of LiDAR data processing tools, such as such as FugroViewer (http://www.fugroviewer.com/), ArcGIS LiDAR Analyst (Kersting and Kersting 2005), ENVI LiDAR (Lausten 2007), GRASS GIS (Neteler et al. 2012), Point Cloud Library (Rusu and Cousins 2011) and FUSION/LDV (McGaughey 2009), have been developed to support Lidar data indexing, conversion, segmentation, object identification. However, most of these applications only work on a standalone machine. In order to process large volume of Lidar data, high performance computing and cloud computing have begun to be applied on big LiDAR data processing. For example, Hanusniak, V., et al (2015) have developed a Hadoop framework for point cloud data storage system. Li, Z., et al. (2017) developed a general-purpose scalable framework coupled with a sophisticated data decomposition and parallelization strategy to efficiently handle "big" LiDAR data by coupling existing LiDAR processing tools LASTools with Hadoop. The geoprocessing applications can also be expedited by processing LiDAR data processing in a highly scalable distributed computing environment. For example, Jian, X., et al. (2015) developed a Hadoop-based algorithm of generating DEM grid from point cloud data. Liu, K. used cloud computing for change detection of mobile Lidar data (Liu, Boehm et al. 2016). Rizki, P. N. M., et al. (2017) investigated the use of in-memory processing with Spark for creating a digital elevation model from massive light detection and ranging (LiDAR) point clouds, which can be considered an iterative process. In this paper, we integrate PCL, a sophisticated C++ library that provide functions for point could data indexing, filtering, registration, segmentation and object identification, with Hadoop distributed file system and Map-reduce process to take advantage of parallel storage and computing.

3. METHODOLOGY

The proposed framework is composed of two parts: data management and data processing as shown in Figure 1. In the data management part, the Lidar data can be directly fetched from other data platforms, and then stored in HDFS in the original data formats. In the data processing part, the input point cloud data will be assigned with a reasonable number of Map tasks considering data locality and workload balancing. The statistical outlier filter functions are embedded into MapReduce to directly process the data in each Map task. In the Reduce part, the outputs from each Map task are collected and analyzed to get the final results.

*Corresponding author

3.1 Data Management

To help users to transfer such big data into HDFS, a data fetching module is developed. The data published in other data platforms can be directly downloaded into HDFS with customized configuration parameters, such as destination path, block size, and replication factor.

The traditional point cloud data processing algorithms focus on the single file level. However, in Hadoop computing architecture, the point cloud file may be larger than default block size. For this situation, the large file might be split into multiple blocks and stored in different data nodes. It would lead to two problems:1) the partitions of the original files cannot be recognized by the Lidar libraries without the splitting metadata; 2) regrouping of the data requires excessive disk and network usage, which will impair the efficiency. Algorithms for reading and regrouping point cloud binary data are also needed, which will add the complexity of the system development and finally affect the efficiency. To solve these problems, we set the block size parameter in Hadoop as big as the images when fetching data, which will keep each file from being divided.

3.2 Data Processing

3.2.1 Data Partition Period: To achieve parallel computation of the input data, MapReduce divides the entire data set into multiple logical partitions, and then allocates them to the corresponding nodes to read and process the data in parallel. How these input data are divided and allocated directly affects the localization of data, which creates a significant difference in system performance. Considering the data location and workload balance, we design a data split strategy to split the queried Lidar data by customizing the FileInputFormat class. In the FileInputFormat class, each Lidar file will be treated as a block and then identify the data nodes where this block is physically stored at. Since each file have several copies in the cluster, the queried Lidar data will be equally split into several groups according to the number of involved data nodes. Therefore, each compute/data node is assigned with a similar volume of data to maintain the workload balancing between the compute nodes.

3.2.2 Map Period: After each compute node receives the allocated splits, it will start a Map task for each split. In the Map task, we will first get the split information, such as the input file path, the point cloud processing operations and the result file path. And then call the corresponding function provided by PCL to remove the outlier noise point for each file in each data node. After statistical filter processing, the resulting point cloud data are stored in local machines and then uploaded into HDFS according to the file path predefined by the user. In addition, the status report for each point cloud processing task is recorded and transferred to the next reduce period.

3.2.3 Reduce Period: The Reduce task will collect all the status reports from the Map period, then analyze which point cloud filtering processing tasks succeed, and which fail. For the failed tasks, the system will re-launch a new MapReduce job.

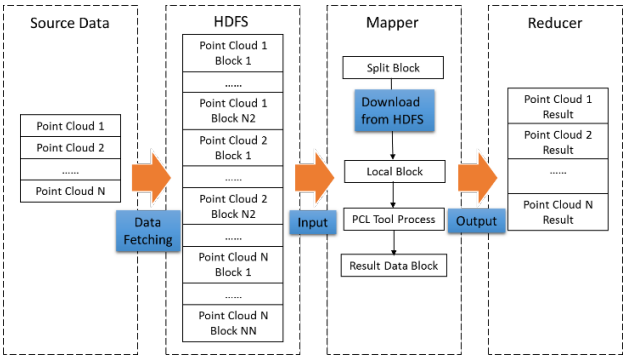

Figure 1. The architecture of the proposed framework

4. EXPERIMENT

4.1 Experiment data

In this study, the LiDAR data for the City of Baltimore acquired on April 15, 2008 is used as the experimental data (approximately 90 square miles). The spatial resolution for LiDAR data is 1 meter and the total data size in laz format is about 800 MB. After changing the datasets to be the PCD format, the data size increases to 6.24GB.

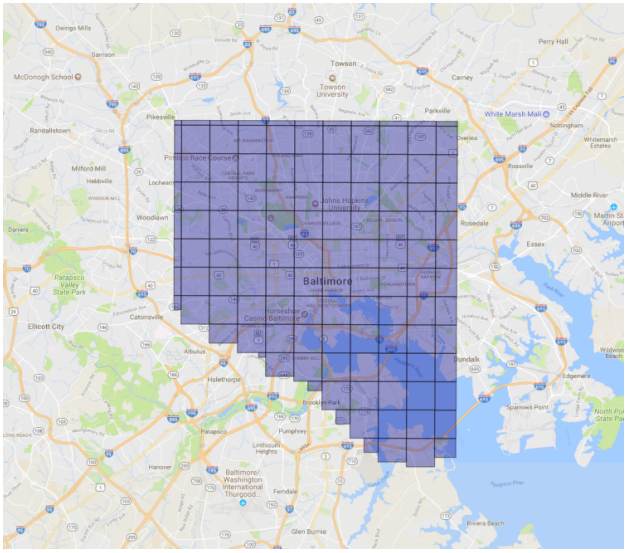

Figure 2. Study Field in Baltimore

These data were acquired by Sanborn Aero Commander 500B with flight height 1400 meter above ground level. Measure unit is US survey feet for both the horizontal and vertical axises, with 0.185-meter vertical accuracy and 1.0-meter horizontal accuracy. By using pcd format conversion tool, all the laz format data are converted to the pcd data format (Figure 3 and Figure 4 show the visualization of data sample in laz format and pcd format). This conversion process contains two steps: first, the las2txt tool of Lastools is utilized for coordinate transformation as the format converted; second, automatically update header files information by a customized script.

PCL provides many point cloud processing tools such as filtering, registration, segmentation, recognition and visualization. In this experiment, we utilize the PCL library to remove outliers using a StatisticalOutlierRemoval filter.

Figure 3. Original Laz. Data and Visualization in ArcScene

Figure 4. Converted pcd Format, Visualization in PCL Viewer

Table 4: Data Groups for Test

Group	File Number	Number of Point Cloud	File Size
G1	1	336, 758	10 MB
G2	1	3, 382, 294	100 MB
G3	10	33, 822, 940	1 GB
G4	100	338, 229, 400	10 GB
G5	200	676, 458, 800	20 GB
G6	1000	3, 382, 294, 000	100 GB

4.2 Cluster Environment

A cluster with three high performance PCs has been setup for the experiments. The cluster is equipped with Hadoop 2.7.0 and consisted of a NameNode and three DataNodes. PCL 1.7.0 is installed on each DataNode. Both NameNode and DataNode use 8 CPU-cores (3.60GHz), 16 GB of RAM and 256 GB of SSD storage. The NameNode and all DataNodes are connected by 1 gigabit internet. Ubuntu 14.04, Hadoop 2.7 and Java 1.8.are installed on both NameNode and DataNodes. Table 2 show the PC and cluster hardware configuration, and Tables 3 shows the cluster software configurations.

Table 2. Cluster hardware configuration

Data Node	3	
Network	-	1 Gigabytes Switch
CPU	1	8 CPU-cores (3.60GHz)
RAM	24	GB
Storage	256	SSD disk

Table 3. Cluster Software Configuration

Name	Version	Details
Hadoop	2.7.0	Installed on each node
Ubuntu	14.04	
Java SDK	1.8.0	
PCL	1.7.0	Provides point cloud processing tools

4.4 Experiment Results

To evaluate the efficiency of the proposed framework, we design two groups of experiments: 1) the first group is to run the referred cloud point processing algorithm on a single node; 2) the second group is to execute the referred algorithm on the Hadoop cluster. In each group, 6 different data file sizes are used for the performance testing. In order to reduce variability and measurement errors, we performed three operations for each group and took the average time values. The average run-time for the two groups are shown in Figure 5.

The figure shows that when the size of dataset is 10 MB and 100MB, the run-time for the single node is less than that for the Hadoop cluster. However, when the dataset's size is larger than 100 MB, the run-time for the cluster is obviously less than that for the single machine. This result is caused by the Hadoop framework's overhead. It requires to run several processes first, such as launching Hadoop client and scheduling map tasks to prepare for the Map and Reduce period, so the whole running procedure would cost more time than single machine if the input data size is small. However, the proposed framework has much better performance when processing larger volume of data as shown in Figure 5. The results illustrate that the proposed Hadoop-based system can efficiently process the point cloud data by using the point cloud library.

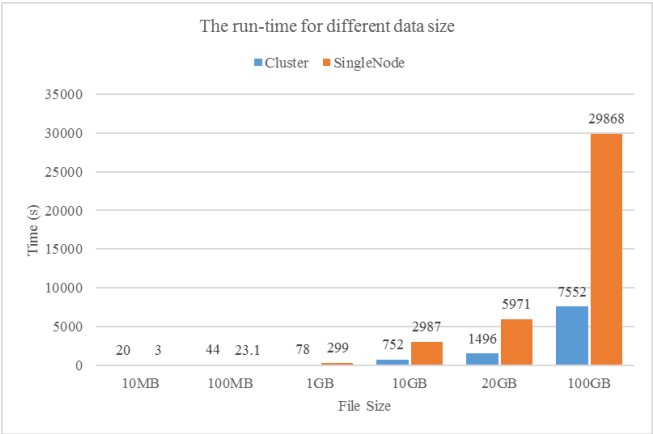

Figure 5. Average run-time for two groups

5. CONCLUSION

In this paper, a Hadoop-based distributed framework is proposed to efficiently manage and process big point cloud data. By integrating point cloud library processing tools into MapReduce, this framework provides various parallel point cloud processing operations. The experiment result shows that

the proposed framework can reduce the run time when dealing with big data volume.

For the future work, the proposed framework can be improved from the following two aspects:

1) the data pipeline in the proposed framework can be improved. As shown in Figure 1, each file needs to be downloaded into local file system for the data processing algorithm. If we could develop a HDFS middleware to enable the Lidar libraries to directly read data from HDFS, the runtime and disk consumption can be improved a lot. Taken the above experiment as an example, the disk size can be saved by 176.72 GB for each data note.
2) More data processing algorithms will be exemplified to improve the design of the proposed framework, such as point cloud data split and result merging strategy to improve the parallelism of data processing.

ACKNOWLEDGEMENTS

The authors are grateful to their colleagues for their constructive comments and suggestions in writing this article.

REFERENCES

Guan, H., Li, J., Zhong, L., Yongtao, Y., & Chapman, M. (2013). Process virtualization of largescale lidar data in a cloud computing environment.Computers & Geosciences, 60, 109-116.

Yonglin, S., Lixin, W., & Zhi, W. (2010). Identification of inclined buildings from aerial LIDAR data for disaster management. In 2010 18th International Conference on Geoinformatics, 1-5, IEEE.

Youn, C., Nandigam, V., Phan, M., Tarboton, D., Wilkins-Diehr, N., Baru, C., ... & Wang, S. (2014). Leveraging XSEDE HPC resources to address computational challenges with high-resolution topography data. InProceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment, 59, CM

Hanusniak, V., et al. (2015). Exploitation of Hadoop framework for point cloud geographic data storage system. Digital Information Processing and Communications (ICDIPC), 2015 Fifth International Conference on, IEEE.

Jian, X., et al. (2015). "A Hadoop-Based Algorithm of Generating DEM Grid from Point Cloud Data." The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 40(7): 1209.

Li, Z., et al. (2017). "A general-purpose framework for parallel processing of large-scale LiDAR data." International Journal of Digital Earth: 1-22.

Liu, K., et al. (2016). Change detection of mobile Lidar data using cloud computing. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives.

Rizki, P. N. M., et al. (2017). "Spark-based in-memory DEM creation from 3D LiDAR point clouds." Remote Sensing Letters 8(4): 360-369.

Rusu, R. B. and S. Cousins (2011). 3d is here: Point cloud library (pcl). Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE.