MASSIVE POINT CLOUD PROCESSING ON HADOOP: CHALLENGES AND PROPOSED SOLUTION

Minh Hieu Nguyen (1), Sanghyun Yoon (1), Sangyoon Park (1), and Joon Heo* (1)

[1] Yonsei University, 50 Yonsei-ro, Seodamun-gu, Seoul, 03722, Korea;
Email: {nguyenminhhieu, yoonssa, parksangyoon}@yonsei.ac.kr, jheo@yonsei.ac.kr

KEYWORDS: Massive point cloud processing; Hadoop-based framework; Map-Reduce application;

ABSTRACT: Because of the remarkable technological development of sensors and algorithms for point cloud data acquisition and registration, collecting point cloud data over large spaces has become more accessible than ever before**.** The result of this process is the generation of massive point cloud data, which can exceed the capacity of a single computer. Efficient visualization of this data is an important issue that needs to be addressed. Using big data platforms, such as Hadoop, could bring benefits in processing big point cloud data. However, due to some barriers, there have not been many studies conducted on this platform to solve problems of big point cloud data, so far. In this study, the potential and challenges of processing big point cloud data using Hadoop will be presented. Thereafter, a comprehensive solution will be proposed to overcome the limitations, which can result in the first Hadoop-based framework for fully processing massive point cloud data.

1. INTRODUCTION

The development of science and technology has provided powerful devices for collecting point cloud data. For example, with a Leica HDS 6100 system, point cloud data can be collected up to 500000+ points per second (Leica HDS 6100 brochure). This data collection results in the generation of big point cloud data up GB or even TB in size, posing great challenges for a single computer. First, in a commodity computer, the storage capacity is only 1~2 TB, therefore, it is hard to keep several big point cloud data, as mentioned. At the same time, the process of backing up these massive data also has required a lot of effort. Second, in many cases, processing these big data is impossible due to memory overflow issues. In conventional solutions, data need to be split manually into sub-datasets that can be processed within the capacity of a single computer. At that time, it does not only require human resource to handle the sub-datasets, but also in the effort to summarize the results or implement post-processing step to achieve the desired result. Therefore, automatic parallel processing has been required. So far, two kinds of parallel processing have been used, including taking the advantages of either supercomputers with multi-cores or commodity computer clusters. While the first solution saves development time, it has also required a higher cost of investment making use of the second solution more common. As one of the pioneering studies in this area, (Han et al. 2009) used a personal computer cluster and a virtual grid for the efficient processing of an enormous amount of ALS (Airborne Laser Scanning) data. In this study, Message Passing Interface was used for data transferring between worker nodes. However, this solution has no ability to fault tolerance, nor is it a data locality solution. In other words, data need to be moved frequently between the worker nodes during the processing. In a conventional approach, the study by (van Oosterom et al. 2015) used a database cluster to optimize the storage as well as the fast archiving results in Range Query and kNN operations. In this solution, instead of storing point cloud in flat table, a big point cloud was split into blocks in binary format before storing in the tables. The queries are then accelerated by indexing strategies such as using Morton code (Martinez-Rubi et al. 2015). Three Database Management Systems including PostgreSQL (https://www.postgresql.org), Oracle (https://www.oracle.com), and MonetDB (https://www.monetdb.org) have been tested in this study, and the results showed that Oracle proved to be a very effective environment both w.r.t data loading and querying. The advantage of this solution is stable, time-saving in the development. However, it still remains expensive, lacking the ability to fault tolerance and the expansion of computation up to model-level such as RANSAC (Li et al. 2017).

In the era of big data, the advent of big data platforms such as Apache Hadoop (https://hadoop.apache.org) or Apache Spark (https://spark.apache.org) have prompted a series of studies to address the problems of processing massive point cloud data. In this series, to overcome the limitation of storage the study by (Růžička et al. 2017) researchers used Hadoop as the optimal storage component, which is demonstrated in the scalability when more computers are added to the cluster, and the fault tolerance when one of the computers fails. However, these studies only exploited the storage capability of Hadoop without taking the power of parallel computing of this platform. Before, point cloud data could be processed by using tools or libraries such as LASTool (https://rapidlasso.com/lastools) or PCL (http://pointclouds.org). They have been combined with Hadoop in the studies by (Li, Hodgson, and Li 2018) and (Wang et al. 2017) respectively. However, these frameworks simply call functions of the tool or library as mentioned above in each worker node instead of using native Map-Reduce diagram of Hadoop. In other words, the process of configuration, computation distribution, and result collections are not seamless. Recently, the study by (Kissling et al. 2017) has proposed eEcoLIDAR as an ambitious proposal for handling massive Lidar data by using Hadoop. However, the main result of this project has not been published. In fact, the development of Map-Reduce applications for the purpose of processing spatial data, in general, and point cloud data, in particular, has required a lot of effort from developers, because Hadoop itself does not support spatial data. Therefore, a large number of studies has taken advantages of a third platform running on top of Hadoop such as HBase (https://HBase.apache.org) to save the cost of development (Boehm and Liu 2015; Vo et al. 2018). From another view, when the power of processing data in memory is considered, (Pajić, Govedarica, and Amović 2018) introduced a point cloud data management model based on Apache Spark. In this solution, HBase is used as storage component, while the applications are developed based on Resilient Distributed Dataset mechanism of Spark. The performance of this solution has been compared with PostgreSQL which has shown outstanding advantages in obtaining data. However, the testing operations such as Range Query or kNN are still limited and need to be expanded for showing the potential use of this model in processing massive point cloud data. At the same time, HBase is also not an optimal storage solution because it often happens bottleneck issue when a large amount of data is written down HBase simultaneously (Azqueta-Alzúaz et al. 2017). Moreover, as most of the previous frameworks, they completely lack the ability to visualize massive point cloud data, which is one of crucial factors to obtain useful information from point cloud data. In this study, a comprehensive solution is introduced to overcome the existing limitations. Thereby, it creates foundations to develop a complete framework for processing massive point cloud data.

2. CURRENT CHALLENGES AND ANALYSIS

2.1. Challenges in the storage

If Hadoop allows the storage can be extended along with ability to fault tolerance through the block replication and distributed storage mechanisms, it also bring difficulties to users by these mechanisms. First, a big file will be split into blocks physically in Hadoop; therefore, only the first block keeps header information of this file, if this is a structured data file. It means that many parts of this file cannot be read locally at Map or Reduce tasks because of missing the header information. Meanwhile, point cloud can be stored in many kinds of structured data files, such as LAS (Isenburg 2013) or E57 (http://www.libe57.org). So far, these big data files need to be split into smaller files within one block (128MB) before being imported into Hadoop (Li, Hodgson, and Li 2018). In this way, developers need to use a customized *FileInputFormat* of Hadoop for reading the data. In another way, data can be stored in a flat file such as *CSV or *.PTS. These file formats take more disk space for storage; however, they can be read by Hadoop without any deep customization. In fact, almost all input file formats have required creating index structures, and this process is more complicated than creating an index on a single computer, because instead of reading data directly from the file, Hadoop needs to refer to the master node to get the address of blocks of data file. Therefore, creating index on Hadoop has required both on the master node and on the worker nodes.

2.2. Challenges in the visualization

The major challenge in visualizing massive point cloud on Hadoop is storing and indexing the multi-resolution point cloud data in the HDFS (Hadoop Distributed File System). Hadoop is designed to process big data sets instead of processing many small files. Meanwhile, a big point cloud data can generate a multi-resolution point cloud data up to millions of data units. The issues of storing small files in Hadoop could be expressed as follows. First, a "block" is the minimum data unit of Hadoop and every data block (including a number of replication for fault tolerance) requires its presence at the master node. Therefore, just one file, only 1KB in size, also requires 3 instances (number of replication) of its presence at the master nodes. Meanwhile, a bulky map of blocks at the master node could significantly reduce the performance of accessing data. Second, the process of creating and closing the file reader in HDFS takes a lot of time. Due to these reasons, previous studies had to move the multi-resolution data outside of the distributed storage model to reduce the time of accessing data (Eldawy, Mokbel, and Jonathan 2016; Yu, Zhang, and Sarwat 2018). To overcome these limitations, the number of presences of blocks at master nodes has to be minimum. The techniques of prefetching, indexing, and caching data have to be fully employed.

2.3. Challenges in data processing

Processing point cloud data often requires data to be processed at the model-level, in which a sequence of single operations is used consecutively to obtain the desired results. In these models, it is essential to keep the intermediate results in memory to reduce the time of reading and writing data on disk. However, keeping data in memory also requires organized tuning of parameters to distribute computer resources to the worker nodes. Specifically, in Hadoop, creating more JVMs (Java Virtual Machines) means that more blocks can be processed at the same time. However, these JVMs will also receive fewer resources, which may result in the incomplete work, because the amount of data generated exceeds the capacity provided. Besides, it is also important to keep the workload balanced between these JVMs, since the final result only be collected after all worker nodes complete their tasks.

3. PROPOSED SOLUTION

3.1. Indexing Methods

Indexing methods are developed for the acceleration of processing data. They are categorized into three groups, including spatial partitioning, global indexing, and local indexing. These methods are summarized in Figure 1.

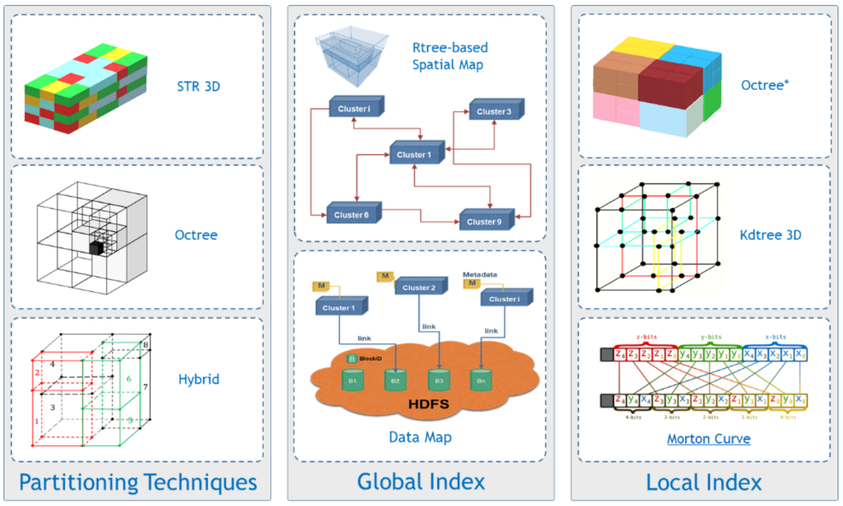

Figure 1. Proposed Model for Indexing Point Cloud Data on Hadoop

Spatial partitioning: This is the very first step in a Map-Reduce application, however, it has played an important role in balancing the workload between the Reduce tasks. Spatial partitioning is a rule or set of rules defined by developers to ensure that objects will be grouped in the same category if they are close to each other and the number of objects between parts is approximately equal. In processing big spatial data, spatial partitioning is usually conducted on sample data at the master node (Eldawy, Alarabi, and Mokbel 2015). In this article, the STR-based method (Leutenegger, Lopez, and Edgington 1997) is proposed for general spatial partitioning, since this method ensures the balancing number of points between parts, and it creates a number of parts which is close to the number of physical blocks. Octree-based method is proposed for spatial partitioning of creating multi-resolution point cloud because this structure is appropriate for the visualization (Scheiblauer 2014; Schütz 2016). In some cases, using only one structure for the spatial partitioning is not enough because it does not ensure minimum number of parts generated, thus, a hybrid structure should be developed (Yang and Huang 2014).

Global indexing: When a file is processed, Hadoop reads all blocks of this file as default. However, Hadoop reads only the blocks that are directly involved in the computation, if a global index is created. Global index manages blocks of data generated by the spatial partitioning step. It works based on two components, including spatial map and data map. Spatial map supports locating exactly which regions of data will be processed, while data map supports reading data within these regions. Spatial partitioning can be implemented in different methods, therefore, global index needs to be adaptive to the geometric structures generated by the spatial partitioning step. Among many kinds of tree-based structures, Rtree is proposed for the global index, since this structure can handle overlap issues and discard empty space (Balasubramanian and Sugumaran 2012).

Local indexing: This step organizes data in each data block of Hadoop. Local index can be Grid File, Rtree, Kdtree, Octree or others. In some cases, the global and local index are merged in a single index (Whitman et al. 2014) or either of them is not created (Yu, Wu, and Sarwat 2015). Among the tree-based structures for indexing point cloud data, the study by (Han et al. 2011; Han 2018; Elseberg, Borrmann, and Nüchter 2013) showed the superiority of Octree compared to the others. In addition to using tree-based structure, point cloud data could be indexed by using space-filling curve (Guan, van Oosterom, and Cheng 2018). In this way, multi-dimensional data are transformed into one-dimensional data which can be indexed by using Btree structure. The local index should be kept in memory to speed up the performance (Kyzirakos, Alvanaki, and Kersten 2016). However, it also be written down HDFS to reuse in other processes (Eldawy and Mokbel 2015).

3.2. Visualizing Massive Point Cloud Data

Visualization is an important factor to achieve insight on point cloud data. In the visualization of big point cloud, which exceeds the capacity of GPU (Graphics Processing Unit), the multi-resolution point cloud should be used along with out-of-core methods to obtain information at only core regions. Between WebGL (https://www.khronos.org/webgl) and OpenGL (https://www.opengl.org) for the rendering, WebGL is proposed since Hadoop undertook most heavy tasks. Hence, the web-based application is only used for showing the multi-resolution data. Besides, a web-based solution can share the results among users without moving the data. Based on Potree (Schütz 2016), which is one of the state-of-the-art studies in this research topic, a model of visualizing massive point cloud data on Hadoop is given in Figure 2. The main difference of this model compared to Potree is in storing the multi-resolution data. Because storing and accessing many small files in HDFS are not optimal, it is necessary to combine them into larger files with the ability to random access, like the mechanism of NoSQL, or directly use a NoSQL data warehouse, such as HBase, MongoDB (Abramova and Bernardino 2013) or Cassandra (Dede et al. 2013), for storing the multi-resolution point cloud data.

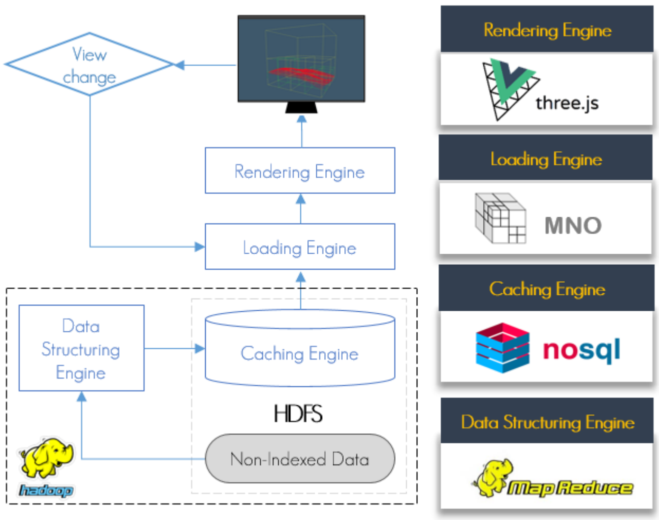

Figure 2. Proposed Model for Visualizing Point Cloud Data on Hadoop

3.3. Processing Point Cloud Data

Based on the analysis in Section 2, the overall architecture of our framework is introduced in Figure 3. This framework consists of three layers: (1) Storage layer, (2) Operation layer, and (3) Interactive layer.

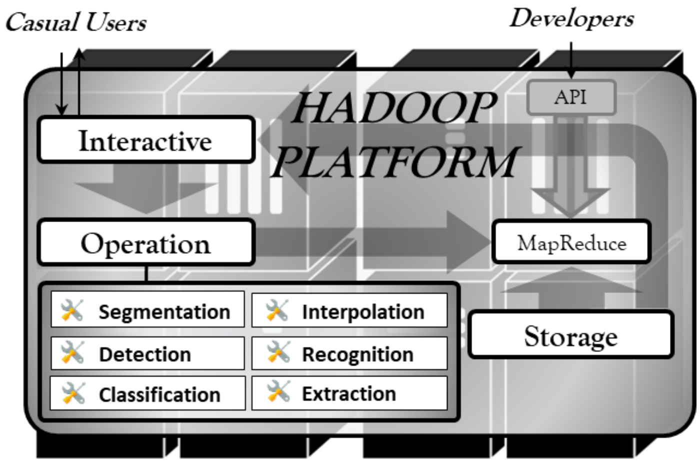

Figure 3. Proposed Architecture for Visualizing Point Cloud Data on Hadoop

The storage layer is optimized by the proposed indexing methods to accelerate the Map-Reduce applications. This layer plays an important role in the operation of the entire system, since data could be accessed randomly instead of sequentially. The operation layer is equipped with Map-Reduce applications to take full advantage of parallel computing in Hadoop. Some of group applications in this proposed framework are also introduced at the operation layer in Figure 3. To enrich the applications in this layer, developers can use an API (Application Programming Interface) to save the development time. The interactive layer, which is also a web-based visualization, will be equipped with high-level user interfaces to support casual users to approach the system without requiring deeply related knowledge of big data technology.

4. CONCLUSION

In this paper, many of the studies related to processing massive point cloud data on Hadoop are reviewed to clarify the potential and challenges in this research topic. Firstly, most studies have not fully taken advantages of Map-Reduce diagram in parallel processing because customizing the native structure of Hadoop for processing spatial data has required expert knowledge. Secondly, no research has been presented to solve thoroughly the problem of visualizing big point cloud data on Hadoop. This can be explained by the obstacles in accessing small files in Hadoop system. Thirdly, the previous studies hardly provided the solutions for extending existing results. Because they only focus on developing end-user solutions, without taking into account the provision of API to support other developers. Based on the analysis, the solutions have been introduced to overcome the existing limitations and challenges, and open clear direction for this research in the future. The efficiency of these solutions will be further analyzed in our next studies.

5. ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT) (No. 2018R1A2B2009160).

6. REFERENCES

Abramova, Veronika, and Jorge Bernardino. 2013. "NoSQL databases: MongoDB vs cassandra." In *Proceedings of the international C* conference on computer science and software engineering*, 14-22. ACM.

Azqueta-Alzúaz, Ainhoa, Marta Patiño-Martinez, Ivan Brondino, and Ricardo Jimenez-Peris. 2017. "Massive data load on distributed database systems over HBase." In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 776-79. IEEE Press.

Balasubramanian, Lakshmi, and M Sugumaran. 2012. 'A state-of-art in R-tree variants for spatial indexing', *International Journal of Computer Applications*, 42: 35-41.

Boehm, J, and K Liu. 2015. 'NoSQL for storage and retrieval of large LiDAR data collections', *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40: 577-82.

Dede, Elif, Bedri Sendir, Pinar Kuzlu, Jessica Hartog, and Madhusudhan Govindaraju. 2013. "An evaluation of cassandra for hadoop." In *2013 IEEE Sixth International Conference on Cloud Computing*, 494-501. IEEE.

Eldawy, Ahmed, Louai Alarabi, and Mohamed F Mokbel. 2015. 'Spatial partitioning techniques in SpatialHadoop', *Proceedings of the VLDB Endowment*, 8: 1602-05.

Eldawy, Ahmed, and Mohamed F Mokbel. 2015. "Spatialhadoop: A mapreduce framework for spatial data." In *2015 IEEE 31st international conference on Data Engineering*, 1352-63. IEEE.

Eldawy, Ahmed, Mohamed F Mokbel, and Christopher Jonathan. 2016. "HadoopViz: A MapReduce framework for extensible visualization of big spatial data." In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, 601-12. IEEE.

Elseberg, Jan, Dorit Borrmann, and Andreas Nüchter. 2013. 'One billion points in the cloud–an octree for efficient processing of 3D laser scans', *ISPRS Journal of Photogrammetry and Remote Sensing*, 76: 76-88.

Guan, Xuefeng, Peter van Oosterom, and Bo Cheng. 2018. 'A parallel N-dimensional Space-Filling Curve library and its application in massive point cloud management', *ISPRS International Journal of Geo-Information*, 7: 327.

Han, Soo-Hee, Seong-Joo Lee, Sang-Pil Kim, Chang-Jae Kim, Joon Heo, and Hee-Bum Lee. 2011. 'A Comparison of 3D R-tree and octree to index large point clouds from a 3D terrestrial laser scanner', *Journal of the Korean Society of Surveying, Geodesy, Photogrammetry and Cartography*, 29: 39-46.

Han, Soo Hee, Joon Heo, Hong Gyoo Sohn, and Kiyun Yu. 2009. 'Parallel processing method for airborne laser scanning data using a PC cluster and a virtual grid', *Sensors*, 9: 2555-73.

Han, Soohee. 2018. 'Towards Efficient Implementation of an Octree for a Large 3D Point Cloud', *Sensors*, 18: 4398.

Isenburg, Martin. 2013. 'LASzip: lossless compression of LiDAR data', *Photogrammetric Engineering and Remote Sensing*, 79: 209-17.

Kissling, W Daniel, Arie C Seijmonsbergen, Ruud PB Foppen, and Willem Bouten. 2017. 'eEcoLiDAR, eScience infrastructure for ecological applications of LiDAR point clouds: reconstructing the 3D ecosystem structure for animals at regional to continental scales'.

Kyzirakos, Kostis, Foteini Alvanaki, and Martin Kersten. 2016. "In memory processing of massive point clouds for multi-core systems." In *Proceedings of the 12th International Workshop on Data Management on New Hardware*, 7. ACM.

Leutenegger, Scott T, Mario A Lopez, and Jeffrey Edgington. 1997. "STR: A simple and efficient algorithm for R-tree packing." In *Proceedings 13th International Conference on Data Engineering*, 497-506. IEEE.

Li, Lin, Fan Yang, Haihong Zhu, Dalin Li, You Li, and Lei Tang. 2017. 'An improved RANSAC for 3D point cloud plane segmentation based on normal distribution transformation cells', *Remote Sensing*, 9: 433.

Li, Zhenlong, Michael E Hodgson, and Wenwen Li. 2018. 'A general-purpose framework for parallel processing of large-scale LiDAR data', *International Journal of Digital Earth*, 11: 26-47.

Martinez-Rubi, Oscar, Peter Van Oosterom, Romulo Gonçalves, Theo Tijssen, Milena Ivanova, Martin L Kersten, and Foteini Alvanaki. 2015. 'Benchmarking and improving point cloud data management in MonetDB', *SIGSPATIAL Special*, 6: 11-18.

Pajić, Vladimir, Miro Govedarica, and Mladen Amović. 2018. 'Model of Point Cloud Data Management System in Big Data Paradigm', *ISPRS International Journal of Geo-Information*, 7: 265.

Růžička, Jan, Lukáš Orčík, Kateřina Růžičková, and Juraj Kisztner. 2017. 'Processing LIDAR Data with Apache Hadoop.' in, *The Rise of Big Spatial Data* (Springer).

Scheiblauer, Claus. 2014. 'Interactions with gigantic point clouds'.

Schütz, Markus. 2016. 'Potree: Rendering large point clouds in web browsers', *Technische Universität Wien, Wiedeń*.

van Oosterom, Peter, Oscar Martinez-Rubi, Milena Ivanova, Mike Horhammer, Daniel Geringer, Siva Ravada, Theo Tijssen, Martin Kodde, and Romulo Gonçalves. 2015. 'Massive point cloud data management: Design, implementation and execution of a point cloud benchmark', *Computers & Graphics*, 49: 92-125.

Vo, Anh-Vu, Nikita Konda, Neel Chauhan, Harith Aljumaily, and Debra F Laefer. 2018. "Lessons learned with laser scanning point cloud management in Hadoop HBase." In *Workshop of the European Group for Intelligent Computing in Engineering*, 231-53. Springer.

Wang, Chunxiao, Fei Hu, Dexuan Sha, and X Han. 2017. 'Efficient LiDAR point cloud data managing and processing in a hadoop-based distributed framework', *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4: 121.

Whitman, Randall T, Michael B Park, Sarah M Ambrose, and Erik G Hoel. 2014. "Spatial indexing and analytics on Hadoop." In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 73-82. ACM.

Yang, Jiansi, and Xianfeng Huang. 2014. 'A hybrid spatial index for massive point cloud data management and visualization', *Transactions in GIS*, 18: 97-108.

Yu, Jia, Jinxuan Wu, and Mohamed Sarwat. 2015. "Geospark: A cluster computing framework for processing large-scale spatial data." In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 70. ACM.

Yu, Jia, Zongsi Zhang, and Mohamed Sarwat. 2018. "Geosparkviz: a scalable geospatial data visualization framework in the apache spark ecosystem." In *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*, 15. ACM.