Analysis of Performance and Optimization of Point Cloud Conversion in Spatial Databases

Aleksandra Chrószcz¹, Piotr Łukasik¹, Michał Lupa¹

¹ AGH University of Science and Technology, Faculty of Geology, Geophysics and Environmental Protection, Department of Geoinformatics and Applied Computer Science, al. Mickiewicza 30, 30-059 Kraków, Poland

E-mail: alekch@student.agh.edu.pl

Abstract. This article compares popular relational database management systems and one non-relational database in the context of storage of cloud points from LIDAR. The authors examine the efficient storage of cloud points in the database and optimization of query execution time. Additionally, there is also a comparison of the impact of SSD and traditional HDD technology on the time taken to perform operations on a point cloud.

1. Introduction

Laser scanning LIDAR (Light Detection and Ranging) is an innovative solution for acquiring digital terrain models (Digital Terrain Model) and digital elevation models (Digital Elevation Model). GIS software provides numerical models of areas such as the pseudoraster model (GRID) in the form of a triangle mesh (Triangular Irregular Network) which can also be acquired by laser scanning. The result of a scan is usually a cloud of points that can be used to create plans, maps, and terrain models after further processing.

2. Technology presentation

2.1. Lidar

LIDAR is a device that can be compared to radar with the difference that it uses light instead of microwaves. Scanning is performed based on the distance from the sensor to the test area as this determines the time delay between sending and receiving the laser pulse. The laser beam is reflected from the surface being tested and the data can be obtained in the form of a point cloud. The laser system is divided into air (on-board) and ground [1].

The data used in the project come from aircraft-based scanning and utilizes a positioning system (GPS), navigation system (INS), distance measuring system (transmitter and receiver), a photogrammetric camera, and a flight planning system. LIDAR is mounted under a plane along with a photogrammetric camera. The number of points obtained per square meter is dependent on flight parameters such as altitude and the number of passes. At low altitudes, 20 points.m⁻² can be achieved. Mesh density is also dependent on weather conditions, terrain obstructions, air movement and pollution [1].

2.2. The use of spatial databases

Point clouds created when scanning terrain with LIDAR technology require storing in appropriate structures. The results of the digitization of cartography are Geographic Information Systems. In order
to ensure the universality of data, two standards have been created: OGC [2] and SQL / MM - Spatial [3]. These afford the following benefits: standardization of data storage formats using WKB (Well-Known Binary); definition of objects using WKT (Well-Known-Text); and functions which represent a spatial extension of the SQL language [3]. The custom approach to storing spatial data that is included in this article is MongoDB. Data is stored as JSON-style documents.

3. Description of the experiments
Databases that have been used for previous academic projects were tested, including four relational databases and one non-relational. A DBMS was also used.

3.1. The use of spatial databases
The following experimental equipment specifications were used:
- Processor Intel Core i7-4790K @ 4.00 GHz, 8 GB RAM, Disc SSD Crucial M550 256 GB,
- Processor Intel Core i7-4790K @ 4.00 GHz, 8 GB RAM, Disc HDD 1TB 5400 RPM .

The management systems relational databases: PostgreSQL 9.5.0, MySQL Community Edition (GPL), Microsoft SQL Server 2016 CTP2.2 32/64-bit, Base X. The non-relational database management system: MongoDB 3.2.

3.2. The research process
a) Binary files with the las extension were converted to extend csv using the BSD licensed LibLAS program (http://www.liblas.org/). Data was obtained from the Central Geodetic and Cartographic Documentation, relating to the area of the city of Warsaw.
b) Tables (MongoDB - collection) designed for pre-converted files were prepared using SQL language (MongoDB - MongoDB Query). Queries are presented in Table 1.

c) Copy the contents of the csv file to created tables or collections.

Table 1. Create tables to store data in the form of a point cloud (using PostgreSQL)

| NAME     | QUERY |
|----------|-------|
| PostgreSQL | CREATE SEQUENCE id_seq;
|           | CREATE TABLE lidar
|           | (id_point INTEGER DEFAULT NEXTVAL('id_seq'),
|           | x double precision,
|           | y double precision,
|           | z double precision, ...); |

Table 2 presents the scan results of the surface copy command generated by LIDAR. Each query has been adapted to the requirements of the database system and its functions. The amount of data that was loaded into each database is shown below.

File size:
- 1.csv: the number of tuples - 6841183, size 324 MB
- 2.csv: the number of tuples - 5029211, size 236 MB
- 3.csv: the number of tuples - 6485280, size 308 MB
- 4.csv: the number of tuples - 5930495, size 280 MB
Total: 24,286,169 rows with a total size of 1,148 MB

---

1 due to the legal provisions concerning commercial solutions, the database name cannot be disclosed
Table 2. The commands for copying the contents of the file to the tables / collection

| NAME       | QUERY                                                                 |
|------------|------------------------------------------------------------------------|
| PostgreSQL | COPY lidar (x, y, z, intensity, returnnumber, numberofreturns, scandirection, edgeofflightline, classification, scananglerank, red,green, blue) FROM 'F:\1.csv' DELIMITERS ' '; |
| MySQL      | LOAD DATA LOCAL INFILE 'F:\1.csv' INTO TABLE lidar FIELDS TERMINATED by ';'; LINES TERMINATED by '\n' (x,y,z,intensity,returnnumber,numberofreturns, scandirection,edgeofflightline,classification,scananglerank,red,green,blue) |
| Microsoft SQL Server | BULK INSERT dbo.lidar FROM 'F:\1.csv' WITH (FIELDTERMINATOR = ',', ROWTERMINATOR = '\n') |
| Base X     | plik loader.ctl: LOAD DATA INFILE 'F:\1.csv' APPED INTO TABLE LIDAR FIELDS TERMINATED BY ';;' (x, y, z, intensity, returnnumber, numberofreturns, scandirection, edgeofflightline, classification, scananglerank, red, green, blue) wiersz polecenia: sqlldr lidar@//localhost:1521/lidar control=loader.ctl |
| MongoDB    | mongoimport.exe --host=127.0.0.1:27017 -d lidar -c lidar --type csv -- file F:\1.csv --headerline |

4. Performance analysis of individual databases using SQL queries/ MongoDB Query

Chapter 4 contains the method and results of the experiments. During the experimental phase, each test was conducted 10 times; the result reflects the average of these trials. Shown below are the times taken to load data into the database (Table 3) and average processing time (Table 4).

Table 3. Average times of loading data into the database [s].

| NAME          | 5029211 rows | 5930495 rows | 6485280 rows | 6841183 rows |
|---------------|--------------|--------------|--------------|--------------|
| PostgreSQL SSD | 20.9         | 23.7         | 26.5         | 30.7         |
| MySQL         | 30.39        | 36.88        | 41.13        | 46.74        |
| MS Server     | 25.28        | 30.03        | 31.67        | 42.47        |
| Base X        | 148.23       | 178.55       | 195.79       | 204.01       |
| MongoDB       | 72.3         | 80.4         | 86.6         | 97.5         |
| PostgreSQL HDD | 64.03        | 80.86        | 84.6         | 89.2         |

Table 4. The average processing time of queries limited by number of N tuples/documents.

| NAME          | N=10 | N = 100 | N= 1000 | N= 10000 | N= 20000 | N= 50000 | N= 100000 | N= 500000 |
|---------------|------|---------|---------|----------|----------|---------|-----------|-----------|
| PostgreSQL SQL | 2.8  | 2.9     | 3.5     | 21.04    | 23.24    | 26.07   | 54.03     |
| MySQL         | 0    | 0       | 0       | 0.016    | 0.031    | 0.094   | 0.171     | 0.86      |
| MS SQL Server | 2.26 | 6.48    | 6.64    | 6.53     | 6.65     | 6.72    | 7.93      |
| Baza X        | 6.38 | 8.63    | 10.55   | 13.39    | 14.95    | 16.17   | 18.86     | 155.07    |
| MongoDB       | 0.15 | 0.45    | 3.26    | 30.25    | 59.15    | 145.9   | 304.2     | 1510.4    |
The best results were achieved by MySQL, but the results are difficult to see on the charts because none of the times exceeds 1 second. The reason for this may be the InnoDB engine used in the latest versions of MySQL (since version 5.5) which incorporates clustering indexes that significantly speed up the database in some cases. The effect of these indexes is that data is also stored within the index [4, 5]. Stable read times are provided by Microsoft SQL Server database, which, despite the increasing volume of returned data, does not slow down as much as the other database servers.

There is a linear increase in read times from the MongoDB database which is associated with the method of writing data to the console.

**Figure 1.** Query processing time [s]
5. Optimizing point cloud management in PostgreSQL
The authors focused on optimizing the PostgreSQL database due to the wide range of tools it offers to support clouds of points. Additionally, the PointCloud extension used is integrated with the popular PostGIS system.

5.1. Optimization process with PointCloud
The PointCloud extension was added for storage optimization and operations on point clouds from LIDAR scanning, thus making it possible to create point clouds of various types that are compressed yet retain all parameters. Point cloud structures are defined in xml. Single points can be optimized by creating data conversion components, thereby reducing the size of the cloud. The PCID identifier defines the type of a created point cloud. The transition to data stored in the form of PcPoint() was performed using an UPDATE of the columns in the table from the loaded LIDAR data. The operation was performed on the SSD and HDD with the same data (Figure 2).

The execution time of an operation on an SSD is seven times faster than a standard HDD; a significant difference if constantly changing sets of data that require frequent updates. The performance of these two types of disk was also tested by copying and creating new tables. In this case, the SSD was about 3 minutes faster than the HDD, which represents a significant advantage.

- SSD: Query returned successfully: 24286169 rows affected, 02:24 minutes execution
- HDD: Query returned successfully: 24286169 rows affected, 14:27 minutes execution

Figure 2. Comparison of conversion points on the HDD and SSD.

5.2. Optimization process by Index B-tree
The database used B-tree index id_point columns in order to accelerate the performance of queries [6, 7]. Indexes were introduced on both the basic version of the table and the optimized version using the Pointcloud extension. Types of optimization: Type 1 - No optimization, Type 2 - Pointcloud extension, Type 3 - Indexing id_point column on a table without the Pointcloud extension, Type 4 - Indexing id_point column on a table with the Pointcloud extension (Fig. 3).

Figure 3. Query execution time after optimization [s]

6. Summary
The research indicates that the most optimal choice is PostgreSQL as it has the best data loading times, is supported by the PostGIS and Pointcloud extensions and is a free system. If only considering read times, MySQL can be recommended as it achieved the best results in this research. MS Server produces stable return times of tuples, but unfortunately is not a free system and its usage in this experiment was based on an academic license. When a greater number of records are returned MS Server has an
advantage over PostgreSQL. Base X cannot match its competitors, as it is limited to only one processor core, up to 1 GB of RAM, and support for up to 4 GB of data. The MongoDB non-relational database demonstrated that it behaves well with a small number of documents, but it becomes inefficient when access times increase linearly. The tests were carried out with returning records to the console, but the results could be different when used by a web application, for example. In some studies, SSD technology obtains results seven times better than a standard HDD. Another significant aspect of SSD technology is its reliability and longer lifespan, however this technology requires more testing and development.

References
[1] Wężyk P. Podręcznik dla uczestników szkoleń z wykorzystania produktów LiDAR. Główny Urząd Geodezji i Kartografii, Warszawa 2015
[2] OpenGIS Implementation Specification for Geographic information - Simple feature access, http://www.opengeospatial.org/standards/sfi
[3] ISO/IEC 13249-3:1999, Information technology – Database languages – SQL 2000.
[4] Helm R., Marriott K., Odersky M.: Constraint-Based Query Optimization for Spatial Databases. Proc. 10th ACM PODS, 1991
[5] Lupa M., Sarlej W., Piórkowski A., Krawczyk A.: Wykorzystanie metod morfologii matematycznej w procesie generalizacji baz danych przestrzennych. Studia Informatica [on-line], Vol. 34 Num. 2B (9 May 2013)
[6] Aref W., Samet H. Optimization for Spatial Query Processing. VLDB ’91 Proceedings of the 17th International Conference on Very Large Data Bases, Pages 81-90
[7] Sevinç E., Coşar A. An Evolutionary Genetic Algorithm for Optimization of Distributed Database Queries. The Computer Journal, 2010