Realistic benchmarks for point cloud data management systems

Peter van Oosterom, Oscar Martinez-Rubi, Theo Tijssen and Romulo Gonçalves

Abstract Lidar, photogrammetry, and various other survey technologies enable the collection of massive point clouds. Faced with hundreds of billions or trillions of points the traditional solutions for handling point clouds usually under-perform even for classical loading and retrieving operations. To obtain insight in the features affecting performance the authors carried out single-user tests with different storage models on various systems, including Oracle Spatial and Graph, PostgreSQL-PostGIS, MonetDB and LAStools (during the second half of 2014). In the summer of 2015, the tests are further extended with the latest developments of the systems, including the new version of Point Data Abstraction Library (PDAL) with efficient compression. Web services based on point cloud data are becoming popular and they have requirements that most of the available point cloud data management systems can not fulfil. This means that specific custom-made solutions are constructed. We identify the requirements of these web services and propose a realistic benchmark extension, including multi-user and level-of-detail queries. This helps in defining the future lines of work for more generic point cloud data management systems, supporting such increasingly demanded web services.

Peter van Oosterom
Section GIS technology, Department OTB, Faculty of Architecture and the Built Environment, TU Delft, The Netherlands, e-mail: P.J.M.vanOosterom@tudelft.nl

Oscar Martinez-Rubi
Netherlands eScience Center, Amsterdam, The Netherlands e-mail: O.Rubi@esciencecenter.nl

Theo Tijssen
Section GIS technology, Department OTB, Faculty of Architecture and the Built Environment, TU Delft, The Netherlands, e-mail: T.P.M.Tijssen@tudelft.nl

Romulo Gonçalves
Netherlands eScience Center, Amsterdam, The Netherlands e-mail: R.Goncalves@esciencecenter.nl

1 Introduction

Traditionally point clouds have been converted to grids, vector objects or other types of data to support further processing in a GIS environment. Today point clouds are also directly used for estimating volumes of complex objects, visibility analysis, roof solar potential analysis, 3D visualizations and other applications. In addition, the usage of point cloud data over the web, usually for visualization purposes, have also increased significantly. For example in archaeology, as shown in figure 1, point clouds are crucial for the 3D documentation and analysis of sites [7][12]. In addition to managing grids, vectors or TINs, it is more and more demanded that data management solutions are able to handle massive point clouds.

Fig. 1 Snapshot of a web application based on point clouds for an archaeology application.

This paper extends the results presented in our previous work [17] where we identified the various systems for managing point cloud data on the market, we defined a conceptual benchmark based on user requirements and we investigated the performance of the various point cloud management systems (PCDMS's) via an executable benchmark. Moreover, several methods for the improvement of the various PCDMS's were suggested and tested [16]. In this paper we present new benchmark results motivated by the latest developments of the Point Data Abstraction Libarary (PDAL). In addition, we study the usage of some web services based on point cloud data which currently rely on specific solutions. Based on this analysis, we identify ways to improve the existing executable benchmark in order to reflect the needs of such services. This can also be used as a guideline for the new developments required by the PCDMS's in order to have more generic solutions which could also be used for these demanding services.

The outline of this paper is as follows: TRehe different PCDMS's are introduced in Section 2. A summary of the executable benchmark defined in our previous work

[17] together with its results (second half 2014) are presented in Section 3, while the results of the new benchmark execution (summer 2015) are detailed in Section 4. In Section 5 we describe two services based on point cloud data, the 'Actueel Hoogtebestand Nederland' (AHN, height model of The Netherlands) 2D viewer [4] and the AHN2 3D web viewer and download tool [5], and we perform an analysis of their usage. Motivated by this study, in Section 6 we propose an extension of the executable benchmark to cover the challenges in supporting visualization tools and multi-user interactions. Finally, the conclusions and future work are described in Section 7. At the end of the document there is an Appendix section which contains extended information on the previous (second half 2014) benchmark execution.

2 Point Cloud Management Systems

The suitability of Database Management Systems (DBMS) for managing point cloud data is a continuous debate. File-based solutions provide efficient access to data in its original format, however, data isolation, data redundancy, and application dependency on such data format are major drawbacks. Furthermore, file-based solutions have also poor vertical and horizontal scalability [13]. We consider and compare both DBMS and file-based solutions for PCDMS's. In the latter points are stored in files in a certain format, and accessed and processed by solution specific software. Within DBMS solutions two storage models can be distinguished:

- Blocks model: nearby points are grouped in blocks which are stored in a database table, one row per block
- Flat table model: points are directly stored in a database table, one row per point, resulting in tables with many rows.

All file-based solutions use a type of blocks model and we have chosen to test LAStools by Rapidlasso [11] as basis for our used file-based PCDMS. The DBMS with native point cloud support based on the blocks model are Oracle Spatial and Graph and PostgreSQL-PostGIS. Even though in principle any DBMS with spatial functionality can be used as a PCDMS with the flat table model, we tested Oracle Spatial and Graph [1], PostgreSQL-PostGIS [14] and the column-store MonetDB [15]. For more information regarding the various PCDMS's and their tuning (block szie, compression), we refer the reader to our previous work [17].

In addition to using native blocks solutions, it is also possible to use third-party blocking solutions. Point Data Abstraction Library (PDAL) [9] is a software for manipulating point cloud data and it is used, in general, as an abstraction layer on management operations. Thus the same operations are available independently on which system (DBMS or file-based) actually contains the data. It has recently released its version 1.0.0. It is compatible with a long list of point cloud file formats including LAS and LAZ and it has a set of commands that can be used to create a stand-alone file-based PCDMS, thus offering an alternative to LAStools.

PDAL has database drivers for Oracle Spatial and Graph, PostgreSQL and SQLite, i.e. tools to load and retrieve data to/from a database using the blocks model. Unfortunately the PDAL point cloud format used for Oracle Spatial and Graph is not fully compatible with the native format provided by the SDO_PC package [1]. On the other hand, PDAL is compatible with the native format provided by the PostgreSQL-PostGIS point cloud extension [19]. SQLite does not have native point cloud support, thus the functionality is limited to the PDAL features.

Based on the LASzip compression [2] the PDAL team has recently included the laz-perf compression [6] which can be used in Oracle Spatial and Graph and SQLite. It achieves a compression rate similar to LAZ.

3 Executable benchmark

The required point cloud functionality and usage profile for governmental institutions, industry and academia was obtained through a set of interviews [20]. The profile showed that spatial selections for data analysis or point cloud data extraction for further analysis by external tools were the ones ranked higher. In addition it was also detected that the data preparation was also a concern or challenge for the users. This information was used to define a conceptual benchmark for PCDMS's. An executable benchmark was derived from it. It has several stages with incremental data sets and tested functionality. The data sets vary from a few million points to several hundred billion points and they were subsets of AHN2 [3], the second National Height Model of the Netherlands, which in its raw format consists of 640 billion points.

The first stage of the executable benchmark is the *mini-benchmark* which goal is to gain experience with the used systems. In this stage each system loads a small data set with 20 million points and performs seven queries (retrieval operations) based on selecting points that are contained in rectangular, circular or polygonal regions.

The second stage of the benchmark is the *medium-benchmark* and it is used to test the scaling performance of the systems as well as more advanced functionality. In this stage each system loads four different data sets, from 20 million, 210 million, 2.2 billion to 23 billion points and executes 20 queries which include, in addition to the *mini-benchmark* queries, elevation queries, such as points with elevation lower than 10 meters in a circular region, queries on large regions or very complex regions and nearest neighbor queries.

The third stage is the *full-benchmark* and its goal is to see the performance of the system in the loading of a massive data set such as the AHN2 and its behavior in the execution of 30 queries which include, in addition to the *medium-benchmark* queries, extra large region queries and aggregated queries like computing the highest point in a province.

The fourth and last stage of the benchmark is the *scale-up-benchmark*, which was only defined and not yet executed, envisioned a test with very massive data sets

with 20 trillion points and even more tested functionality. In appendix A there is more information on the used data sets and the benchmark queries.

Several stages of the benchmark were executed for the various existing PCDMS's using data sets. We tested the blocks model with the native support of Oracle Spatial and Graph and PostgreSQL-PostGIS, in the latter the loading was done using PDAL. We tested flat table models in Oracle Spatial and Graph, PostgreSQL-PostGIS and MonetDB. They were compared with a file-based PCDMS based on LAStools with both LAZ and LAS formats.

All systems run on the same platform, a HP DL380p Gen8 server with 128 GB RAM and 2 x 8 Intel Xeon processors E5-2690 at 2.9 GHz, RHEL 6 as operative system and different disks directly attached including 400 GB SSD, 5 TB SAS 15 K rpm in RAID 5 configuration (internal), and 2 x 41 TB SATA 7200 rpm in RAID 5 configuration (in Yotta disk cabinet).

We also tested an implementation of the flat table model with Oracle Spatial and Graph in an Oracle Exadata X4-2 hardware [8], an Oracle SUN hardware designed for the Oracle database with an advanced architecture including hardware hybrid columnar compression (HCC), massive parallel smart scans/predicate filtering and less data transfer.

3.1 Storage, Preparation and Loading

In appendix B the reader can find the tables with the loading results of the execution of the *medium-benchmark* and the *full-benchmark*. Compared to flat table systems the blocks model DBMSs are faster and compress the data better during preparation and loading. Flat table systems enable modifications of the table definition or the data values as in any database table which is more complicated in the blocks model. For both, the integration with other types of data is straight and all the key features of DBMSs are present, i.e. data interface through the SQL language, remote access and advanced security.

LAStools prepares data faster than any DBMS since no loading is needed, only resorting and indexing. The storage requirements of the LAZ format are lower than those of the DBMSs, but with its fixed file format the data model loses flexibility as one is restricted to the specified format. For example the standard LAS format allows only one byte for user data. In addition, when dealing with large data sets (stored in many LAS/LAZ files), LAStools needs to be aided by a DBMS to maintain its performance. The DBMS is used to store the extent of the files and it is used as a pre-filtering stage in the retrieval operations. Storage requirements and speed of loading in Oracle Exadata is comparable to LAStools. However, the used hardware is different.

3.2 Querying

In appendix C the reader can find the tables with the retrieval/querying results of the execution of the *medium-benchmark* and the *full-benchmark*. As previously stated, data retrieval was tested by selecting points within rectangles, circular areas, simple and complex polygons, and nearest neighbor and aggregated queries. The DBMS results were obtained by CTAS (create table as select) queries, while the LAStools results were obtained by storing the selection in an output LAS file.

Blocks model DBMSs perform well on larger areas or complex polygons, independent of the number of points. However, the blocks model adds an overhead which affects simple queries most. The flat table model DBMSs perform well for simple queries on small point clouds; for large point clouds the native indexing methods become inefficient. Alternative flat table models based on space filling curves provided nearly constant response times, independent of number of points [16]. The file-based solution using LAStools performs best for simple queries. The queries to LAZ data are slower than to LAS data because of the need of uncompressing the data. In addition, massive point clouds require an external DBMS to maintain good performance.

Data retrieval in Oracle Exadata is comparable to LAStools but complex queries run significantly better because of massive parallelization. However and as previously stated, the systems run in different hardware.

4 New benchmark execution

From our previous benchmark execution we could conclude that when a file-based solution fulfills the user requirements it is effective to use that solution. However, if more flexibility and/or more advanced functionality are required DBMSs offer a good alternative. At the time of the previous benchmark execution (second half 2014), the *full-benchmark* stage (loading 640 billion points and executing 30 queries) could only be executed with decent performance in LAStools, Oracle Exadata and PostgreSQL-PostGIS (results of the latter were not presented), thus limiting the choice of PCDMS. Moreover, most systems miss two important features. Firstly, though data preparation and loading can be easily parallelized with additional tools, only MonetDB supports parallel processing. For the DBMS for which we applied out-of-the-core parallel queries for data retrieval the performance improved significantly.

The developer teams of Oracle Spatial and Graph, PDAL and MonetDB have been actively improving their systems and these are reaching a more mature and robust state. In order to assess the recent improvements a new execution of the benchmark is required. In the previous execution of the *full-benchmark* stage we compared PCDMS's running in different systems. In this paper we present the results of a new *full-benchmark* execution of LAStools with two file formats, LAS and LAZ, and Oracle Spatial and Graph interfaced with PDAL and the the new laz-perf

compression, but all running in the same hardware, the HP DL380p Gen8 previously described. Another *full-benchmark* run also considering MonetDB and its the new developments and PostgreSQL-PostGIS with the latests PDAL improvement is planned for the near future.

4.1 Storage, Preparation and Loading

Contrary to the previous benchmark execution we used a cleaned version of the AHN2 data set where duplicate and erroneous points are deleted. The cleaned data set is distributed in 37,588 LAZ files and contains 638 billion points instead of the raw format with 640 billions points.

In the case of LAStools we prepare (resort and index) the data and, as previously explained, we require a DBMS which contains the spatial extent of the files. In the case of Oracle Spatial and Graph we use the external third-party PDAL library to load the data from the LAZ files into Oracle using the blocks model. Note that in both cases the loading is not natively parallel but can be easily done since each file can be loaded independently. In table 1 we present the results of the loading in the tested systems, in all the cases we used 16 simultaneous processes:

system	LAStools/LAS	LAStools/LAZ	Oracle/PDAL
Total load time	22:54 hours	19:41 hours	33:53 hours
total size	12.18 Tb	1.66 Tb	2.07 Tb
#points	638,609,393,087	638,690,670,516	638,860,225,350

Table 1 *Full-benchmark* loading results for LAStools with both LAS and LAZ and Oracle Spatial and Graph with PDAL. In all cases 16 processes were used.

The loading of the data is faster in LAStools. However, the required time for in Oracle has enormously decreased with PDAL. The speed is now 18,854 million points per hour while with the loading method used before it was around 431 million points per second.

In storage terms we can observe the laz-perf compression with PDAL is almost as efficient as the LAZ, thus there is not much penalty anymore in storage terms by loading the data in a DBMS. The estimation of storage terms in Oracle Spatial and Graph without using laz-perf compression is of 20 Tb for the AHN2. Also note that the number of points after loading differs for the various PCDMS's, this is due to several LAStools processes crashing for files large than 60 million points (issue reported to developer).

4.2 Querying

We use lasmerge and lasclip commands to perform the queries in the LAStools PCDMS's and we use PDAL approach for performing the queries in Oracle/PDAL approach. In this new benchmark, the output was store in LAS files (also in case of Oracle/PDAL). Note that we can not use native SQL/SDO_PC Oracle features because of the compatibility issue between the PDAL format and the native Oracle Spatial and Graph format.

A subset of the *full-benchmark* queries was executed with the tested systems, they consist on selecting the points in rectangular, circular or polygonal regions. Both systems currently have limited functionality and the nearest neighbor and aggregated queries could not be executed.

In table 2 we present the result of the executed queries in the tested systems. For each query we detail the number of returned points as well as the response time. The queries are executed in sequential order and each query is executed twice, the value in the table is the response time of the second execution (as this value is usually a bit more stable). Both LAStools and PDAL processes are single-core processes.

Query	LAStools/LAS			LAStools/LAZ			Oracle/PDAL		
	#points	Time[s]	#pts/s	#points	Time[s]	#pts/s	#points	Time[s]	#pts/s
1	74850	0.03	2495000	74850	0.11	680455	74818	0.25	299272
2	717959	0.08	8974488	717959	0.42	1709426	717869	0.97	740071
3	34691	0.02	1734550	34691	0.09	385456	34667	0.23	150726
4	563037	0.09	6255967	563037	0.43	1309388	563013	1.16	485356
5	182861	0.15	1219073	182861	0.36	507947	182861	0.57	320809
6	460096	1.40	328640	460096	1.79	257037	387134	1.26	307249
7	45811	0.24	190879	45811	0.68	67369	45813	1.49	30747
8	2365590	3.72	635911	2365590	5.52	428549	2273056	6.86	331349
9	620390	2.02	307124	620390	3.34	185746	620392	4.2	147712
13	896803	280.52	3197	896803	330.96	2710	896802	581.37	1543
14	765961	87.38	8766	765961	133.65	5731	765951	874.35	876
15	3992098	0.38	10505521	3992098	1.60	2495061	3991903	4.29	930514
17	2202833	0.21	10489681	2202833	0.94	2343439	2201079	3.33	660985
21	382284	0.48	796425	382284	9.15	41780	378454	12.40	30520
24	2468151	238.31	10357	2468151	512.66	4814	787912	5515.86	143
27	27453	0.05	549060	27453	0.14	196093	27452	0.36	76256

Table 2 Full benchmark query results of LAStools with both LAS and LAZ and and Oracle/PDAL

From the results of table 2 we observe that LAStools with LAS offers the best performance. However, it deals with uncompressed data which is not the case in the other PCDMS's. Hence, the most fair comparison is between LAStools/LAZ and Oracle/PDAL and in this one LAStools/LAZ is faster in most of the cases but the response times are in the same order of magnitude. We noticed that the first execution of the queries (not shown in the table) in Oracle/PDAL was, in general, about 30% slower that the second execution while this difference was almost not

noticeable in the LAStools solutions. Moreover, a closer look at the table reveals some interesting issues:

- The number of returned points differ in Oracle/PDAL and LAStools. There are two issues that may cause this difference: (i) PDAL uses GEOS library for spatial operations which is different compared to the library used in LAStools. Concretely, they have different rules for points on the edges. (ii) Some of the input files in LAStools failed in importing (for being too large) and this may cause that some points are missing.
- Query #24 (long diagonal area) in Oracle/PDAL has less points and requires much more time that LAStools solutions. The execution of this query is severely damaged by the high amount of affected blocks in this query which produces that the PDAL process performs wrongly. This issue is under investigation.
- The number of points in LAStools solutions for queries #6 and #8 are higher than in Oracle/PDAL. The spatial region of such queries have holes and LAStools clipping methods ignore holes. This issue has already been reported.

This comparison demonstrates that a DBMS-based PCDMS can be almost as efficient as a specific and tailor-made file-based solution and, in theory, with all the benefits of DBMS systems. Obviously, much work is still to be done in the lines of (i) better cross-compatibility between PDAL and Oracle Spatial and Graph in order to actually exploit all the DBMS features and of (ii) a better exploitation of multi-process architectures to solve retrieval operations.

5 Point cloud web services

In this section we describe two web services based on point cloud data and we analyze their usage based on their generated log files.

5.1 AHN viewer

The first service that we analyze is based on ESRI ArcGIS and provides a 2D map of the Netherlands in which the user can add layers with elevation data derived from the various AHN data sets (AHN2 and also some parts of the upcoming AHN3) [4]. The data is structured in a multi-resolution imaging tiling structure.

We analyze the log file of the year 2012 produced by the server that provides the tiles for the AHN2 data set in order to identify the usage pattern of this service. Each row in the log file contains one connection to the server, i.e. the IP address of the connection, the date and the four coordinates of the bounding box of tile that was requested.

The analysis of the log file was done with Python/pandas [10]. There are 3,788,770 rows/connections in the file from 3,353 different IP's/users. From the

coordinates of the bounding box we can compute the area for each requested tile. In table 3 we present the number of requested tiles for each level of the multi-resolution structure where the level is derived from the area of the requested tile.

Level	Area[km^2]	#requests
0	65536.000	7069
1	16384.000	203053
2	4096.000	205947
3	1024.000	214446
4	256.000	239431
5	64.000	286880
6	16.000	326066
7	4.000	652608
8	1.000	509057
9	0.250	483089
10	0.062	366020
11	0.016	210958

Table 3 Number of requests per level which is derived from the area of each requested tile.

We focus in the period of time where the service was most used, this is the first two weeks of August 2012. In figure 2 we depict the number of simultaneous users of the service in that period. We can observe that such a service can get up to 25 simultaneous users. The number of simultaneous users is defined as the number of different IP's addresses in 10 minutes intervals.

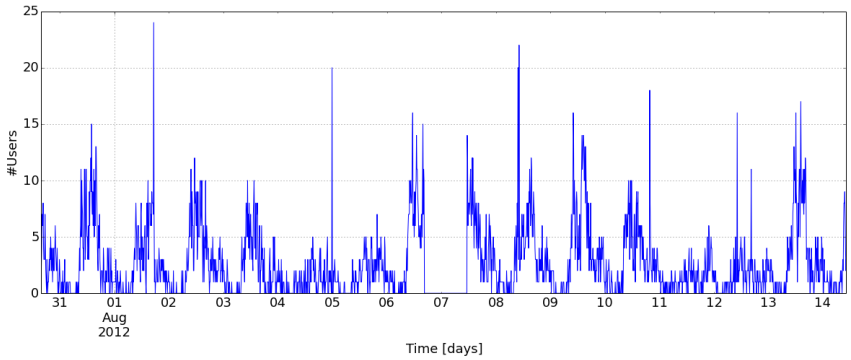

Fig. 2 Number of simultaneous users using the service during August 2012

When analyzing the typical usage pattern of a single user we detected that, in general, a user starts with requesting a tile with the highest area, which is the tile with all Netherlands, and then zooms in and out to smaller areas. A typical user can make up to 35 tile requests per second, thus the response time of each request should be below 28 milliseconds (in order to stay within 1 second response for all requests).

5.2 AHN2 3D web viewer and download tool

The second service that we analyze is the AHN2 3D web viewer and download tool [5] developed by the author team. Several renderers exploiting WebGL have become available for the point cloud web visualization such as plasio (http://plas.io/) or potree (http://potree.org/). In our project we extended potree to be able to visualize massive point clouds such as the complete AHN2 data set.

We have developed a publicly available web service with free and open-source tools for the 3D visualization of AHN2. In addition the service also has a multi-resolution download tool, a search bar, a measurement toolkit, a 2D overview map with field of view depiction, a demo mode and the tuning of the visualization parameters. In figure 3 we show an snapshot of the web service.

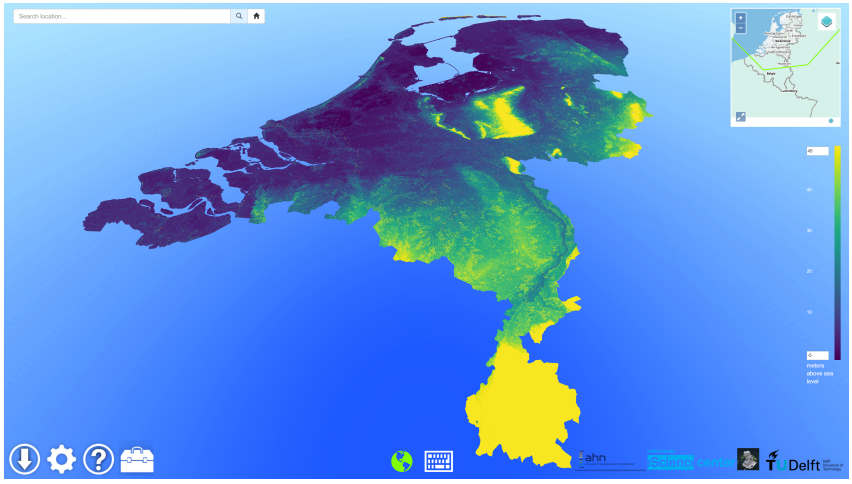

Fig. 3 Snapshot of the AHN2 3D web viewer and download tool

Potree uses a specific data structure, a multi-resolution octree, that requires the data set to be reorganized. This reorganization is very time consuming for large data sets. We have created a free and open-source tool that divides the generation of a massive octree into the generation of smaller octrees which can later be combined. The small octrees generation tasks are independent and can be distributed on different systems. Their combination is possible because the extent of all the nodes of the octrees is known and fixed.

The creation of the whole octree for AHN2 took around 15 days with processing distributed in different machines and processes. In table 4 we show an overview of the octree structure with the number of LAZ files and points per level, and the ratios of these for consecutive levels. Due to the flat nature (in elevation) of such country-wise point clouds (and more in the Netherlands where the highest point is at 380 meters) this octree structure is more similar to a quadtree structure (the ratios are

in most cases similar to 4). Note that the total number of points is not 638 billion points, 6.5 % of the points are dropped because of the way the tree is created. This can be decreased by tuning the tree creation parameters but the processing time would increase as well. Future work is also expected on trees that contain all the points.

level	#files	files_fact	#points	points_fact
0	1		34045	
1	4	4,00	134786	3,96
2	14	3,50	541973	4,02
3	41	2,93	2205484	4,07
4	143	3,49	8833283	4,01
5	499	3,49	36081908	4,08
6	1804	3,62	155411383	4,31
7	6767	3,75	668597511	4,30
8	25939	3,83	2834989373	4,24
9	101057	3,90	11355433955	4,01
10	398423	3,94	39911483676	3,51
11	1584598	3,98	112993998398	2,83
12	6671815	4,21	259014500658	2,29
13	29442790	4,41	170207571211	0,66
Total	38233895		597189817644	

Table 4 Overview of the AHN2 octree

With Python/pandas [10] we analyze the usage of the AHN2 web viewer and download tool by analyzing a log file with activity information of the server that contains the octree structure around the day of its official release announcement, on the 28th July 2015. The log file contains the requests to the data server, each line contains one connection to the server, i.e. the IP address of the connection, the date, the file that was requested, its size and the number of points in the file. Each LAZ file contains a node of an octree and each node contains points in cubic areas with certain density depending on the node level, from the file name we can derive the level of the node/file and the location.

In the three days which are contained in the log file a total of 54 IP's/users used the service (with 7 simultaneous users). In table 5 we show the number of requests for each level, we observe that most of the requested files are of levels 3 and 4. In figure 4 we show an histogram for the number of points of the requested files. We see that the majority of files have a number of points of less than 10.000 or between 80.000 and 100.000.

Like on the 2D web viewer, we observed that a typical user starts with requesting level 0 (all NL extent), then zooms in and out to deeper levels. For a smooth visualization experience the user requests up to 18 files per second, i.e. the response time of each request should be below 55 milliseconds.

Level	#requests
13	1036
12	635
11	607
10	690
9	904
8	1001
7	993
6	993
5	1120
4	2088
3	2656
2	1510
1	455
0	153

Table 5 Number of requests per level

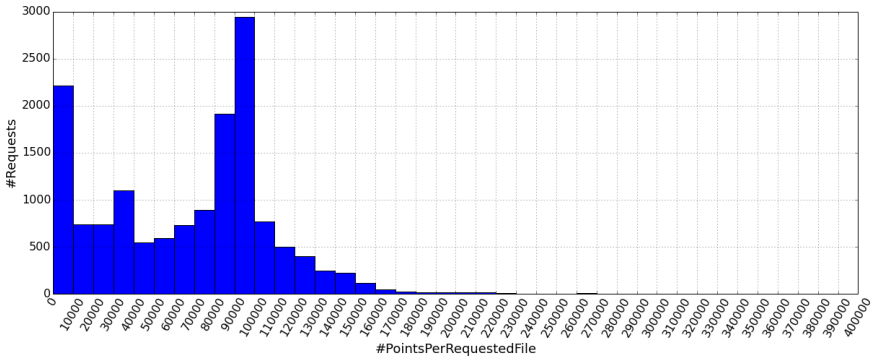

Fig. 4 Histogram for the number of points of the requested files

6 Executable benchmark extension

Web services such as the previously described are becoming very popular. Currently for these services the data is reorganized in multi-resolution data structures such as the octree used in the AHN2 3D web viewer and download tool. These services usually have simultaneous users. The typical user requests are to nodes from different level-of-detail's (LoD's) of the data structures. From our previous analyses we extracted an indication of their desired per-user performance to provide a smooth visualization experience.

Due to the lack of the required functionality and/or decent performance of the more generic PCDMS's introduced in Section 2 these services currently rely on specific solutions for their data management layers. In order to change this trend new developments are required. In addition, the executable benchmark should be extended to reflect the needs of these services.

6.1 LoD queries

We propose to extend the executable benchmark with LoD queries, i.e. queries where we select a representative sub-sample of the points in the queried region. For example get a representative 1% of the points of the whole point cloud (the Netherlands in our case) or queries such as the illustrated in figure 5 where a perspective view query is depicted, i.e. several rectangular regions with different detail/density of points are queried.

However, note that the current web services approaches and their related data structures have a major drawback, they use a discrete number of LoDs and the viewer may notice the difference in the point density between neighboring tiles/nodes at different levels. In [17] we present suggestions to solve this issue based on the vario-scale LoD research [18].

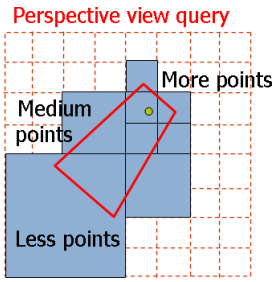

Fig. 5 Perspective view query

As previously stated most of the tested PCDMS's have no native support for multi-resolution/LoD data structures for point cloud data, only Oracle Spatial and Graph (see Usage notes of the SDO_PC_PKG.INIT method in the Oracle SDO_PC_PKG documentation [1]). Hence, at the moment in order to test the LoD performance in most of the PCDMS's hybrid solutions or specifically developed solutions have to be developed and used.

6.2 Multi-user queries

The second extension that we propose is to add multi-user queries, i.e. how the system reacts when multiple users are using it. Ideally the multi-user queries should be combined with LoD queries but due to the lack of LoD support on most of the PCDMS's this is currently complicated. Our proposed starting point is to define a set of similar queries and to simulate a pool of several users that simultaneously query the set. For example, in figure 6 we show a set of equal-area rectangular queries spread over the extent of the data set with 23 billion points (subset of AHN2) used in the *medium-benchmark* stage of the current executable benchmark. In this test there

should be several executions with different number of users and for each execution we determine which is the average query time. In addition to these synthetically generated queries, it is also possible to 're-run' the queries from the log files (at various speeds).

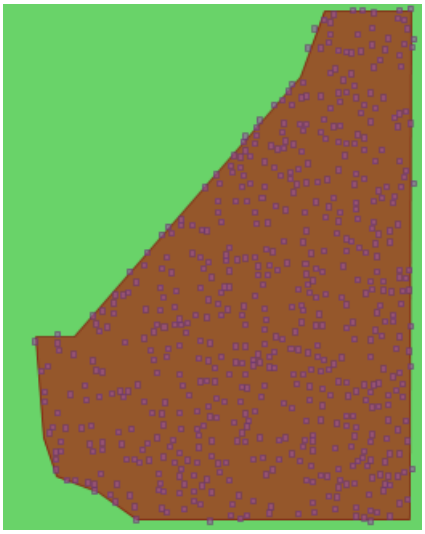

Fig. 6 Multi-user queries

The results of this test could be used to determine for which number of users the performance decreases to an unacceptable level, thus giving an indication of when scaling-out (horizontal scalability) is required. Cloud solutions are perfect candidates for such scaling operations. As part of our research, and together with the Microsoft Azure research team, such solutions are currently being investigated.

7 Conclusions and future work

In our previous work we defined a conceptual benchmark from which we derived an executable benchmark that was executed for various PCDMS's. In this paper, and motivated by the recent developments in PDAL, we have re-executed the *full-benchmark* stage with LAStools and Oracle/PDAL to find out that DBMS solutions for PCDMS's can be a real and attractive alternative to specific file-based solutions such as the efficient LAS/LAZ file format with the LAStools. However, several improvements are still required in the Oracle/PDAL approach: exploitation of multi-process architectures for faster data retrieval operations (also true for LAStools) and full compatibility between Oracle Spatial and Graph and PDAL in order to fully benefit from all the features of DBMS systems. In the future we will perform more

full-benchmark executions with other systems such as MonetDB, which point cloud support has also been recently improved, PostgreSQL or the native Oracle Spatial and Graph point cloud that is also being currently improved.

We described two web services based on point cloud data. The first one is based on ESRI ArcGIS and provides a 2D visualization while the second one, which is developed by the authors team, delivers a novel 3D web visualization tool and a download tool. We analyzed the usage patterns of such services as well as their used data structures and extracted which performance is required by the systems feeding them.

Due to the lack of efficient LoD support in the more generic PCDMS's, usually the web services rely on specific solutions for their data management layer. New developments are required in the PCDMS's in order to be usable for such applications. We proposed an extension of the executable benchmark to take into account the needs of the web services based on point clouds, more concretely we added LoD and multi-user queries. In the future, and as the required functionality becomes available, the extended executable benchmark will be re-executed.

Several activities for point cloud standardization have been recently initiated. Several initiatives, in which the authors team participate, have been recently started with focus on standardization for point cloud data and their usage. The OGC (Open Geospatial Consortium) has created a Point Cloud Domain Working Group (DWG) to address issues on the interoperability when sharing and processing point cloud data. There have been discussions on a possible cooperation with ISO TC211, the ISO technical committee for Geographic information/Geomatics. In parallel the OSGEO PointDown initiative was started with the aim of creating an overview on the usage of point cloud data over the web in order to provide a generic service definition.

Acknowledgements We thank all the members of the project Massive Point Clouds for eSciences, which is supported in part by the Netherlands eScience Center under project code 027.012.101. Also special thanks for their assistance to Mike Horhammer, Daniel Geringer, Siva Ravada (all Oracle), Markus Schütz (developer of potree), Martin Isenburg (developer of LAStools), and to Howard Butler, Andrew Bell and the rest of PDAL developers.

References

1. Oracle Database Online Documentation 12c Release 1 (12.1): Spatial and Graph Developer's Guide / SDO_PC_PKG Package (Point Clouds). https://docs.oracle.com/database/121/SPATL/sdo_pc_pkg_ ref.htm (2014)
2. rapidlasso GmbH LASzip - free and lossless LiDAR compression. http://www.laszip.org/ (2014)
3. Actueel Hoogtebestand Nederland (AHN). http://www.ahn.nl/ (2015)
4. AHN viewer. http://ahn.maps.arcgis.com/apps/webappviewer/index.html (2015)
5. AHN2 3D viewer and download tool. http://ahn2.pointclouds.nl/ (2015)
6. laz-perf. https://github.com/verma/laz-perf.git (2015)
7. Mapping the Via Appia in 3D. http://mappingtheviaappia.nl/4dgis/ (2015)

8. Oracle Exadata Database Machine X4-2. https://www.oracle.com/ engineered-systems/exadata/database-machine-x4-2/ index.html (2015)
9. PDAL. http://www.pdal.io/ (2015)
10. Python/pandas. http://pandas.pydata.org/ (2015)
11. rapidlasso GmbH. http://rapidlasso.com/ (2015)
12. De Kleijn, M., De Hond, R., Martinez-Rubi, O., Svetachov, P.: A 3d geographic in-formation system for mapping the via appia. Tech. Rep. Research Memorandum (VU-FEWEB) 2015-1, VU University Amsterdam, Amsterdam, The Netherlands (2015)
13. Fiore, S., DAnca, A., Palazzo, C., Foster, I., Williams, D., Aloisio, G.: Ophidia: Toward big data analytics for escience. Procedia Computer Science **18**, 2376 – 2385 (2013). DOI http://dx.doi.org/10.1016/j.procs.2013.05.409. URL http://www.sciencedirect.com/science/article/pii/S1877050913005528. 2013 International Conference on Computational Science
14. Group, T.P.G.D.: PostgreSQL 9.3.5 Documentation. Tech. Rep. 9.3.5, The PostgreSQL Global Development Group (2014)
15. Martinez-Rubi, O., Kersten, M., Goncalves, R., Ivanova, M.: A column-store meets the point clouds. FOSS4GEurope (2014)
16. Martinez-Rubi, O., van Oosterom, P., Gonçalves, R., Tijssen, T., Ivanova, M., Kersten, M.L., Alvanaki, F.: Benchmarking and improving point cloud data management in monetdb. SIGSPATIAL Special **6**(2), 11–18 (2015). DOI 10.1145/2744700.2744702. URL http://doi.acm.org/10.1145/2744700.2744702
17. van Oosterom, P., Martinez-Rubi, O., Ivanova, M., Horhammer, M., Geringer, D., Ravada, S., Tijssen, T., Kodde, M., Gonçalves, R.: Massive point cloud data management: Design, implementation and execution of a point cloud benchmark. Computers and Graphics **49**, 92 – 125 (2015). DOI http://dx.doi.org/10.1016/j.cag.2015.01.007. URL http://www.sciencedirect.com/science/article/pii/S0097849315000084
18. van Oosterom, P., Meijers, M.: Vario-scale data structures supporting smooth zoom and progressive transfer of 2d and 3d data. International Journal of Geographical Information Systems **28**(3), 455–478 (2014)
19. Ramsey, P.: A PostgreSQL extension for storing point cloud (LIDAR) data. https://github.com/pramsey/pointcloud (2015)
20. Suijker, P.M., Alkemade, I., Kodde, M.P., Nonhebel, A.E.: User requirements Massive Point Clouds for eSciences (WP1). Tech. rep., Delft University of Technology (2014). http://repository.tudelft.nl/view/ir/uuid%3A351e0d1e-f473-4651-bf15-8f9b29b7b800/

Appendixes

Appendix A: Executable benchmark data sets and queries

Data sets

Tables 6 and 7 contains information on the used data sets in the executable benchmark and their usage in the different stages. Figure 7 shows the extent of the used data sets.

Data set name	Benchmark	Points	Files	Format	Size (GB)
20M	Mini/Medium	20,165,862	1	LAS	0.4
210M	Medium	210,631,597	17	LAS	4.0
2201M	Medium	2,201,135,689	153	LAS	42.0
23090M	Medium	23,090,482,455	1,492	LAS	440.4
639478M	Full	639,478,217,460	60,185	LAZ	986.7
638860Mc	Full and clean	638,860,225,350	37,588	LAZ	2534.2

Table 6 Data sets name, benchmarks in which they are used, number of files and disk size

Data set name	Area (km^2)	Description
20M	1.25	TU Delft campus
210M	11.25	Major part of Delft city
2201M	125	City of Delft and surroundings
23090M	2,00	Major part of Zuid-Holland province
639478M	40,000	The Netherlands (full AHN2)
638860Mc	40,000	The Netherlands (full cleaned AHN2)

Table 7 Data sets area and description

Note that for the full AHN2 we include two versions of the data set. The first one, *639478M* was used in our previous *full-benchmark* execution while the second one, *638860Mc* is the one used for in the new execution and does not contain erroneous and duplicate points that are found in the first version. Also note the difference in the data sets sizes. This is due to the fact that for the cleaning process that was required to generate the second (cleaned) version of the full AHN2 data set the points in the files needed to be resorted and that affected dramatically the compression performance of LAZ. In the first version the files were separated according to their nature, in object and terrain files, and that improved the compressor performance. However, as part of the cleaning processes, these files were joined and the compression ratio was affected. The compression ratio improves when the data is resorted by LASTools as part of the benchmark execution (see table 1) but even in that case it is not optimal due to the mixing of point cloud from different nature (total size 1,66 Tb).

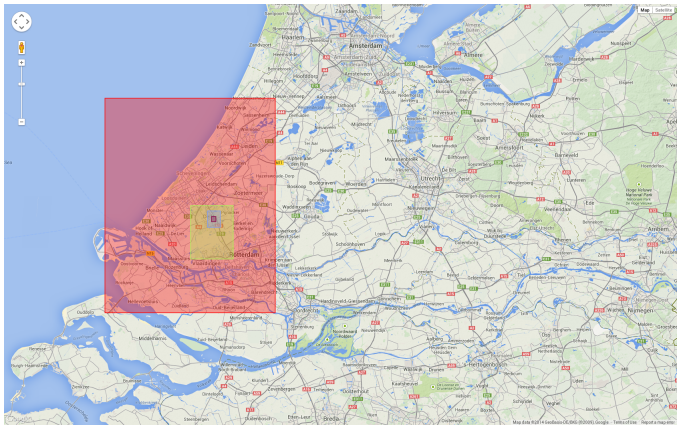

Fig. 7 Approximated projection of the extents of the used datasets in Google Maps: Purple area is for 20M dataset, cyan area is for 210M dataset, green area is for 2201M dataset and red area is for 23090M dataset.

Queries

Figures 8 and 9 show the first 20 query geometries that were used in the several benchmark stages. Table 8 describes all of them, their ID, the number of points in the boundary of the query geometry (*Pnts*) and the test data set name in which the query geometry is located.

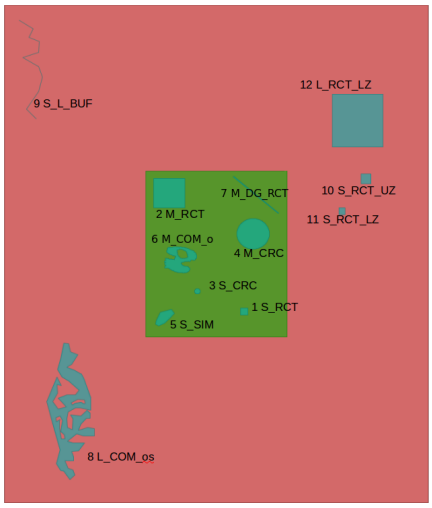

Fig. 8 Queries used in the medium benchmark (up to 210M extent)

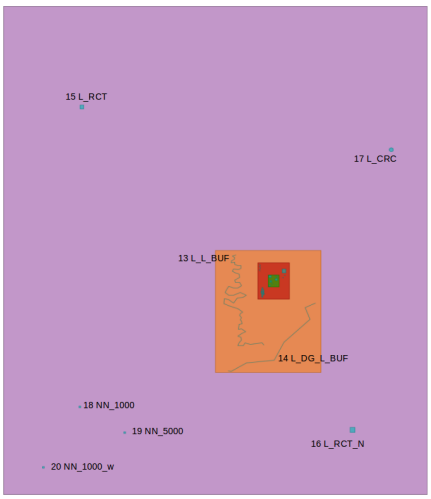

Fig. 9 Queries used in the medium benchmark

ID	Key	Pnts	Test	Area [km2]	Description
1	S_RCT	5	20M	0.0027	Small axis-aligned rectangle
2	M_RCT	5	20M	0.0495	Medium axis-aligned rectangle
3	S_CRC	97	20M	0.0013	Small circle, radius 20 m.
4	M_CRC	379	20M	0.0415	Medium circle, radius 115 m
5	S_SIM	9	20M	0.0088	Small, simple polygon
6	M_COM_o	792	20M	0.0252	Medium, complex polygon, 1 hole
7	M_DG_RCT	5	20M	0.0027	Medium, narrow, diagonal rectangular area
8	L_COM_os	89	210M	0.1341	Large, complex polygon, 2 holes
9	S_L_BUF	94	210M	0.0213	Small polygon (10 m buffer in line of 11 pts)
10	S_RCT_UZ	5	210M	0.0021	Small axis-aligned rectangle, cut in max. z
11	S_RCT_LZ	5	210M	0.0051	Small axis-aligned rectangle, cut in min. z
12	L_RCT_LZ	5	210M	0.1419	Large axis-aligned rectangle, cut in min. z
13	L_L_BUF	237	2201M	0.0475	Large polygon (1 m buffer in line of 61 pts)
14	L_DG_L_BUF	39	2201M	0.0499	Large polygon (2 m buffer in diag. line of 8 pts)
15	L_RCT	5	23090M	0.2342	Large axis-aligned rectangle
16	L_RCT_N	5	23090M	0.1366	Large axis-aligned rectangle in empty area
17	L_CRC	93	23090M	0.1256	Large circle
18	NN_1000	1	23090M	0.0000	Point for NN query, 1000 nearest points
19	NN_5000	1	23090M	0.0000	Point for NN query, 5000 nearest points
20	NN_1000_w	1	23090M	0.0000	Point in water for NN query, 1000 nearest points
21	L_NAR_RCT	5	639478M	0.0236	Large narrow axis-aligned rectangle, points
22	L_NAR_DG_RCT	5	639478M	0.6399	Large narrow diagonal rectangle, min(Z)
23	XL_NAR_DG_RCT	5	639478M	42.5573	Very large narrow diagonal rectangle, max (Z)
24	L_NAR_DG_RCT_2	5	639478M	0.1208	Large narrow diagonal rectangle, points
25	PROV_DG_RCT	5	639478M	3022.0427	Provincial size diagonal rectangle, min(Z)
26	MUNI_RCT	5	639478M	236.7744	Municipality size diagonal rectangle, max(Z)
27	STREET_DG_RCT	5	639478M	0.0016	Street size diagonal rectangle, points
28	VAALS	1565	639478M	23.8972	Municipality Vaals, avg(Z)
29	MONTFERLAND	1565	639478M	106.6520	Municipality Montferland, avg(Z)
30	WESTERVELD	1569	639478M	282.7473	Municipality Westerveld, avg(Z)

Table 8 Description of the different queries

.

Appendix B: Executable benchmark loading results

Table 9 contains the loading details of the *medium-benchmark* execution for various PCDMS's and data sets. The results of LAStools are when using LAS (instead of LAZ). The PCDMS using the blocks model were using the compression available at that time (second half 2014) and with optimal block sizes previously computed. Note that all the Oracle Exadata approaches (*oe** on the table) run in a different hardware than the other approaches.

Table 10 contains the loading details of the *full-benchmark* execution that was done with LAStools and Oracle Exadata PCDMS's. Note that for this execution the *6394784M* data set was used, i.e. the AHN2 version with duplicate and erroneous points. For an in-deep analysis of these results we refer the reader to our previous work [17].

Approach	Time[s]				Size[MB]		Points	Points/s
	Total	Init.	Load	Close	Total	Index		
pf20M	44.07	2.00	13.86	28.21	1558.71	551.83	20,165,862	457,587
pf210M	771.63	2.09	71.44	698.10	16249.24	5762.20	210,631,597	272,970
pf2201M	9897.37	1.15	722.91	9173.31	169776.13	60214.46	2,201,135,689	222,396
pf23090M	95014.05	3.64	8745.90	86264.51	1780963.91	631663.71	23,090,482,455	243,022
of20M	128.43	0.51	123.92	4.00	879.50	453.85	20165862	157018
of210M	275.72	0.32	230.01	45.39	9124.50	4739.94	210,6315,97	763,933
of2201M	1805.68	0.31	1228.81	576.56	95825.00	49533.74	2,201,135,689	1,219,007
of23090M	15825.53	0.14	9226.37	6599.02	997062.50	519621.48	23,090,482,455	1,459,065
mf20M	7.15	1.14	3.70	2.31	475.78	14.09	20,165,862	2,820,400
mf210M	29.15	1.13	16.63	11.39	4888.05	66.95	210,631,597	7,225,784
mf2201M	304.14	4.91	198.76	100.47	50661.30	281.17	2,201,135,689	7,237,245
mf23090M	8490.90	0.93	5466.99	3022.98	529448.70	949.37	23,090,482,455	2,719,439
pb20M	153.41	22.03	130.53	0.85	101.77	0.69	20,165,862	131,451
pb210M	129.14	0.81	121.00	7.33	1009.13	5.18	210,631,597	1,631,033
pb2201M	754.22	0.86	687.49	65.87	10245.61	53.05	2,201,135,689	2,918,427
pb23090M	12263.05	0.87	7450.10	4812.08	106781.48	552.77	23,090,482,455	1,882,931
ob20M	296.22	0.35	228.17	67.70	226.50	0.20	20,165,862	68,077
ob210M	1246.87	0.25	557.70	688.92	2244.50	1.44	210,631,597	168,928
ob2201M	16737.02	0.86	7613.39	9122.77	21220.50	13.25	2,201,135,723	131,513
ob23090M	192612.07	0.31	96148.06	96463.70	220085.50	165.55	23,090,482,953	119,881
lt20M	9.49	0.03	9.46	0.00	384.65	0.02	20,165,862	2,124,959
lt210M	30.07	0.02	28.68	1.37	4021.37	3.88	210,631,597	7,004,709
lt2201M	218.06	0.02	216.18	1.86	41992.78	9.40	2,201,135,689	10,094,174
lt23090M	2129.30	0.03	2116.64	12.63	440484.48	68.04	23,090,482,455	10,844,166
oenc21090M	508.71				537600.00	0.00	21,090,482,455	41,458,753
oeql21090M	619.14				218504.00	0.00	21,090,482,455	34,064,157
oeqh21090M	928.84				94240.00	0.00	21,090,482,455	22,706,269
oeal21090M	1149.55				93992.00	0.00	21,090,482,455	18,346,729
oeah21090M	1767.53				59096.00	0.00	21,090,482,455	11,932,177

Table 9 Times and sizes of the data loading procedure for the different PCDMSs and datasets. The names of approaches encode the PCDMS name (o for Oracle, p for PostgreSQL, etc), flat or blocked model (f and b, respectively), and the dataset name. For example *ob2201M* stands for the dataset 2201M loaded in the Oracle blocks PCDMS

system	LAStools	Oracle Exadata
Total load time	22:54 hours	4:39 hours
total size	12,181 Tb	2,240 Tb
#points	638,609,393,087	639,478,217,460

Table 10 *Full-benchmark* loading results for the LAStools and Oracle Exadata PCDMSs.

*

Appendix C: Executable benchmark querying results

Table 11 contains the number of returned points and the response times of the first seven queries for the different PCDMS's and data sets. Note that each query was executed twice, the numbers in the table are from the second execution, usually called hot query because of the fact that the PCDMS may be able to reuse cached data either by the PCDMS itself or the file system or the operative system (OS).

Table 12 contains the number of returned points and the response times of the execution of the 30 *full-benchmark* queries for the LAStools and Oracle Exadata PCDMS. Note that for LAStools two columns are given. The first one is when using a DBMS in a pre-filtering step for the queries and the other is without it. For an in-deep analysis of these results we refer the reader to our previous work [17].

Approach	Number of points							Time[s]						
	1	**2**	**3**	**4**	**5**	**6**	**7**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
pf20M	74947	718131	34637	562919	182792	387134	45805	0.35	2.25	0.24	1.90	1.18	1.72	0.84
pf210M	74947	718131	34637	562919	182792	387135	45805	0.42	2.50	0.27	2.26	0.91	1.65	1.34
pf2201M	74947	718131	34637	562919	182792	387135	45805	4.92	19.03	2.90	18.28	9.37	17.71	10.16
pf23090M	74947	718131	34637	562919	182792	387134	45805	5.17	18.02	3.32	17.75	9.42	15.46	13.50
of20M	74872	718021	34691	563037	182861	387145	45813	0.24	0.37	0.28	1.85	0.75	1.32	1.32
of210M	74872	718021	34691	563037	182861	387145	45813	0.45	0.58	0.52	1.27	1.12	1.47	1.79
of2201M	74872	718021	34691	563037	182861	387145	45813	1.47	3.87	1.29	4.26	4.38	6.60	5.24
of23090M	74872	718021	34691	563037	182861	387145	45813	1.25	18.20	2.34	22.75	6.99	27.18	635.06
mf20M	74872	718021	34691	563037	182861	387134	45813	0.06	0.13	0.06	0.20	9.96	187.16	38.71
mf210M	74872	718021	34691	563037	182861	387135	45813	0.13	0.26	0.15	0.28	9.95	185.65	38.56
mf2201M	74872	718021	34691	563037	182861	387135	45813	0.64	0.90	0.64	0.77	10.37	186.38	39.17
mf23090M	74872	718021	34691	563037	182861	387134	45813	7.21	16.74	9.70	9.88	17.94	198.51	43.96
pb20M	74947	718131	34697	563108	182930	387142	45821	0.32	2.14	0.20	1.69	0.61	1.72	0.41
pb210M	74947	718131	34697	563108	182930	387142	45821	0.32	2.15	0.20	1.65	0.64	1.62	0.46
pb2201M	74947	718131	34697	563108	182930	387142	45821	0.31	2.19	0.21	1.67	0.67	1.63	0.41
pb23090M	74947	718131	34697	563108	182930	387142	45821	0.32	2.19	0.21	1.68	0.68	1.68	0.44
ob20M	74947	718131	34697	563110	182930	387145	45821	0.41	1.38	0.34	1.21	0.62	1.38	0.53
ob210M	74947	718131	34697	563110	182930	387145	45821	0.38	1.28	0.36	1.22	0.62	1.29	0.54
ob2201M	74947	718131	34697	563110	182930	387145	45821	0.39	1.36	0.36	1.23	0.60	1.33	0.50
ob23090M	74947	718131	34697	563110	182930	387145	45821	0.40	1.30	0.34	1.21	0.60	1.40	0.53
lt20M	74840	717931	34695	563049	182849	460068	45834	0.04	0.12	0.03	0.09	0.66	1.48	0.51
lt210M	74840	717931	34695	563049	182849	460068	45834	0.06	0.14	0.05	0.14	0.67	1.51	0.51
lt2201M	74840	717931	34695	563049	182849	460068	45834	0.06	0.12	0.05	0.14	0.70	1.51	0.51
lt23090M	74840	717931	34695	563049	182849	460068	45834	0.05	0.15	0.05	0.14	0.68	1.50	0.52
oeqh21090M	40368	369352	19105	290456	132307	173927	9559	0.18	0.35	0.59	0.72	0.66	0.67	0.46

Table 11 Comparison of number of points returned and response times by the hot queries 1 to 7 in the different approaches

Query	LAStools			Oracle Exadata	
	#points	Time[s]		#points	Time[s]
		DB	No DB		
1	74861	0.07	0.90	74863	0.48
2	718057	0.16	0.87	718070	0.79
3	34700	0.07	0.78	34675	1.22
4	563119	0.16	0.92	563082	1.69
5	182871	0.70	33.24	182875	1.43
6	460140	1.52	32.79	387201	1.29
7	45831	0.55	32.29	45815	1.71
8	2365925	3.72	36.21	2273469	2.86
9	620568	2.34	34.76	620719	1.58
10	2413	0.08	0.88	2434	0.40
11	591	0.05	0.84	591	0.44
12	343168	0.26	1.03	343171	0.60
13	897042	412.29	829.49	897359	23.34
14	765989	102.19	424.91	766029	15.05
15	3992330	0.49	1.39	3992290	2.23
16	0	0.04	0.75	0	0.00
17	2203066	0.32	1.18	2201280	2.51
21	382395	2.28	20.74	382335	0.95
22	12148049	142.09	1115.27	12147802	113.37
23	691422551	313.14	828.51	691422526	330.85
24	2468239	234.40	4261.77	2468367	393.21
25	-	-	-	$3.5319 \cdot 10^10$	1193.10
26	2124162497	282.89	1124.04	2124162754	25.79
27	27443	0.13	923.59	27459	1.23
28	809097723	1553.87	1885.54	866802585	67.67
29	1509662615	3438.34	5697.79	1509662411	120.02
30	-	-	-	$1.3443 \cdot 10^10$	3569.54

Table 12 Full benchmark query results of LAStools and Oracle Exadata. Notes: (a.) Nearest neighbours queries (#18, #19 and #20) were not executed as functionality was not implemented, and (b.) Oracle Exadata query #25 was also re-run using an MBR instead of a geometry close to an MBR with and improved the time to 353.93 seconds with 3.6546E+10 selected points