Point cloud data management benchmark: Oracle, PostgreSQL, MonetDB and LAStools

Oscar Martinez-Rubi, Peter van Oosterom, Romulo Gonçalves , Theo Tijssen
(TU Delft and Netherlands eScience Center)

Management of massive point cloud data: wet and dry (2), Delft, 8 December 2015

Content overview

0. *Background*
1. Conceptual benchmark
2. Executable benchmark
3. Conclusion and future work

Rijkswaterstaat
Ministerie van Infrastructuur en Milieu

NL eScience Point cloud project

- TU Delft:
  1. GIS technology
  2. TU Delft, Library, contact with research & education users, dissemination & disclosure of point cloud data
  3. 3TU.Datacentrum, long-term provision of ICT-infra
  4. TU Delft Shared Service Center ICT, storage facilities
- NL eScience Center, designing and building ICT infrastructure

- Oracle Spatial, New England Development Centre (USA), improving existing software
- Rijkswaterstaat, data owner (and in-house applications)
- Fugro, point cloud data producer
- CWI, Amsterdam, MonetDB group

User requirements

- report user requirements, based on structured interviews with
  - Government community: RWS (Ministry)
  - Commercial community: Fugro (company)
  - Scientific community: TU Delft Library
- report at MPC public website http://pointclouds.nl
- basis for conceptual benchmark, with tests for functionality, classified by importance (based on user requirements and Oracle experience)

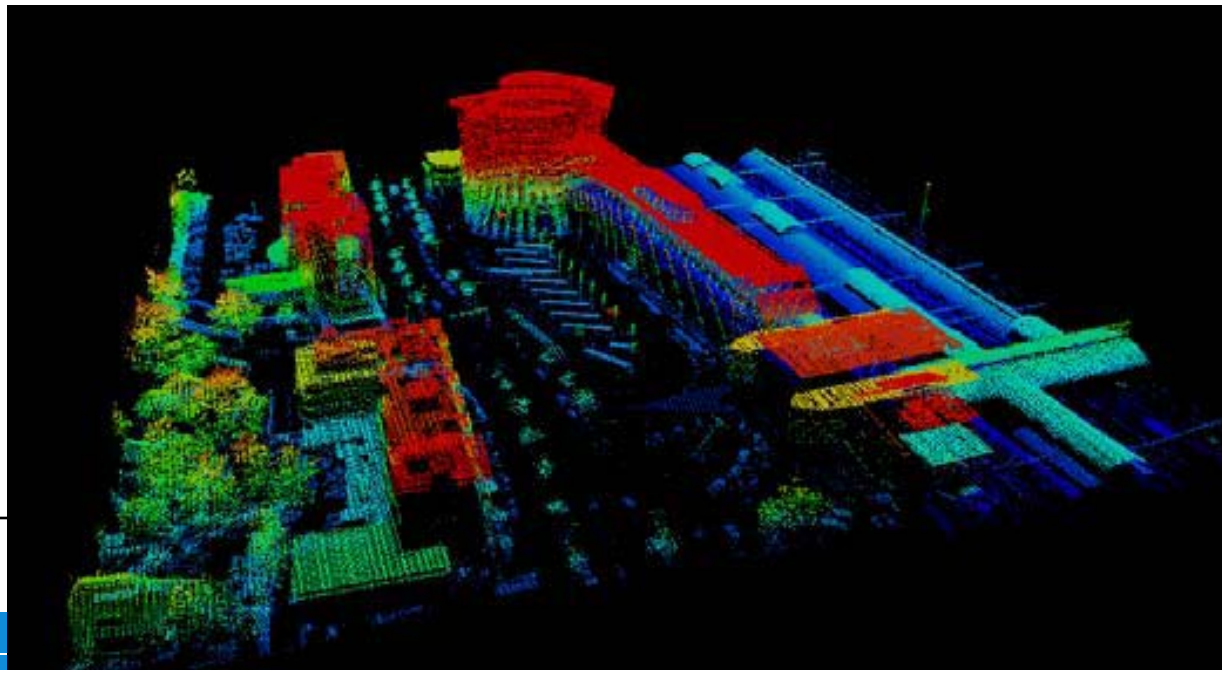
Point cloud data

- Not new, but growing rapidly (with increasing data management problems)
- Some producing technologies:
  - Laser scanning (terrestrial, airborne)
  - Multi-beam echo sounding
  - Stereo photogrammetry (point matching)

- Results are huge data sets, very detailed (precise), information rich

- AHN (2009):
  1 point per 16 m^2
  →AHN2 (2014):
  10 points per 1 m^2

Many Terabytes



Case AHN2 (open data in NL)

- TU Delft Library: distribution point for (geo-)data, including AHN2
- Users include:
  - Architecture (Urbanism, Landscape architecture),
  - Civil Engineering and Geosciences (Water Management, Geo-engineering)
  - Aerospace Engineering (Mathematical Geodesy & Positioning)
  - Electrical Engineering, Mathematics and Computer Science (Computer Graphics & Visualisation)
  - 'Outside' the TU Delft (but on campus): Deltares
  - More and more students are using this data
- Challenge: how to make this big data useful to the users? (AHN2: 640,000,000,000 points with 3 cm hor./vert. accuracy)

Applications, often related to the environment

- examples:
  - flood modeling,
  - dike monitoring,
  - forest mapping,
  - generation of 3D city models, etc.
- it is expected that future data sets will feature an even higher point density (Cyclomedia announced a 35 trillion pts NL data set)
- because of a lack of (processing) tools, most of these datasets are not being used to their full potential (e.g. first converting to 0.5 m grid or 5 m grid, the data is losing potentially significant detail)

Approach

- develop infrastructure for the storage, the management, ... of massive point clouds (note: no object reconstruction)
- support range of hardware platforms: normal/ department servers (HP), cloud-based solution (MS Azure), Exadata (Oracle)
- scalable solution: if data sets becomes 100 times larger and/or if we get 1000 times more users (queries), it should be possible to configure based on same architecture
- generic, i.e. also support other (geo-)data and standards based, if non-existent, then propose new standard to ISO (TC211/OGC): Web Point Cloud Service (WPCS)
- also standardization at SQL level (SQL/SFS, SQL/raster, SQL/PC)?

Why a DBMS approach?

- today's common practice: specific file format (LAS, LAZ, ZLAS,...) with specific tools (libraries) for that format
- point clouds are a bit similar to raster data: sampling nature, huge volumes, relatively static
- specific files are sub-optimal data management:
  - multi-user (access and some update)
  - scalability (not nice to process 60,000 AHN2 files)
  - integrate data (types: vector, raster, administrative)
- 'work around' could be developed, but that's building own DBMS
- no reason why point cloud cannot be supported efficiently in DBMS
- perhaps 'mix' of both: use file (or GPU) format for the PC blocks

Content overview

0. Background
1. *Conceptual benchmark*
2. Executable benchmark
3. Conclusion and future work

Benchmark organization

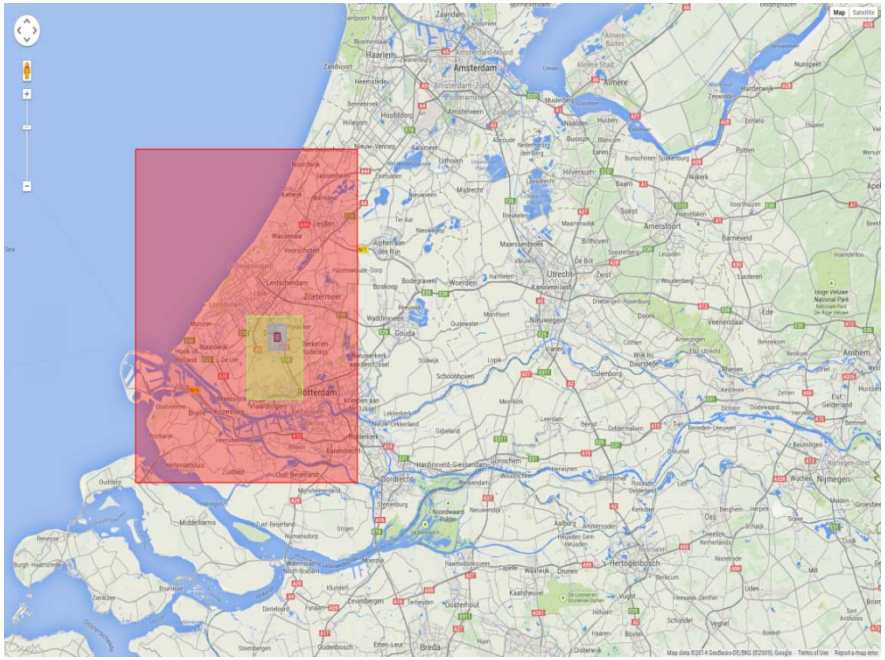
- mini-benchmark, small subset of data
  (20 million points) + limited functionality
  → get experience with benchmarking, platforms
  → first setting for tuning parameters: block size, compression

- medium-benchmark, larger subset
  (20 billion points) + more functionality
  → more serious testing, first feeling for scalability
  → more and different types of queries (e.g. nearest neighbour)

- full-benchmark, full AHN2 data set
  (640 billion points) + yet more functionality
  → LoD (multi-scale), multi-user test

- scaled-up benchmark, replicated data set
  (20 trillion points) → stress test

Test data: AHN2 (subsets)



Name	Points	LAS files	Disk size [GB]	Area [km²]	Description
20M	20,165,862	1	0.4	1.25	TU Delft campus
210M	210,631,597	16	4.0	11.25	Major part of Delft city
2201M	2,201,135,689	153	42.0	125	City of Delft and surroundings
23090M	23,090,482,455	1,492	440.4	2,000	Major part of Zuid-Holland province
639478M	639,478,217,460	60,185	11,644.4	40,000	The Netherlands

HP DL380p Gen8

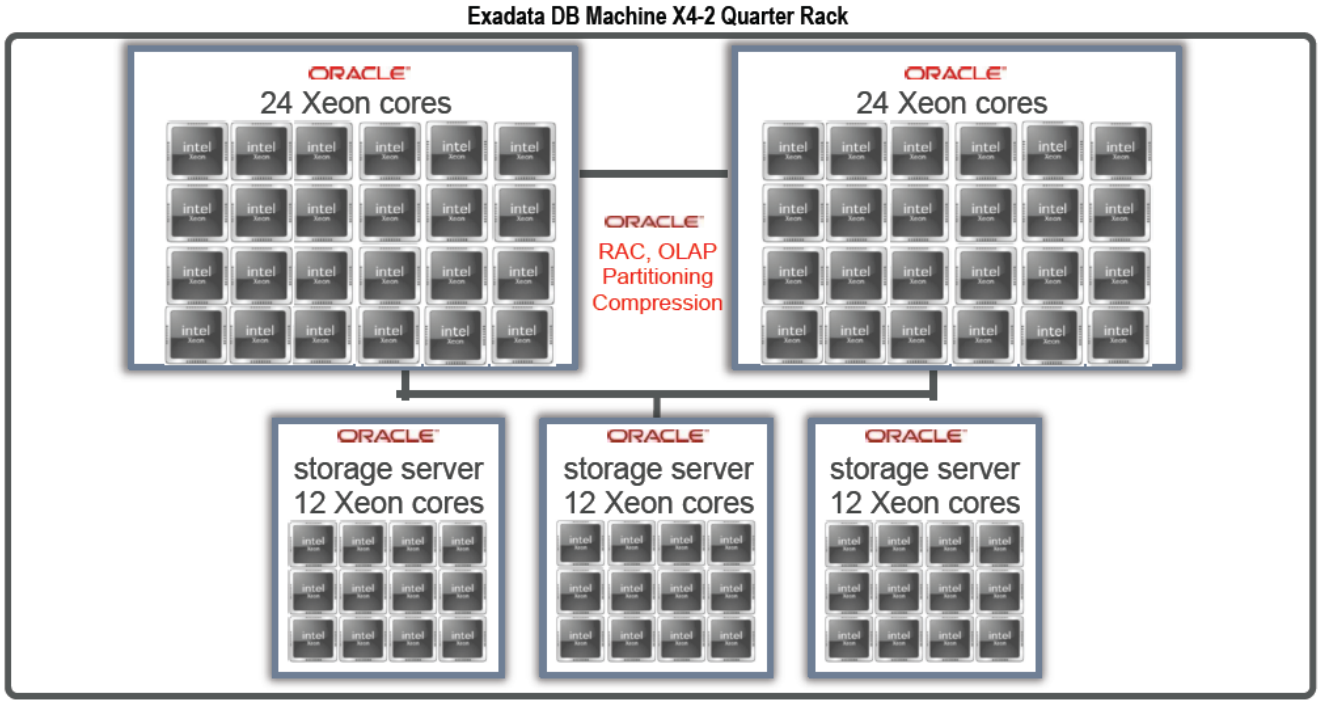
‘Normal’ server hardware configuration:

- HP DL380p Gen8 server
  1. 2 x 8-core Intel Xeon processors (32 threads), E5-2690 at 2.9 GHz
  2. 128 GB main memory (DDR3)
  3. Linux RHEL 6.5 operating system

- Disk storage – direct attached
  1. 400 GB SSD (internal)
  2. 6 TB SAS 15K rpm in RAID 5 configuration (internal)
  3. 2 x 41 TB SATA 7200 rpm in RAID-5 configuration (external in 4U rack 'Yotta-III' box, 24 disks)

Exadata X4-2: Oracle SUN hardware for Oracle database software

- database Grid: multiple Intel cores, computations
  Eight, quarter, half, full rack with resp. 24, 48, 96, 192 cores
- storage Servers: multiple Intel cores, massive parallel smart scans
  (predicate filtering, less data transfer, better performance)
- hybrid columnar compression (HCC): query and archive modes



Content overview

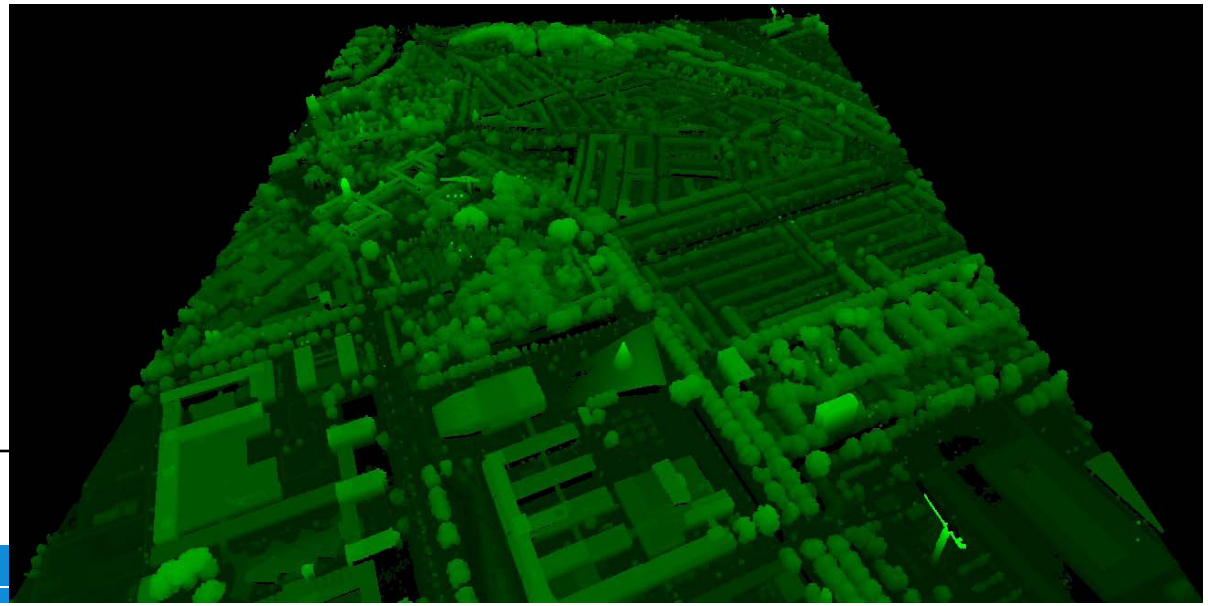
0. Background
1. Conceptual benchmark
2. *Executable benchmark*
3. Conclusion and future work

netherlands eScience center

First executable mini-benchmark

- load small AHN2 dataset (one of the 60,000 LAS files) in:
  1. Oracle PointCloud
  2. Oracle flat (1 x,y,x attribute per row, btree index on x,y)
  3. PostgreSQL PointCloud
  4. PostgreSQL flat (1 2D point + z attribute per row, spatial index)
  5. MonetDB flat (1 x,y,z attribute per “row”, no index)
  6. LASTools (file, no database, tools from Rapidlasso, Martin Isenburg)
- no compression, PC block size 5000, one thread, xyz only
- input 20,165,862 XYZ points (LAS 385 MB, LAZ 37 MB)

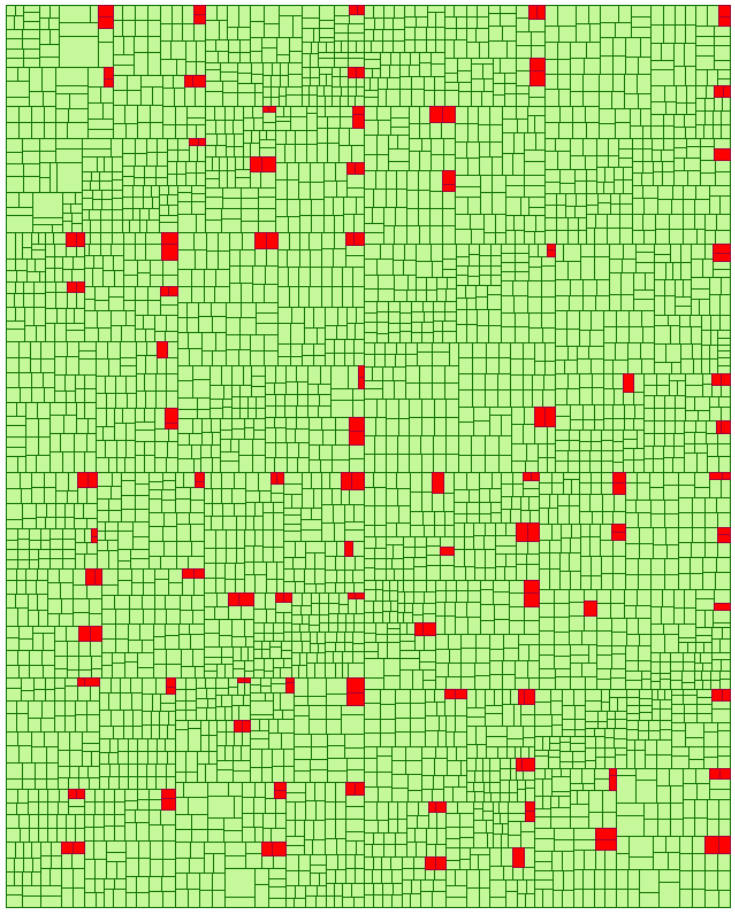
- Blocked – Flat DBMS approach (and files)



Oracle 12c PointCloud (SDO_PC)

- point cloud metadata in SDO_PC object
- point cloud data in SDO_PC_BLK object (block in BLOB)
- loading: text file X,Y,Z,... using bulk loader (from LAS files) and use function SDO_PC_PKG.INIT and SDO_PC_PKG.CREATE_PC procedure (time consuming)
- block size 5000 points
- various compression options (initially not used)

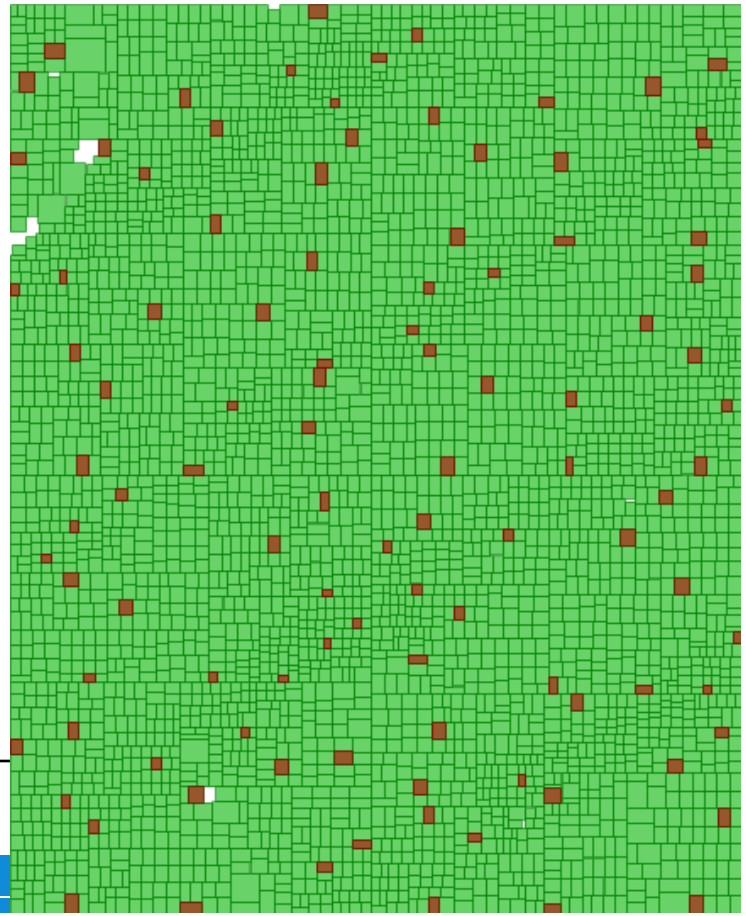
- no white areas
- non-overlapping blocks
- 4037 blocks:
  - 4021 with 5000 points
  - some with 4982 - 4999 points
  - some others with 2501 - 2502 points



PostgreSQL PointCloud

- use PointCloud extension by Paul Ramsey
  https://github.com/pramsey/pointcloud
- also PostGIS extension (query)
- loading LAS(Z) with PDAL pcpipeline
- block size 5000 points
- spatial GIST index for the blocks

- white areas
- 4034 blocks:
  - 3930 blocks with 4999 points
  - 104 blocks with 4998 points



MonetDB

- MonetDB: open source column-store DBMS developed by Centrum Wiskunde & Informatica (CWI), the Netherlands
- MonetDB/GIS: OGC simple feature extension to MonetDB/SQL
- no support for blocked model → only flat model tested
- no need to specify index (will be created on-the-fly when needed by first query...)

LASTools (use licensed/paid version)

- programming API LASlib (with LASzip DLL) that implements reading and writing LiDAR points from/to ASPRS LAS format (http://lastools.org/ or http://rapidlasso.com/)
- LAStools: collection of tools for processing LAS or LAZ files; e.g. lassort.exe (z-orders), lasclip.exe (clip with polygon), lasthin.exe (thinning), las2tin.exe (triangulate into TIN), las2dem.exe (rasterizes into DEM), las2iso.exe (contouring), lasview.exe (OpenGL viewer), lasindex.exe (index for speed-up),...

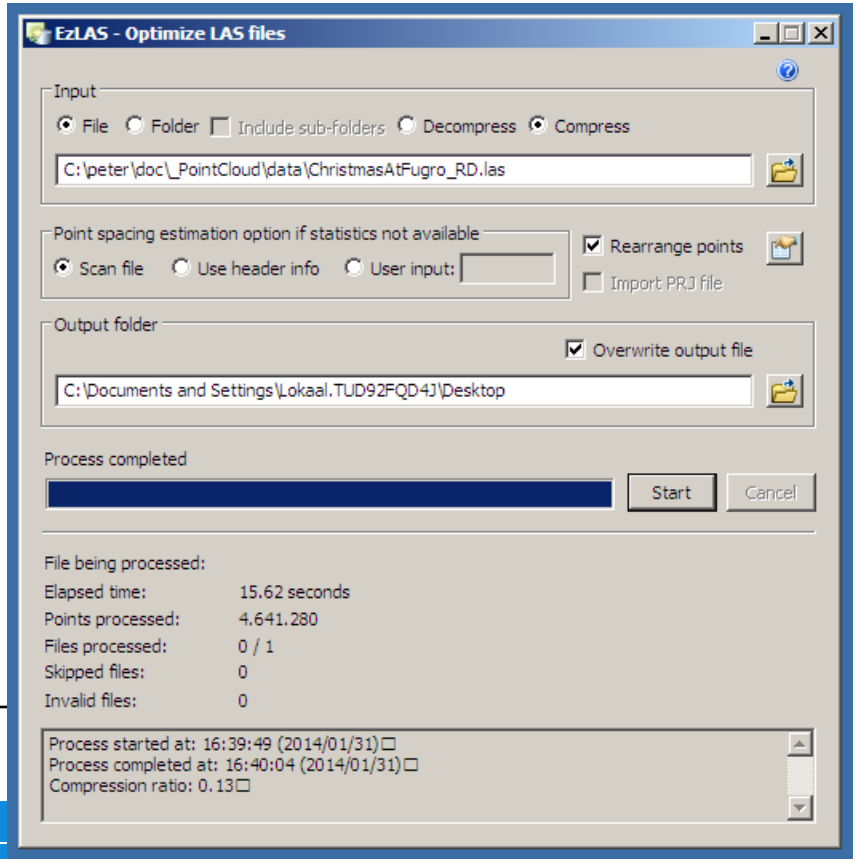
- command: **`lasindex [LAS File path]`**
  create LAX file per LAS file with spatial indexing info
- some tools only work in Windows,
  for Linux Wine (http://www.winehq.org)
- note: file based solution, inefficient for large number of files; AHN2 data sets consists of over 60,000 LAZ (and LAX) files

Esri's LiDAR file format: ZLAS

- Esri LAS Optimizer/Compressor into ZLAS format
- standalone executable, ArcGIS not required
- same executable EzLAS.exe for compression and decompression

- compression a bit disappointing: from 385 MB to 42 MB (factor 9) compared to LAZ 36 MB (factor 10)
- perhaps the 'use' performance is better (in Esri tools)

- not further tested in benchmark



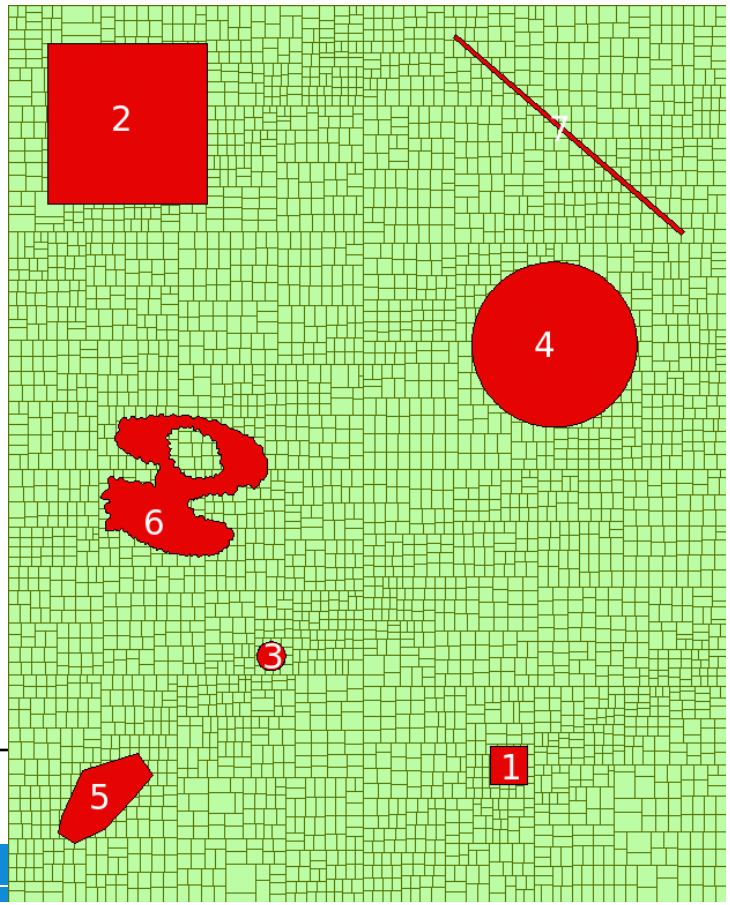
From mini- to medium-benchmark: load (index) times and sizes

p=Postgres, o=Oracle, m=MonetDB, lt=LAStools
f=flat model, b=blocked model
20, 210, 2201, 23090M = millions of points

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

Query geometries (mini-benchmark)

1. small rectangle, axis aligned, 51 x 53 m
2. large rectangle, axis aligned, 222 x 223 m
3. small circle at (85365 446594), radius 20 m
4. large circle at (85759 447028), radius 115 m
5. simple polygon, 9 points
6. complex polygon, 792 points, 1 hole
7. long narrow diagonal rectangle



SQL Query syntax (geometry 1)

- PostgreSQL PointCloud:
```
CREATE TABLE query_res_1 AS
SELECT PC_Explode(PC_Intersection(pa,geom))::geometry
FROM patches pa, query_polygons
WHERE pc_intersects(pa,geom) AND query_polygons.id = 1;
```
note, actually points have been converted to separate x,y,z values

- Oracle PointCloud:
```
CREATE TABLE query_res_1 AS
SELECT * FROM table (sdo_pc_pkg.clip_pc(SDO_PC_object,
(SELECT geom FROM query_polygons WHERE id = 1),
NULL, NULL, NULL, NULL));
```
note SDO_PC_PKG.CLIP_PC function will return SDO_PC_BLK objects, actually have been converted via geometry (multipoint) with SDO_PC_PKG.TO_GEOMETRY function to separate x,y,z values

- LASTools:
```
lasclip.exe [LAZ File] -poly query1.shp
-verbose -o query1.laz
```

Queries: returned points + times (note flat model: increasing times)

- Scalability flat model: an issue

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

Full AHN2 benchmark: loading 640 B (Exadata: different hardware)

system	Total load time [hours]	Total size [TB]
LAStools LAS	22:54	12.18
LAStools LAZ	19:41	1.66
Oracle Exadata	4:39	2.24
Oracle / PDAL	33:53	2.07
MonetDB	17:21	15.0

Query performance full AHN2 (all on same hardware)

- LAS and LAZ solutions with database wrapper for the 60,000 files
- LAS (uncompressed) fastest in query
- LAZ and Oracle/PDAL in same league
- LAS and LAZ do not support query with holes (query 6)

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

Content overview

0. Background
1. Conceptual benchmark
2. Executable benchmark
3. *Conclusion and future work*

Conclusion

- Designed and executed point cloud benchmark
- Influenced point cloud data management developers (systems improved, sometimes dramatically, orders of magnitude)
- Developed an interactive 3D point cloud webservice (and viewer)

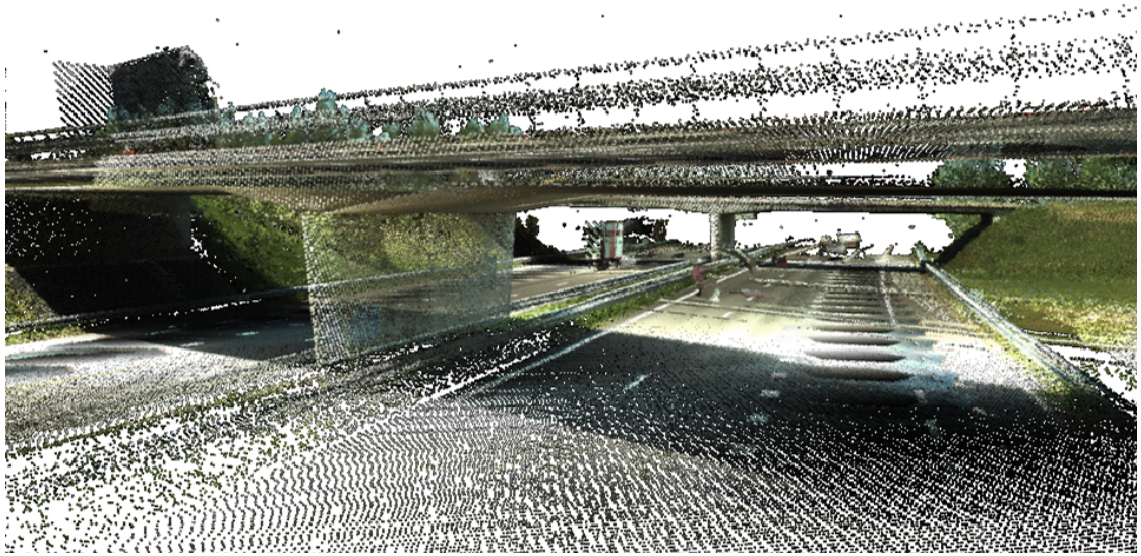
- All code developed open source (majority in SQL + Python)
  - https://github.com/NLeSC/pointcloud-benchmark
  - https://github.com/NLeSC/Massive-PotreeConverter

- A lot of future work:
  - Multi-user testing (based on collected use patterns)
  - Discrete LoD testing (perspective views)
  - Investigate continuous LoD
  - Add more countries ("OpenPointCloudMap", with upload facility)

→nD-PointCloud (submitted H2020 FET Open): http://nd-pc.org

Acknowledgements

- The massive point cloud research is supported by Netherlands eScience Center, the Netherlands Organisation for Scientific Research (NWO) (project code: 027.012.101)



Interested?

- More reading in Springer paper:
  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, pp. 92–125 (2015)

- Join OGC's Point Cloud DWG
  http://www.opengeospatial.org/projects/groups/pointclouddwg

- Try our 640,000,000,000 points web-based 3D point cloud viewer at http://ahn2.pointclouds.nl (comments welcome)