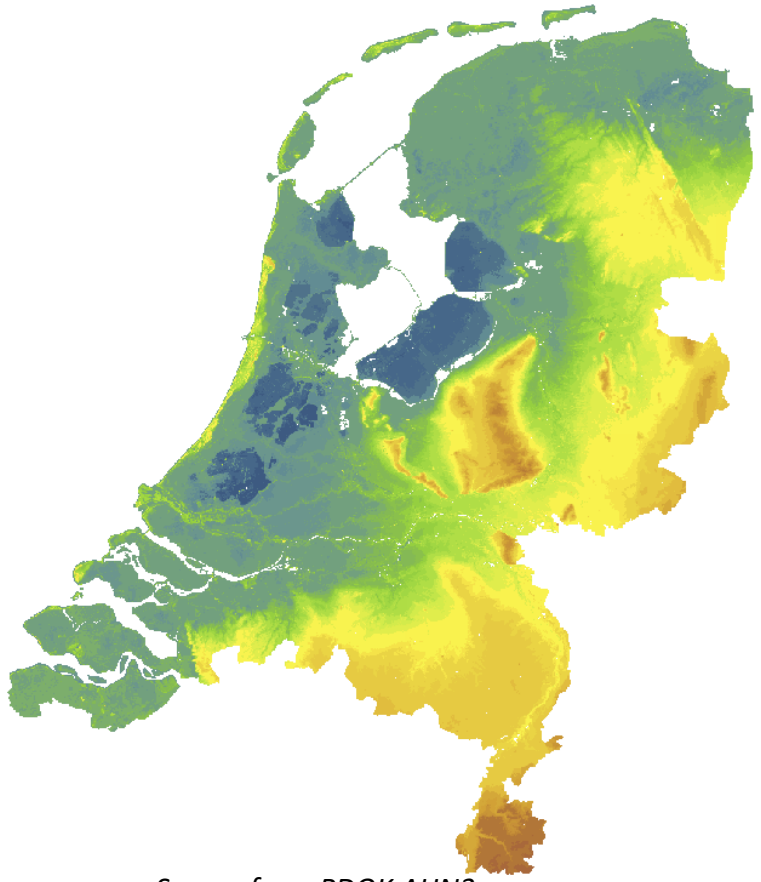
Using Foreign Data Wrapper in PostgreSQL to Expose Point Cloud on File System

Mutian Deng

Mentor #1: Martijn Meijers
Mentor #2: Peter van Oosterom



Source from PDOK AHN3

Outline

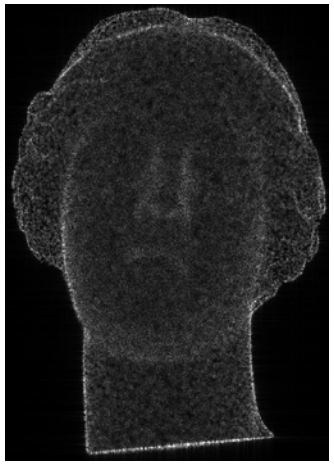
- Introduction
- Related work
- Methodology & Implementation
- Results & Analysis
- Conclusions & Future work

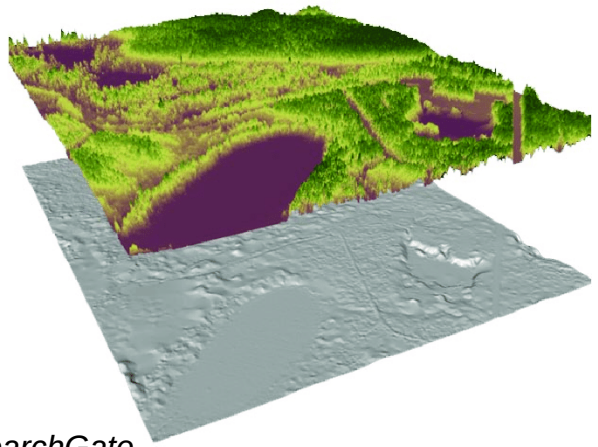
Outline

- Introduction
- Related work
- Methodology & Implementation
- Results & Analysis
- Conclusions & Future work

Point Cloud Data

- Data acquisition technologies
- Wide range of applications





Source from ResearchGate

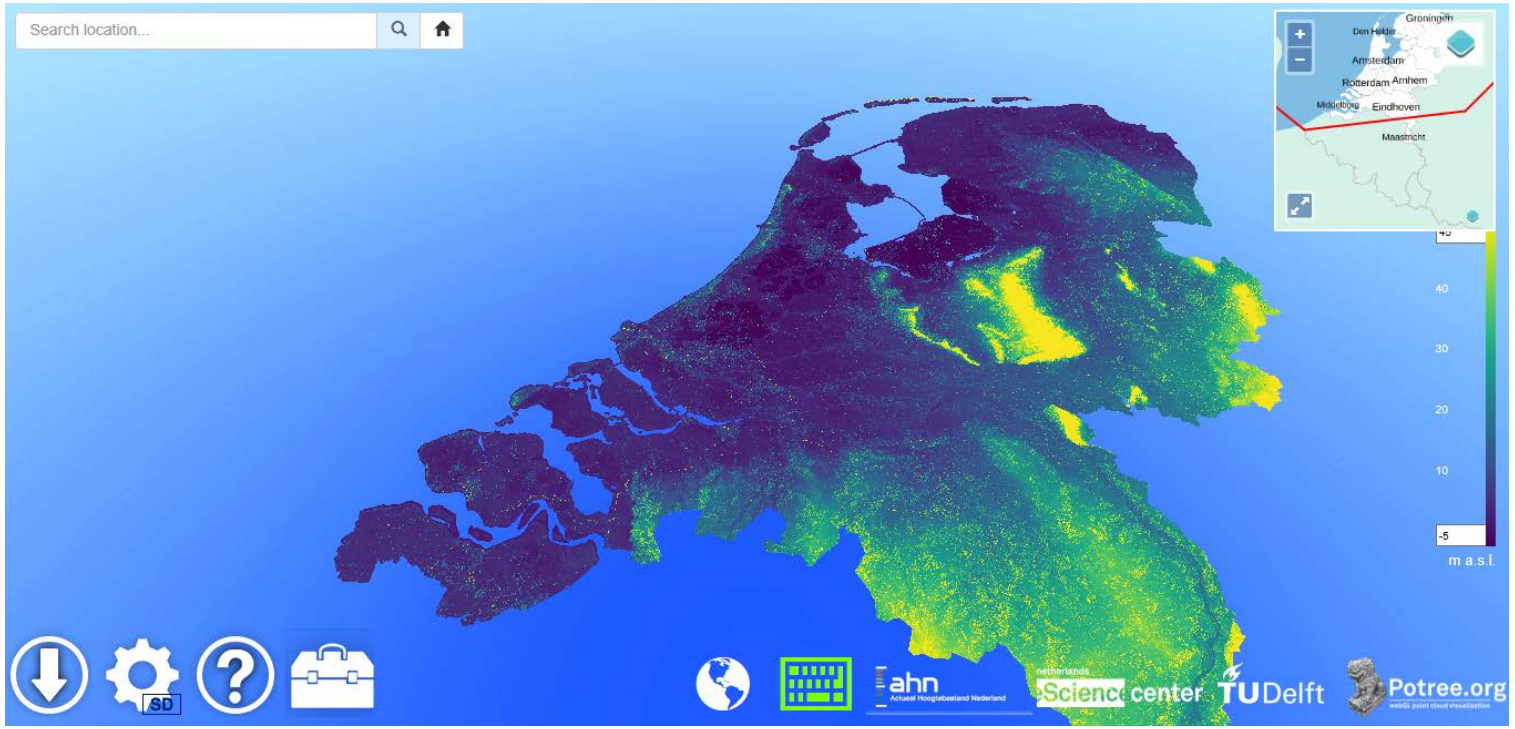
Source from wiki

Point Cloud Data Features

- Large number of points
- Multi-dimensions

Item	Format	Size	Required
X	long	4 bytes	*
Y	long	4 bytes	*
Z	long	4 bytes	*
Intensity	unsigned short	2 bytes	
Return Number	3 bits (bits 0 – 2)	3 bits	*
Number of Returns (given pulse)	3 bits (bits 3 – 5)	3 bits	*
Scan Direction Flag	1 bit (bit 6)	1 bit	*
Edge of Flight Line	1 bit (bit 7)	1 bit	*
Classification	unsigned char	1 byte	*
Scan Angle Rank (-90 to +90) – Left side	char	1 byte	*
User Data	unsigned char	1 byte	
Point Source ID	unsigned short	2 bytes	*

Source from LAS specification



Source from http://ahn2.pointclouds.nl

Problem statement

Efficient management solution of point clouds data is required for efficient use

- Storage
- Access
- Processing (like organization, manipulation)
- Query
- Combination with other GI
- Representation

Point clouds data management

- **File system solution**
  - Storage using file or collection of files in dedicated formats.
  - Read/Process/Write data using appropriate softwares.

+ Pros: 1. Efficient and standard storage
  2. Powerful processing tools
  3. Lossless compression algorithm
  4. Ease of use
- Cons: 1. Limited to the selection like accessing points in AoI.
  2. Not support data combination

Point clouds data management

- **DBMS solution**

  - Storing and Retrieving by database management system with extensions

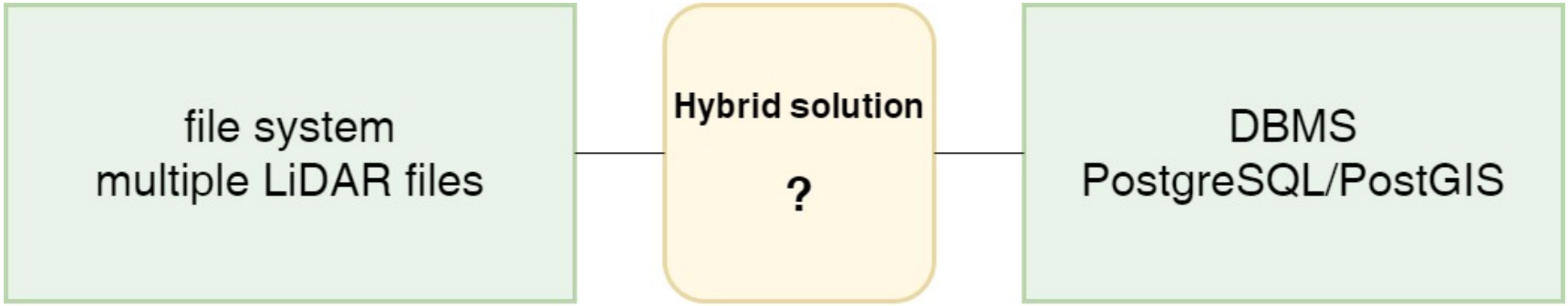
+ Pros: 1. SQL language for query
  2. combination with other GI
  3. update, insert and deletes
  4. multi-users access
- Cons: 1. Efficient storage model is still needed
  2. No fast data importing

Problem statement

Both file system method and DBMS method:

- Have its own advantages and disadvantage
- Satisfy the part of requirements point clouds data management

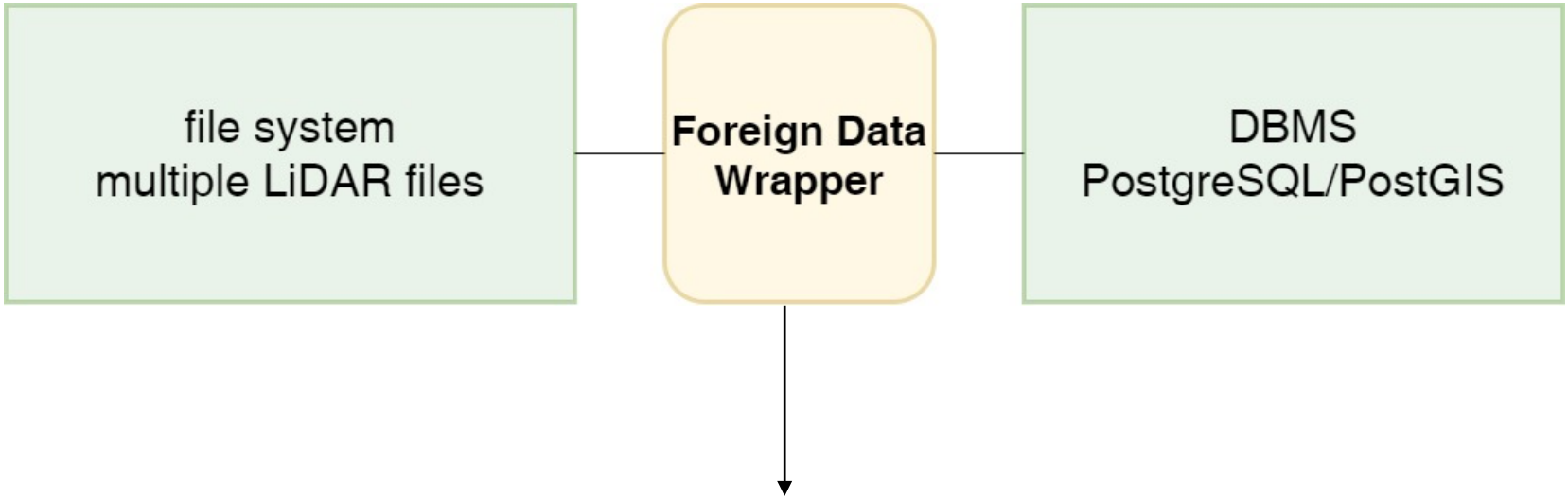
Hybrid solution?



- Store the LiDAR data on file system
- Expose and use LiDAR data on DBMS

Problem statement

- How to realize the Hybrid solution?
- By means of **Foreign Data Wrapper**



Library of PostgreSQL that can communicate with the external data source.

Research question

- Main research question and sub-questions

To what extent we can use LiDAR point clouds directly in the PostgreSQL by means of FDW?

- How does FDW build the connection between LiDAR file system and DBMS?
- What kinds of queries and what levels of selection can be executed upon LiDAR data?
- What sizes of AHN3 datasets can work well by FDW solution?
- What are the benefits and problems when using FDW method?

Outline

- Introduction
- Related work
- Methodology & Implementation
- Results & Analysis
- Conclusions & Future work

File system solution

File formats:

- LAS
- LAZ
- ASCII

File tools

- LAStools
- PDAL
- laspy

Database Management System solution (DBMS)

Databases:

- PostgreSQL (PostGIS)
- Oracle (SDO_PC_PKG)
- NoSQL

Storage model

- Flat table
- Blocks

Database Management System solution (DBMS)

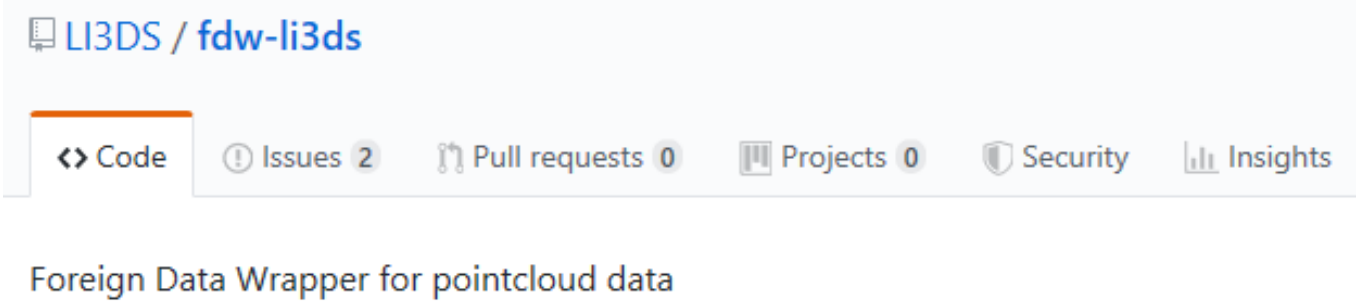
The extensions of PostgreSQL for point cloud data

- PgPointcloud
  Storing point cloud data in PostgreSQL
  two data type: PcPoint and PcPatch

- PostGIS
  Querying geographic objects in PostgreSQL.

Developed FDW for point cloud - LI3DS FDW

- 3 data types supported:
- Sbet
- ROSbag
- EchoPulse



Source from https://github.com/LI3DS/fdw-li3ds

- Not for LAS/LAZ point clouds
- Read-only
- One wrapper only reads one file
- Different formats use different wrappers

- Use Multicorn as implementation tool

Outline

- Introduction
- Related work
- Methodology & Implementation
- Results & Analysis
- Conclusions & Future work

Methodology & Implementation

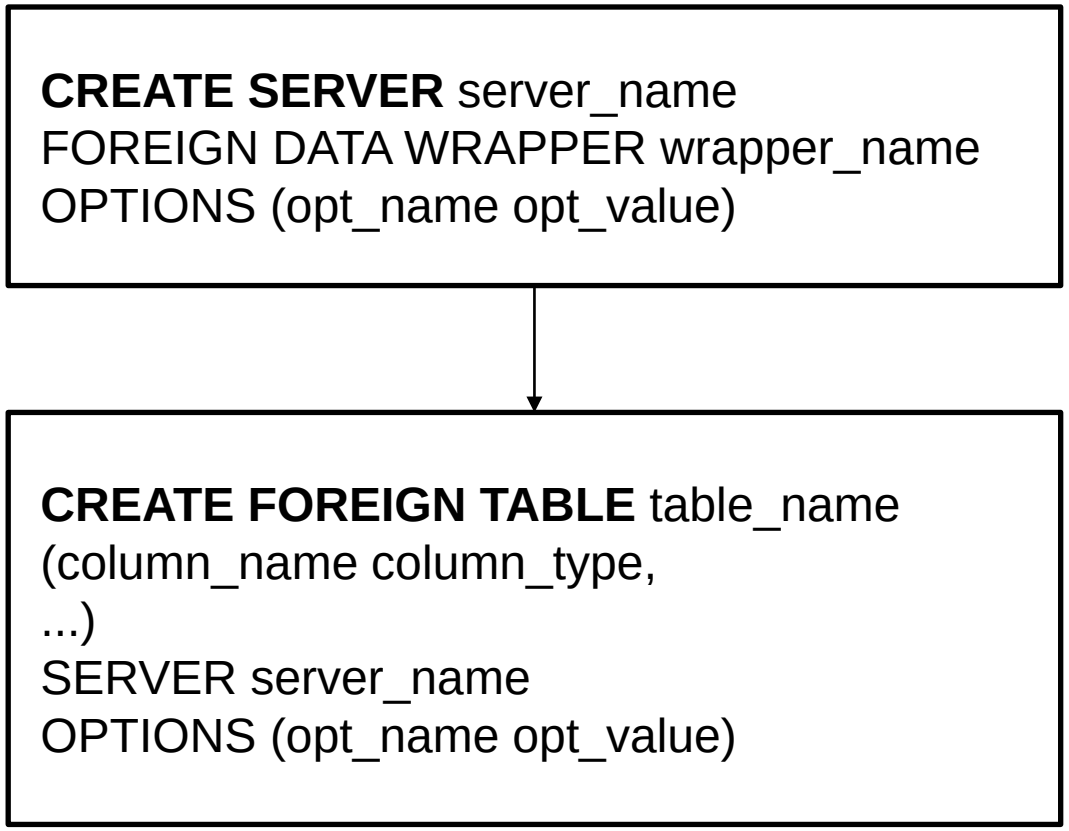
- Foreign Data Wrapper
- FDW for point clouds data
- System Architecture
- Benchmark

Foreign Data Wrapper

Use FDW

To access foreign data:

- Foreign server defines how to connect to a particular external data source.
- Foreign table defines the structure of remote data represent external data.



Foreign Data Wrapper

Write FDW

Implementation tool: Multicorn

https://multicorn.org/

Implementation tools: Multicorn

- FDW library for PostgreSQL
- Multicorn FDW is not different from others (same usage)
- Develop FDW in Python language

```
CREATE SERVER pc_server
FOREIGN DATA WRAPPER multicorn
OPTIONS (wrapper 'SystemFdw')
```

↓

```
CREATE FOREIGN TABLE pc_table
(x double precision,
y double precision,
z double precision
…)
SERVER pc_server
OPTIONS (filepath '…/pcfilesystem')
```

Foreign Data Wrapper

Implementation

interface: multicorn.ForeignDataWrapper

➢ **__init__()**

Args:

(from table creation)

- *options*
- *columns*

```
CREATE FOREIGN TABLE pc_table
(x double precision,
y double precision,
z double precision
…)
SERVER pc_server
OPTIONS (filepath '…/pcfilesystem')
```

➢ **execute()**

Args:

- *quals*: a list of Qual instances (PostgreSQL qualifiers)

```
WHERE x > 1
:
Qual.field:  x
Qual.operator:  >
Qual.value: 1
```

- *columns*

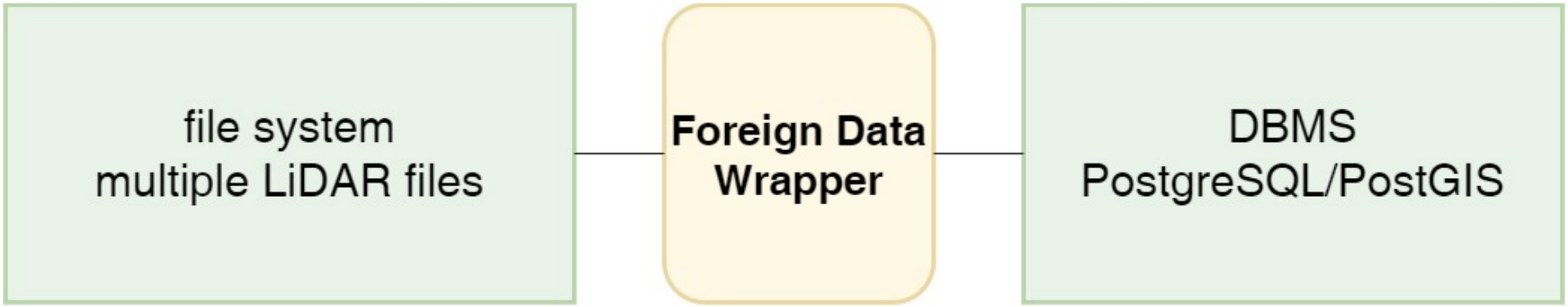
Returns: an iteration python objects

- Sequence
- Dictionary

Methodology & Implementation

- Foreign Data Wrapper
- FDW for point clouds data
- System Architecture
- Benchmark

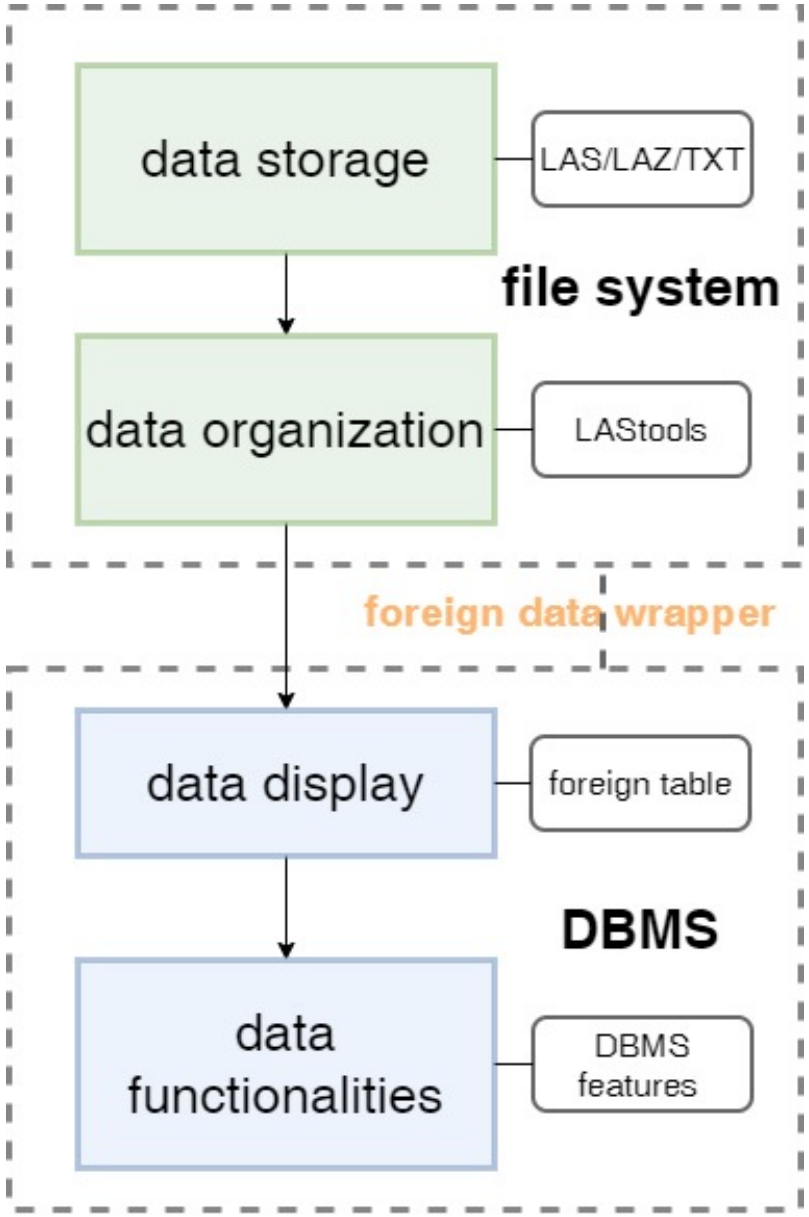
FDW for point clouds data



A foreign data wrapper for LiDAR data needs to realize that

- The LiDAR data can be stored in their original file format on file system and can be processed by powerful file I/O tools
- In the meantime, the point records of data can be exposed on DBMS and the data can be used by DBMS facilities

FDW for point clouds data



FDW for point clouds data

Data access

- File system side
- DBMS side

- Data storage
- Data organization
- Data obtain
- Data display

Data storage

- TXT
- LAS
- LAZ

Data organization

- lassort: sort the LAS/LAZ/TXT point clouds file into z-order (Morton code) arranged cells based on square quad tree

- lasindex: create an LAX file besides LAS/LAZ point clouds file, which contains spatial indexing information. with the presence of LAX, the spatial selection can be speed up

FDW for point clouds data

Data access

- File system side
- DBMS side

- Data storage
- Data organization
- Data obtain
- Data display

Data obtain

FDW

- The point records of the data are obtained and fetched by foreign data wrapper.
- “las2txt” is used to parse each point record.

Data display

Foreign table

x	double precision
y	double precision
z	double precision
classification	integer

FDW for point clouds data

Data functionalities

- **Data manipulation**
- Type conversion
- Querying

- Update
- Delete
- Insert

- Data can be updated, deleted, inserted synchronized both in file and foreign table
- This is realized in foreign data wrapper implementation

FDW for point clouds data

Data functionalities

- Data manipulation
- **Type conversion**
- Querying

X and Y coordinates in number type

↓

ST_SetSRID (ST_MakePoint (x , y) , 28992)

↓

PostGIS geometry Point type

After retrieving the data to foreign table, the data type can be cast for “Point-in-Polygon” query

FDW for point clouds data

Data functionalities

- Data manipulation
- Type conversion
- **Querying**

With the support of PostgreSQL and PostGIS, the possible queries on the foreign table representing the LiDAR data:

- - select the ground/ water/ building… points insides a region
- - select nearest neighbor of one location with a buffer
- - select all the points within a rectangle of different sizes
- - select all the points within a circle of different sizes
- - select all the points within a irregular polygon of different sizes
- - select the highest point of a region
- - select the maximal, minimal and average elevation value of points in a region
- - select the total point density and local point density of a region

Methodology & Implementation

- Foreign Data Wrapper
- FDW for point clouds data
- System Architecture
- Benchmark

System Architecture

Point Cloud Data Management System

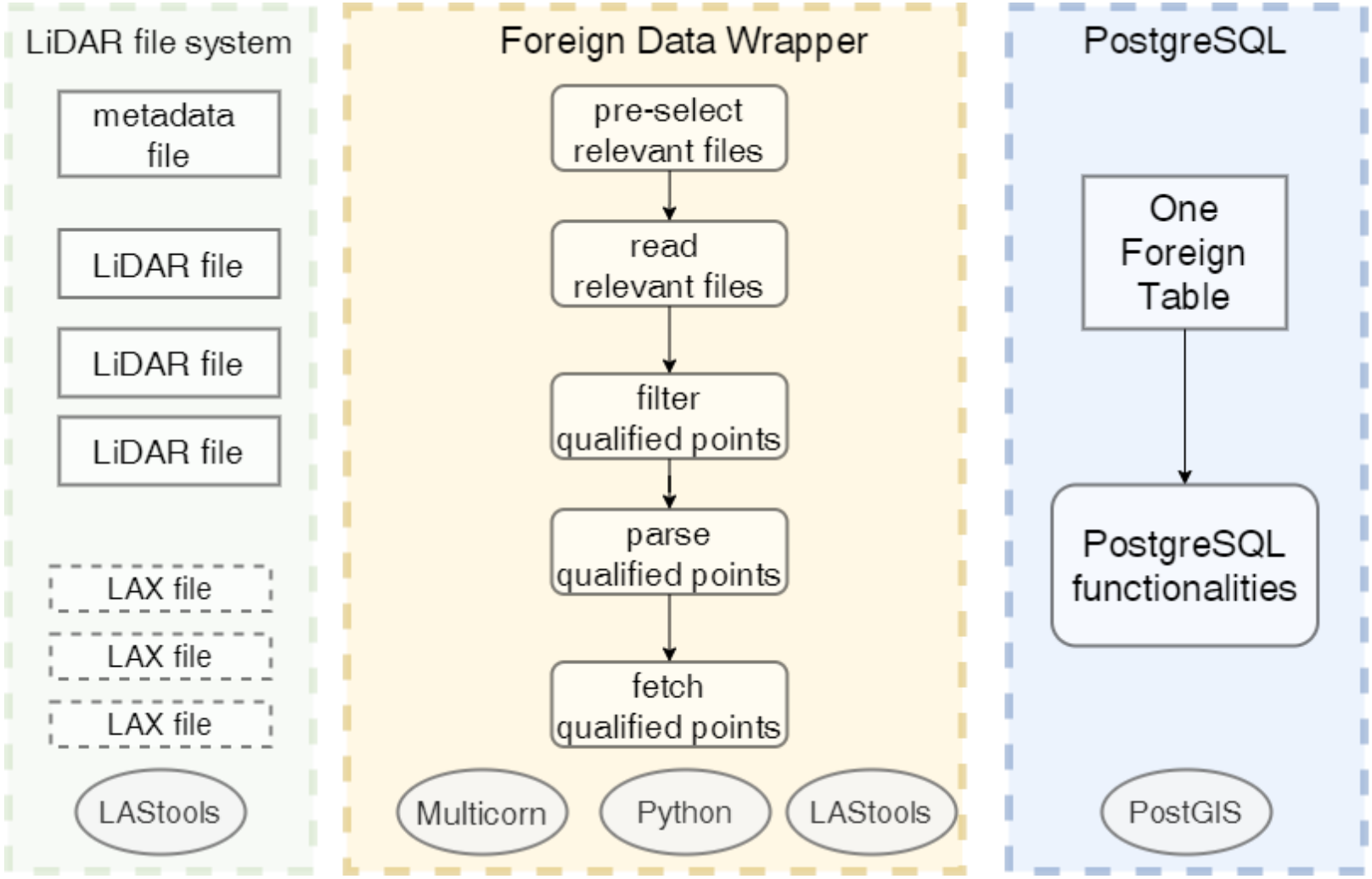
- **Motivation**
  - The point clouds dataset always store data in several distributed files on the file system, rather than only one LiDAR file due to its large volume, like AHN dataset.
  - These LiDAR files on one file system can have different originated formats and spatial extent.

- **FDW solution**
  - The core of FDW for Point Cloud Data Management System is handling multiple LiDAR files.

System Architecture

Point Cloud Data Management System

System components



System Architecture

Point Cloud Data Management System

System process

- System building

Multiple LiDAR files with different formats and extent are collected together.
A metadata file needs to be created and stored in same file system, besides these LiDAR files

id	filename	format	min_x	max_x	min_y	max_y
1	C_37EN1_0000048_0000213.laz	laz	79999.998	80833.351	443749.998	444791.597
2	C_37EN1_0000048_0000214.laz	laz	79999.998	80833.351	444791.598	446874.930
3	C_37EN1_0000048_0000215.laz	laz	79999.998	80833.351	446874.931	448958.263
4	C_37EN1_0000048_0000216.laz	laz	79999.998	80833.351	448958.264	450000.001
5	C_37EN1_0000049_0000213.laz	laz	80833.349	82500.018	443749.998	444791.597
6	C_37EN1_0000049_0000214.laz	laz	80833.348	82500.018	444791.598	446874.930
7	C_37EN1_0000049_0000215.laz	laz	80833.348	82500.018	446874.931	448958.263
8	C_37EN1_0000049_0000216.laz	laz	80833.348	82500.018	448958.264	450000.001
9	C_37EN1_0000050_0000213.laz	laz	82500.015	84166.685	443749.998	444791.597
10	C_37EN1_0000050_0000214.laz	laz	82500.015	84166.685	444791.598	446874.930
11	C_37EN1_0000050_0000215.laz	laz	82500.015	84166.685	446874.931	448958.263
12	C_37EN1_0000050_0000216.laz	laz	82500.015	84166.685	448958.264	450000.001
13	C_37EN1_0000051_0000213.laz	laz	84166.682	85000.001	443749.998	444791.597
14	C_37EN1_0000051_0000214.laz	laz	84166.682	85000.001	444791.598	446874.930
15	C_37EN1_0000051_0000215.laz	laz	84166.682	85000.001	446874.931	448958.263
16	C_37EN1_0000051_0000216.laz	laz	84166.682	85000.001	448958.264	450000.000

System Architecture

Point Cloud Data Management System

System process

- Data access

Since the core of the Point Cloud Data Management System is to handle multiple files

- One single foreign table needs to represent the data content fetched by one FDW from different LiDAR files (on the same file system), based on one selection condition.

System Architecture

Point Cloud Data Management System

System process

- Querying – Rectangle selection

1. Get selection condition box from quals
2. Read metadata file to pre-select the relevant files whose extent overlaps with selection box
3. Use appropriate reader to read relevant files one by one
4. Retrieve the LiDAR file and only parse the points inside the selection box
5. Fetch the qualified points to PostgreSQL

System Architecture

Point Cloud Data Management System

System process

- Querying – polygon selection

➢ 1. Get selection condition box from quals

➢ 2. Read metadata file to pre-select the relevant files whose extent overlaps with selection box

➢ 3. Use appropriate reader to read relevant files one by one

➢ 4. Retrieve the LiDAR file and only parse the points inside the selection box

➢ 5. Fetch the qualified points to PostgreSQL

➢ 6. PostGIS selects the points that are exactly inside the polygon

Methodology & Implementation

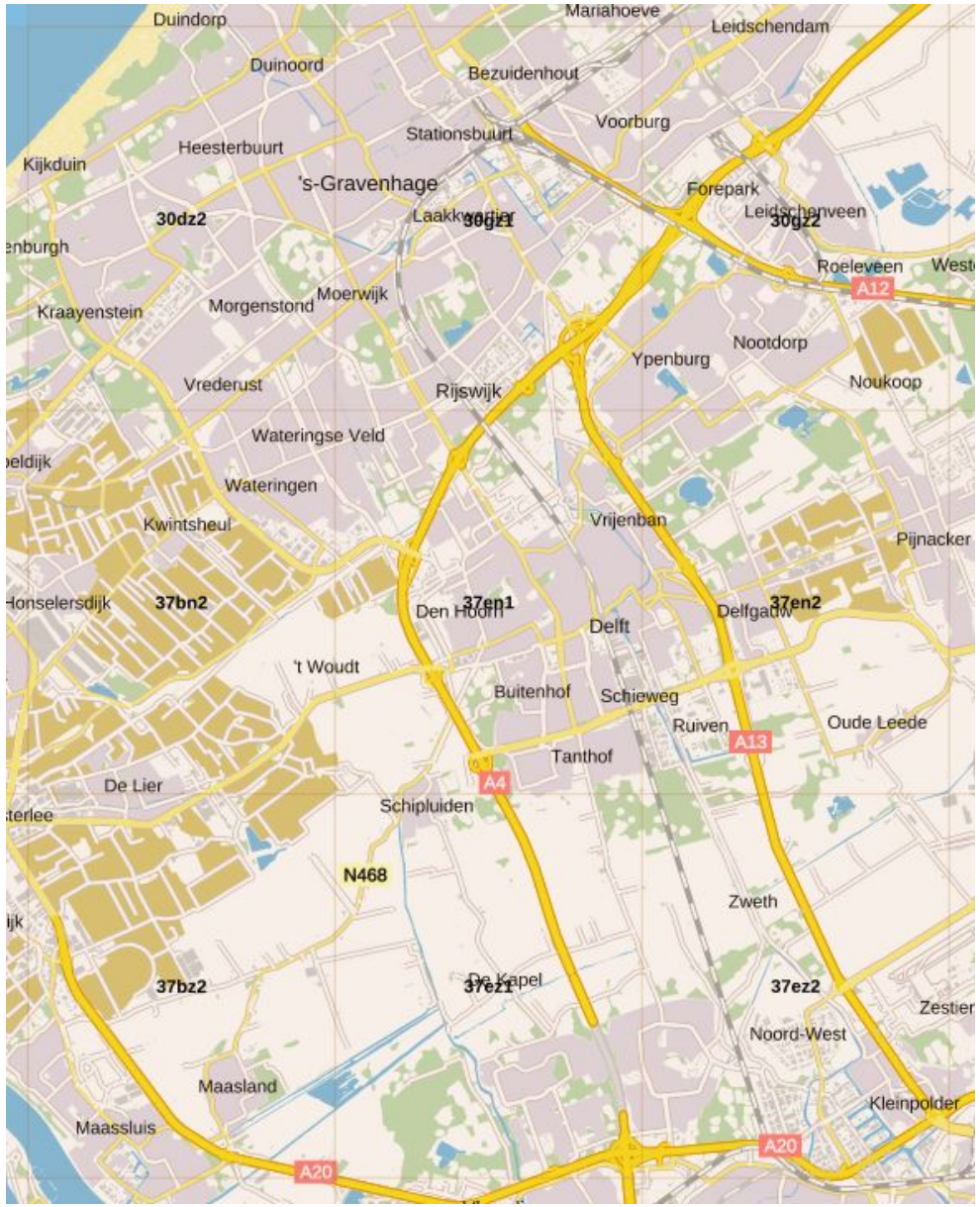
- Foreign Data Wrapper
- FDW for point clouds data
- System Architecture
- Benchmark

Benchmark

Dataset: AHN3

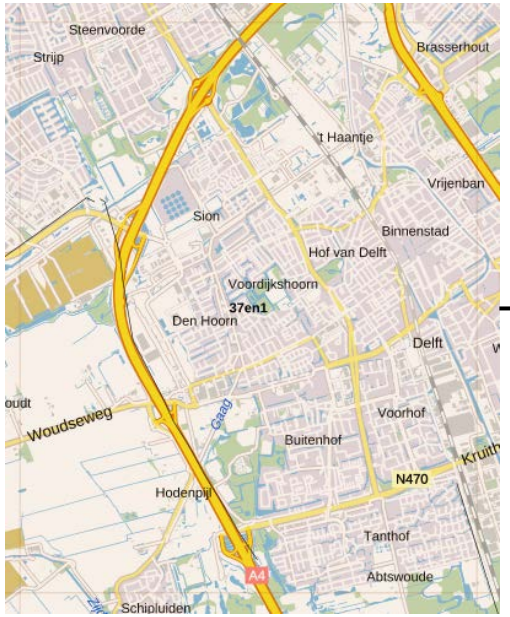
Test aspects:

- Feasibility
- Efficiency
- Scalability



Benchmark

- Test datasets



lassplit

48.216	49.216	50.216	51.216
48.215	49.215	50.215	51.215
48.214	49.214	50.215	51.214
48.213	49.213	50.213	51.213

id	filename	format	min_x	max_x	min_y	max_y
1	C_37EN1_0000048_0000213.laz	laz	79999.998	80833.351	443749.998	444791.597
2	C_37EN1_0000048_0000214.laz	laz	79999.998	80833.351	444791.598	446874.930
3	C_37EN1_0000048_0000215.laz	laz	79999.998	80833.351	446874.931	448958.263
4	C_37EN1_0000048_0000216.laz	laz	79999.998	80833.351	448958.264	450000.001
5	C_37EN1_0000049_0000213.laz	laz	80833.349	82500.018	443749.998	444791.597
6	C_37EN1_0000049_0000214.laz	laz	80833.348	82500.018	444791.598	446874.930
7	C_37EN1_0000049_0000215.laz	laz	80833.348	82500.018	446874.931	448958.263
8	C_37EN1_0000049_0000216.laz	laz	80833.348	82500.018	448958.264	450000.001
9	C_37EN1_0000050_0000213.laz	laz	82500.015	84166.685	443749.998	444791.597
10	C_37EN1_0000050_0000214.laz	laz	82500.015	84166.685	444791.598	446874.930
11	C_37EN1_0000050_0000215.laz	laz	82500.015	84166.685	446874.931	448958.263
12	C_37EN1_0000050_0000216.laz	laz	82500.015	84166.685	448958.264	450000.001
13	C_37EN1_0000051_0000213.laz	laz	84166.682	85000.001	443749.998	444791.597
14	C_37EN1_0000051_0000214.laz	laz	84166.682	85000.001	444791.598	446874.930
15	C_37EN1_0000051_0000215.laz	laz	84166.682	85000.001	446874.931	448958.263
16	C_37EN1_0000051_0000216.laz	laz	84166.682	85000.001	448958.264	450000.000

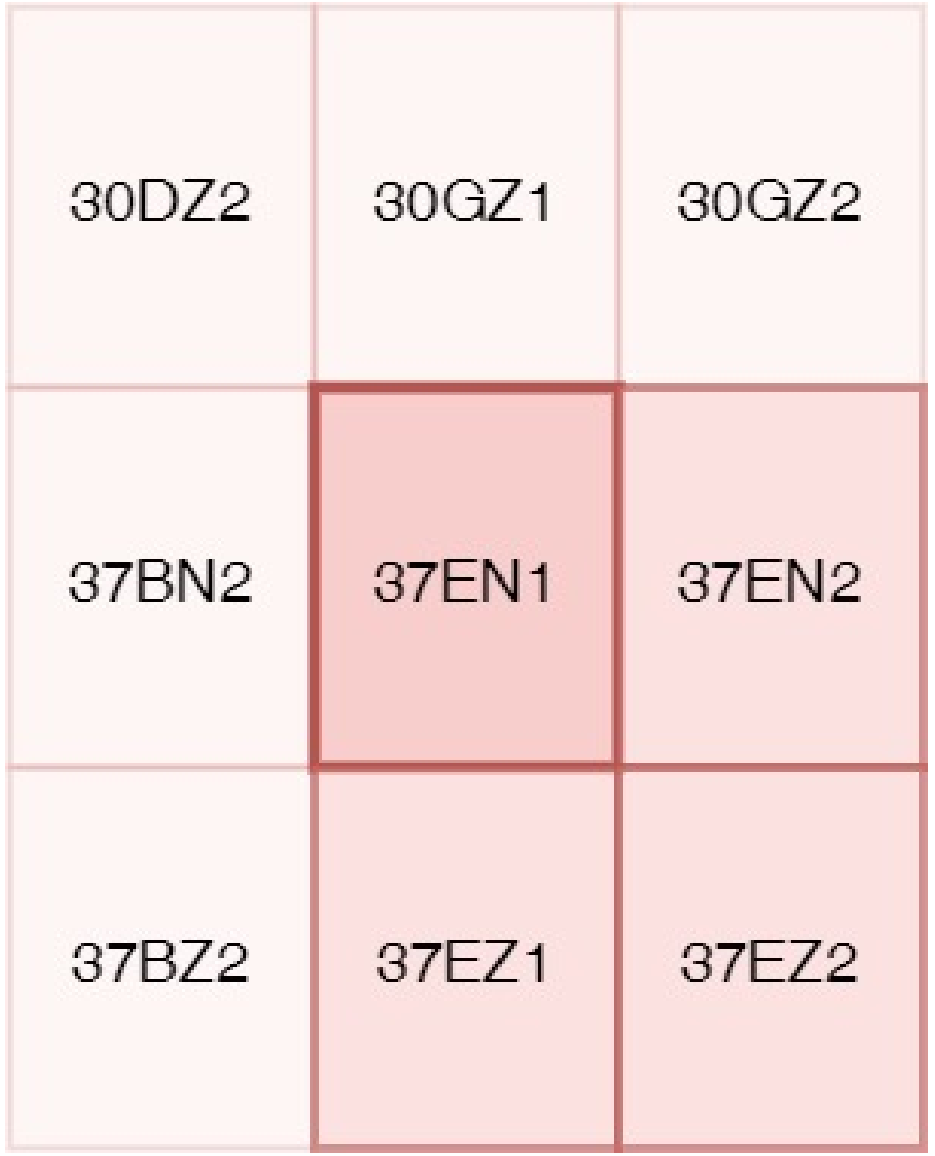
- Use ‘lassplit’ to split one AHN file into 16 files based on x and y interval
- The output files are named in x and y order

Benchmark

- Dataset scale

Data scale	AHN3 files	Test files
Small	1	(1*16) **16**
Medium	4	(4*16) **64**
Large	9	(9*16) **144**

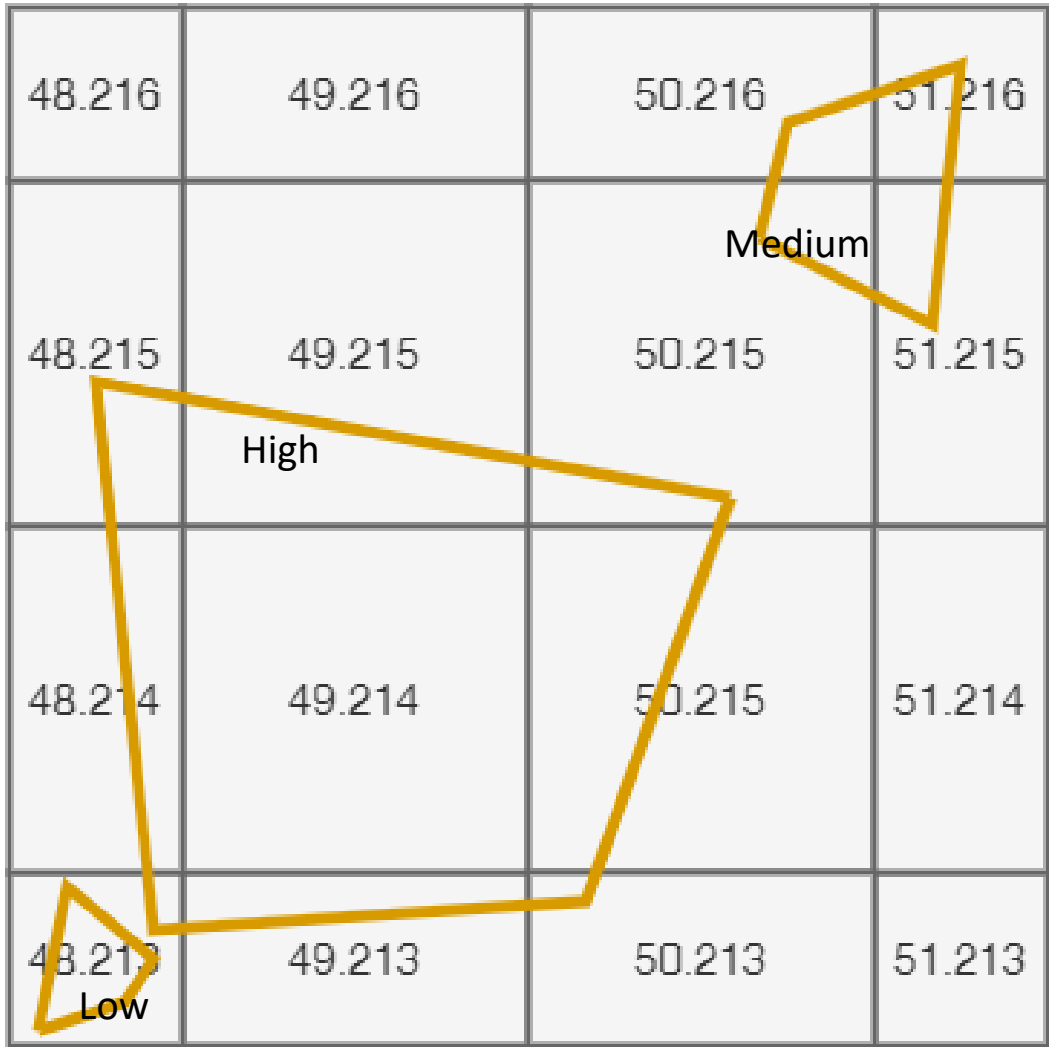
One AHN file has 16 test files



Benchmark

- Query level

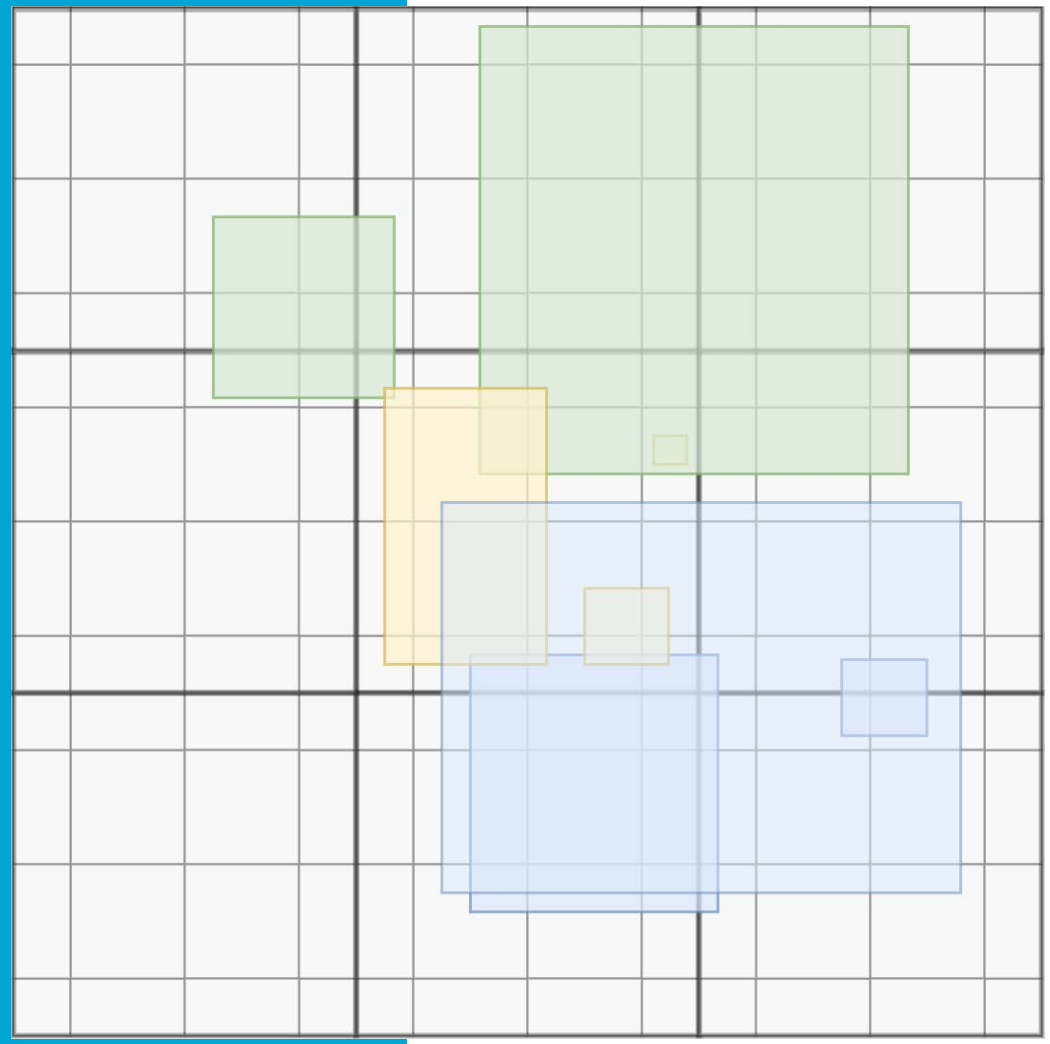
Query level	overlap files in each AHN file
Low	1
Medium	4
High	9



Small scale: One AHN file has 16 test files

Benchmark

- **Scalability**
- Different level queries on different scale of datasets



Data scale	Query level	Relevant files
Small (1 AHN)	low	1
	medium	4
	high	9
Medium (4 AHN)	low	4
	medium	16
	high	36
Large (9 AHN)	low	9
	medium	36

Benchmark

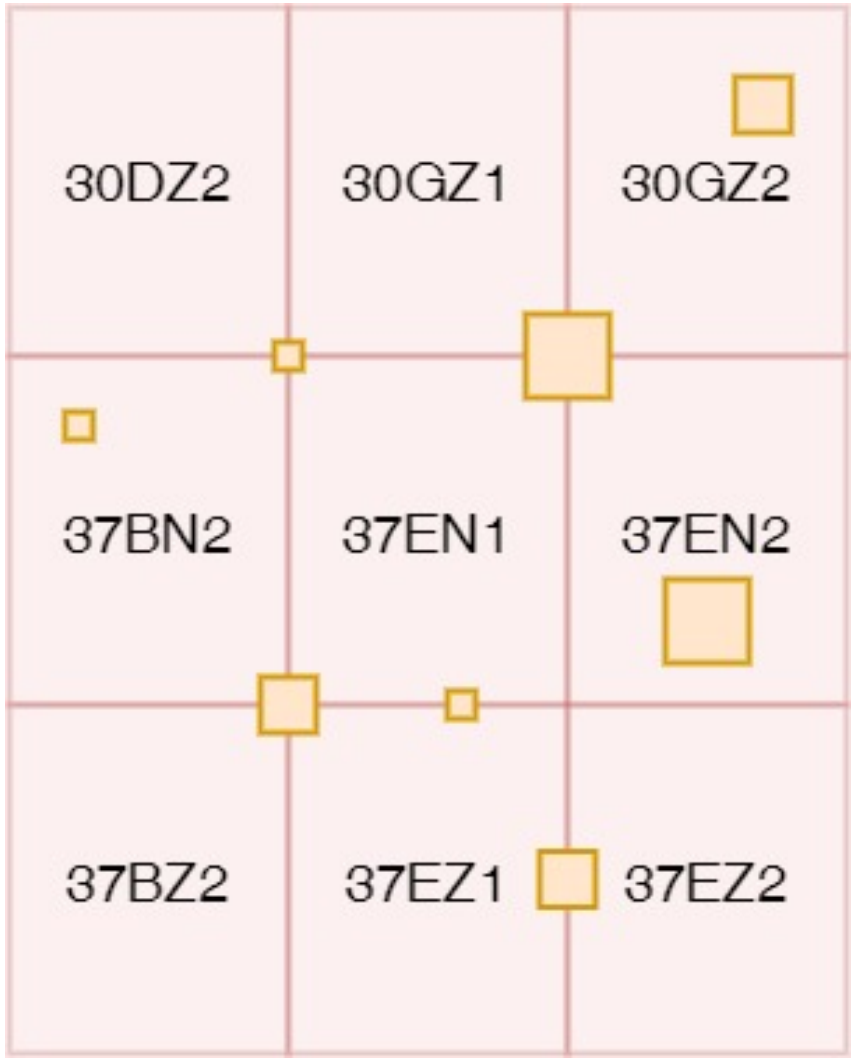
- **Efficiency**
- Mini level queries on small dataset

- The query region only overlaps with one file
- Area of query region is really tiny

Small scale: One AHN file has 16 test files

Benchmark

- **Feasibility**
- Mini level queries on large dataset



Area	Relevant files
100	1
	2
	4
400	1
	2
	4
900	1
	2
	4

- The query region only overlaps with 1, 2, 4 files
- Area of query region is really tiny
- Like querying a house region on the city dataset

Outline

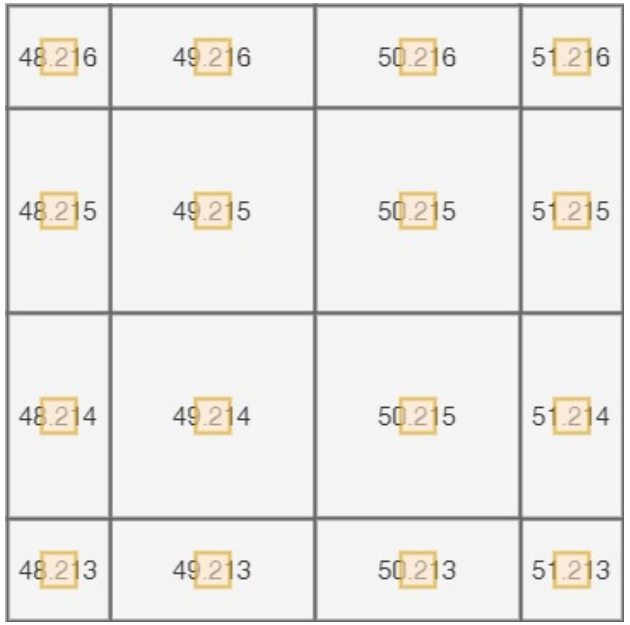
- Introduction
- Related work
- Methodology & Implementation
- Results & Analysis
- Conclusions & Future work

Efficient test results

data organization

Mini level queries on small scale dataset			
system		After organization	
Time(s)	count	Time(s)	count
4	124	0.08	124
8	238	0.09	238
11	165	0.11	166
5	114	0.13	114
9	116	0.07	116
17	253	0.14	252
18	177	0.12	177
11	128	0.13	128
10	140	0.15	140
26	270	0.16	270
26	179	0.05	179
15	88	0.07	87
6	122	0.08	122
11	1022	0.11	1022
15	219	0.14	219
5	129	0.14	129

Query region is tiny compared to the extent of relevant files



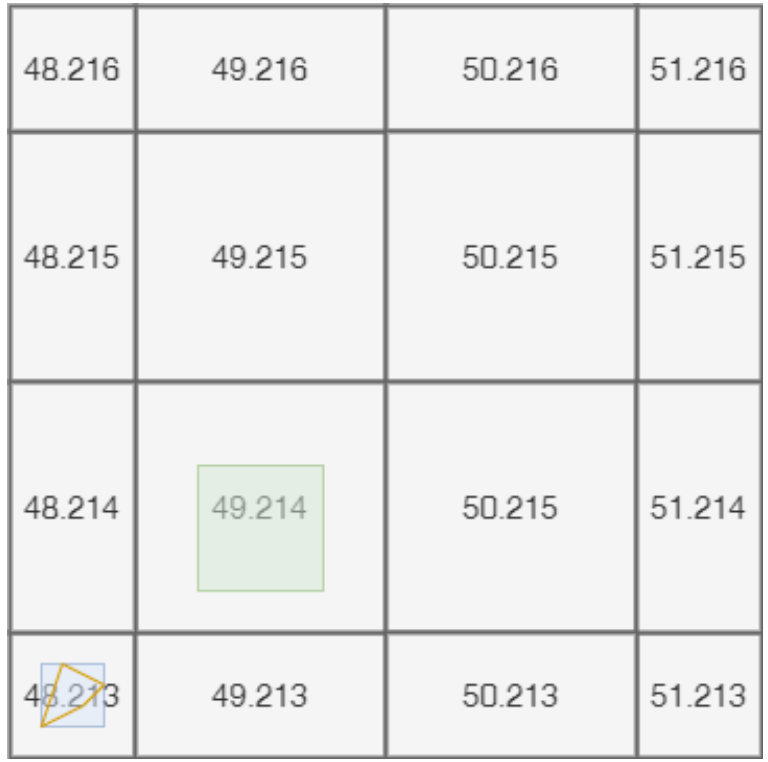
- Time greatly reduced
- Error imported

Efficient test results

data organization

Mini level queries ⟹ small level queries
: Area of query region get larger

		low level queries on small scale dataset			
		system		After organization	
		Time(s)	count	Time(s)	count
all	bbox	55	4724593	49	4724588
	polygon	63	2263124	61	2263118
	rectangle	150	13000976	138	10537689
ground	bbox	58	3841360	54	3841359
	polygon	65	2263124	64	2263118
	rectangle	162	10537701	152	10537689



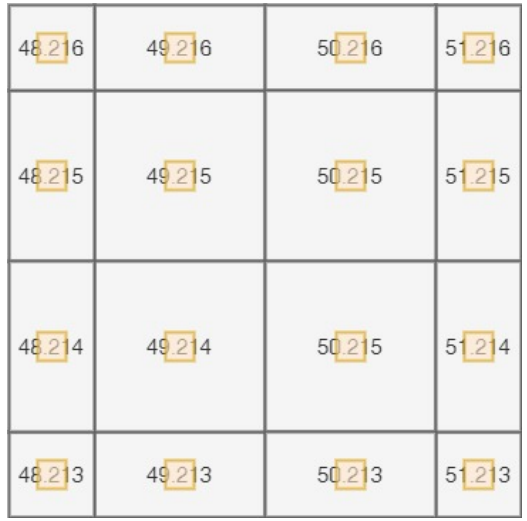
Query region is of half area to the extent of relevant file

- Time difference smaller
- Error larger

Efficient test results

Multicorn qualifiers

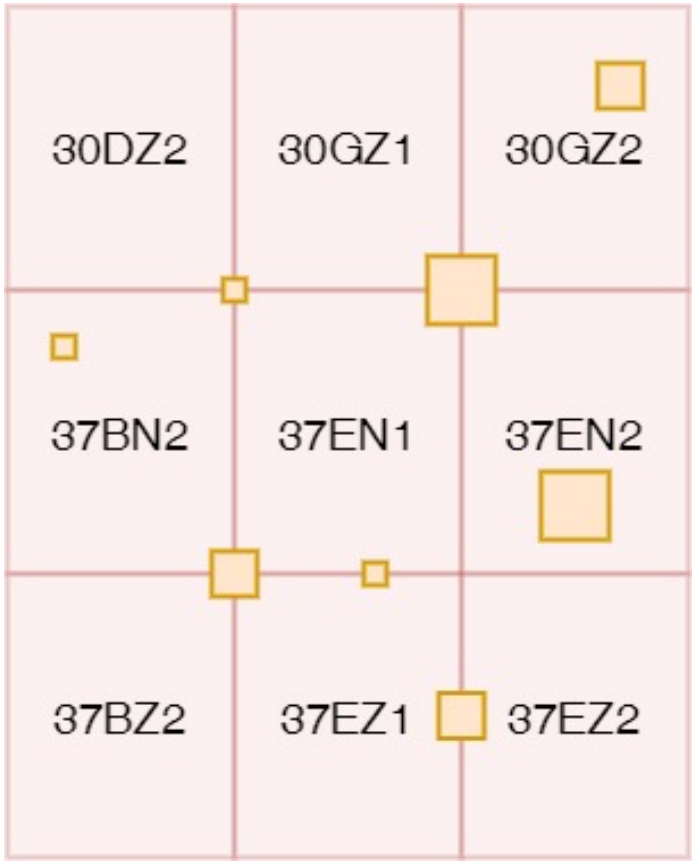
	Mini level queries on small scale dataset				
	Without quals		Using quals		Count /million
	Time(s)	count	Time(s)	count	
48.213	113	124	4	124	11
48.214	221	238	8	238	21
48.215	284	165	11	165	26
48.216	127	114	5	114	12
49.213	241	116	9	116	22
49.214	526	253	17	253	48
49.215	521	177	18	177	49
49.216	302	128	11	128	28
50.213	266	140	10	140	25
50.214	772	270	26	270	66
50.215	713	179	26	179	66
50.216	426	88	15	88	40
51.213	170	122	6	122	15
51.214	302	1022	11	1022	29
51.215	427	219	15	219	37
51.216	130	129	5	129	12



- Using quals: read and fetch the points satisfying the condition
- Without quals: read and fetch all the points in the relevant files

Feasibility test results

relevant	large-mini-bounding box					large-mini-polygon				
	area	system		organization		area	system		organization	
		Time (s)	count	Time (s)	count		Time (s)	count	Time (s)	count
1	100	27	2336	0.48	2326	54.5	27	1279	0.47	1279
2	100	17	1522	0.26	1522	52	17	774	0.25	774
4	100	20	58	0.34	58	45.5	20	1	0.31	1
1	400	17	11894	0.50	11891	220	16	7211	0.53	7211
2	400	23	4101	0.33	4101	230	22	2524	0.33	2524
4	400	17	3801	0.27	3801	186	18	1771	0.29	1771
1	900	23	37780	0.74	37784	458	24	20927	0.85	20930
2	900	44	29478	0.87	29478	318	43	10475	0.93	10475
4	900	18	8383	0.34	8384	441	19	4299	0.36	4300



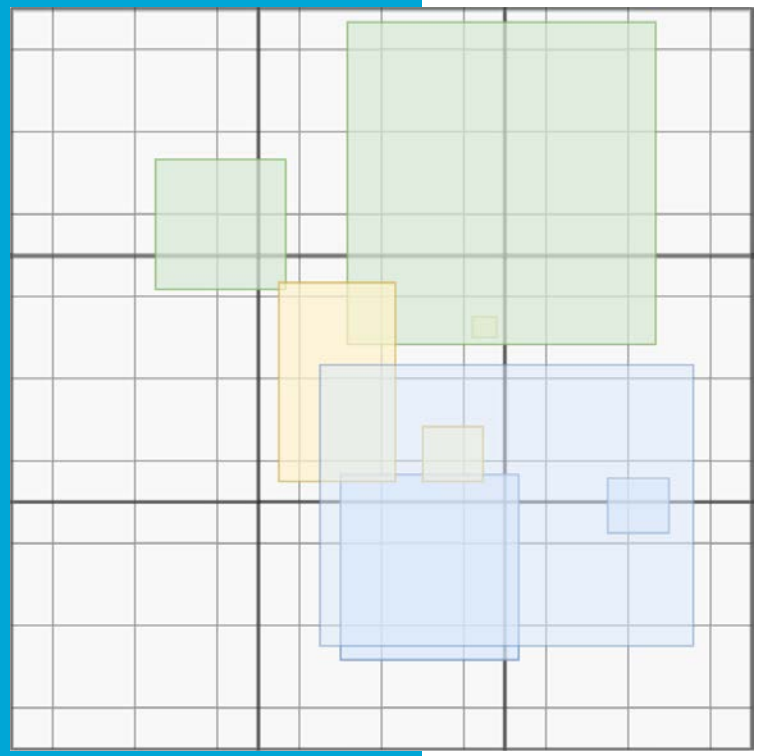
- Time is remarkably reduced by the data organization, when region is tiny.

Feasibility test results

relevant	large-mini-bbx					large-mini-polygon				
	area	system		organization		area	system		organization	
		time(s)	count	time(s)	count		time(s)	count	time(s)	count
1	100	27	2336	0.48	2326	54.5	27	1279	0.47	1279
2	100	17	1522	0.26	1522	52	17	774	0.25	774
4	100	20	58	0.34	58	45.5	20	1	0.31	1
1	400	17	11894	0.50	11891	220	16	7211	0.53	7211
2	400	23	4101	0.33	4101	230	22	2524	0.33	2524
4	400	17	3801	0.27	3801	186	18	1771	0.29	1771
1	900	23	37780	0.74	37784	458	24	20927	0.85	20930
2	900	44	29478	0.87	29478	318	43	10475	0.93	10475
4	900	18	8383	0.34	8384	441	19	4299	0.36	4300

- Query is done in several seconds, not all the files in the underlying file system need to be read, instead of only overlapping files, because of the pre-selection of relevant files.
- Although several huge point clouds files are relevant, only part of the points (not all the points) are required to be read, because of the indexing and qualifiers.

Scalability test results



data scale	query level	relevant	bbx				Time/count	polygon				Time/count
			Area (km2)	Count (million)	Time (s)	Time (min)	(s/million)	Area (km2)	Count (million)	Time (s)	Time (min)	(s/million)
small	low	1	0.4	5	55	1	11.7	0.2	2	63	1	27.8
	medium	4	3.8	76	952	16	12.5	2.0	42	1164	19	27.9
	high	9	3.5	49	718	12	14.7	3.5	49	891	15	18.3
medium	low	4	1.2	30	383	6	12.9	0.6	17	458	8	27.5
	medium	16	8.5	137	1832	31	13.4	7.6	122	2224	37	18.3
	high	36	31.5	502	6399	107	12.8	31.4	501	8197	137	16.4
large	low	9	13.0	203	2490	42	12.3	8.3	132	3145	52	23.9
	medium	36	31.4	653	8108	135	12.4	31.3	653	11265	188	17.3

- Time for each polygon selection is more than the time for its bounding box .
- The speed is represented as “how many seconds needed for 1 million returned points”. The speed of bounding box selection is fast than polygon.

Scalability test results

data scale	query level	relevant	bbx				polygon			
			Area (km2)	Count (million)	Time (s)	Time (min)	Area (km2)	Count (million)	Time (s)	Time (min)
small	low	1	0.4	5	55	1	0.2	2	63	1
	medium	4	3.8	**76**	952	**16**	2.0	**42**	1164	**19**
	high	9	3.5	**49**	718	**12**	3.5	**49**	891	**15**
medium	low	4	1.2	30	383	6	0.6	17	458	8
	medium	16	8.5	137	1832	31	7.6	122	2224	37
	high	36	31.5	502	6399	107	31.4	501	8197	137
large	low	9	13.0	203	2490	42	8.3	132	3145	52
	medium	36	31.4	653	8108	135	31.3	653	11265	188

- In the small scale data test, the high level query overlaps with more files than medium level query, but it costs less time, it is because its query region bounding box count is smaller, no relevance to the query level (number of overlapping files)

Scalability test results

data scale	query level	relevant	bbx				polygon			
			Area /km2	Count /million	time/s	Time /min	Area /km2	Count /million	time/s	Time /min
small	low	1	0.4	5	55	1	0.2	2	63	1
	medium	**4**	3.8	**76**	952	**16**	2.0	**42**	1164	**19**
	high	9	3.5	49	718	12	3.5	49	891	15
medium	low	**4**	1.2	**30**	383	**6**	0.6	**17**	458	**8**
	medium	16	8.5	137	1832	31	7.6	122	2224	37
	high	36	31.5	502	6399	107	31.4	501	8197	137
large	low	9	13.0	203	2490	42	8.3	132	3145	52
	medium	36	31.4	653	8108	135	31.3	653	11265	188

- In the test of 4 relevant files, the query on the larger scale data costs less time than query on the small scale data, it is because the bound box count of query region is smaller, no relevance to the data scale

Scalability test results

(time polygon selection – time bbx selection)

data scale	query level	relevant	bbx				polygon				
			Area (km2)	Count (million)	Time (s)	Time (min)	Area (km2)	Count (million)	Time (s)	Time (min)	Time PostGIS (s)
small	low	1	0.4	5	55	1	0.2	2	63	1	8
	medium	4	3.8	76	952	16	2.0	42	1164	19	212
	high	9	3.5	49	718	12	3.5	49	891	15	173
medium	low	4	1.2	30	383	6	0.6	17	458	8	75
	medium	16	8.5	137	1832	31	7.6	122	2224	37	392
	high	36	31.5	502	6399	107	31.4	501	8197	137	1798
large	low	9	13.0	203	2490	42	8.3	132	3145	52	655
	medium	36	31.4	653	8108	135	31.3	653	11265	188	3157

- Time difference between the polygon selection and bounding box selection(time polygon selection – time bbx selection) is the time cost by PostGIS to do the query on the PostgreSQL foreign table.
- This time is also relevant to returned points count.

Outline

- Introduction
- Related work
- Methodology & Implementation
- Results & Analysis
- Conclusions & Future work

Conclusions

- Answer to the main research question

To what extent we can use LiDAR point clouds directly in the PostgreSQL by means of Foreign Data Wrapper?

- Multiple LiDAR point clouds are stored on file system, with a metadata file.
- By means of foreign data wrapper, they can be exposed and queried on foreign table in the PostgreSQL / PostGIS. (foreign table can be queried same as local table and join with local table)
- Spatial selection, attributes selection, data manipulation, aggregate functions are possible.

Conclusions

- Answer to the sub research questions

- **How does FDW build the connection between LiDAR file system and DBMS?**
- What kinds of queries and what levels of selection can be executed upon LiDAR data ?
- What sizes of AHN datasets can work well by FDW solution?
- What are the benefits and problems when using FDW method?

The connection information is based on the file path of the LiDAR file system, besides the LiDAR files there is a metadata file storing header information including filename, spatial extent, file format, etc. Therefore, the data from the LiDAR files on this file system can be exposed on PostgreSQL.

Conclusions

- Answer to the sub research questions

- How does FDW build the connection between LiDAR file system and DBMS?
- **What kinds of queries and what levels of selection can be executed upon LiDAR data?**
- What sizes of AHN datasets can work well by FDW solution?
- What are the benefits and problems when using FDW method?

The operations includes data manipulation; spatial selection of different query region like rectangle, irregular polygons and circle; attributes selection; aggregate function like maximal, minimal height can be executed.

In the scalability results, different sizes of tested regions can be queried on LiDAR data, time is relevant to the returned points.

Conclusions

- Answer to the sub research questions

- How does FDW build the connection between LiDAR file system and DBMS?
- What kinds of queries and what levels of selection can be executed upon LiDAR data?
- **What sizes of AHN datasets can work well by FDW solution?**
- What are the benefits and problems when using FDW method?

Different scales of tested datasets can work well by FDW solution, if the storage space for the LiDAR file system is enough.

Conclusions

- Answer to the sub research questions

➢ How does FDW build the connection between LiDAR file system and DBMS?
➢ What kinds of queries and what levels of selection can be executed upon LiDAR data ?
➢ What sizes of AHN datasets can work well by FDW solution?
➢ **What are the benefits and problems when using FDW method?**

- The benefit of this solution is there is no need to load LiDAR data from files into PostgreSQL and take local and storage, they can be used by DBMS features like query. Therefore, it is efficient to execute the query like selecting the points inside a building on the datasets of whole province.
- The problems can be it takes time to register header information into metadata file and retrieve the metadata file to search for relevant files.

Future work

- Metadata management
- Data display
- Data organization
- Comparison

➢ Database management system can be a good alternative for header information, because one LiDAR file system can have a large number of different files, and the header information of all these files need to be retrieved during the process of FDW, while the index can be built on the file extent column of metadata table to save the retrieving time.

Future work

- Metadata management
- Data display
- Data organization
- Comparison

➢ When fetching and representing the content (point records) on the PostgreSQL foreign tables, it can be a more efficient way to use the data type **PcPatch** of PgPointcloud to organize and display the content of LiDAR data.

Future work

- Metadata management
- Data display
- Data organization
- Comparison

➢ Data organization is proved to be useful, more spatial access methods and underlying parameters can be researched to improve the feature of the foreign data wrapper solution.

Future work

- Metadata management
- Data display
- Data organization
- Comparison

➢ The hybrid solution of foreign data wrapper can be compared to the both solutions:

- File system solution: use las2las to filter and query the points inside an area of interest.
- DBMS solution: use PgPointcloud to store the LiDAR data in PostgreSQL

Thanks for attention!