Storing and querying nD-PointClouds in Apache Parquet

dr.ir. Martijn Meijers

GIS-technology, Digital Technologies, Architectural Engineering + Technology,
Faculty of Architecture and the Built Environment, Delft University of Technology

nD-PointCloud consortium meeting
Monday, October 30, 2023
13:00–17:15 CET
Faculty of Architecture and the Built Environment, Room F

Previous nD PC presentation (jan ‘23)

- Load point clouds in database + web proxy
  - Huge amount of disk space:
    Row-oriented storage of PostgreSQL
  - Mapping to Octree takes (quite some) time → Directer use of Space Filling Curve based organization?
  - Not easy to create database on Delft Blue HPC → Switch to files?

Suitable file format?

- Evaluated some column stores:
  MonetDB, Citus, Clickhouse

  → Storage: Nicely compressed: Column storage!
- Other requirements
  - Well-supported in diverse programming languages (→ Produce on HPC)
  - Still possible to load as external table in database
    (combine with other vector / raster data)
  - Read (in parts) over `http(s)` directly (‘cloud native’)

Apache Parquet

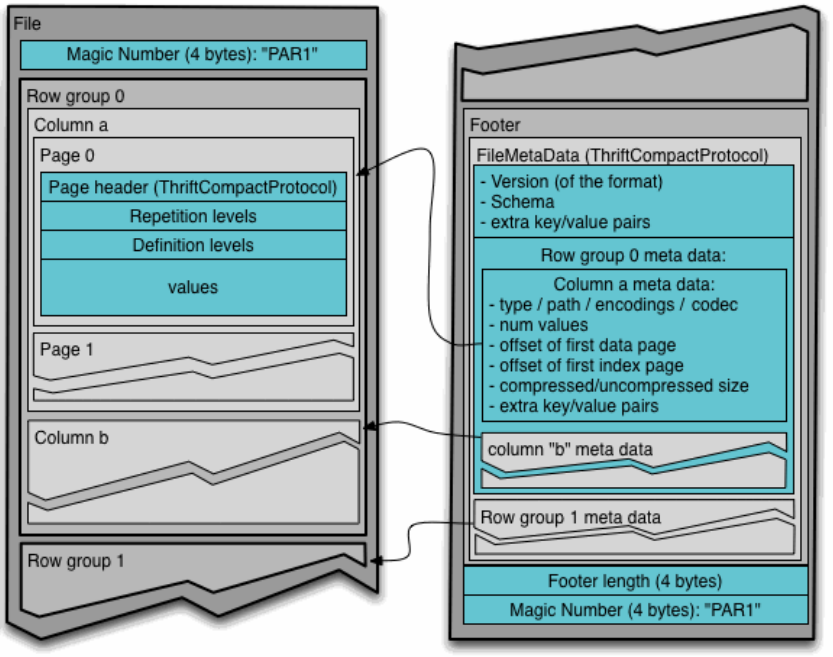
- Columnar store format, for serialized data (*disk storage*)
- Structured (+ nested) data, compressed
- Has a main memory cousin (uncompressed columnar)
  - Apache Arrow → Data frame libraries can read Parquet from disk into Arrow (2D, heterogeneous tabular data)
- You can interact via SQL with the Parquet files using query engine (DuckDb, Apache Drill)
- Note, we do *not* use GeoParquet
  (Recent OGC development to support Simple Feature Specification [SFS] Geometry as column type inside Parquet)

nD Point Cloud

- nD Points become location on Space Filling Curve, stored as SFC key
  - SFC key contains the *"geometry"* for multiple dimensions (x,y [, z, t, cloi]) – *not* SFS
- Use SFC key to both store the *primary* dimensions and retrieve the data efficiently
  Attributes kept as separate properties
- Steps (per nD point):
  - (x,y [, z, t, cloi]), (attributes) → quantize → encode → (1D key [unsigned BIG int of (*mBits* * *nDims*) bits, stored as blob / byte array type]), (attributes)
    - quantize: scale / translate (max resolution of largest data dimension) - mBits
    - encode: bitwise interleave (Morton/Hilbert) - nDims
- Sort and store nD points
  - Sorted records in table, with SFC key as primary key
- For querying
  - Opposite route: (sfc key, attributes) → decode → dequantize → (x,y [, z, t, cloi]), (attributes)

Apache Parquet - details

- Columnar (hybrid)
- Table split over:
  - → Row groups → Columns
  - → Column Chunks → Pages
- Encoding data + Compression at Page level
- Metadata on Min/Max of Row groups (+ recent addition: Pages), stored at end of file (block ranges index, *if* data sorted!)
- Can be fetched in parts with http head followed by http range requests (if server support)
- C++, Java, Rust libraries read / write
- https://parquet.apache.org/



Apache Parquet – Some nD PC data

SFC key

Attributes

sfc_key blob	gps_time double	scan_angle float	intensity uint16	return_number uint8	number_of_returns uint8	classification uint8	scan_direction uint8	is_edge_of_flight_line uint8
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\x8ES\xC…	494147.3131107338	27.0	170	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFC\x04…	493868.6779278455	16.002	118	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFC\xCD…	494146.86342112836	27.0	22	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFC\xF9…	494146.83669732255	27.0	37	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFF,2\xF8	493869.0259201808	16.002	64	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFF\xCB…	494146.41376256285	28.002	30	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFF\xD7…	493868.97246252943	15.0	184	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,O\xF7X\x0…	493870.1446405349	16.002	230	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0E\x8C…	493870.0054991533	15.0	66	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0E\xA0…	493870.1339485246	16.002	195	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0E\xBB…	493870.0858279384	15.0	48	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0F;\xF1&	493870.03758022806	16.002	200	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x10o\x8A}	493869.855603322	16.002	47	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x10r\x93=	493869.74327855057	13.998	200	1	1	2	1	0
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7ws…	164798893.0589545	-10.002	171	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wt…	164798893.02829105	-10.002	16	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wt…	164798893.07124695	-10.002	39	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wt…	164798893.0466985	-10.002	129	1	1	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu…	164798892.99145013	-10.002	49	2	2	2	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wupz	164798892.9668992	-10.002	60	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu…	164798892.96690187	-10.002	18	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu…	164798892.9668992	-10.002	133	2	2	2	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu…	164798892.98529118	-9.0	95	1	3	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wv…	164798892.99755764	-9.0	164	2	2	2	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wvy(	164798892.99755764	-9.0	43	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7ww3M	164798893.0466725	-9.0	23	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7ww7g	164798893.0589441	-9.0	155	1	1	2	1	0

6083217 rows (40 shown) 9 columns

Apache Parquet – Some nD PC data

SFC key **Attributes**

sfc_key blob	gps_time double	scan_angle float	intensity uint16	return_number uint8	number_of_returns uint8	classification uint8	scan_direction uint8	is_edge_of_flight_line uint8
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\x8ES\xC…	494147.3131107338	27.0	170	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFC\x04…	493868.6779278455	16.002	118	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFC\xCD…	494146.86342112836	27.0	22	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFC\xF9…	494146.83669732255	27.0	37	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFF,2\xF8	493869.0259201808	16.002	64	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFF\xCB…	494146.41376256285	28.002	30	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFF\xD7…	493868.97246252943	15.0	184	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,O\xF7X\x0…	493870.1446405349	16.002	230	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0E\x8C…	493870.0054991533	15.0	66	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0E\xA0…	493870.1339485246	16.002	195	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0E\xBB…	493870.0858279384	15.0	48	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0F;\xF1&	493870.03758022806	16.002	200	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x10o\x8A}	493869.855603322	16.002	47	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x10r\x93=	493869.74327855057	13.998	200	1	1	2	1	0
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7ws…	164798893.0589545	-10.002	171	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wt…	164798893.02829105	-10.002	16	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wt…	164798893.07124695	-10.002	39	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wt…	164798893.0466985	-10.002	129	1	1	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu…	164798892.99145013	-10.002	49	2	2	2	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wupz	164798892.9668992	-10.002	60	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu…	164798892.96690187	-10.002	18	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu…	164798892.9668992	-10.002	133	2	2	2	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu…	164798892.98529118	-9.0	95	1	3	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wv…	164798892.99755764	-9.0	164	2	2	2	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wvy(	164798892.99755764	-9.0	43	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7ww3M	164798893.0466725	-9.0	23	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7ww7g	164798893.0589441	-9.0	155	1	1	2	1	0

6083217 rows (40 shown) 9 columns

Row groups

Apache Parquet – Some nD PC data

SFC key **Attributes**

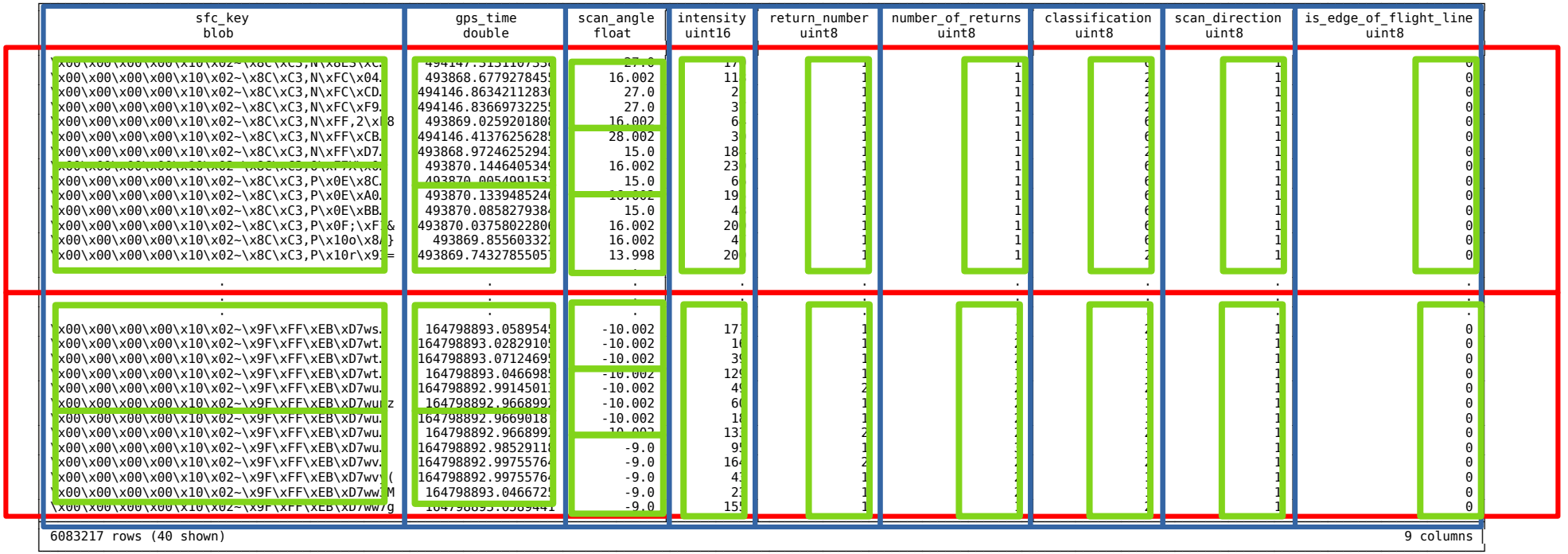
sfc_key blob	gps_time double	scan_angle float	intensity uint16	return_number uint8	number_of_returns uint8	classification uint8	scan_direction uint8	is_edge_of_flight_line uint8
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\x8ES\xC…	494147.3131107338	27.0	170	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFC\x04…	493868.6779278455	16.002	118	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFC\xCD…	494146.86342112836	27.0	22	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFC\xF9…	494146.83669732255	27.0	37	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFF,2\xF8	493869.0259201808	16.002	64	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFF\xCB…	494146.41376256285	28.002	30	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFF\xD7…	493868.97246252943	15.0	184	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,O\xF7X\x0…	493870.1446405349	16.002	230	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0E\x8C…	493870.0054991533	15.0	66	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0E\xA0…	493870.1339485246	16.002	195	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0E\xBB…	493870.0858279384	15.0	48	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0F;\xF1&	493870.03758022806	16.002	200	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x10o\x8A}	493869.855603322	16.002	47	1	1	6	1	0
\x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x10r\x93=	493869.74327855057	13.998	200	1	1	2	1	0
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7ws…	164798893.0589545	-10.002	171	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wt…	164798893.02829105	-10.002	16	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wt…	164798893.07124695	-10.002	39	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wt…	164798893.0466985	-10.002	129	1	1	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu…	164798892.99145013	-10.002	49	2	2	2	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wupz	164798892.9668992	-10.002	60	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu…	164798892.96690187	-10.002	18	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu…	164798892.9668992	-10.002	133	2	2	2	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu…	164798892.98529118	-9.0	95	1	3	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wv…	164798892.99755764	-9.0	164	2	2	2	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wvy(	164798892.99755764	-9.0	43	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7ww3M	164798893.0466725	-9.0	23	1	2	1	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7ww7g	164798893.0589441	-9.0	155	1	1	2	1	0

6083217 rows (40 shown) 9 columns

Row groups + **Columns** = Column Chunks

Apache Parquet – Some nD PC data

SFC key **Attributes**



sfc_key blob	gps_time double	scan_angle float	intensity uint16	return_number uint8	number_of_returns uint8	classification uint8	scan_direction uint8	is_edge_of_flight_line uint8
\x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFC\x04.	493868.677927845	16.002	11	1	1	2	1	0
x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFC\xCD.	494146.8634211283	27.0	2	1	1	2	1	0
x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFC\xF9.	494146.8366973225	27.0	3	1	1	2	1	0
x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFF,2\x[illegible]8	493869.0259201808	16.002	6	1	1	6	1	0
x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFF\xCB.	494146.4137625628	28.002	3	1	1	6	1	0
x00\x00\x00\x00\x10\x02~\x8C\xC3,N\xFF\xD7.	493868.9724625294	15.0	18	1	1	2	1	0
[illegible]	493870.1446405349	16.002	23	1	1	6	1	0
x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0E\x8C.	493870.0054991532	15.0	6	1	1	6	1	0
x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0E\xA0.	493870.1339485240	16.002	19	1	1	6	1	0
x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0E\xBB.	493870.0858279384	15.0	4	1	1	6	1	0
x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x0F;\xF <	493870.0375802280	16.002	20	1	1	6	1	0
x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x10o\x8 }	493869.855603321	16.002	4	1	1	6	1	0
x00\x00\x00\x00\x10\x02~\x8C\xC3,P\x10r\x9 =	493869.7432785057	13.998	20	1	1	2	1	0
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7ws.	164798893.0589545	-10.002	17	1	1	2	1	0
x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wt.	164798893.0282910	-10.002	1	1	1	2	1	0
x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wt.	164798893.0712469	-10.002	3	1	1	2	1	0
x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wt.	164798893.046698	-10.002	12	1	1	2	1	0
x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu.	164798892.9914501	-10.002	4	2	2	2	1	0
x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu z	164798892.966899	-10.002	6	1	1	2	1	0
x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu.	164798892.9669018	-10.002	1	1	1	2	1	0
x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu.	164798892.966899	-10.002	13	2	2	2	1	0
x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wu.	164798892.9852911	-9.0	9	1	1	2	1	0
x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wv.	164798892.9975576	-9.0	16	2	2	2	1	0
x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7wv (	164798892.9975576	-9.0	4	1	1	2	1	0
x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7ww M	164798893.046672	-9.0	2	1	1	2	1	0
\x00\x00\x00\x00\x10\x02~\x9F\xFF\xEB\xD7ww/g	164798893.058944	-9.0	15	1	1	2	1	0

6083217 rows (40 shown) 9 columns

Column Chunks divided into **Pages**
(Encoding and Compression can be different per column dependent on data type)

Converter (.laz) to Apache Parquet

- Read input point cloud points and produce SFC key
  - Quantize, Encode → (SFC Key + original record index); Extra attributes as additional columns
  - Internal column store format: Vec of Vecs
- Sort SFC Key vec
- Generate nD histogram (once sorted data): distribution of points over SFC
- Write Parquet output (write key column and fetch from internal column store format relevant attributes by their row index in correct order)
- [more steps if data does not fit all in main memory … not discussed today]

Querying Parquet – Steps in theory

- Formulate integrated dimension query (e.g. space + level of detail) as set of half spaces (nD convex polytope)
- Use obtained histogram and implicit n-ary tree to find relevant parts of curve (tree traversal)
- Join query ranges to data table
  - In database use SQL: select * from data, ranges where data.sfc_key between ranges.start and ranges.end;

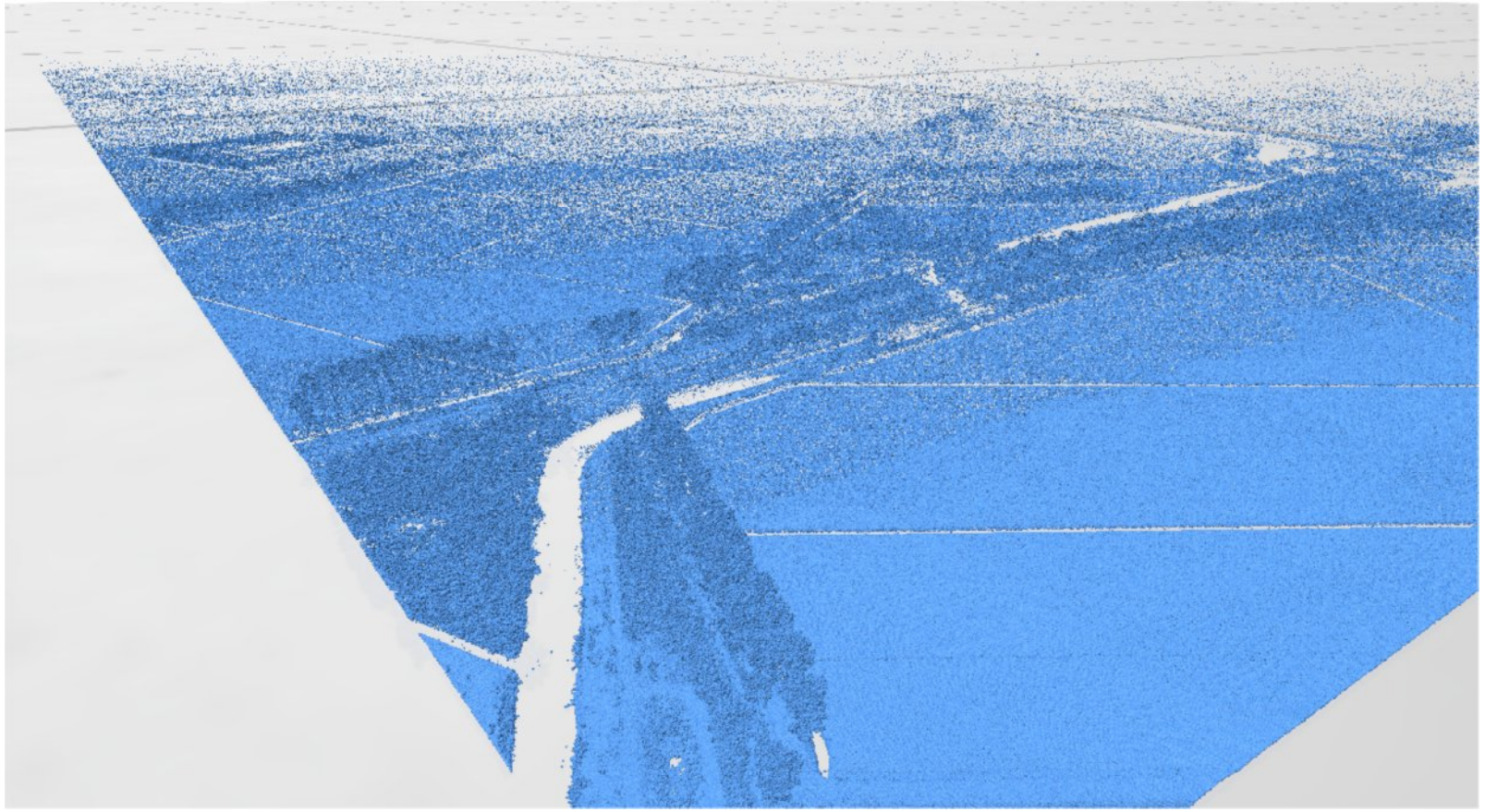
Querying Parquet – Off-the-shelf merge-join with query engine

- Up to now, I did not get it to work with off-the-shelf query engines (DuckDB/Datafusion) for somewhat larger files via SQL
  - Engine starts to re-sort the data, while data and ranges are sorted: efficient sort-merge-join best possibility (no need for sort as both join sides are already sorted)
- Own implementation, not via SQL

Querying Parquet – Custom Implementation Merge-Join

- With generated query ranges:
  - Custom merge join with half spaces → query ranges and data (SFC keys)
    Read relevant parts from Parquet file, that contains sorted data
  - Heavy use of Min/Max index data on both Row groups as well as Column pages in footer of Parquet file
  - Fetch in phases:
    1) Exact matching records based on SFC key column (a. Row Group → b. Column page)
    2) Retrieve additional attributes for query result points from disk
  - Decode / de-quantize keys
  - Exact test of nD points against query hyper planes → needs coordinates (if query range on boundary), but application also need coordinates
- Granularity of answer: Precise to individual Point record

Query result – Integrated Space / Level of Detail query



Querying Parquet – Block level

- *Note*: idea – Still to implement/experiment with it
- Row groups (column chunks) give Blocks of points – nD box around blocks, with all blocks same number of points(!)
- nD boxes can be hierarchically indexed (separate from Parquet file)
  - Construct Tree based on points in SFC curve order (iteratively ‘merge’ small nD box with closest & largest nD box neighbor on the curve) → Binary tree
  - With added additional Level of Importance in SFC key large curve parts (=big nD bounding box) need to end up at top of tree
- Use this tree structure inside client (e.g. Potree, will require client side modification) to request Column chunks from Parquet file
- Answers to queries will not be precise up to point level (but if this is necessary depends on application – viewing points might not need this)

Discussion

- Apache Parquet features very promising for storing + querying nD points
  - Compression works [but works better with native types (signed int64) than with blob's; still to quantify]
  - Can be directly queried / linked to database as external table (SQL access)
  - Retrieval (in theory) possible over HTTP(S) – 'cloud native'
  - Row groups can provide nD Point Clouds as blocks, will need additional structure for retrieval on the web
- Further investigate
  - Query performance
    Individual record or block level query results
  - Off the shelf query engines that can read Parquet –
    Expressing that data is already sorted seems bottleneck
  - Split SFC key in parts (low res/high res parts): fit in multiple columns of native type – current MSc Geomatics thesis project

Questions?

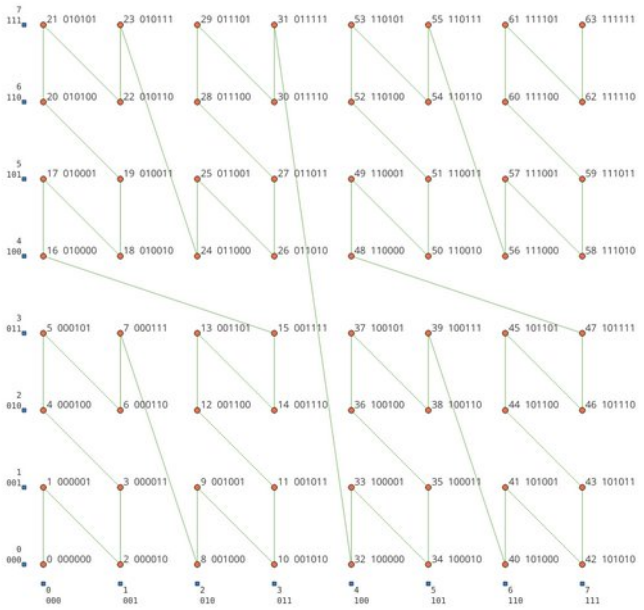
dr.ir. Martijn Meijers

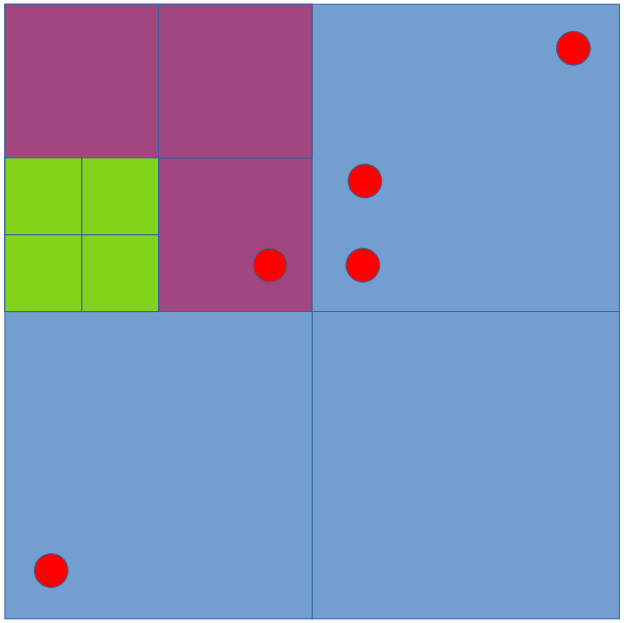
Delft University of Technology
Faculty of Architecture and the Built Environment
Architectural Engineering + Technology
Digital Technologies
GIS-technology

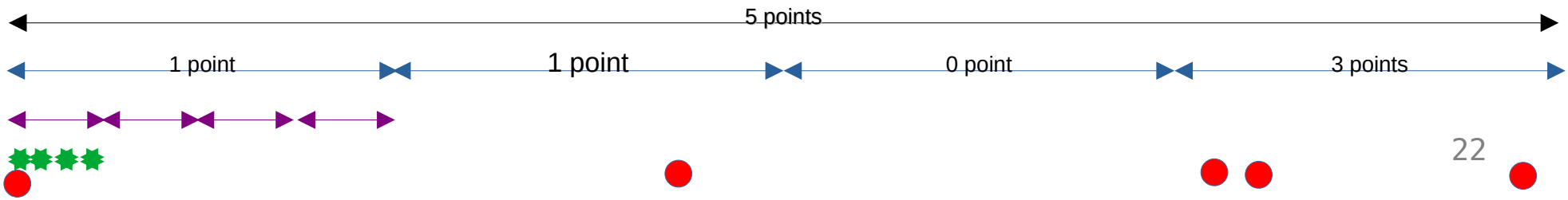
tel: (+31) 15 278 56 42
mailto:b.m.meijers@tudelft.nl
https://www.tudelft.nl/en/staff/b.m.meijers/

Extra slides

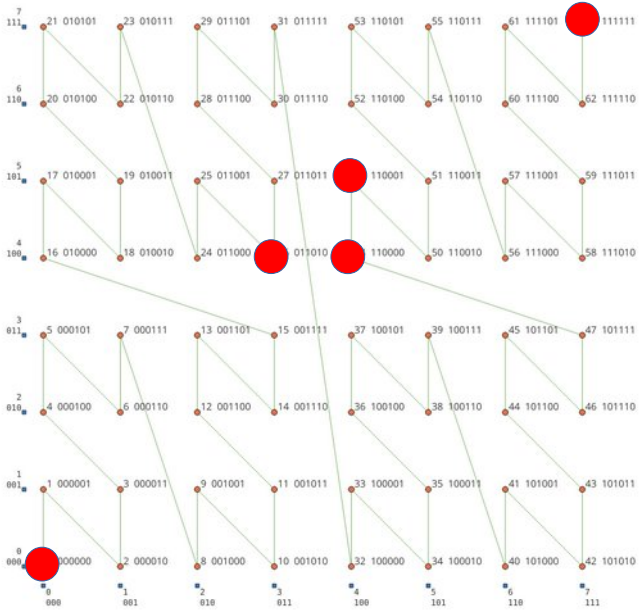
Curve + Histogram (after quantize)

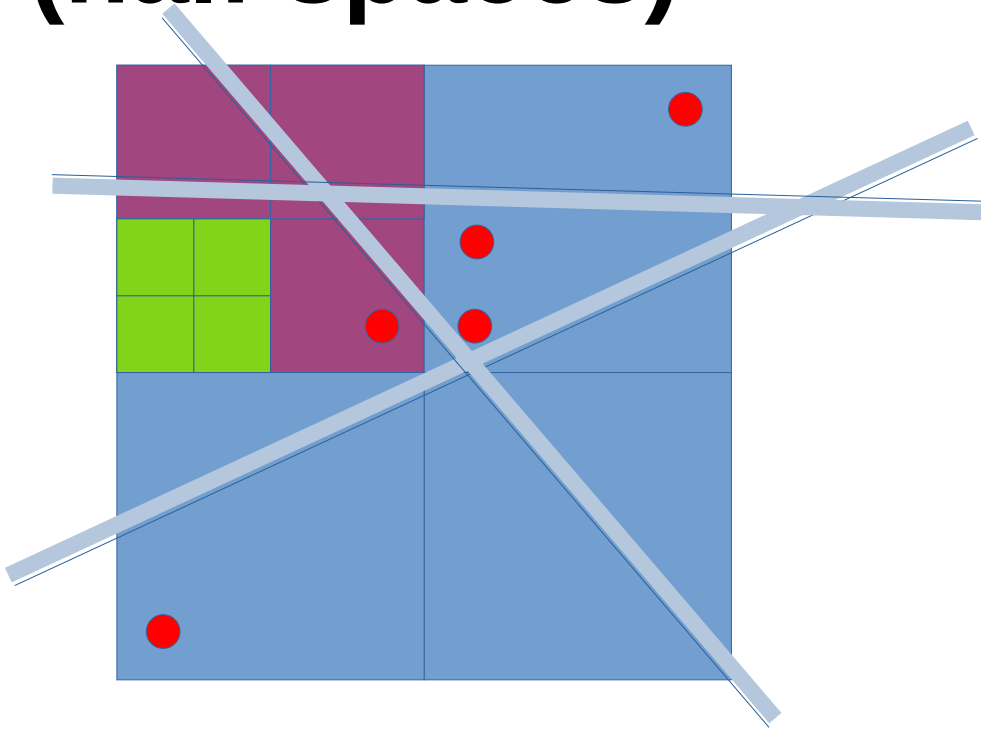


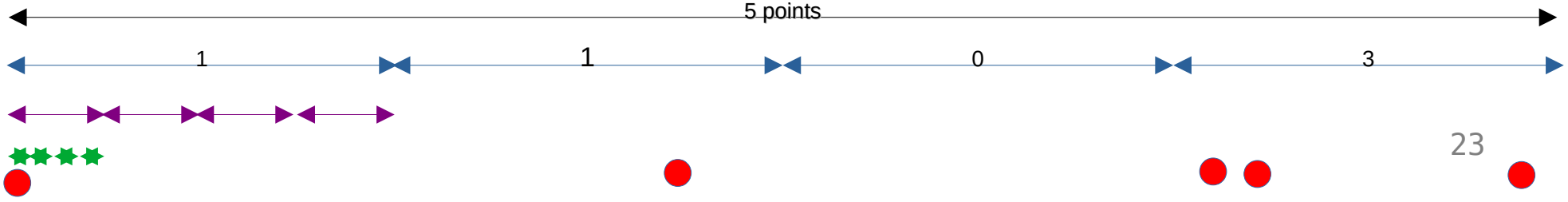




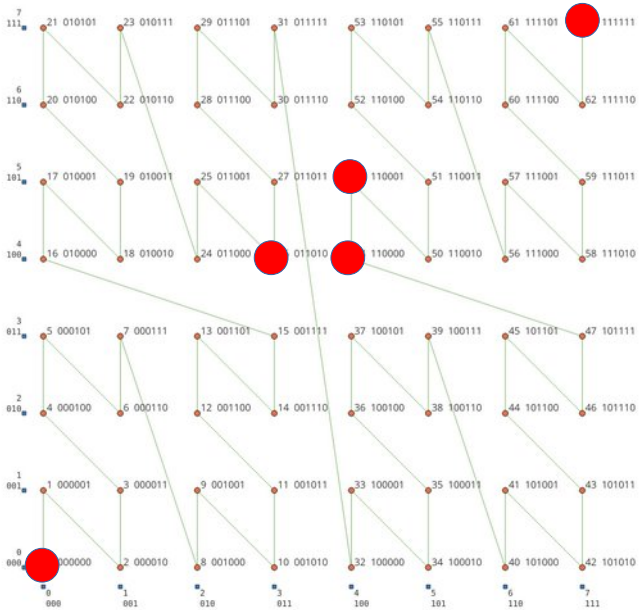
Query with Polytope (half spaces)

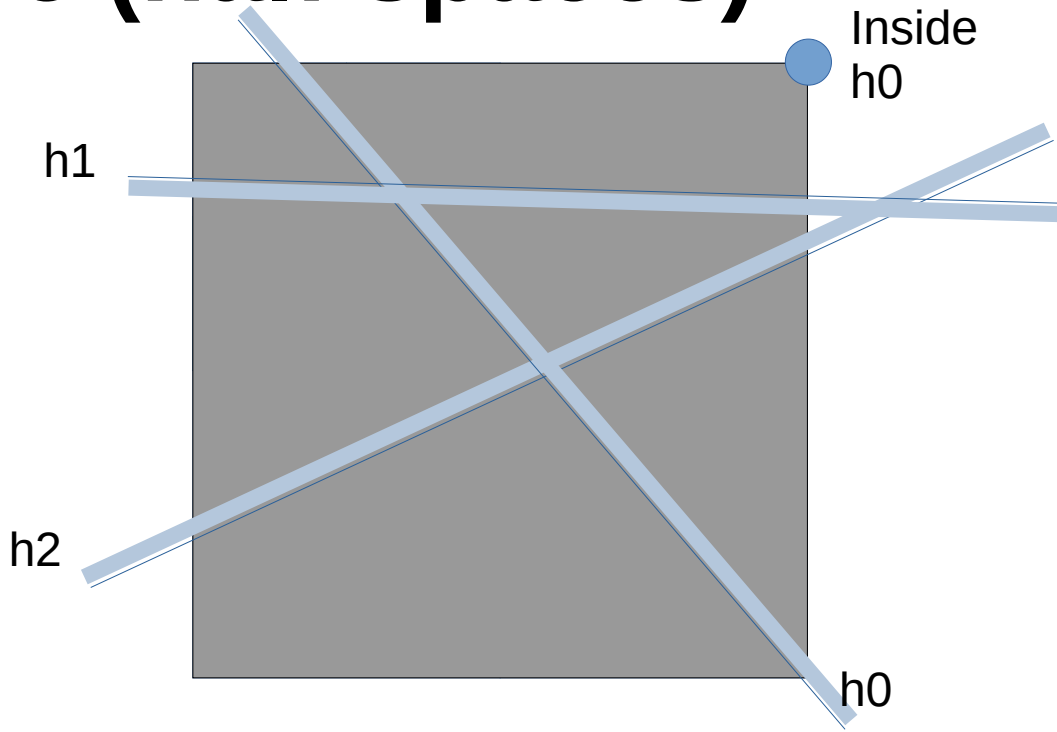


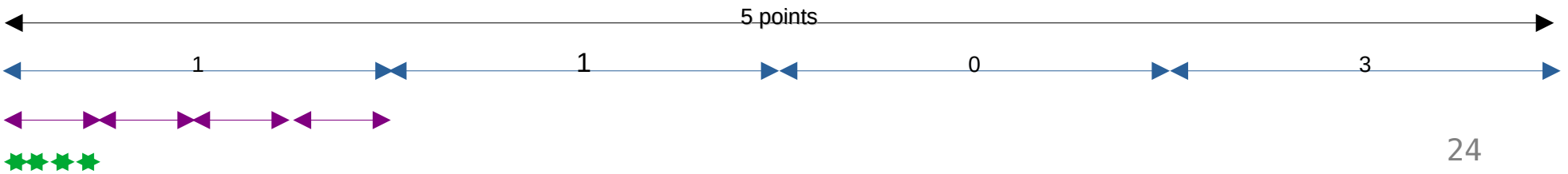




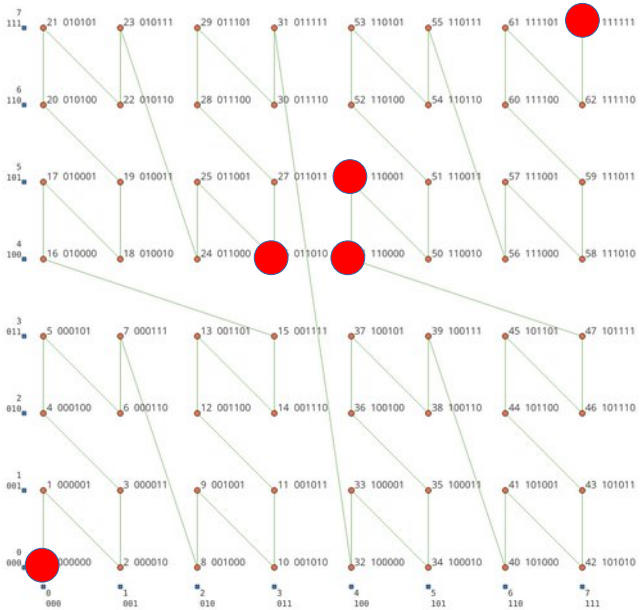
Query with Polytope (half spaces)







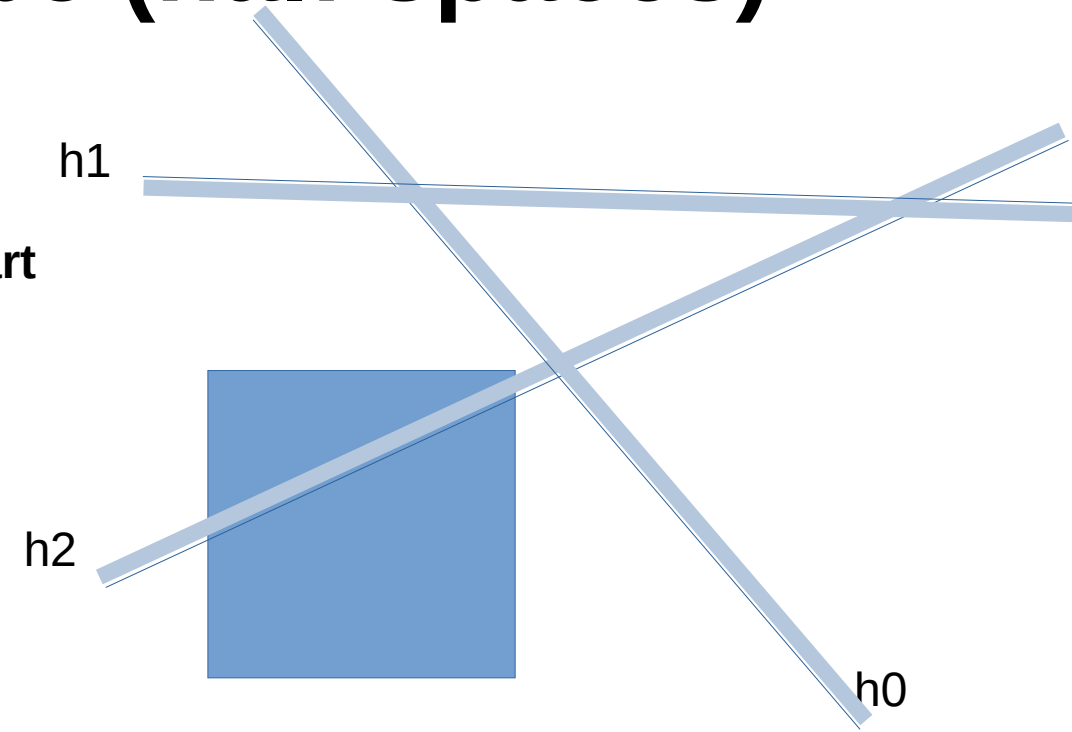
Query with Polytope (half spaces)

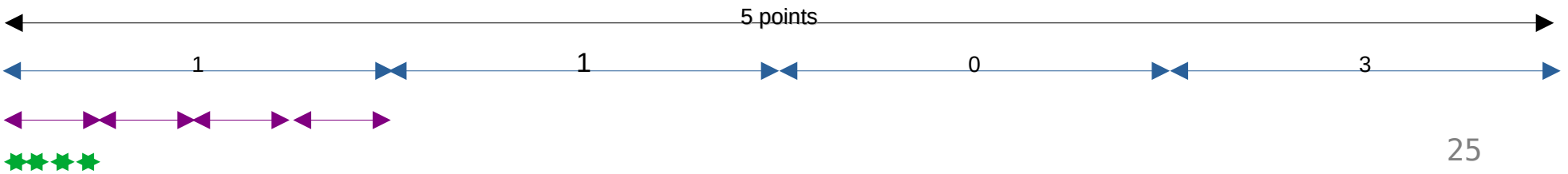


Exclude Curve part

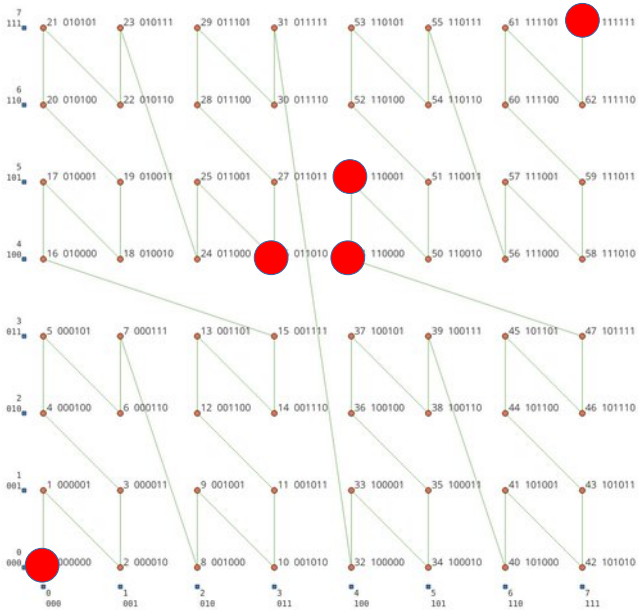
→

Fully Outside h0

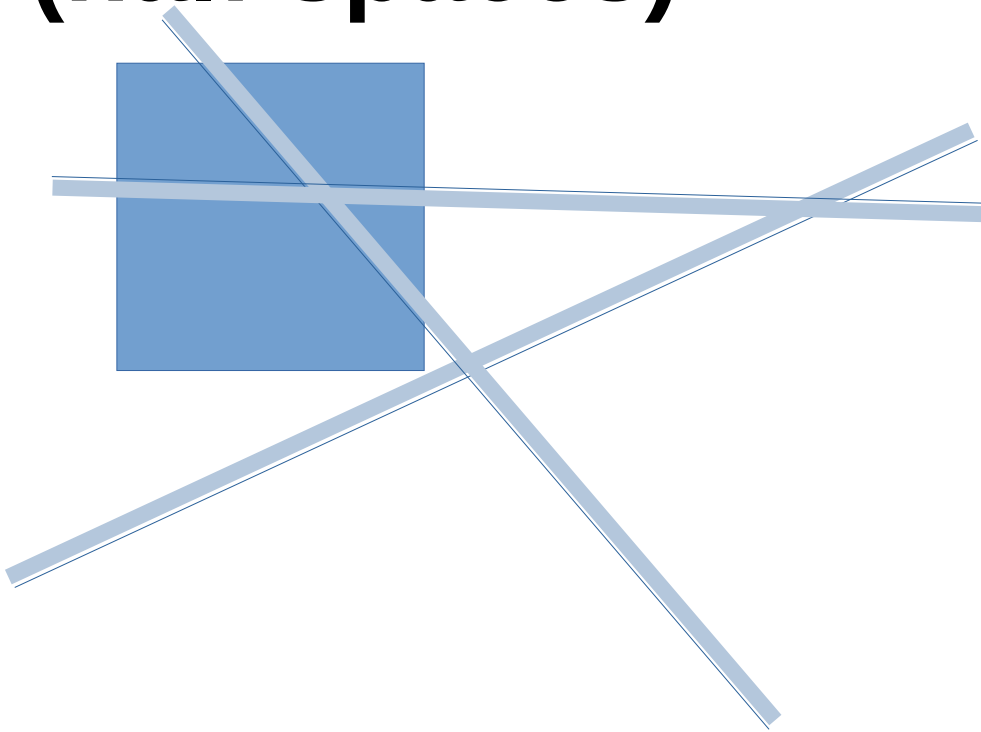


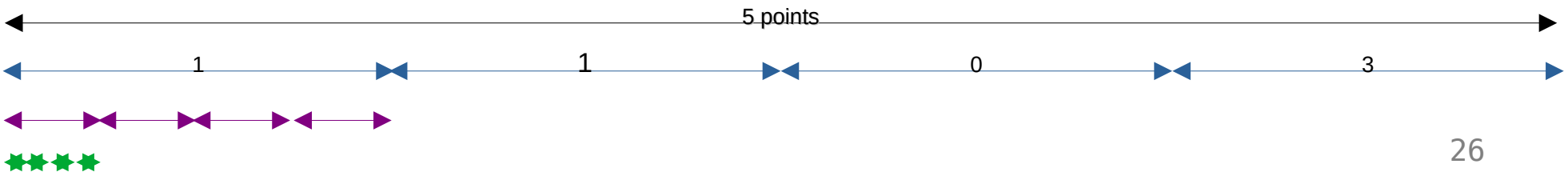


Query with Polytope (half spaces)

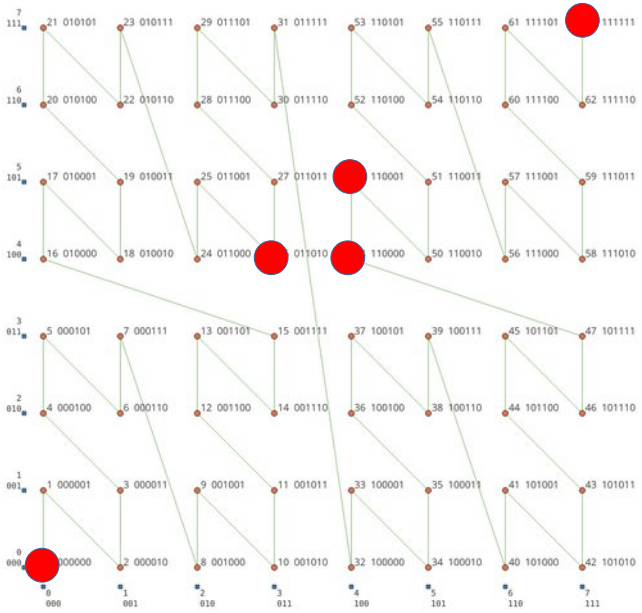


Accept
Curve part

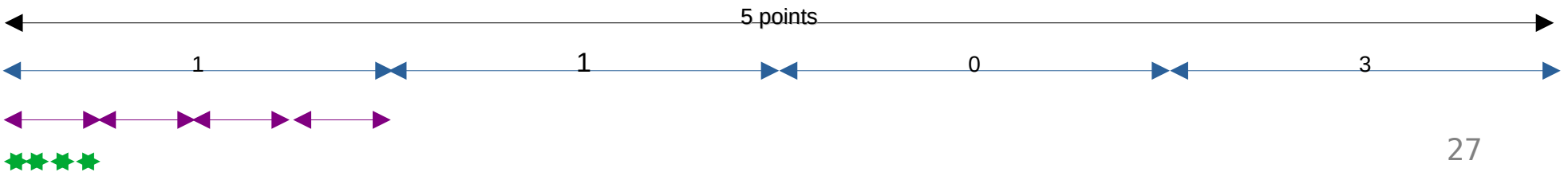




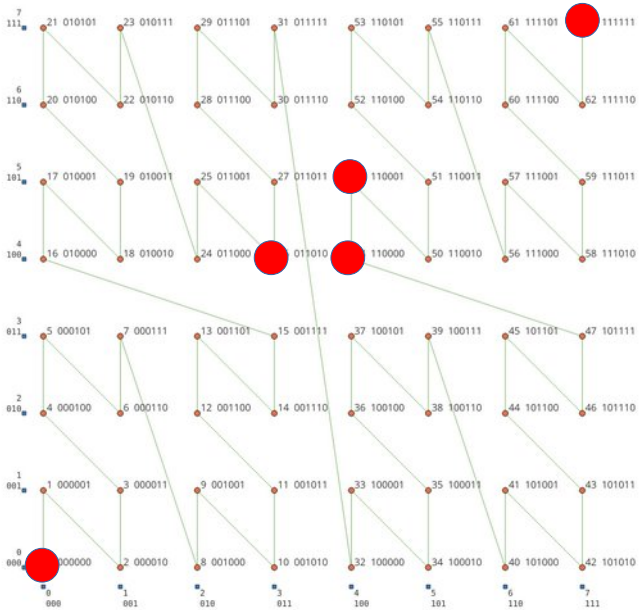
Query with Polytope (half spaces)



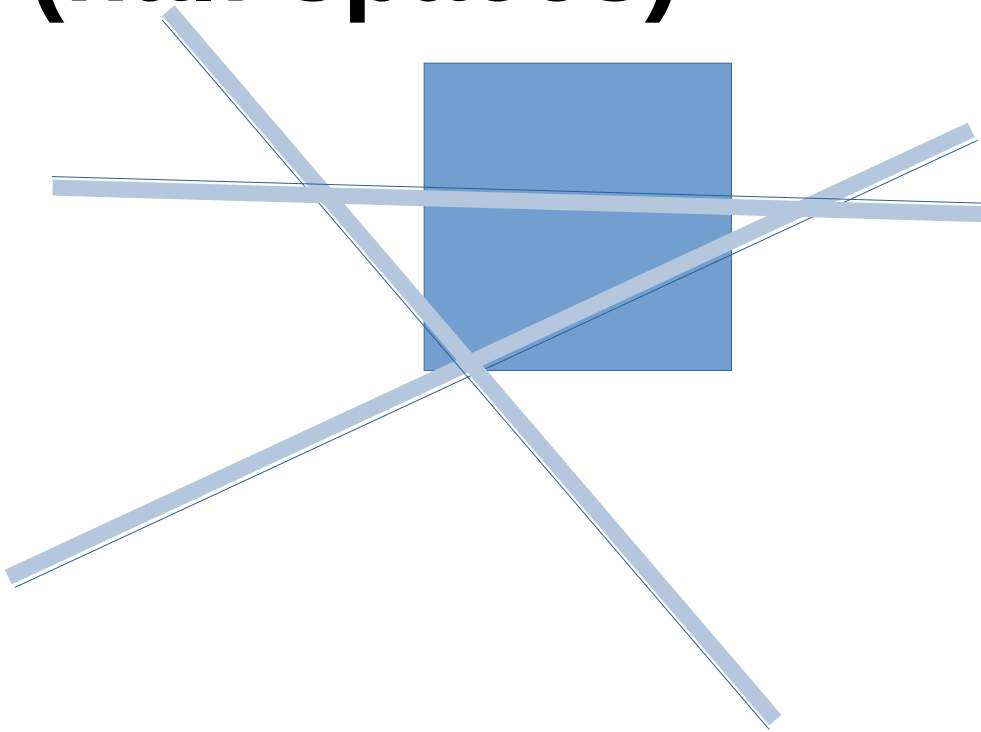
Exclude
Curve part
→
Histogram
0 points

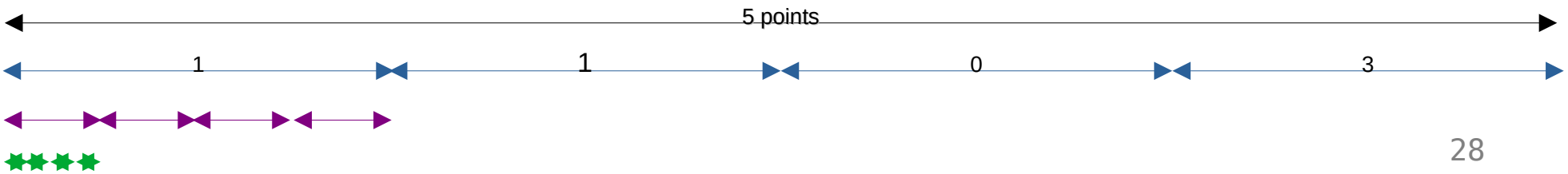


Query with Polytope (half spaces)

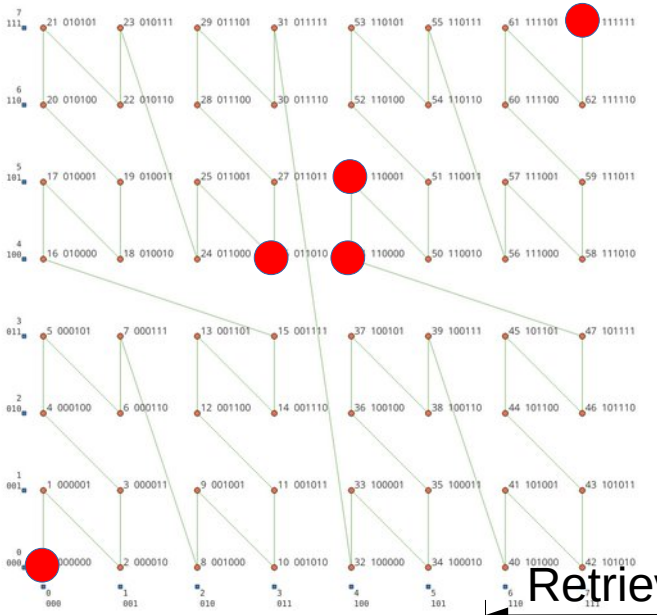


Accept
Curve part



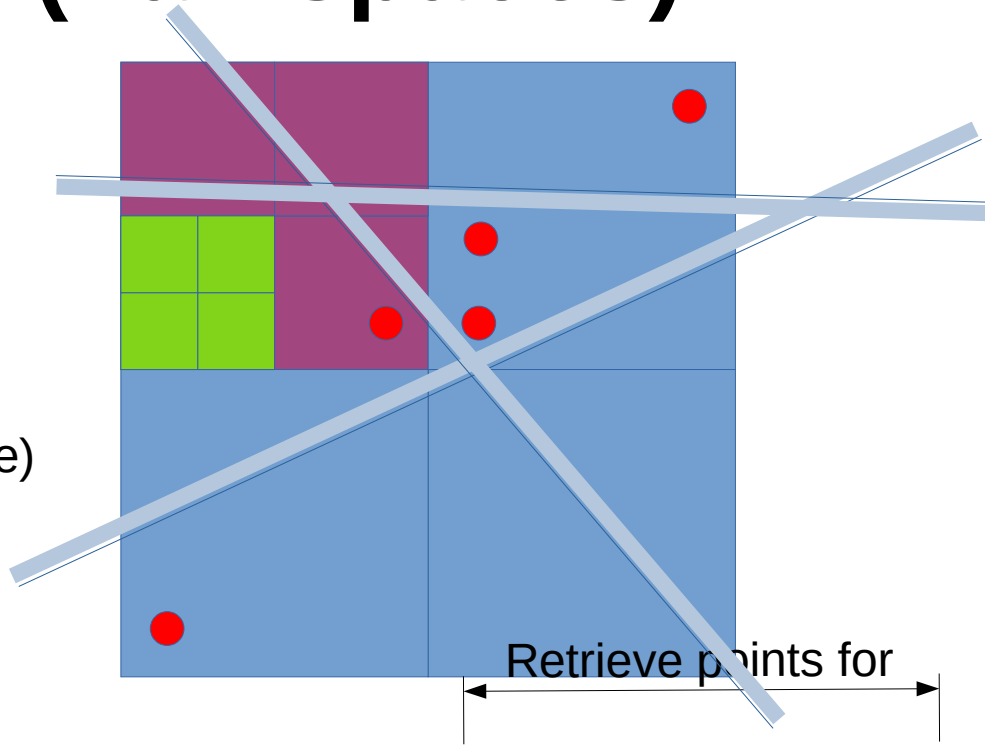


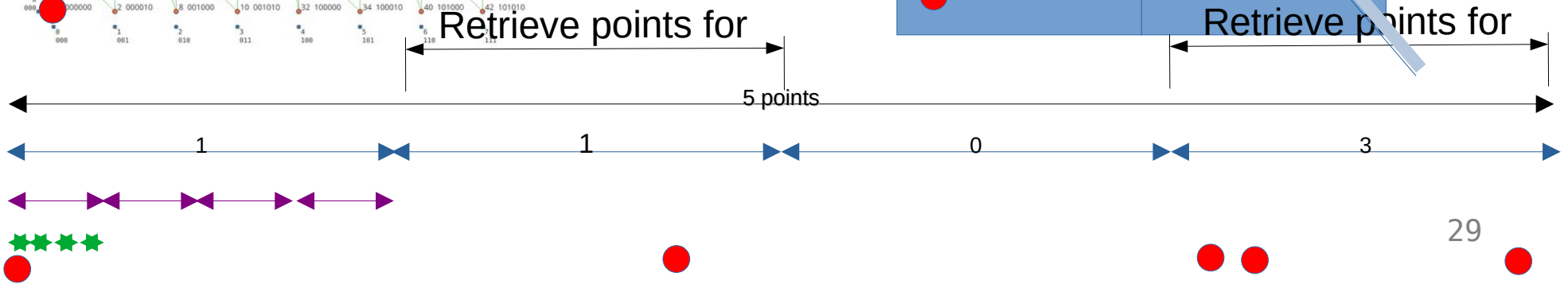
Query with Polytope (half spaces)



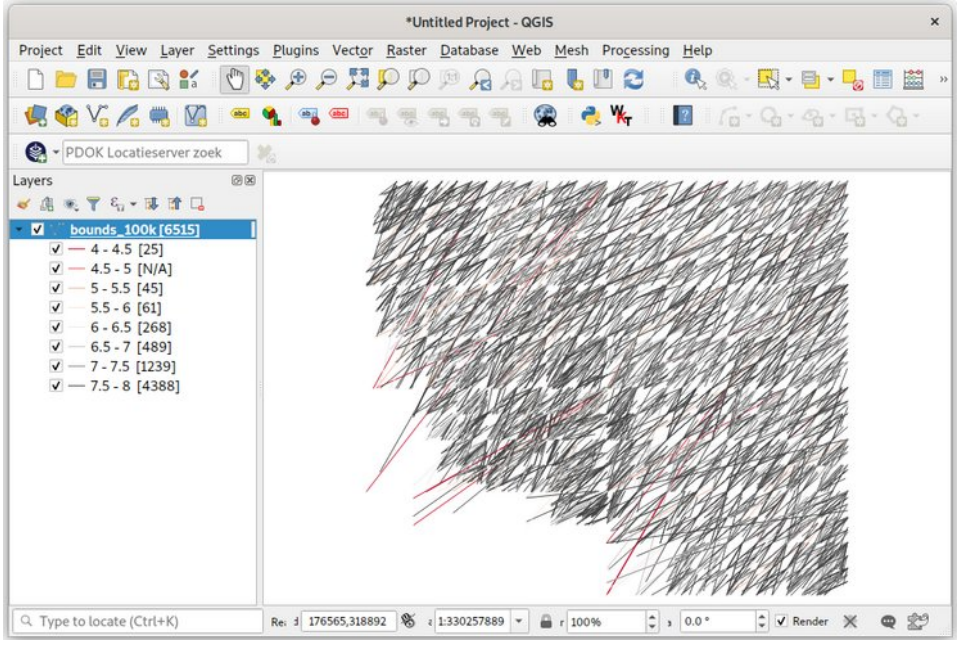
Retrieve Points

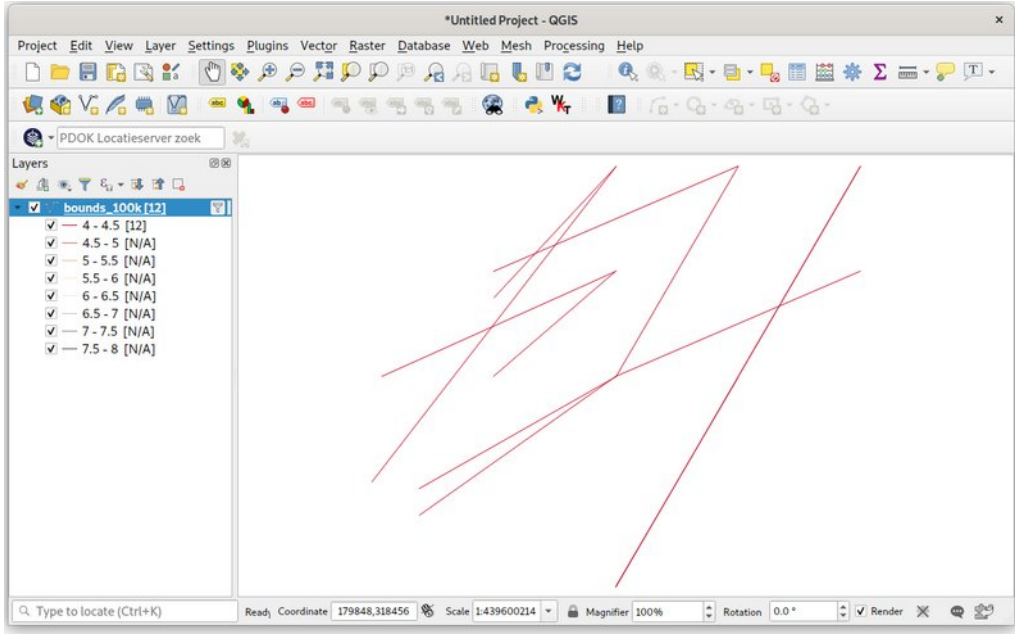
Evaluate against Polytope (iff curve part not fully inside)





Blocks (2D + cLoI) – Look from above





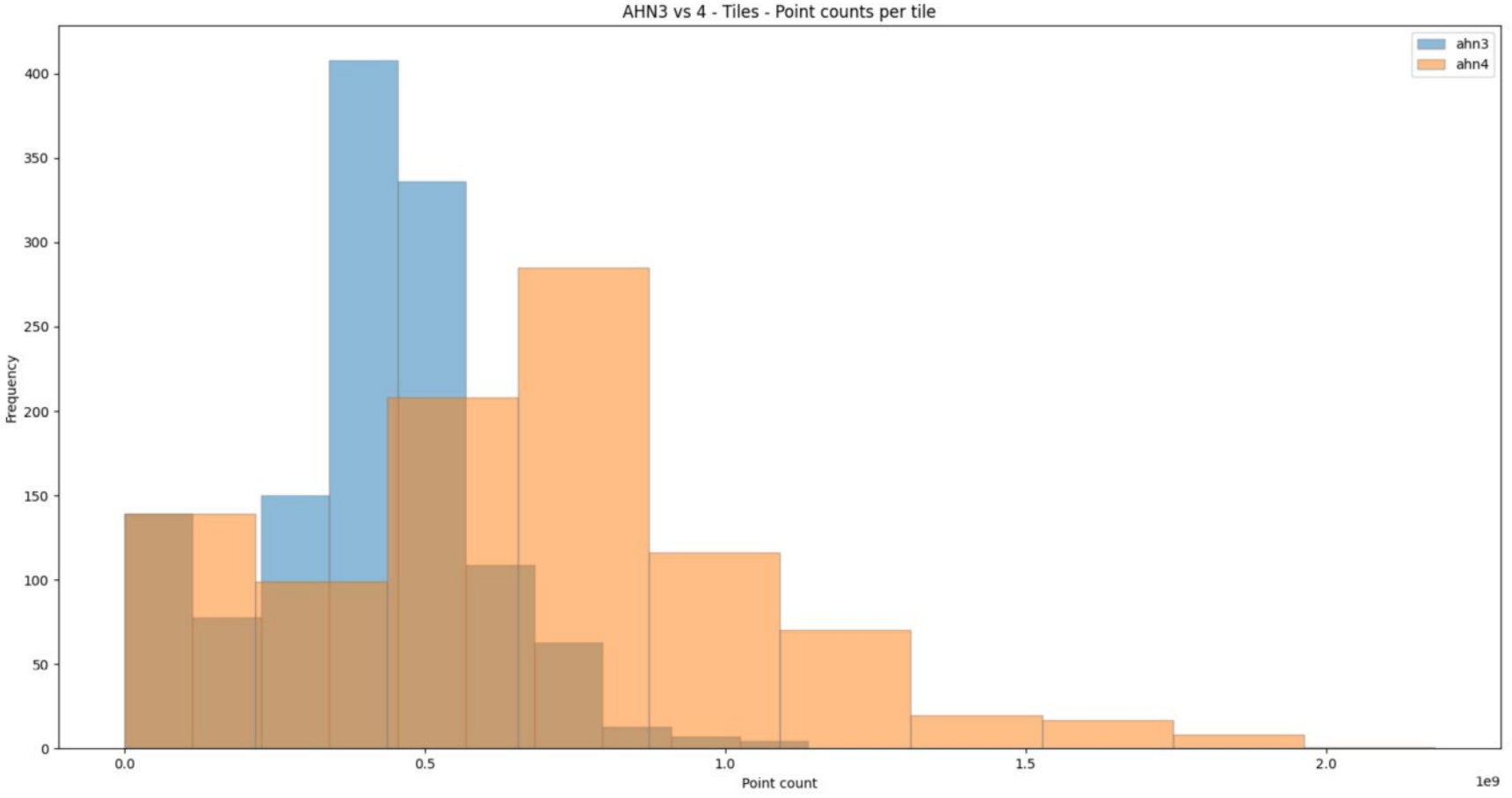
AHN3 – Size – Total dataset

- Wrote Python script to download only headers of 1374 .laz files
- Statistics AHN3:
  - Bytes (laz): 2_566_998_161_919 (≈2.5TB)
  - Point count: 557_925_797_136
  - Uncompressed (estimation): 16_737_773_914_080 (≈16.7TB)
    - with Point format 6: 30 bytes per point
  - Compression factor: ≈6.5x

AHN4 – Size – Subset of dataset

- Fetched metadata (May 2022) – not yet whole country
  1010 tiles
- Sizes:
  - Bytes (laz) 4_888_768_203_224 (≈5TB)
  - Point count 665_838_075_757
  - Compression factor ≈7.3x
- Estimation:
  - ≈ 6.6TB total compressed (1374 tiles .laz)

    ≈ 48TB uncompressed


AHN3 vs 4 - Tiles - Point counts per tile
ahn3
ahn4
Frequency
0
50
100
150
200
250
300
350
400
0.0
0.5
1.0
1.5
2.0
Point count
1e9