SplitSFC: A database solution for massive point cloud data management

Yuduan Cai

Martijn Meijers
Peter van Oosterom

18-01-2024

TU Delft

Content

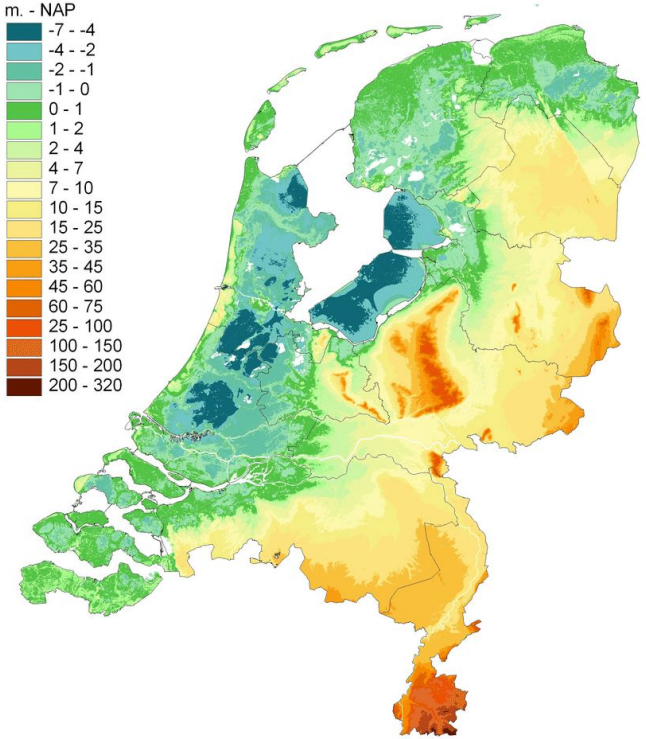
- Introduction: Point Cloud
- Background: Space Filling Curve
- Methodology
- Results
- Conclusion

01

Point Cloud

TU Delft

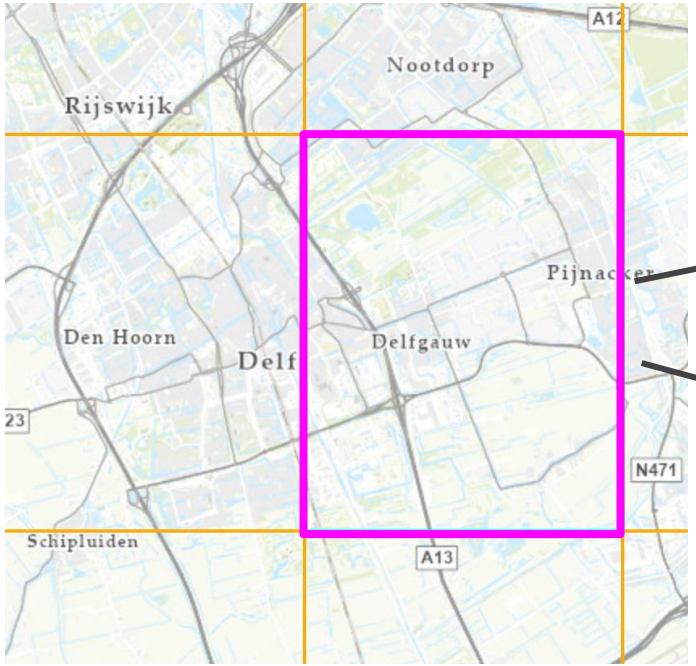
Example: Digital Surface Model



Elevation of the Netherlands

Downloading AHN3 from PDOK

Example: Digital Surface Model



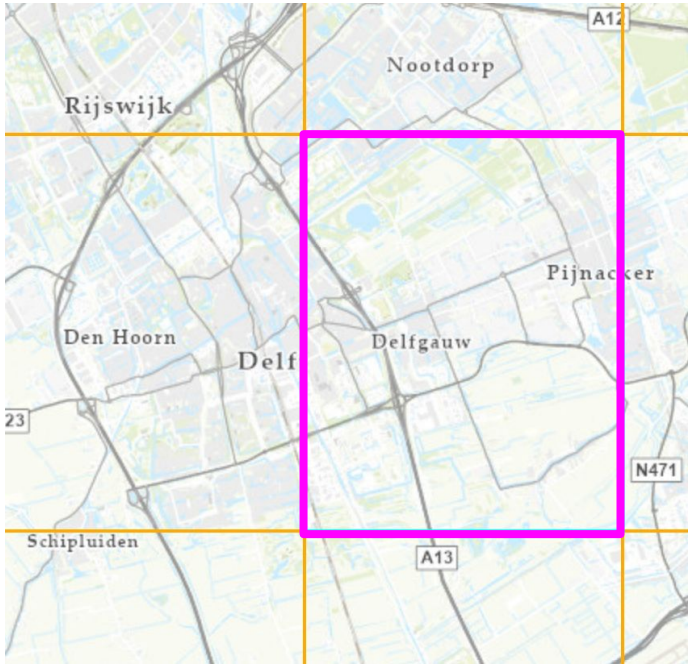
0.5m Raster

Point Cloud

AHN3, Tile 37en2

TU Delft

Example: Digital Surface Model



AHN3, Tile37en2

	Format	Size
0.5m Raster	TIF	0.49 GB
Point Cloud	LAZ	**2.47 GB**

Over **5 times** larger!

510 millions points in one tile!

TU Delft

Example: Digital Surface Model

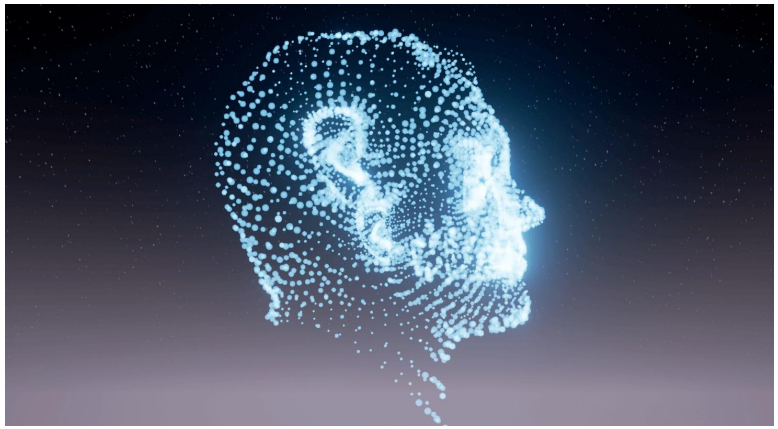
AHN3 from PDOK

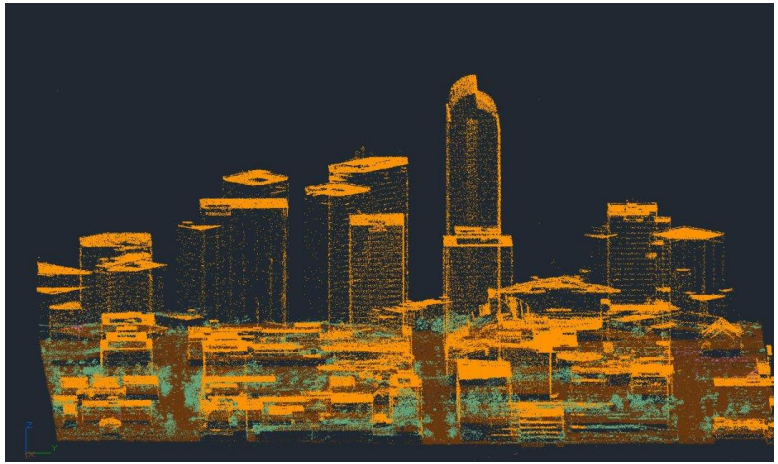
- **~640 billion points** in total!
- How about global DSM?

- How to store them?
- How to provide functionalities?

What is a point cloud?

- A set of points in 3D space
  - XYZ
  - Classification, RGB, GPS time, etc.
- A third representation of spatial data
  - vector, raster, PC
- Wide usage
  - 3D reconstruction
  - Autonomous driving
  - VR, AR, Gaming





TU Delft

However, the data management is challenging!

However, the data management is challenging!

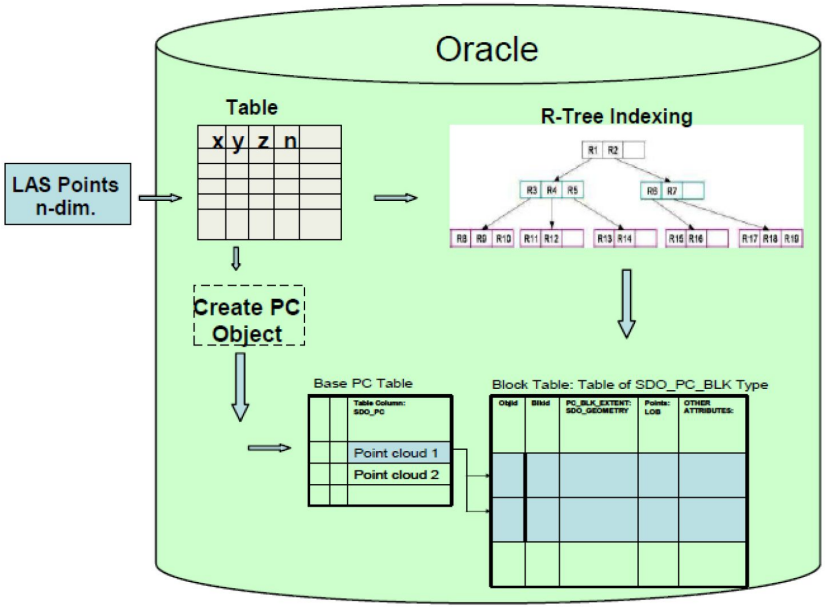
- Point clouds are a set of points, **massive** in size
- **File formats** like LAS are dominating
- But, **DBMS solutions** are also **crucial**
  - Functionalities
  - Optimised disk IO strategies
  - Auto parallelization in the query executions

Types of database storage

- Flat table
  - flexible, but large in size
  - suitable for small point cloud
- Block-based model
  - compact and scalable
  - suitable for massive point clouds

    Oracle SDO_PC

    pgPointCloud



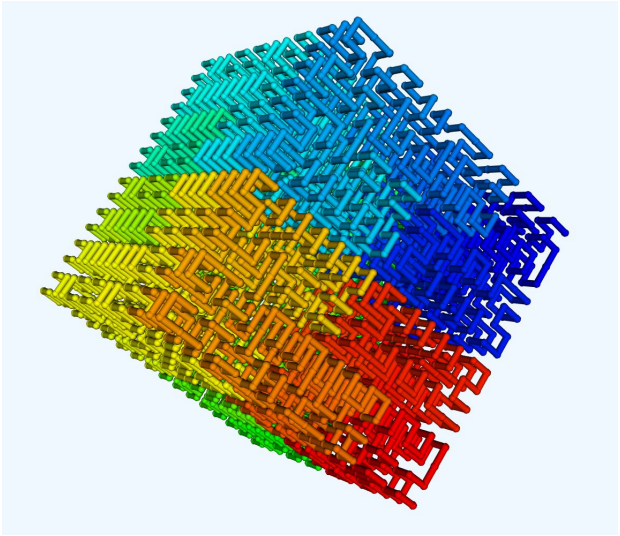
Storage Model for Oracle SDO_PC type
Ravada et al., 2010

TU Delft

Spatial Access Methods

- Space Filling Curve
  - PlainSFC (van Oosterom, 2015)
  - DynamicSFC (Psomadaki, 2016)
  - HistSFC (Liu, 2020)

- But, PlainSFC uses a flat table
- A SFC-based block model is required!



3D Morton Curve
Source: https://www.hongheiniao.com

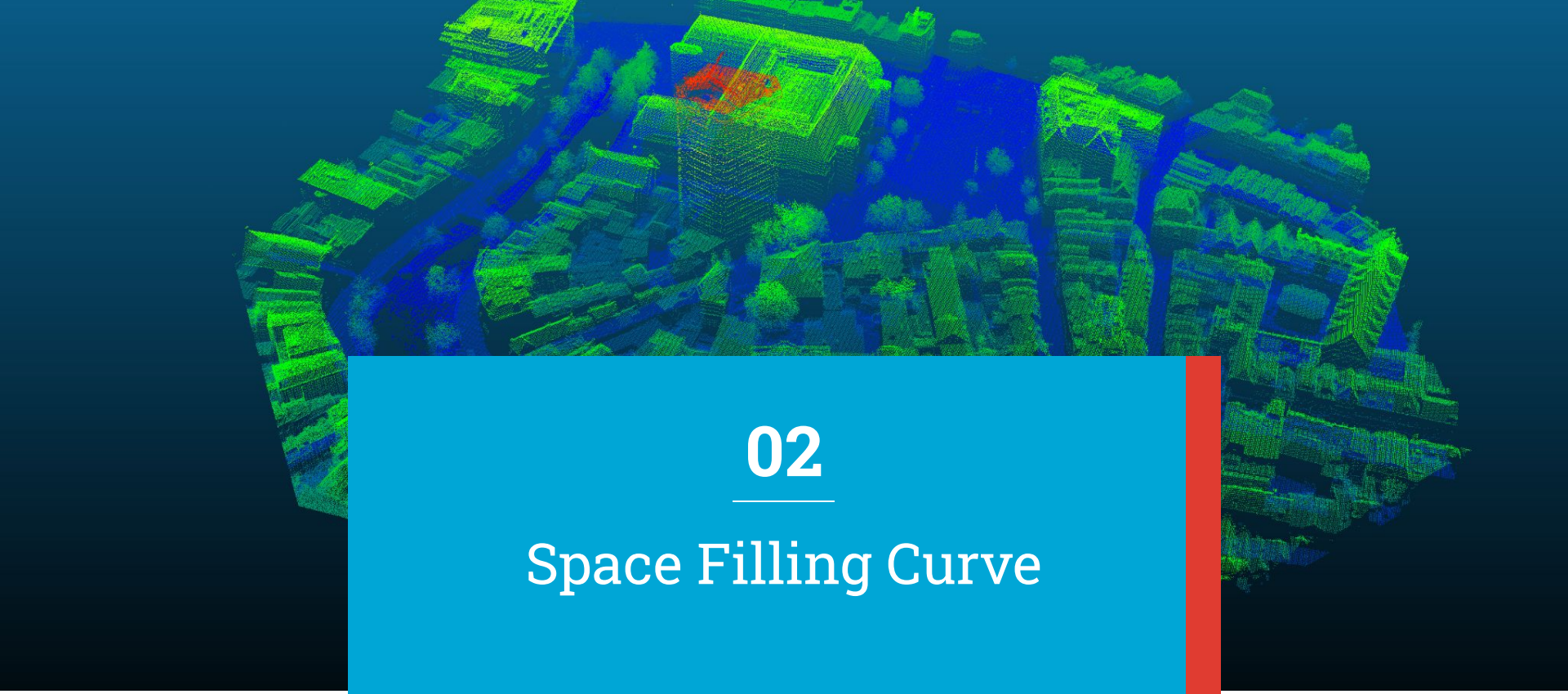
TU Delft

Objective & Question

- Objective of the research
  - To investigate a database solution for massive points clouds
  - The storage model is block-based, and use SFC to organize points
  - Implement the prototype and test it

- Core question:
  - **Does employing the split Space Filling Curve approach constitute an effective strategy for managing massive point clouds in a relational database?**

TU Delft



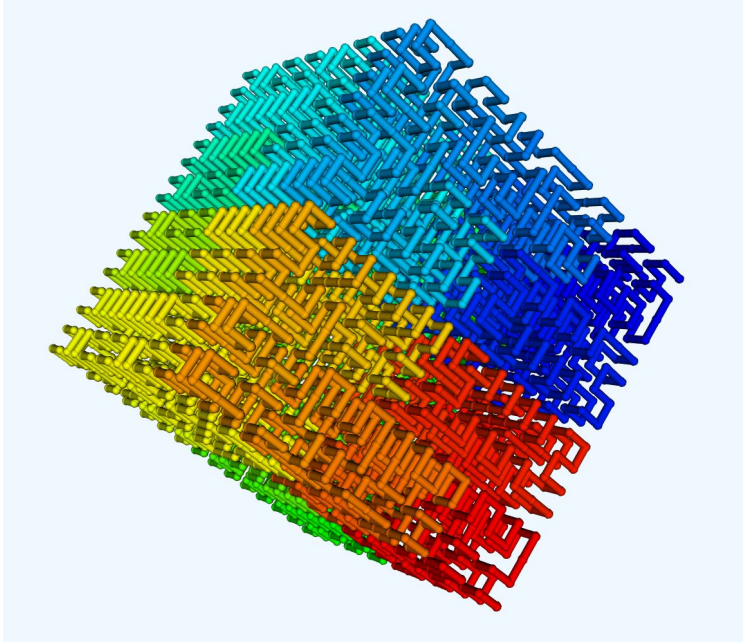
02

Space Filling Curve

TU Delft

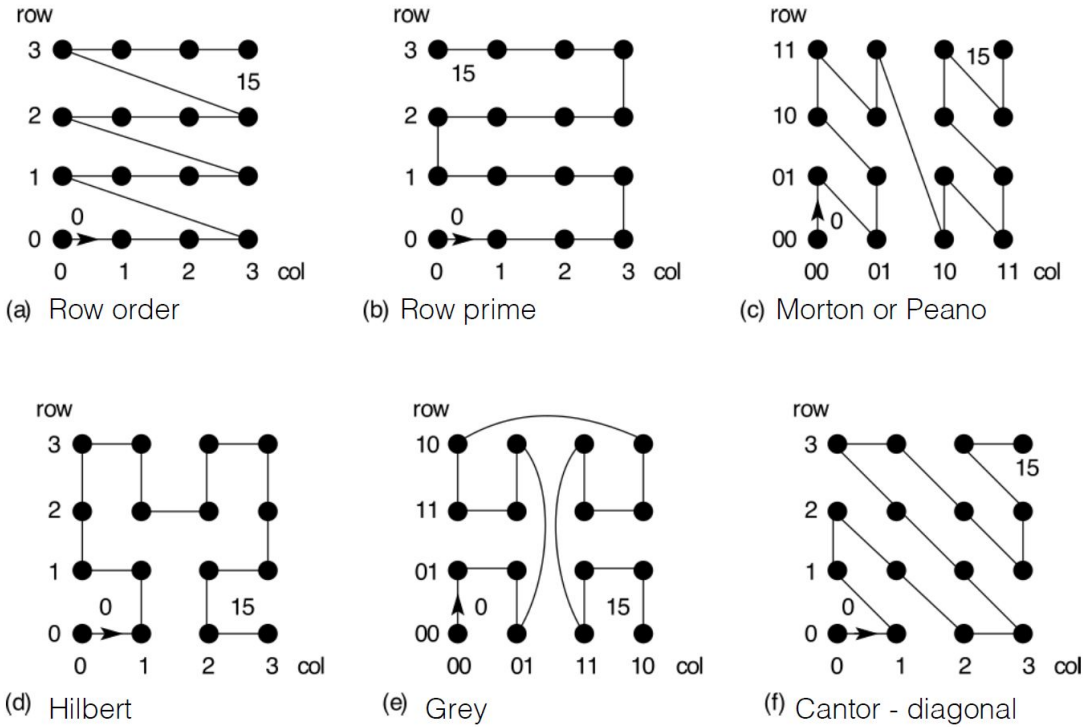
Space Filling Curve

- What is SFC?
  - **Applying a linear order to multi-dimensional data**
- Why SFC?
  - Dimensionality reduction (sorting, indexing)
  - Full resolution curve (compression)
  - Spatial clustering (physics storage)



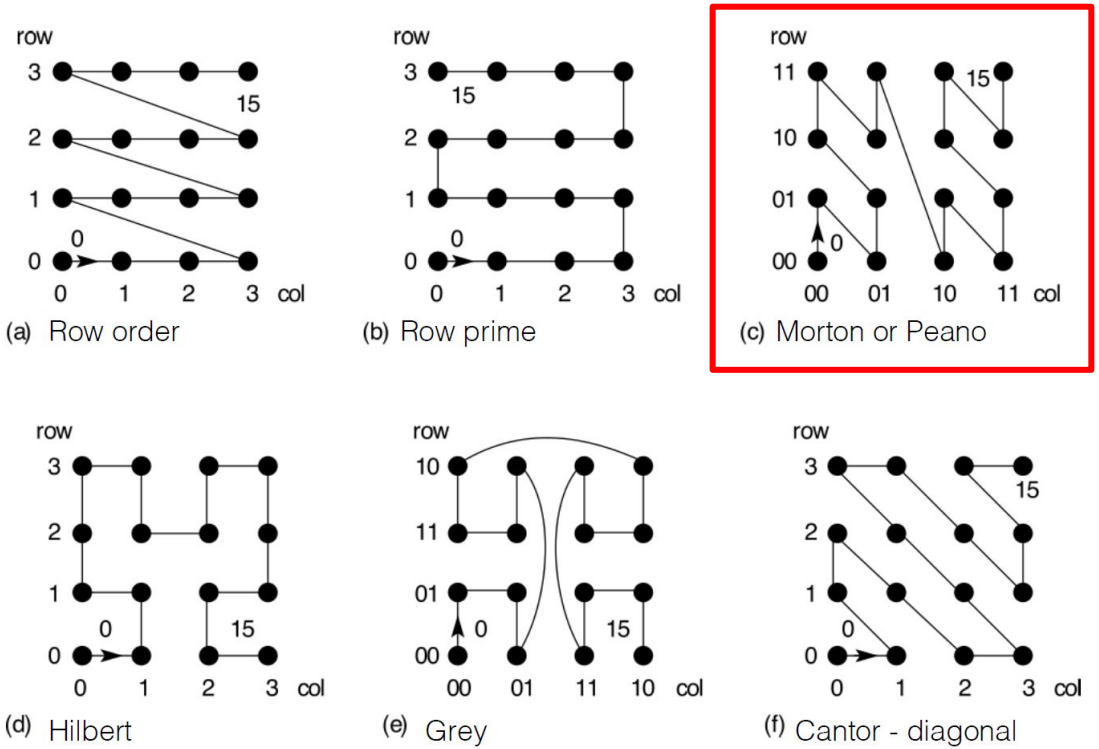
Source: https://www.hongheiniao.com

Space Filling Curve



Source: Van Oosterom, Spatial Access Methods, 1999

Space Filling Curve - We choose Morton!

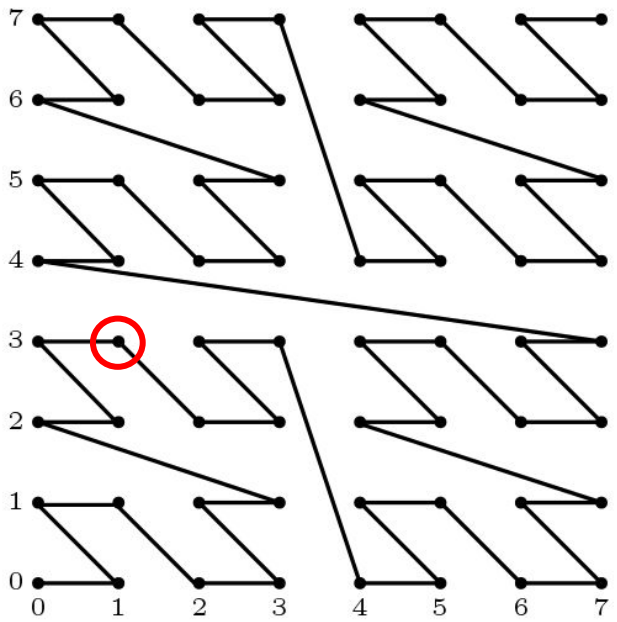


- Simplicity
- 2n-tree hierarchy

Source: Van Oosterom, Spatial Access Methods, 1999

Morton Encoding

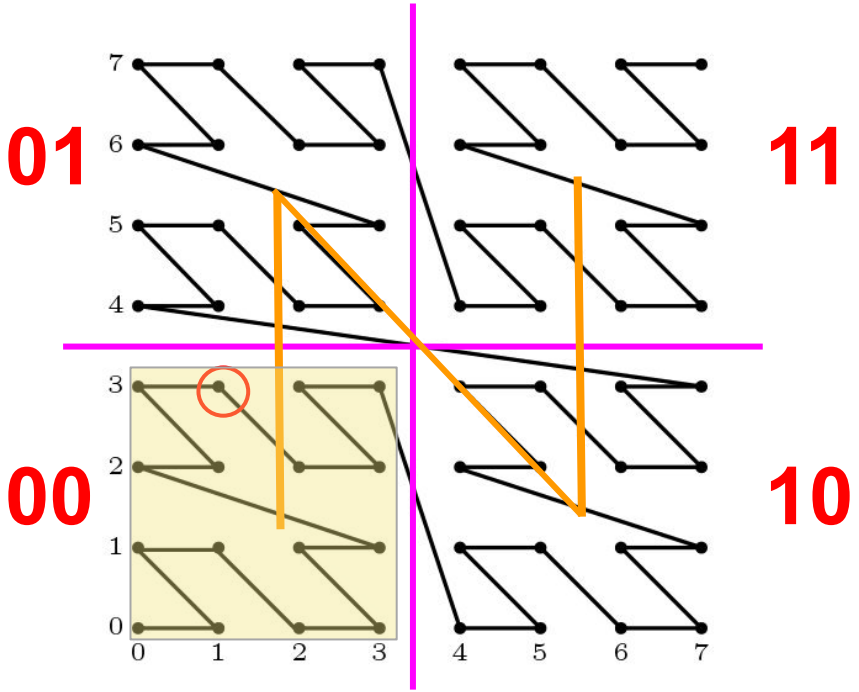
- Bitwise interleaving
  - Point (1,3)
  - x= 1 or **001** in binary
  - y= 3 or **011** in binary
  - Morton = **000111**



TU Delft

Quadtree structure in Morton keys

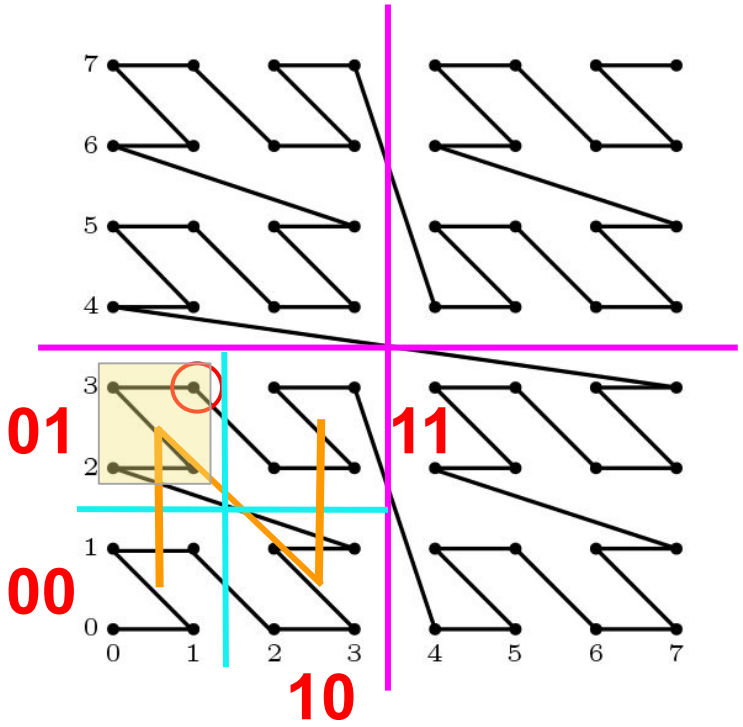
- Bitwise interleaving
  - Point (1, 3) / (**001**, **011**)
  - Morton = **000111**
- A hierarchy equivalent to a Quadtree
  - 00



TU Delft

Quadtree structure in Morton keys

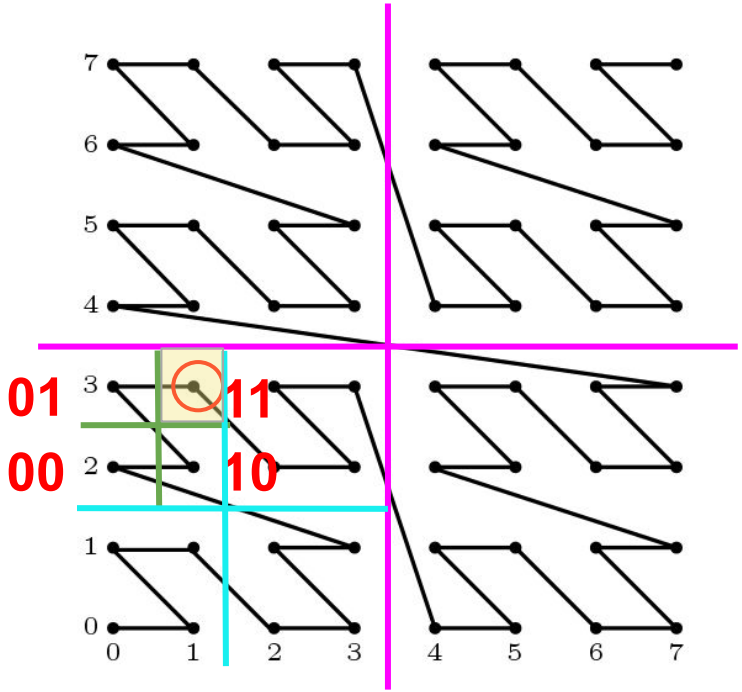
- Bitwise interleaving
  - Point (1, 3) / (**001**, **011**)
  - Morton = **000111**
- A hierarchy equivalent to a Quadtree
  - 00 01



TU Delft

Quadtree structure in Morton keys

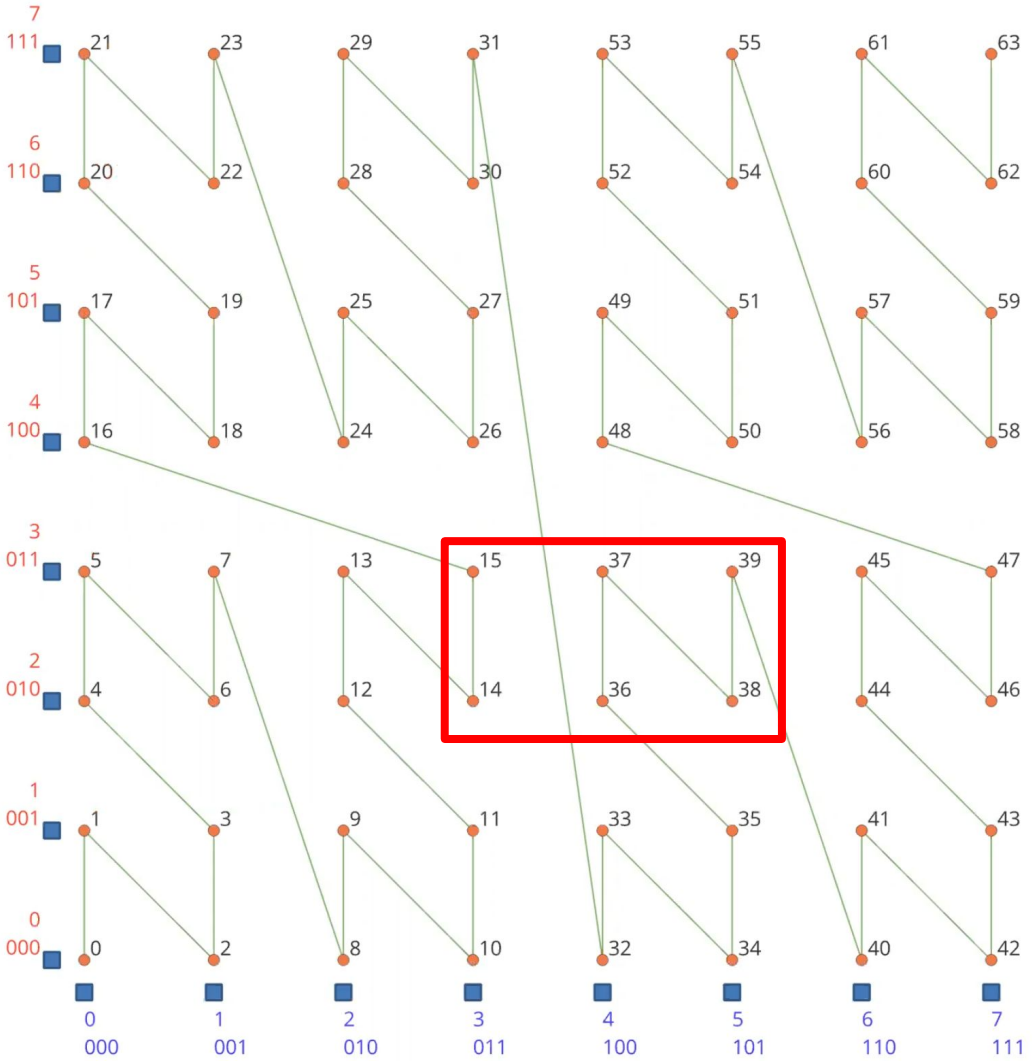
- Bitwise interleaving
  - Point (1, 3) / (**001**, **011**)
  - Morton = **000111**
- A hierarchy equivalent to a Quadtree
  - 00 01 11



TU Delft

Query

Box ((3,2), (5,3))



Overlaps

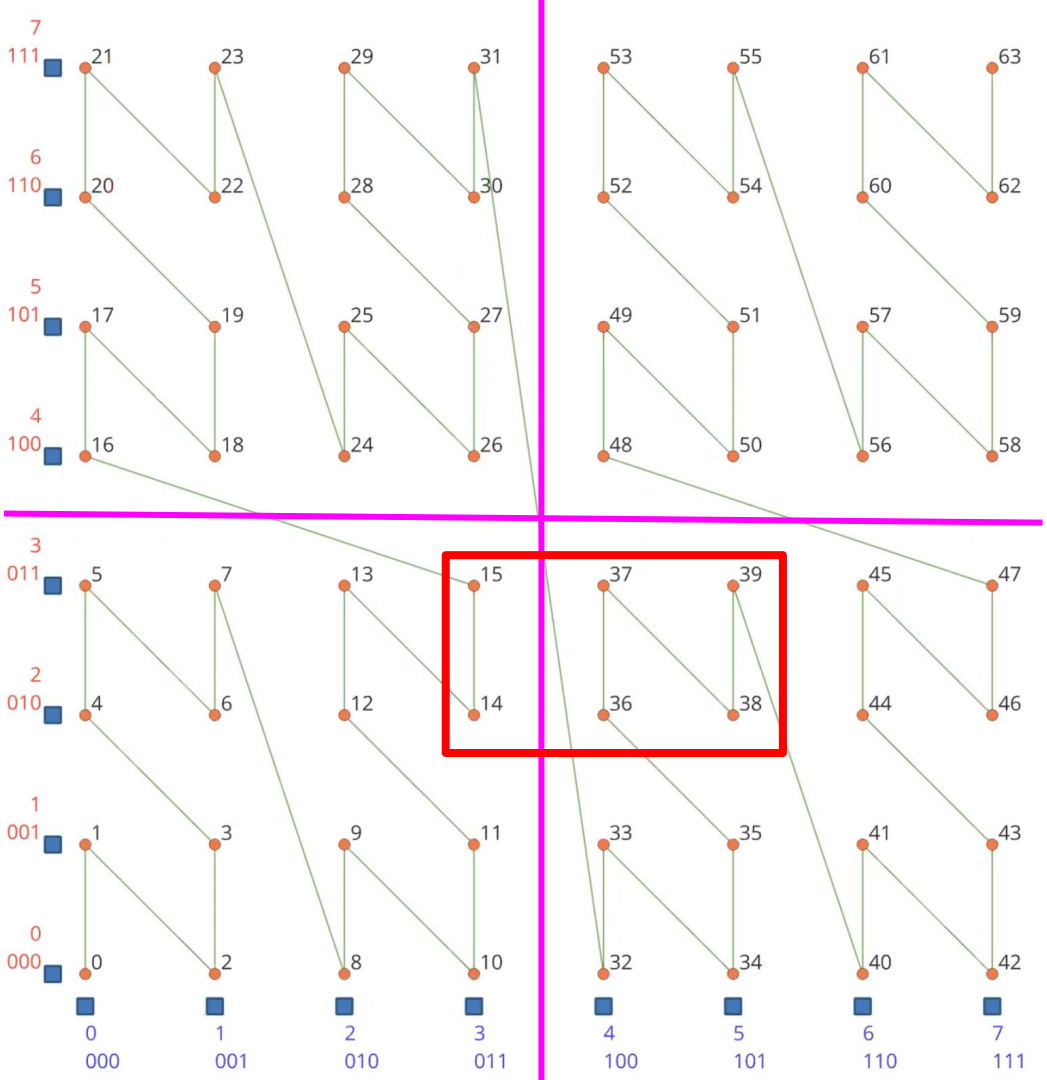
No overlappings

Fully contained

TU Delft

Query

Box ((3,2), (5,3))



Overlaps

No overlappings

Fully contained

TU Delft

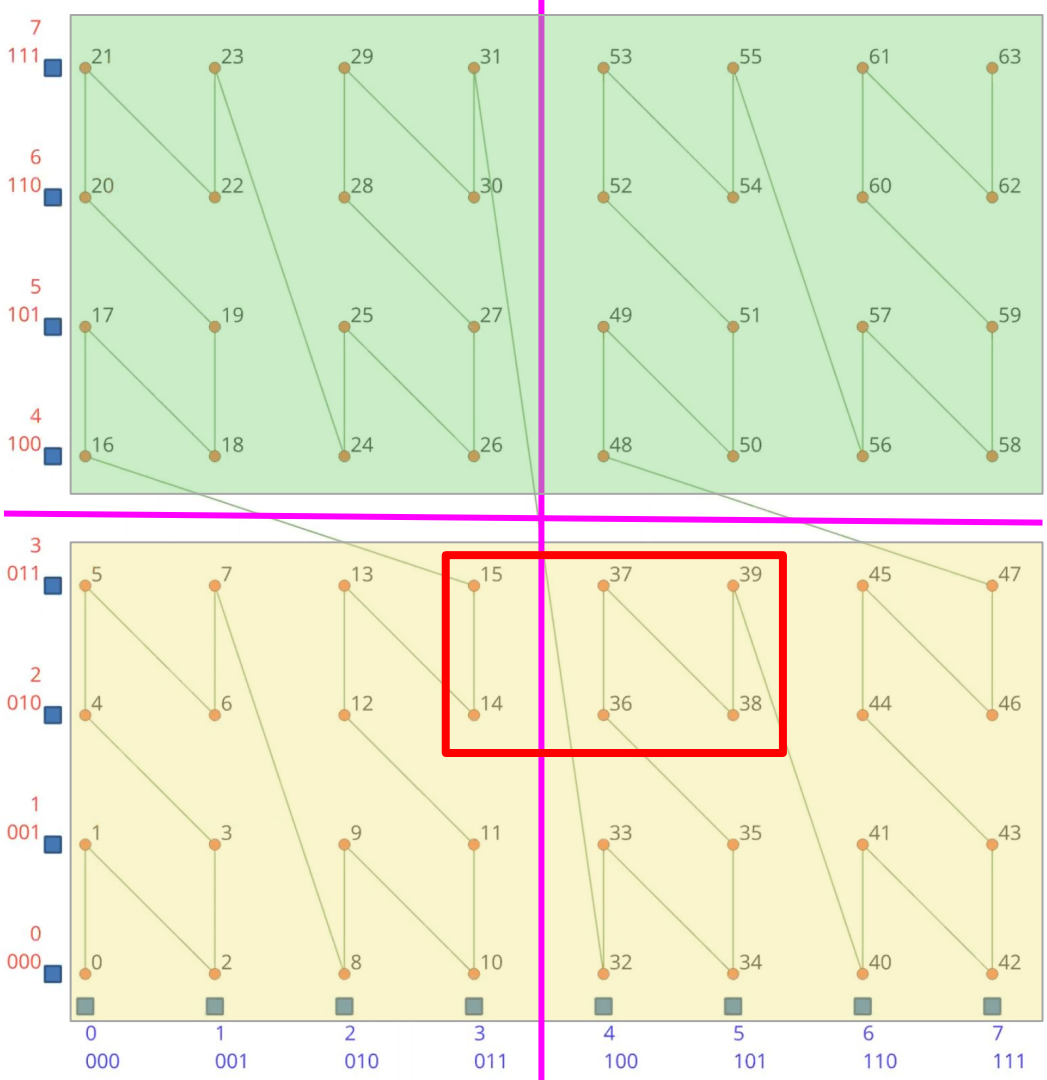
Query

Box ((3,2), (5,3))

Ranges

[0, 16)

[32, 47)



Overlaps

No overlappings

Fully contained

TU Delft

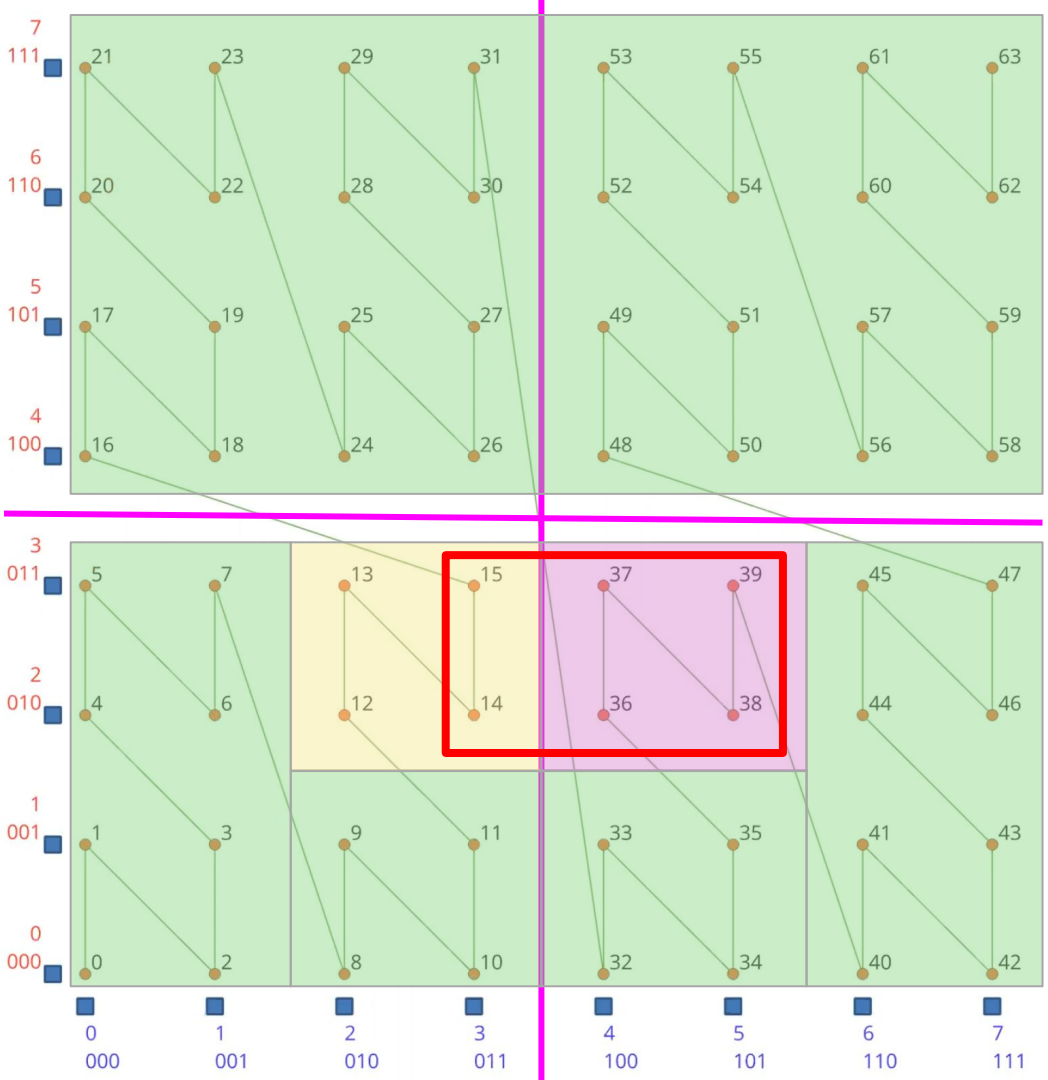
Query

Box ((3,2), (5,3))

Ranges

[12, 16)

[36, 40)



Overlaps

No overlappings

Fully contained

TU Delft

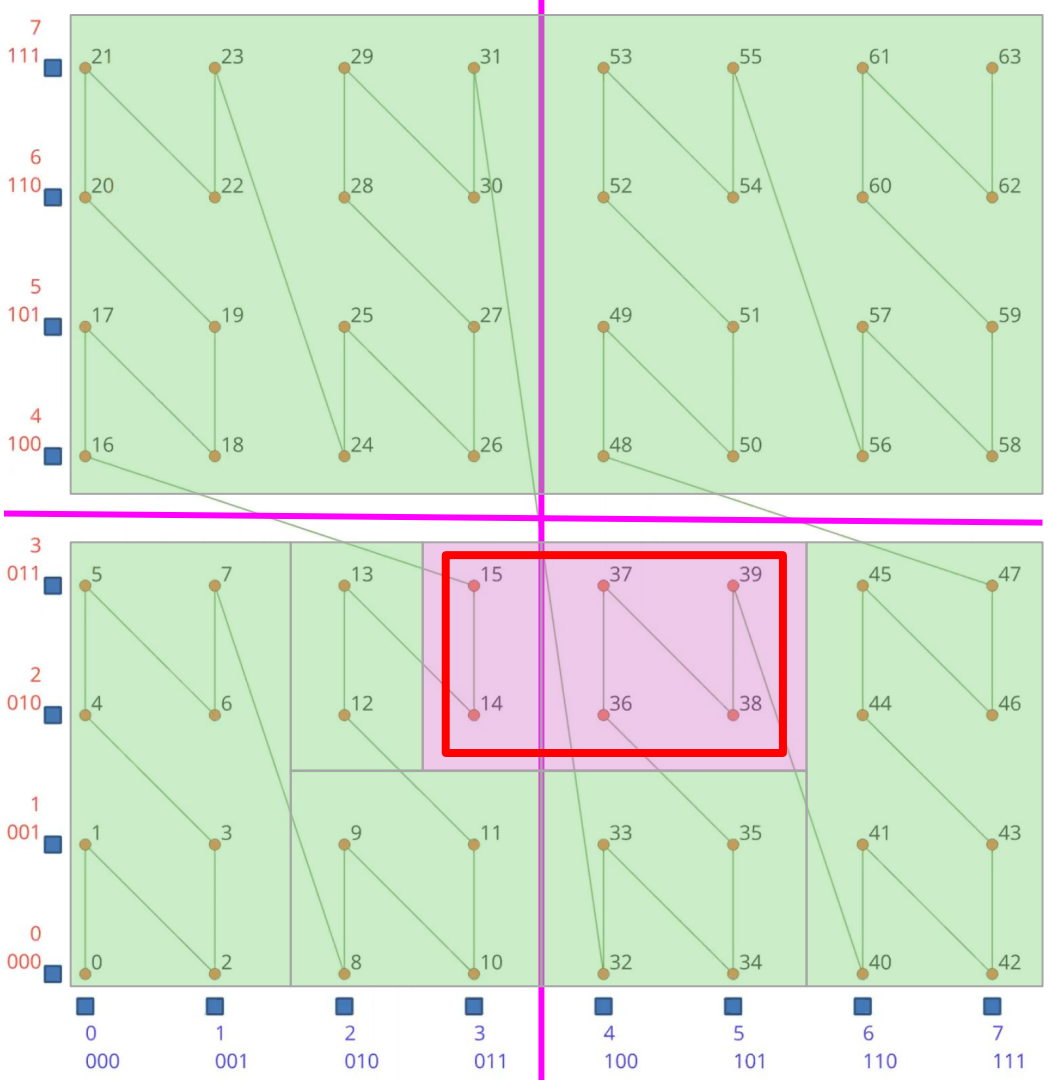
Query

Box ((3,2), (5,3))

Ranges

[14, 16)

[36, 40)



Overlaps

No overlappings

Fully contained

TU Delft

Using SFC to organize points in flat table

X	Y	Z	...
1	1	3.5	...
0	3	0.4	...
1	5	10.8	...
3	4	5.9	...
6	4	2.7	...

Normal

Morton key	Z	...
00000011	3.5	...
00000101	0.4	
00010011	10.8	...
00011010	5.9	...
00111000	2.7	...

PlainSFC

Van Oosterom et al., 2015

- Point are clustered
- Query efficiency improved
- Still flat table

TU Delft

03

The SplitSFC Method

TU Delft

Numerical meaning of the split

Morton key	Z	...
00000011	3.5	...
00000101	0.4	
00010011	10.8	...
00011010	5.9	
00111000	2.7	...

Numerical meaning of the split

Morton key	Z	...
00000011	3.5	...
00000101	0.4	
00010011	10.8	...
00011010	5.9	
00111000	2.7	...

- Noticing same parts in Morton keys
- **Can we split the Morton key and make blocks out of it?**

Numerical meaning of the split

Morton key	Z	...
00000011	3.5	...
00000101	0.4	
00010011	10.8	...
00011010	5.9	
00111000	2.7	...

SFC head	SFC tile	Z	...
0000	[0000,0001, 0011,1000]	[3.5, 0.4, 2.9, 5.0]	...
0001	[0000,0001, 0011,1000]	[7.5, 1.6, 10.8, 4.3]	...
0011	[0000,0001, 0011,1000]	[12.4, 2.7, 0.2, 6.1]	...

TU Delft

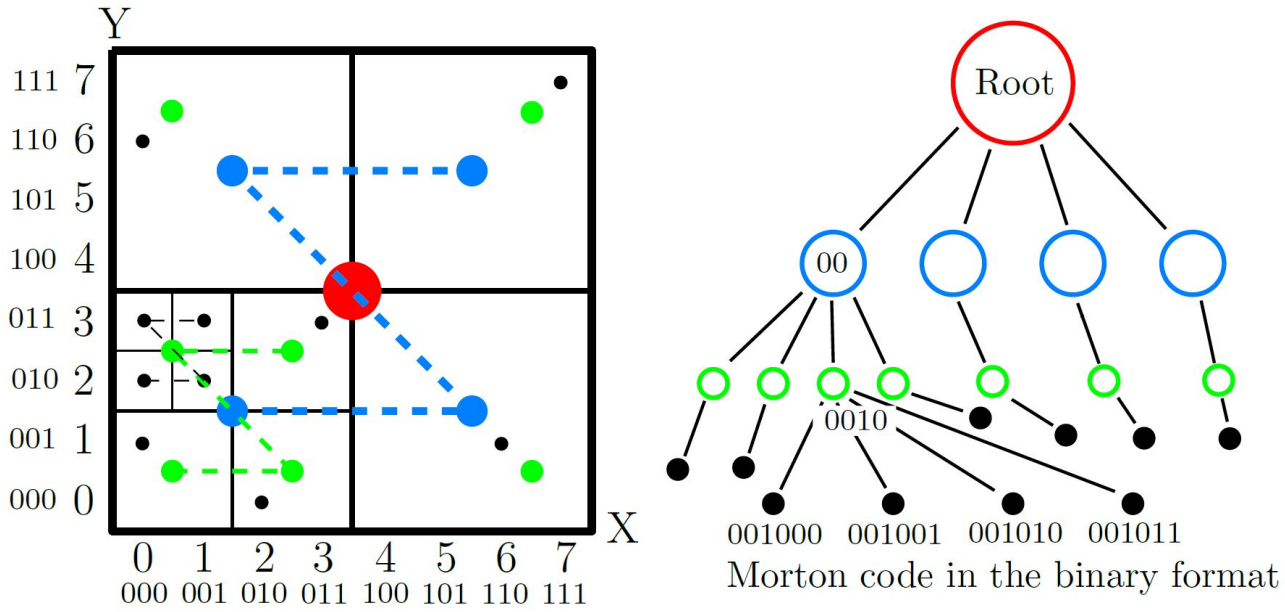
Numerical meaning of the split

- Storage
  - Block model, compact & compressed
  - SFC head stored once, shared with many
- Querying
  - The geometric meaning of the split?

SFC head	SFC tile	Z	…
0000	[0000,0001, 0011,1000]	[3.5, 0.4, 2.9, 5.0]	…
0001	[0000,0001, 0011,1000]	[7.5, 1.6, 10.8, 4.3]	…
0011	[0000,0001, 0011,1000]	[12.4, 2.7, 0.2, 6.1]	…

TU Delft

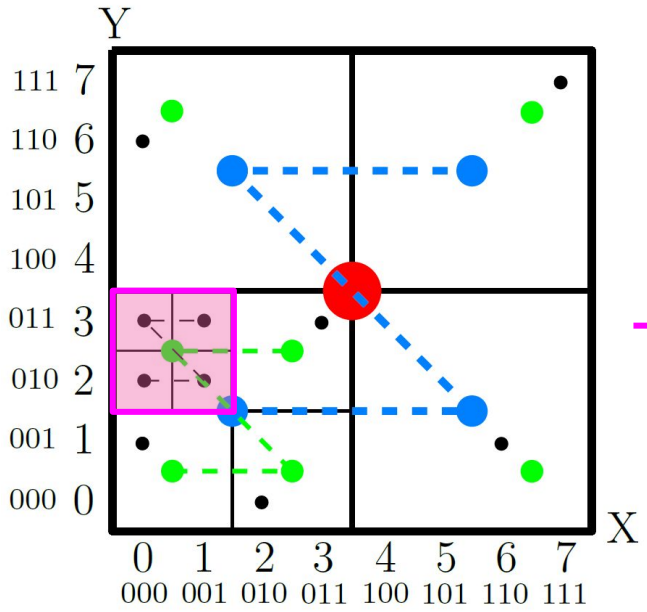
Geometric meaning of the split

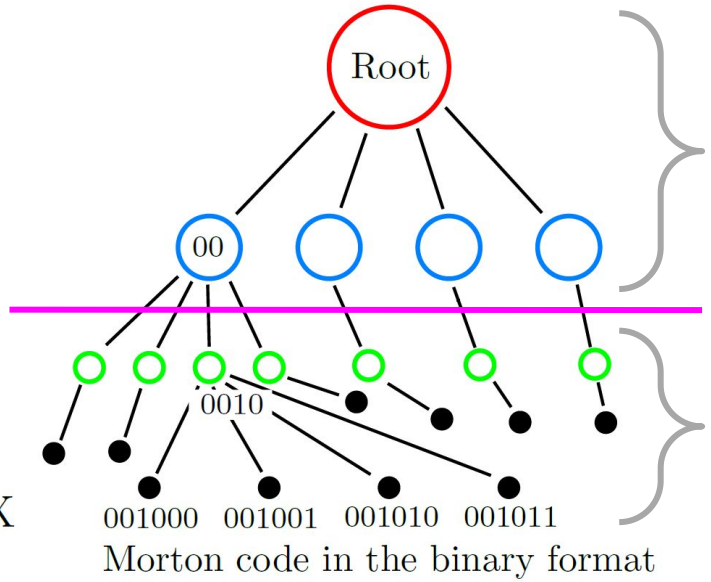


Source: H Liu, 2022. nD-PointCloud Data Management.

TU Delft

Geometric meaning of the split





SFC head:

Location of the quadrant

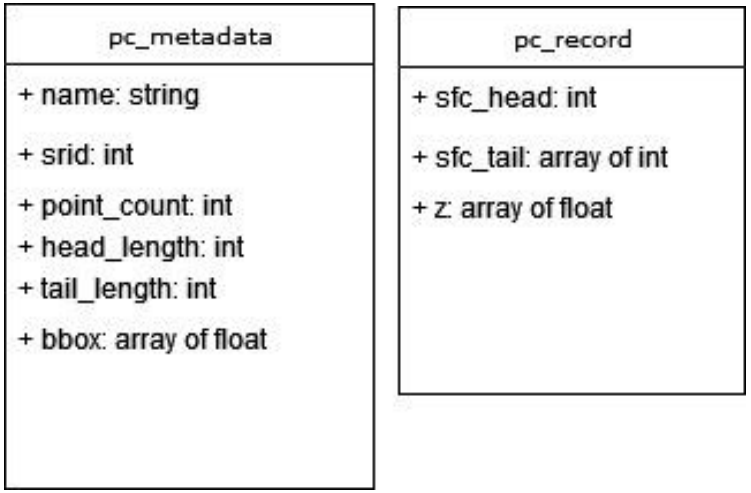
SFC tail:

Specific location of a point inside the quadrant

Source: H Liu, 2022. nD-PointCloud Data Management.

TU Delft

Storage Model



Points
- Organizing dimensions → SFC key (Morton)
  - SFC head
  - SFC tail
- Associate dimensions

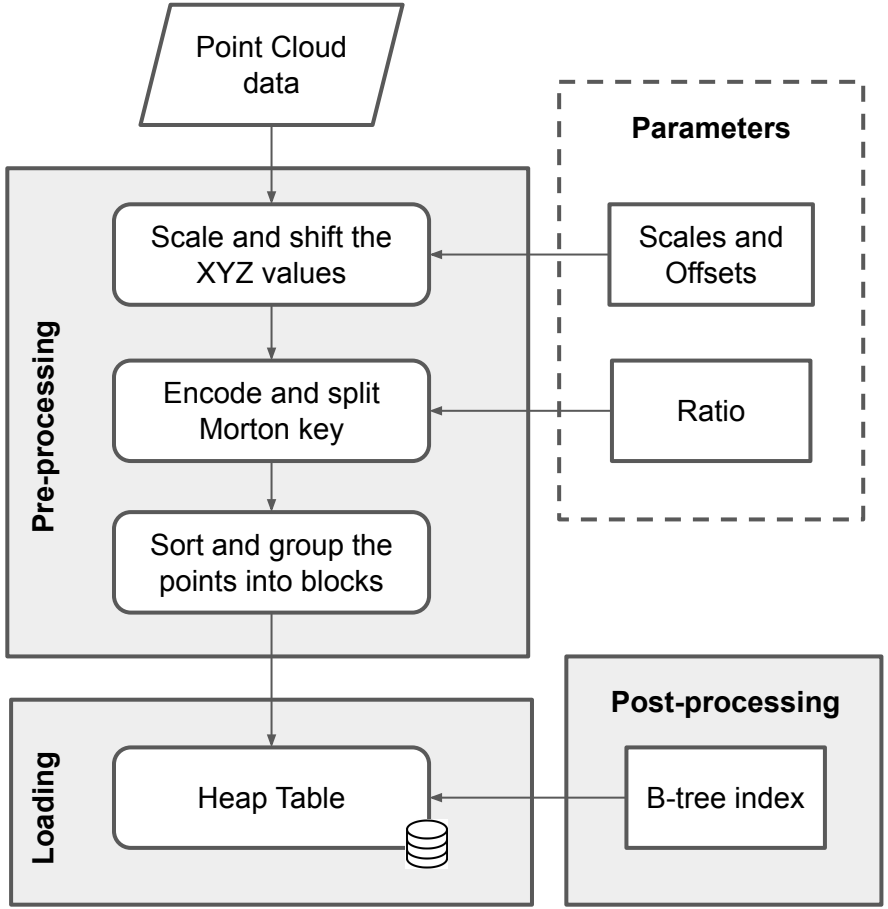
A block
- Points with the same SFC head
- Sorted by SFC tail

TU Delft

Storage Model

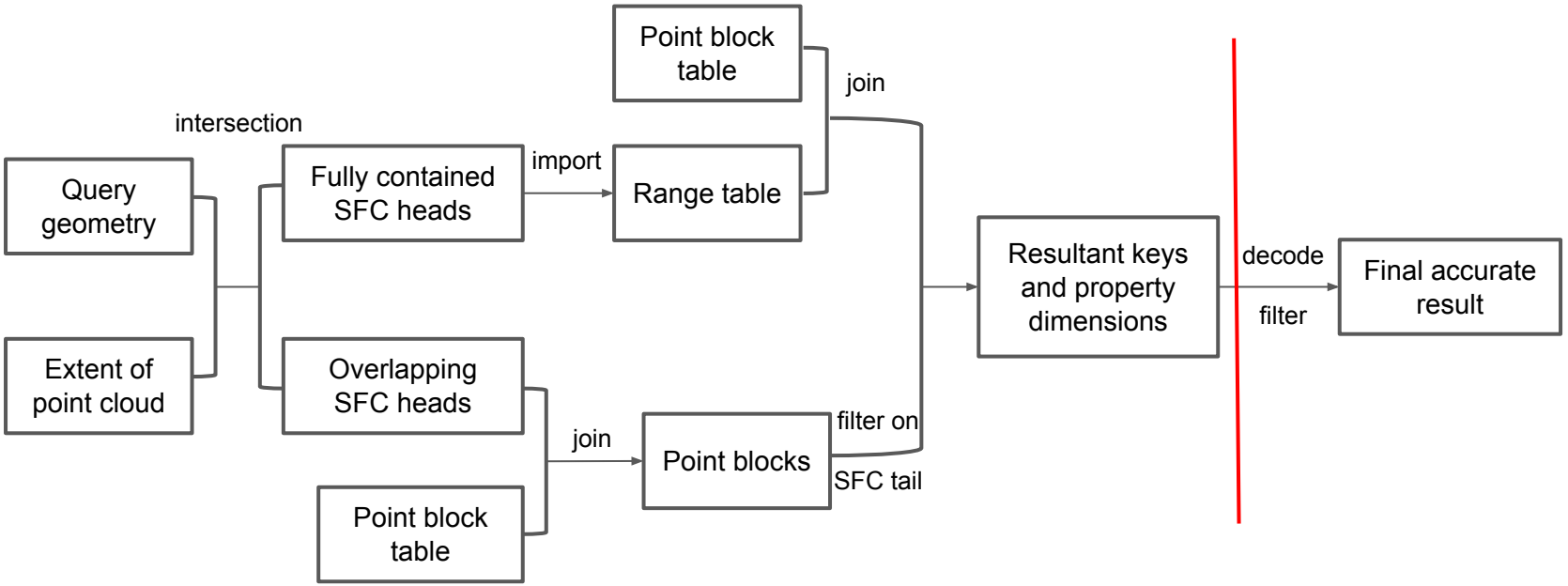
SFC head	SFC tile	Z	…
0000	[0000,0001,0011,1000]	[3.5, 0.4, 2.9, 5.0]	…
0001	[0000,0001,0011,1000]	[7.5, 1.6, 10.8, 4.3]	…
0011	[0000,0001,0011,1000]	[12.4, 2.7, 0.2, 6.1]	…

Loading Procedure



TU Delft

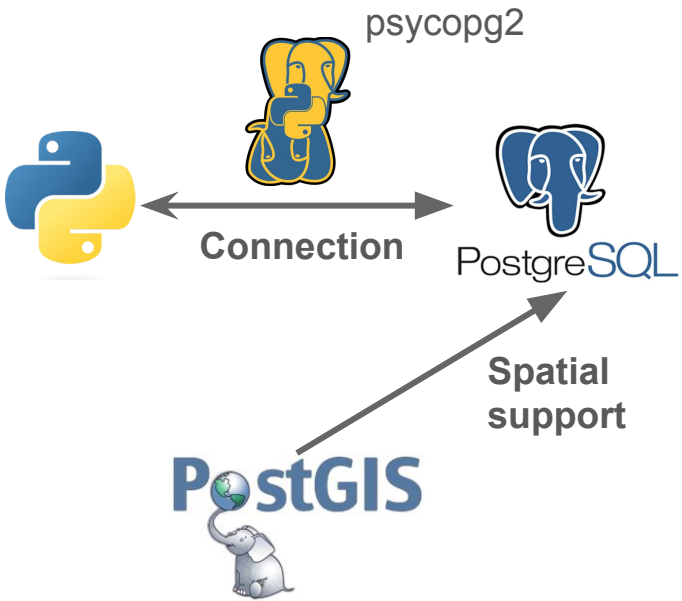
Query Procedure



TU Delft

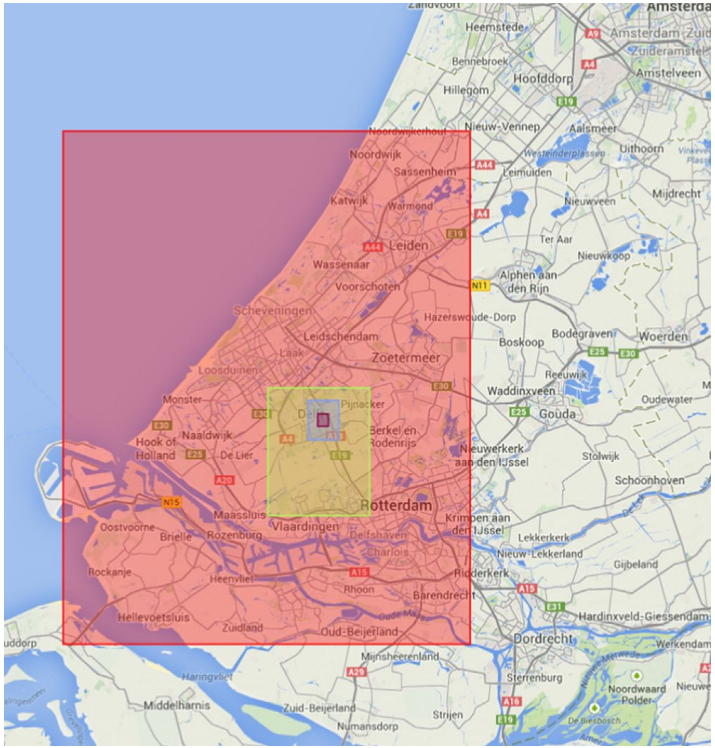
Implementation

- Hardware & software
  - PostgreSQL, PostGIS: DBMS
  - Python: complex algorithms
  - Pakhuis: the supercomputer of GIST Group
- Products
  - Importer (las2pg)
  - Querying tools
  - Exporter (pg2las)



Benchmark

- A scalable framework
  - AHN2
  - From Delft Campus to South Holland
  - Proposed by van Oosterom et al (2015)
- Why?
  - Reproductive, as AHN2 is open data
  - Well-designed and comprehensive
  - Easy to compare with other solutions

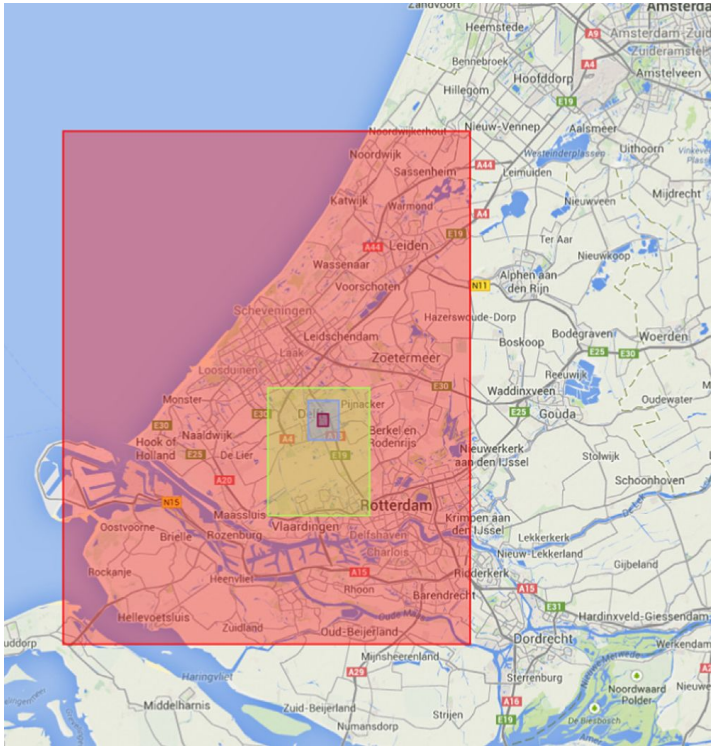


Source: van Oosterom et al., 2015. Massive point cloud data management: Design, implementation and execution of a point cloud benchmark.

Benchmark

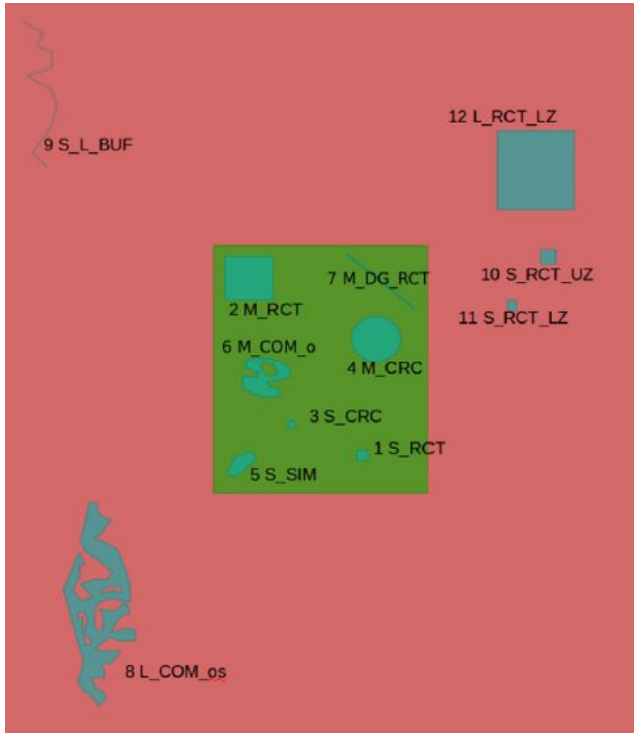
- Measure performance
  - Importing time
  - Storage size
  - Query response time

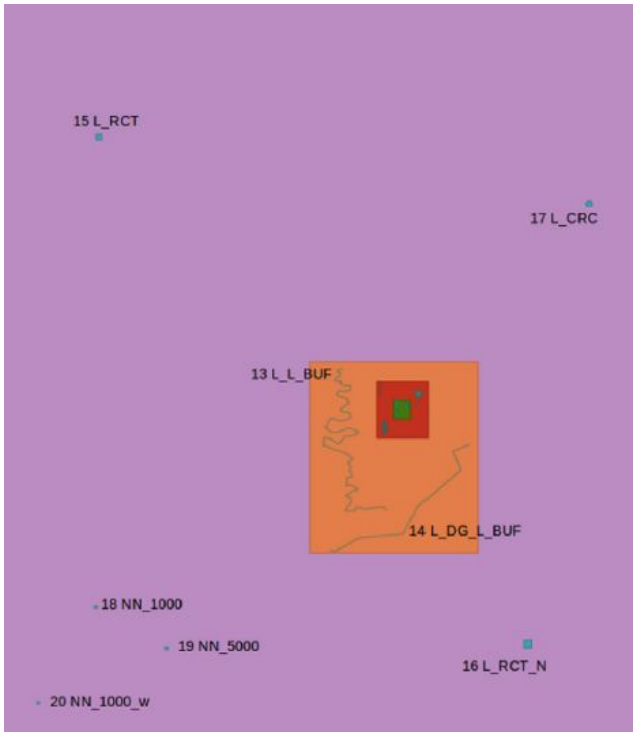
Point	File	Size	Description
20M	1	0.4	TU Delft Campus (purple)
210M	1	4	Major of Delft city (cyan)
2201M	153	42	Delft city and surroundings (green)
23090M	1492	440	Major of South Holland (red)



Source: van Oosterom et al., 2015. Massive point cloud data management: Design, implementation and execution of a point cloud benchmark.

Benchmark





TU Delft

Source: van Oosterom et al., 2015. Massive point cloud data management: Design, implementation and execution of a point cloud benchmark.

04

Results

TU Delft

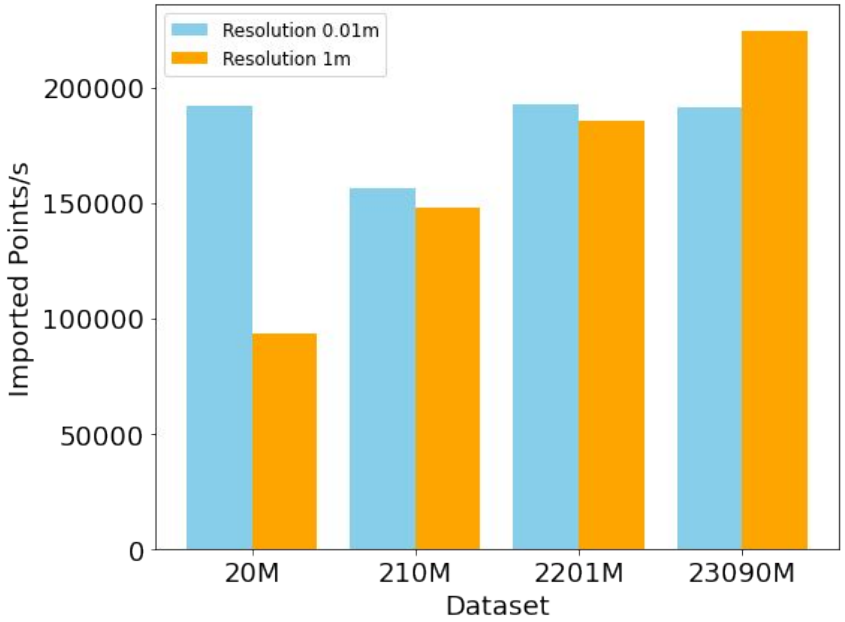
Mini-benchmark

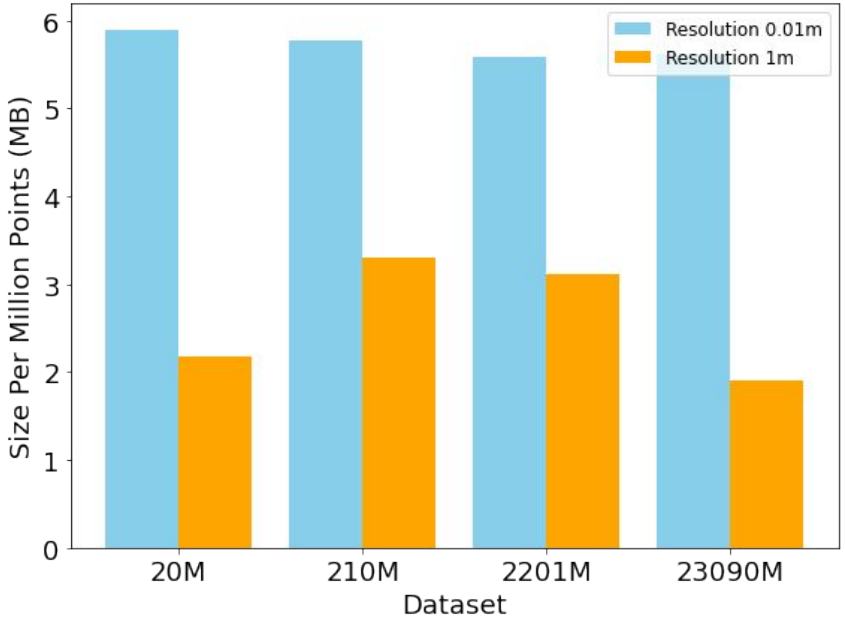
- To explore the effects of the parameters, including scales, offsets and ratios
- 20M, 210M dataset

	Explanation	Scales	Offsets
Solution 1	Default value	[0.01, 0.01]	[0, 0]
Solution 2	The points are shifted to the corner of their bounding box	[0.01, 0.01]	[min_x, min_y]
Solution 3	The resolution is reduced to 1m	[1, 1]	[0, 0]

TU Delft

Scales



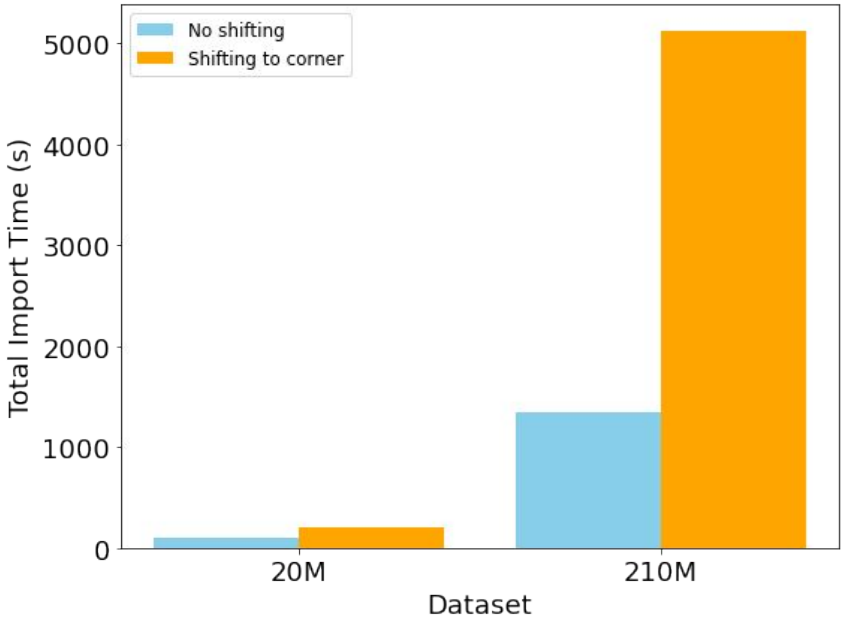


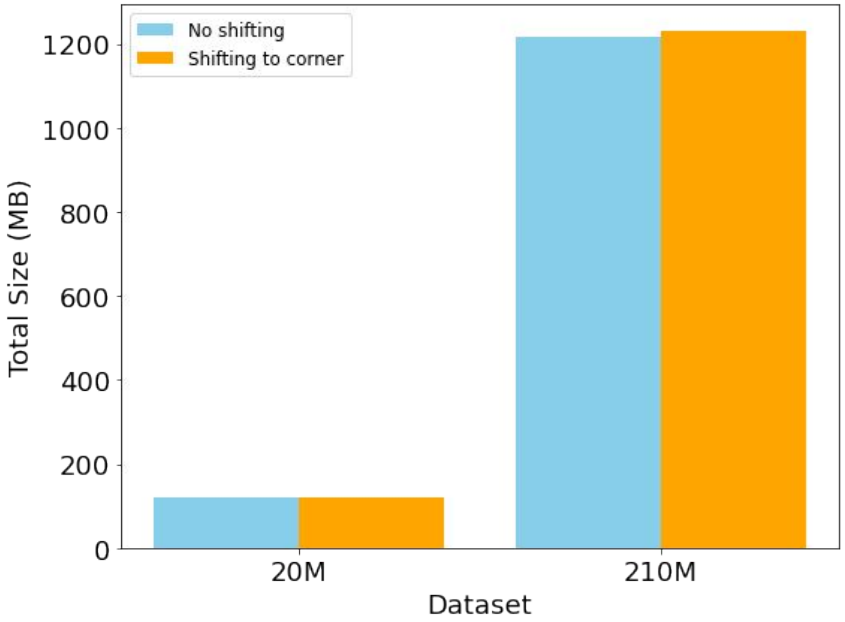
TU Delft

☺ Saves storage space

☹ Little help with importing time

Offsets

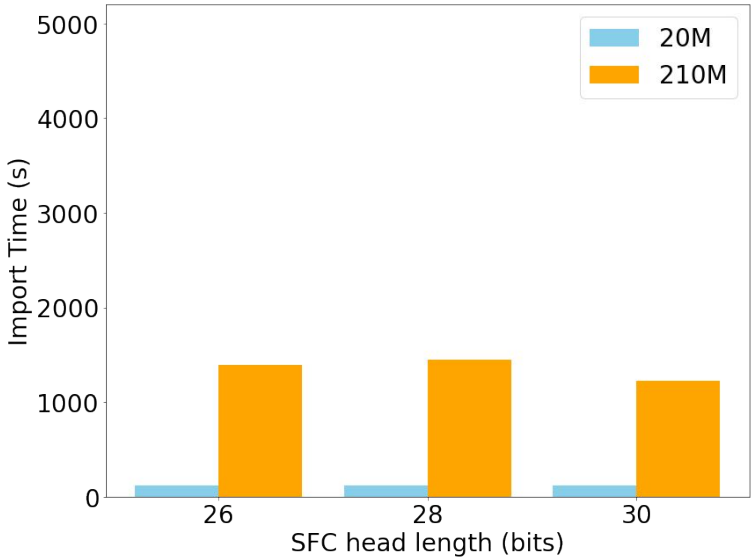


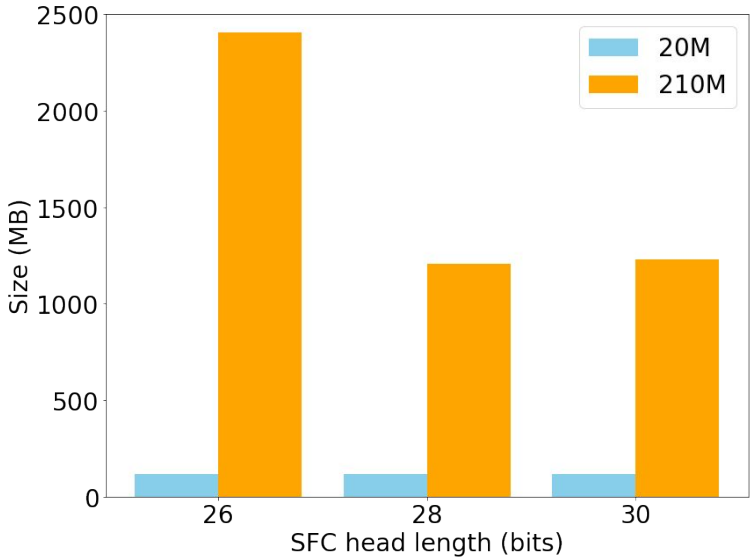


☹ No significant differences for the storage size

TU Delft

Ratios (default mode)



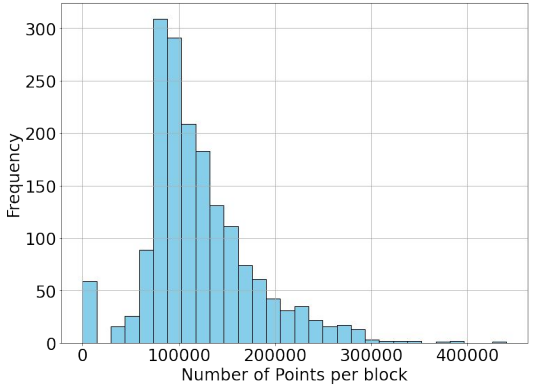


The ratio is dataset specific, and its tuning needs experiments.

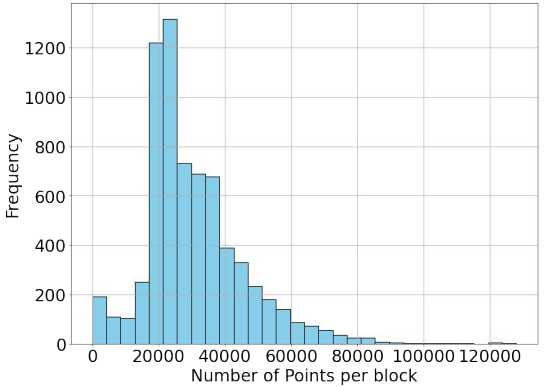
The details effects are not clear yet, but the ratio of AHN2 can be 50%~60%.

TU Delft

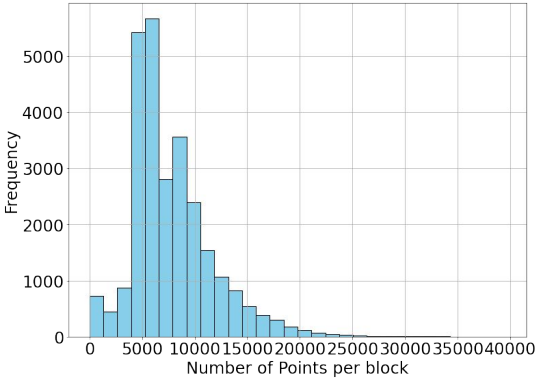
Point distribution (210M dataset)



50% split



55% split



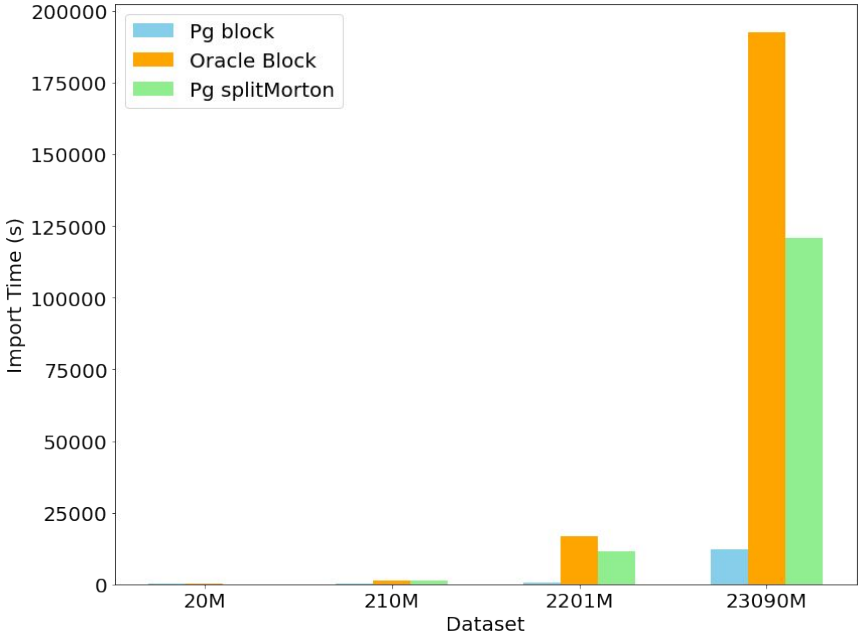
60% split

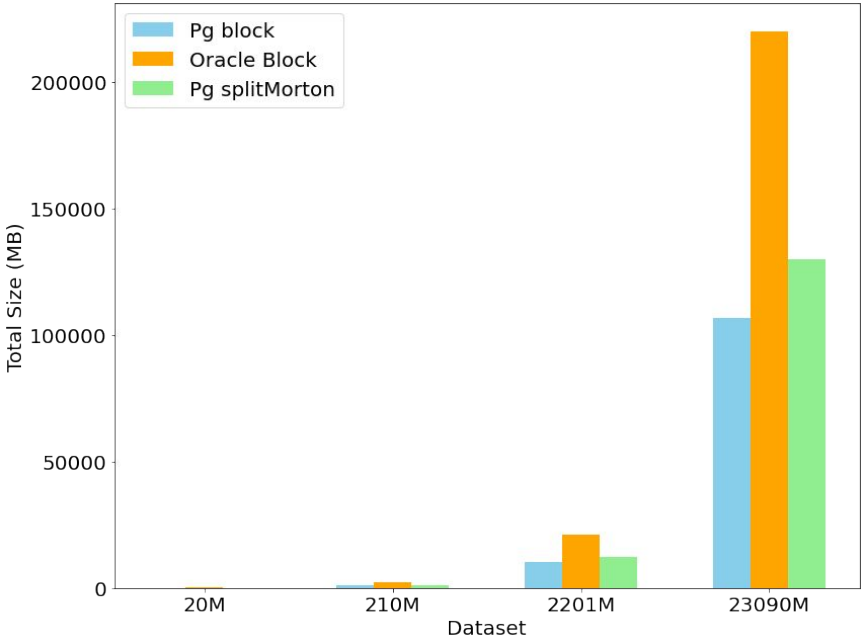
TU Delft

Medium-benchmark

- To explore the scaling behaviour
- To evaluate and compare the performance
  - SplitSFC
  - pgPointCloud
  - Oracle SDO_PC

Importing time & Storage Size

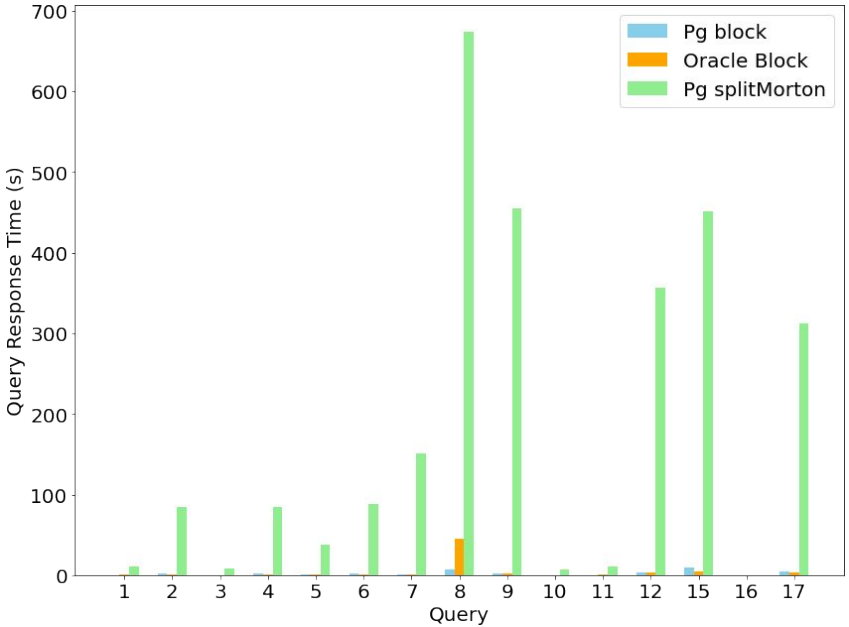




pgPointCloud > SplitSFC > Oracle SDO_PC

TU Delft

Data retrieval



pgPointCloud ~ Oracle SDO_PC >> SplitSFC

Possible cause: Morton decoding, Python implementation

TU Delft

Medium-bench In Summary

- **Importing Time.** The SplitSFC approach takes longer time to import, because of the extra time in Morton key computation, sorting and grouping and the nature of Python implementation.
- **Storage size.** pgPointCloud and SplitSFC approach occupy comparable storage space, and the Oracle SDO_PC approach occupies more storage space. Notably, if the algorithm and the implementation are improved, there is a chance that SplitSFC can have a very good performance compared to other state-of-the-art DBMS solutions.
- **Querying response time.** SplitSFC is much slower in querying. However, this may also be due to the nature of Python implementation.

05

Conclusion & Discussion

TU Delft

Conclusions

Does employing the split Space Filling Curve approach constitute an effective strategy for managing massive point clouds in a relational DBMS?

 SplitSFC approach has potential advantages for massive point clouds in DBMS

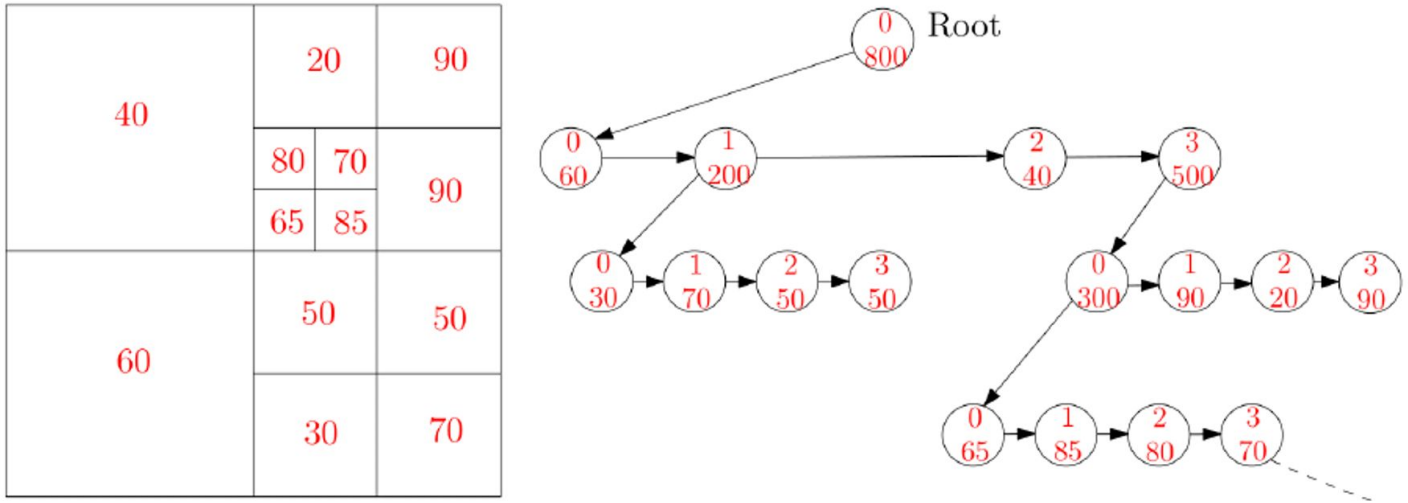
 The immediate implementation may not yield a definitive improvement

- Fixed-length SFC key splitting
- Interpreted nature of Python
- Parallel Loading

TU Delft

Future work

- Adaptive splitting algorithm
  - HistogramTree: the distribution of the points



Source: H Liu, 2022. nD-PointCloud Data Management.

Future work

- Adaptive splitting algorithm
  - HistogramTree: the distribution of the points
- 3D Morton key
  - How to encode the z dimension?
- Multi-split algorithm
  - i.e. head, body, tail
- Implementation with C++

Thank you for listening!