https://geospatialworld.net/news/lizardtech-
warded-us-patent-lidar-point-cloud-compression-2/

FractalCloud:

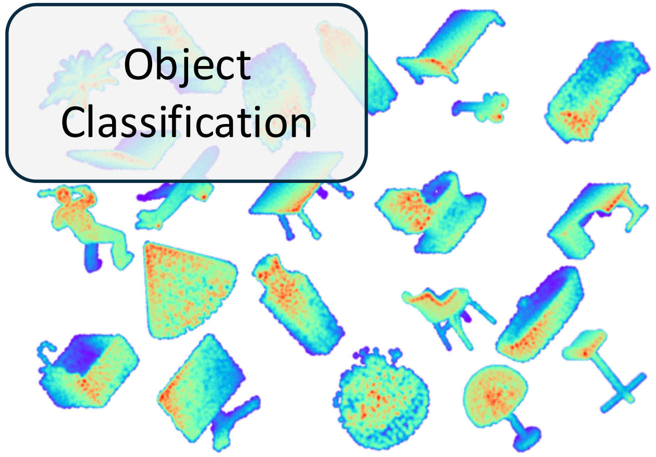
A **Fractal**-Inspired Architecture for Efficient Large-Scale Point **Cloud** Processing

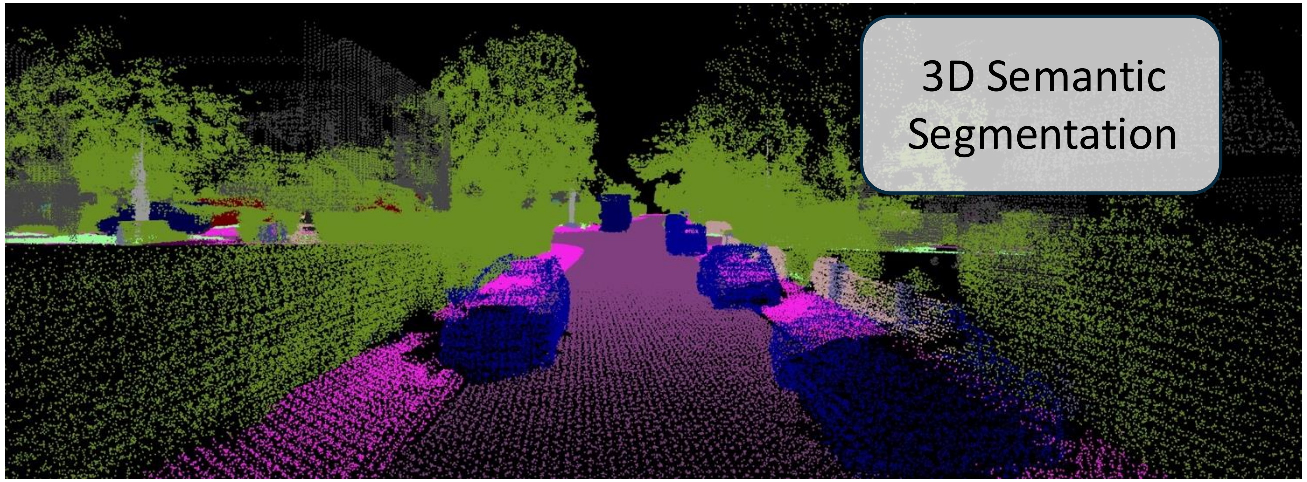
HPCA 2026

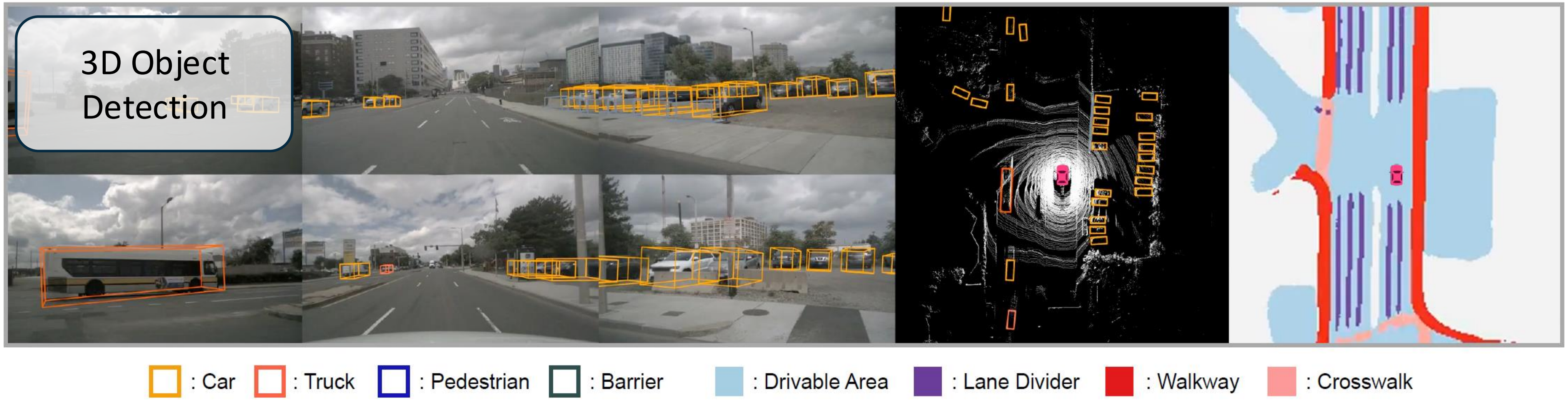
Yuzhe Fu, Changchun Zhou, Hancheng Ye, Bowen Duan, Qiyu Huang, Chiyue Wei, Cong Guo, Hai Li, and Yiran Chen

Department of Computer Science, Duke University
Department of Electrical and Computer Engineering, Duke University
Duke University Center for Computational Evolutionary Intelligence (CEI)

Point cloud in deep learning: PNN







Qi et al., PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, CVPR 2017
Vallet et al., TerraMobilita/IQmulus urban point cloud analysis benchmark, CG 2015.
Tang et al., Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution, ECCV 2020.

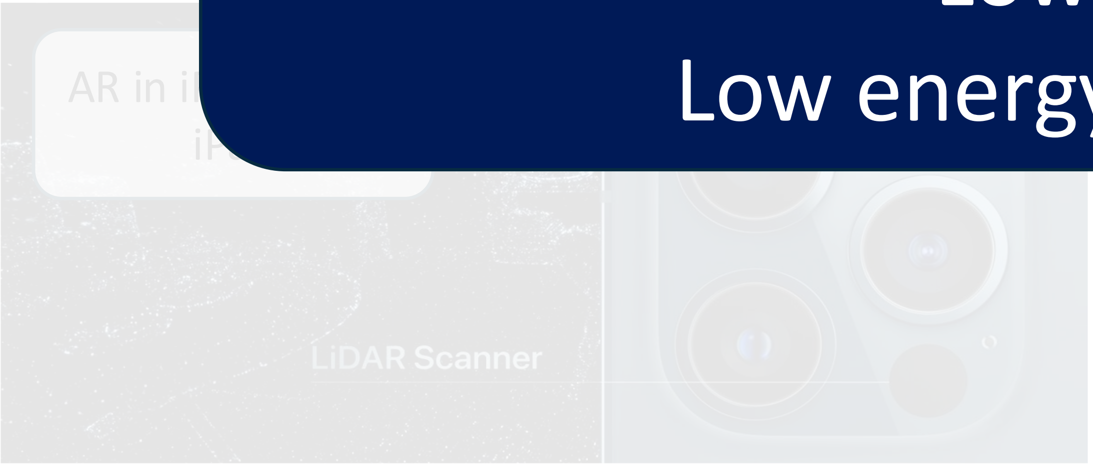
Point cloud in daily life

https://www.softwareone.com/en/insights
iPad - Apple

5 ways LiDAR is transforming the world before our eyes
Tourists scaling the Great Wall of China can now get takeout delivered by drone | CNN Business

Duke

Point cloud in daily life



Efficiency is important

Low latency

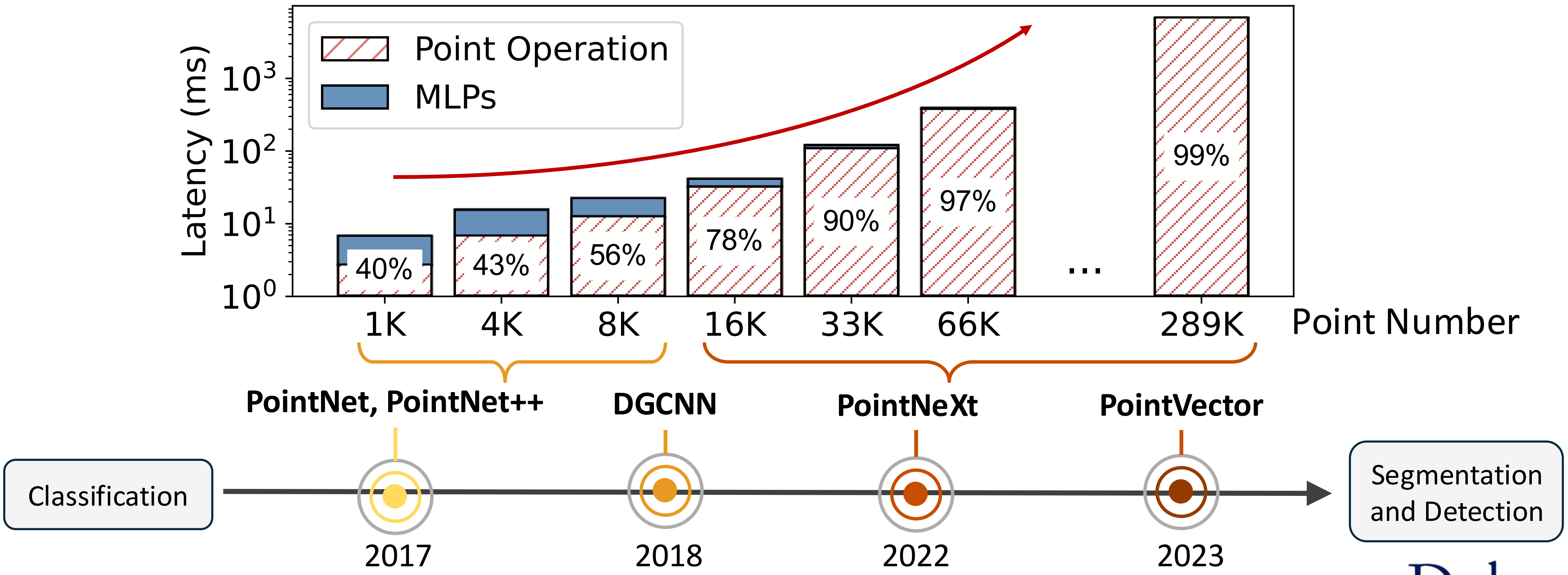
Low energy consumption

https://www.softwareone.com/en/insights
iPad - Apple

5 ways LiDAR is transforming the world before our eyes
Tourists scaling the Great Wall of China can now get takeout delivered by drone | CNN Business

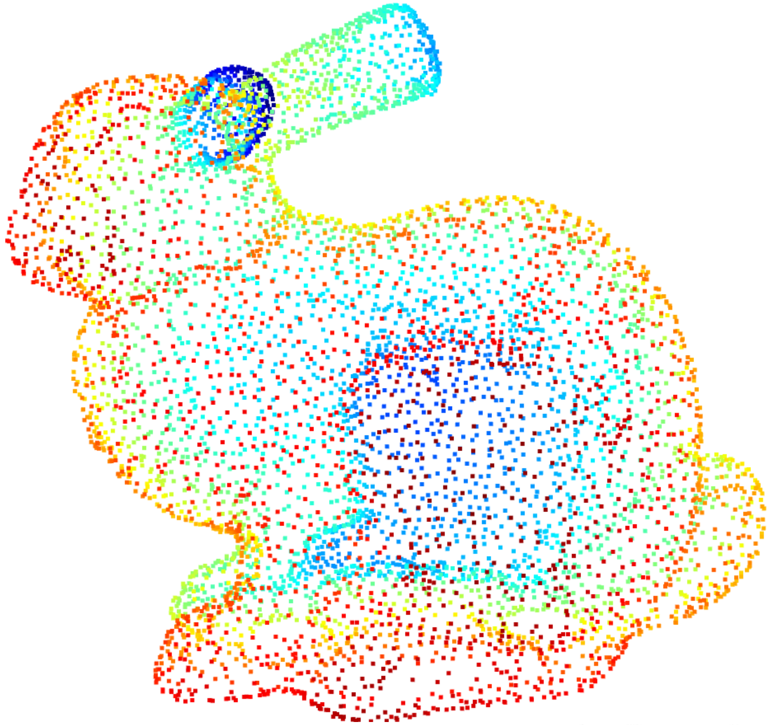
Poor scaling in PNNs

Bottleneck shift: from MLPs to Point Operations



Duke

Point cloud Data



2D Image

Pixels: RGB values

Structured in memory

idx	Pixels
0	p0
1	p1
2	p2
3	p3
…	……
62	p62
63	p63

Neural Network

3D Point Cloud

Points: (x, y, z), Feature, …

Unordered in Memory

idx	Points
0	p16
1	p0
2	p27
3	p2
…	……
62	p1
63	p22

Neural Network

Zhou et al., Open3D: A modern library for 3D data processing. arXiv, 2018.
https://pixabay.com/zh/photos/rabbit-nature-wildlife-animal-5469252/

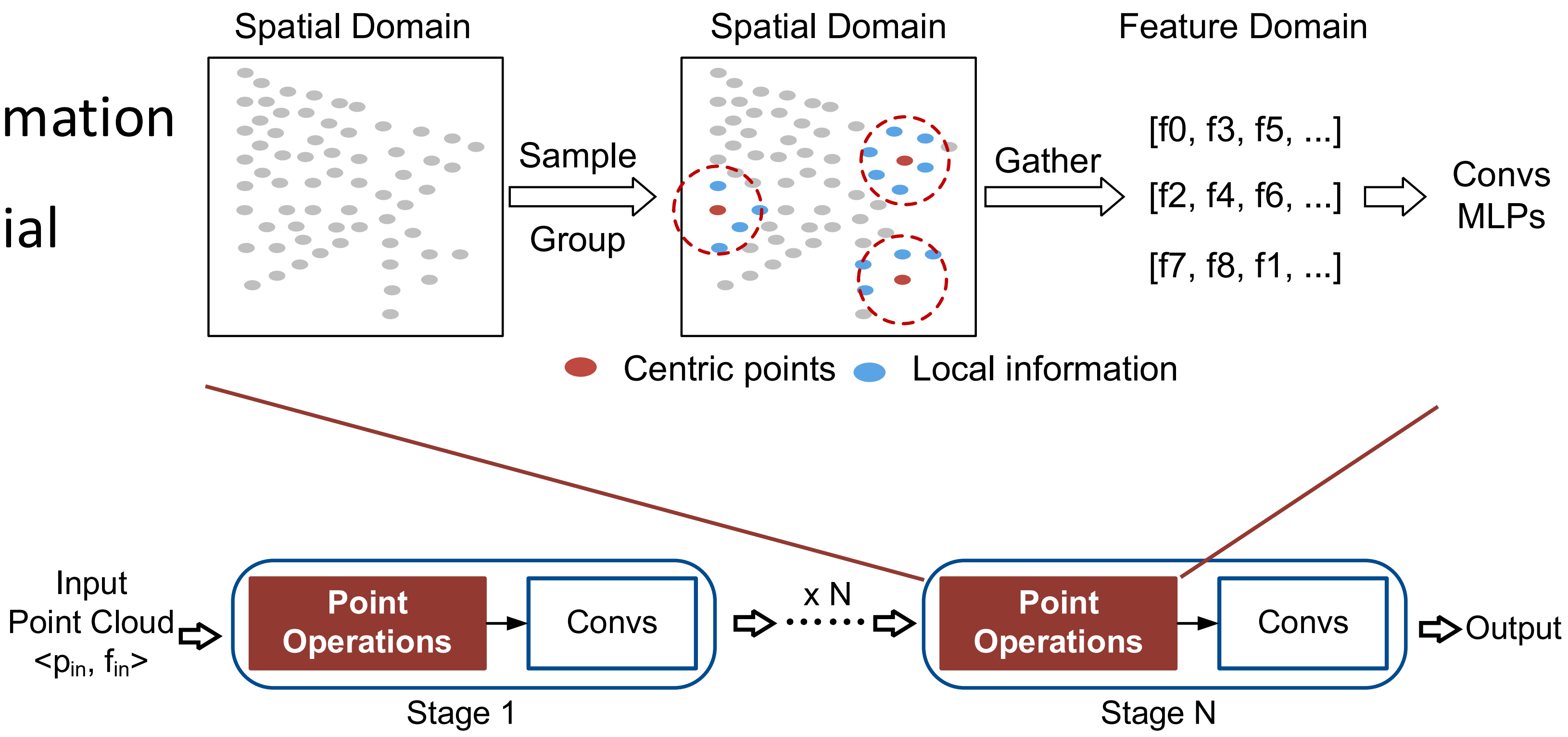
Duke

Point operations in PNNs

Bottleneck shift: from MLPs to Point Operations

- **Sample**: Centric points
- **Neighbor Search**: local information
- **Gather**: Map data from spatial domain to feature domain

- **All-to-All Computing**
- **Global Memory Scan**
- **Iterative Computing**



The backbone of PNNs

Duke

Point operations in PNNs

Bottleneck shift: from MLPs to Point Operations

- Samp
- Neig
- Gath

dom

Bottleneck shifts to point operations.
Partition can help.

- Irregu
- Iterative Computing
- All-to-All Computing

The backbone of PNNs

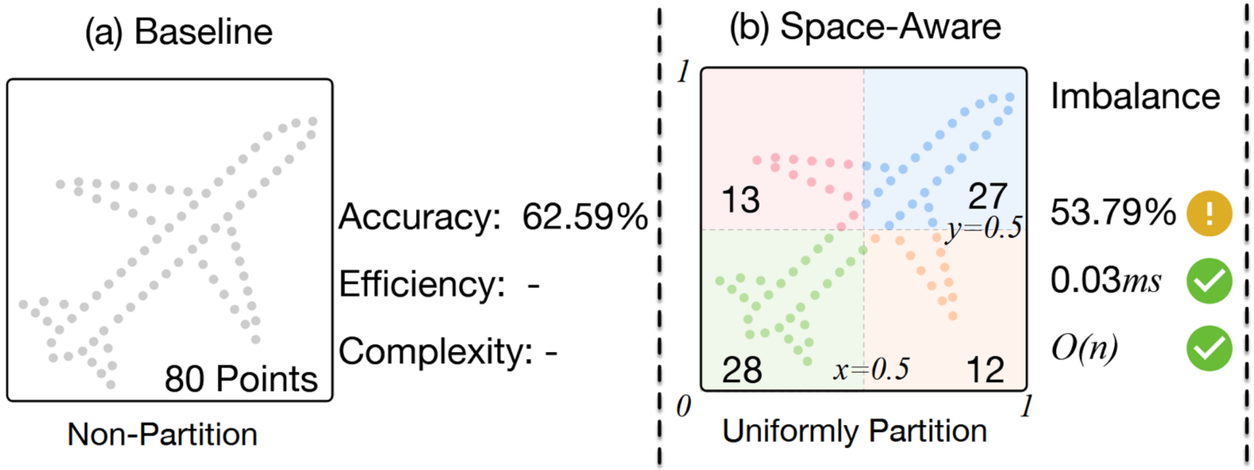
Duke

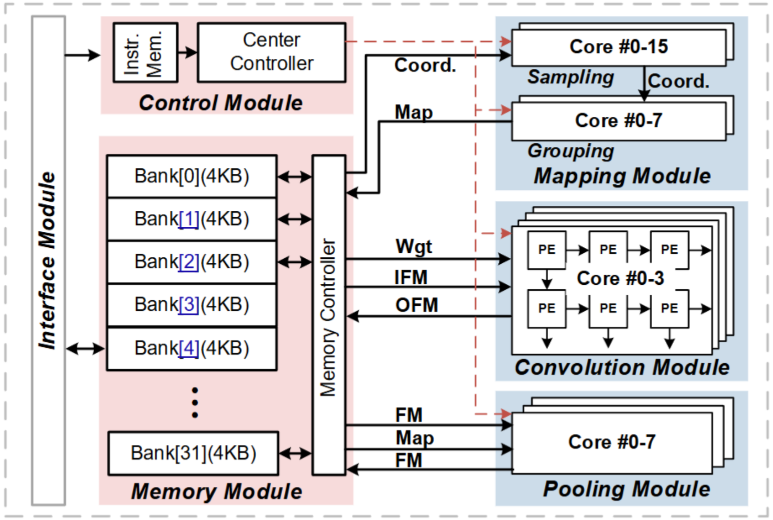
Current Hardware Architecture

- **Space-Aware Partition [VLSI’21, ICCAD’23]**
  - Example: Uniformly Partition
  - Hardware friendly
  - Streamed memory access

Imbalanced point distribution

Fail to guarantee accuracy





Kim et al., Pnnpu: A 11.9 tops/w highspeed 3d point cloud-based neural network processor with block-based point processing for regular dram access. VLSI, 2021.
Zhou et al., An Energy-Efficient 3D Point Cloud Neural Network Accelerator with Efficient Filter Pruning, MLP Fusion, and Dual-Stream Sampling. ICCAD, 2023.

Duke

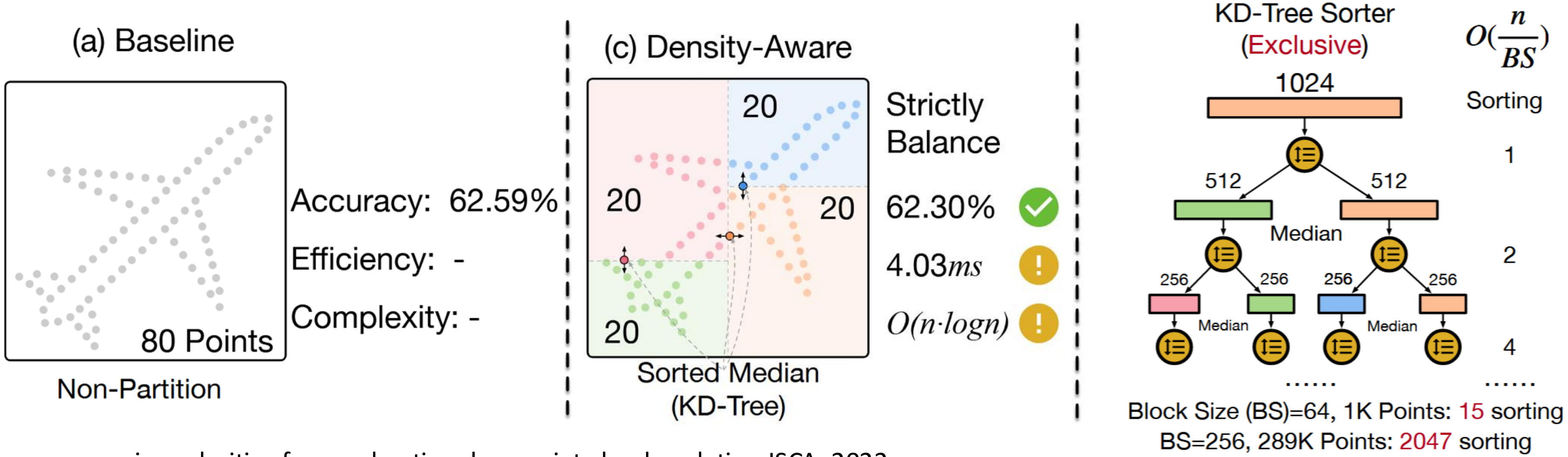
Current Hardware Architecture

- **Density-Aware Partition [ISCA'22, ASPLOS'25]**
  - Example: KD-Tree
  - Guaranteed accuracy
  - Streamed and balanced memory access

Exclusive hardware

Acceptable when small-scale process

New bottleneck for large-scale process



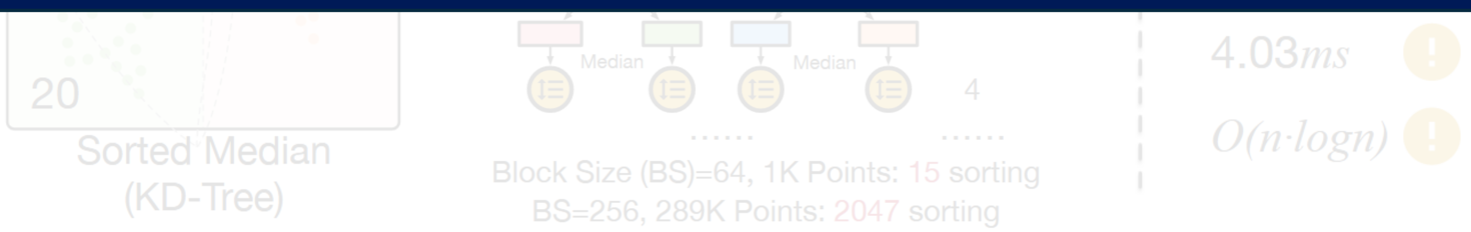
Feng et al., Crescent: taming memory irregularities for accelerating deep point cloud analytics. ISCA, 2022.
Feng et al., StreamGrid: Streaming Point Cloud Analytics via Compulsory Splitting and Deterministic Termination. ASPLOS, 2025.

Duke

Three roads for Current Architecture

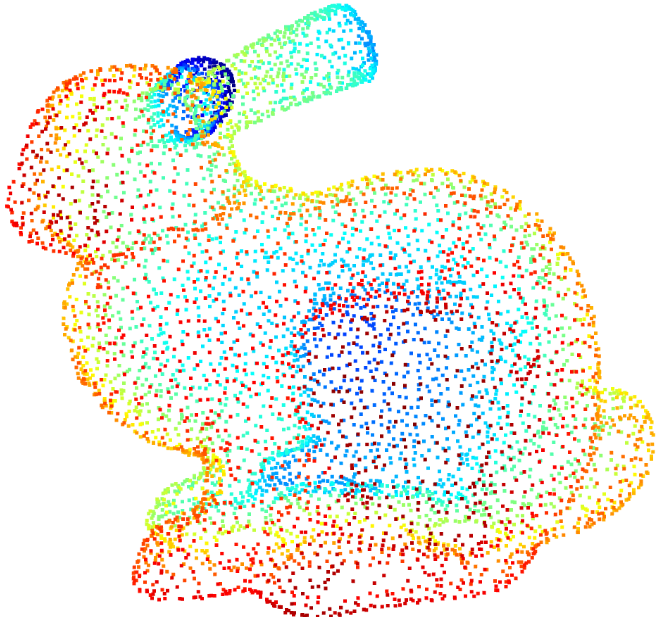
We hope partition could be

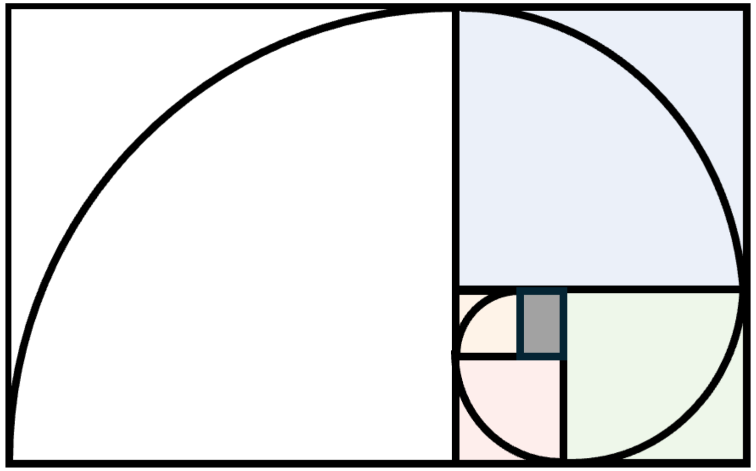
Accurate **&** Efficient



Feng et al., Crescent: taming memory irregularities for accelerating deep point cloud analytics. ISCA, 2022.
Feng et al., StreamGrid: Streaming Point Cloud Analytics via Compulsory Splitting and Deterministic Termination. ASPLOS, 2025.

Fractal Insight





Real point clouds follows geometry

Inspired by fractal geometry

- Traverse shape, not sort

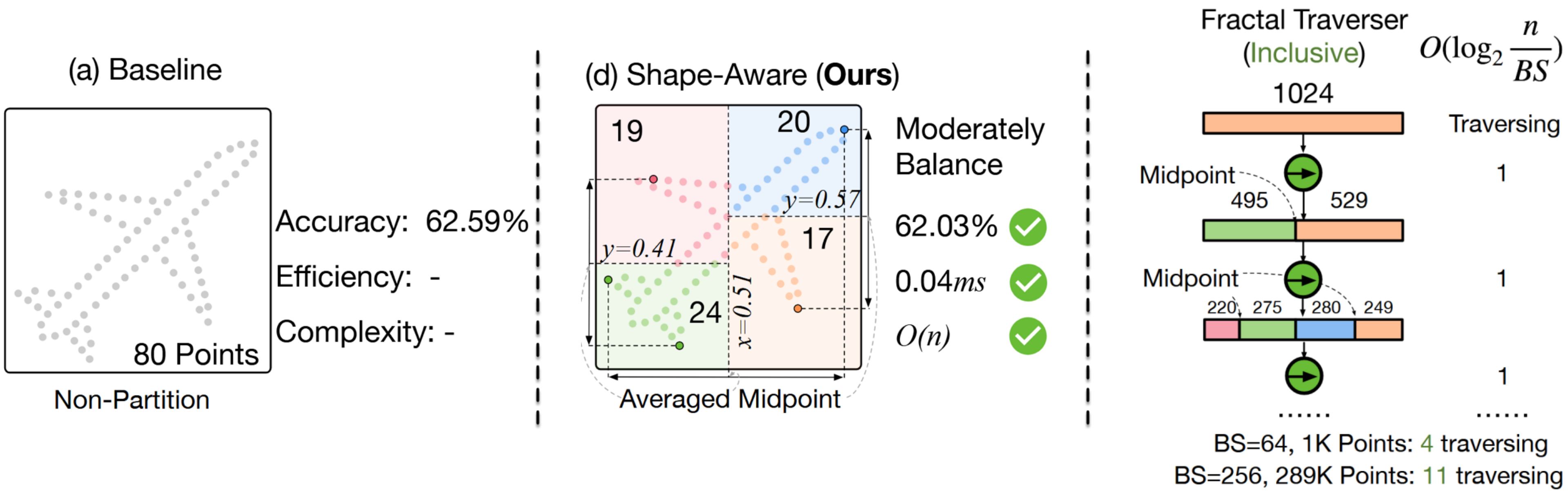
Duke

Fractal: Accurate and Efficient

- **Shape-Aware Partition**
  - Streamed memory access
  - Guaranteed accuracy

Inclusive hardware

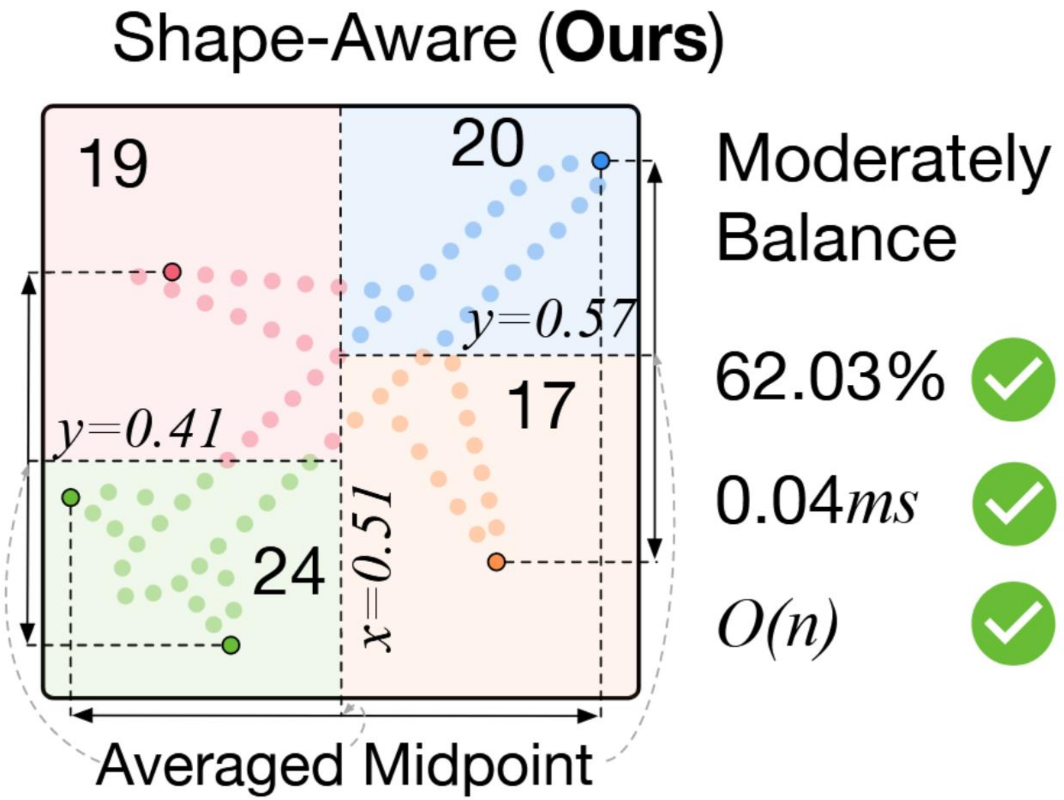
Efficient for all-scale process



Duke

Fractal: Iterative Shape-Aware Partitioning

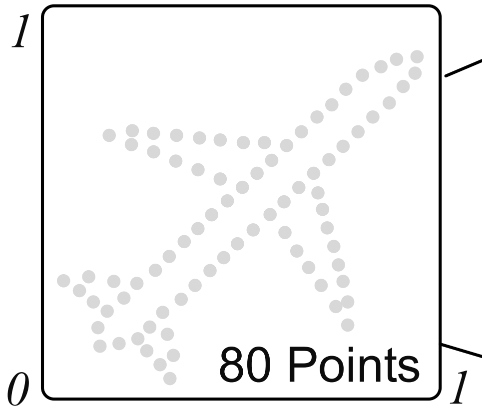
- **Inputs:**
  - Point cloud
  - Threshold (controls block size)
- **Each iteration:**
  - If block size > threshold
    - **Traverse** points along one axis
    - Compute midpoint from min & max
    - Partition
  - Alternate partition axis (x -> y -> z)



Duke

Example for Fractal – 80 Points, threshold 24

Original Point Cloud



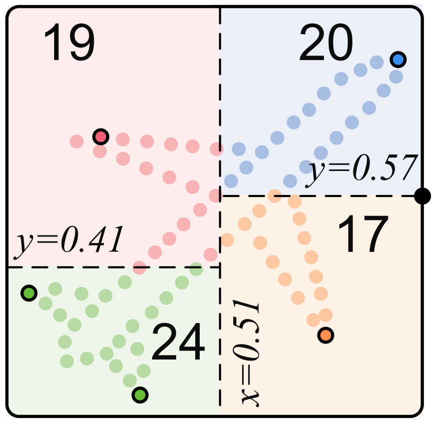
Start Fractal, with th=24

Data Layout in Memory

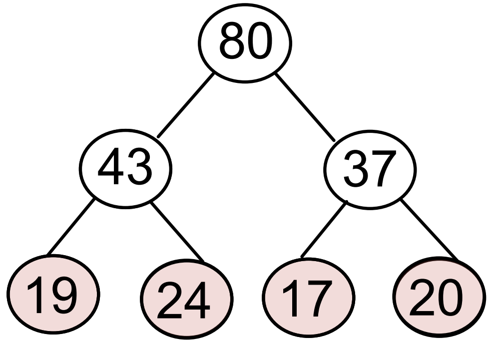
idx	Coordinates
1	(x_0,y_0,z_0)
2	(x_{40},y_{40},z_{40})
3	(x_{63},y_{63},z_{63})
···	······
79	(x_8,y_8,z_8)
80	(x_{56},y_{56},z_{56})

Unordered

With Fractal



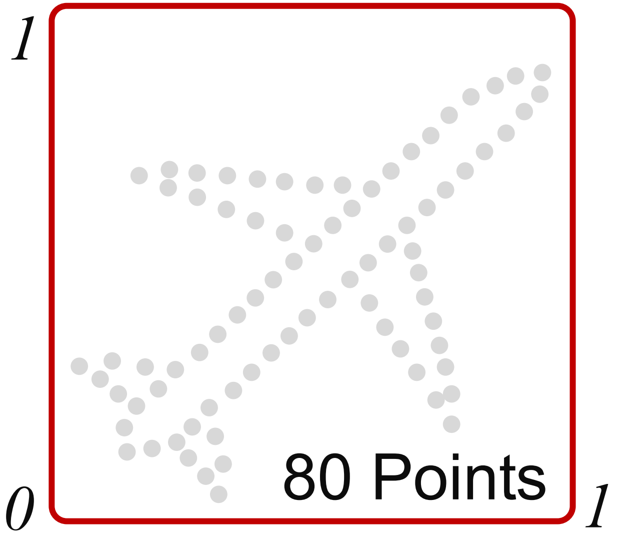
Binary Tree Flow



After 3 Fractal Iterations, 4 blocks, all blocks < 24

Example for Fractal – 80 Points, threshold 24

Check Fractal



80 > 24, **do Fractal**

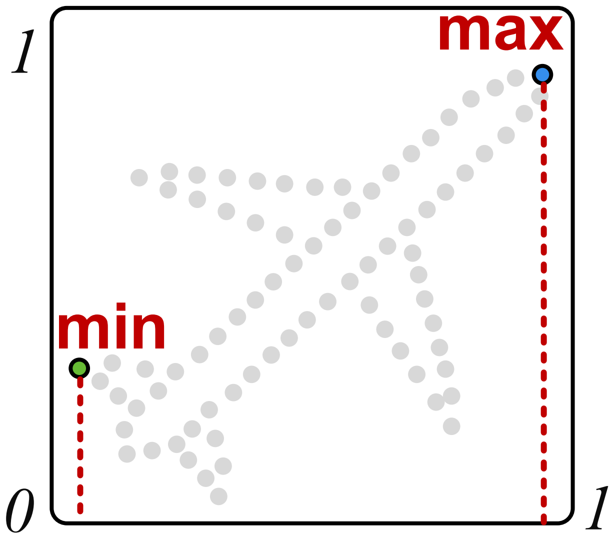
Binary Tree Flow

Check 80

Duke

Example for Fractal – 80 Points, threshold 24

1st Fractal Iteration



Step1: Find min & max along x-axis

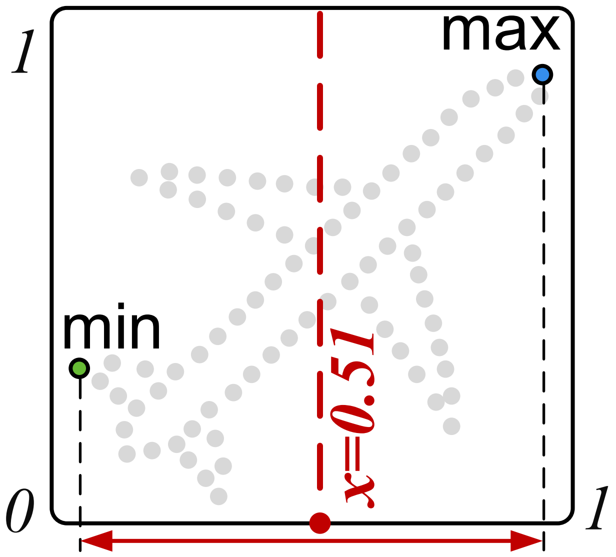
Binary Tree Flow

Process 80

Duke

Example for Fractal – 80 Points, threshold 24

1st Fractal Iteration



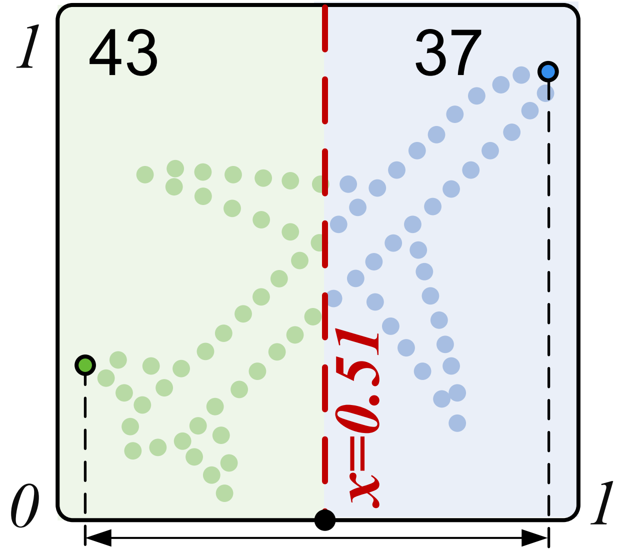
Step2: Find midpoint by (min+max)/2

Binary Tree Flow

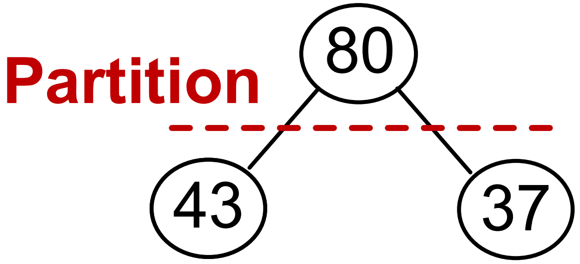
Process 80

Example for Fractal – 80 Points, threshold 24

1st Fractal Iteration



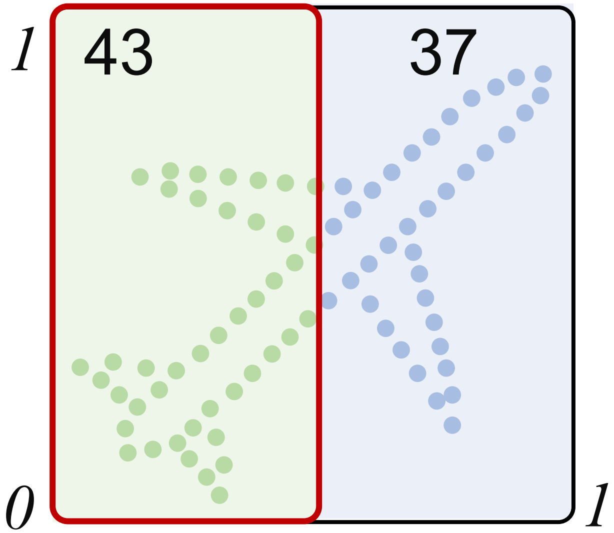
Binary Tree Flow



Step3: Partition 80 into 43- and 37- point blocks

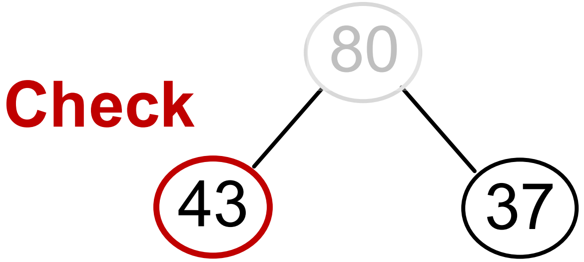
Example for Fractal – 80 Points, threshold 24

Check Threshold



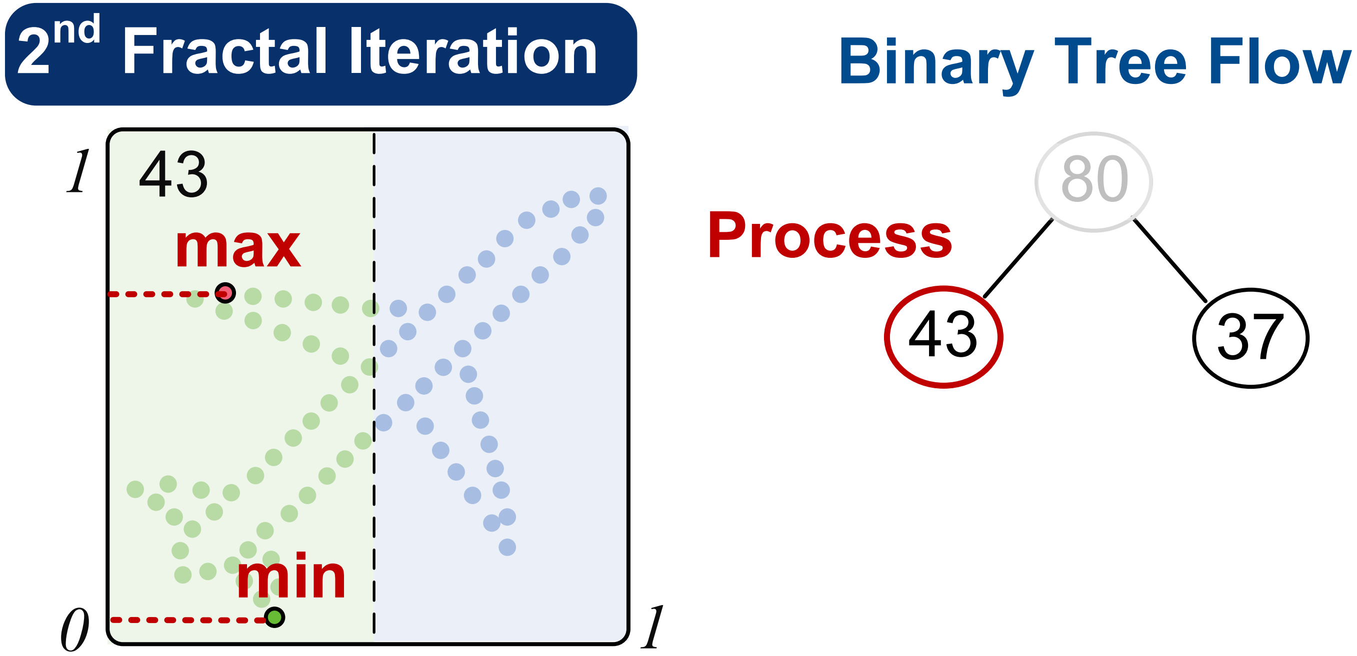
43 > 24, **do Fractal**

Binary Tree Flow



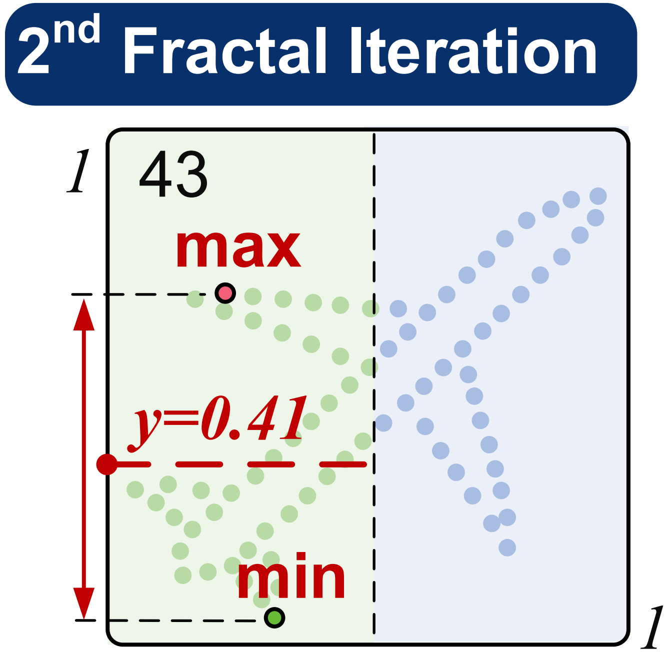
Duke

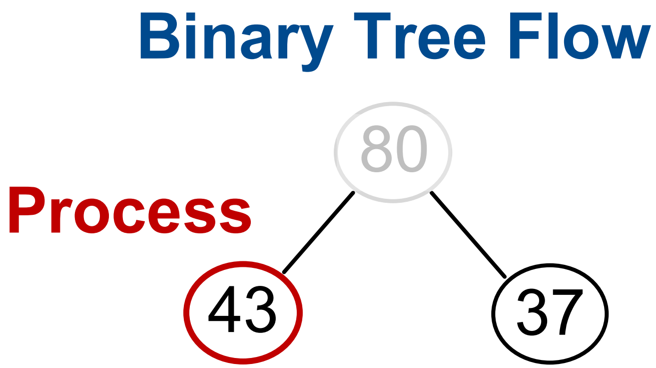
Example for Fractal – 80 Points, threshold 24



Step1: Find min & max along y-axis

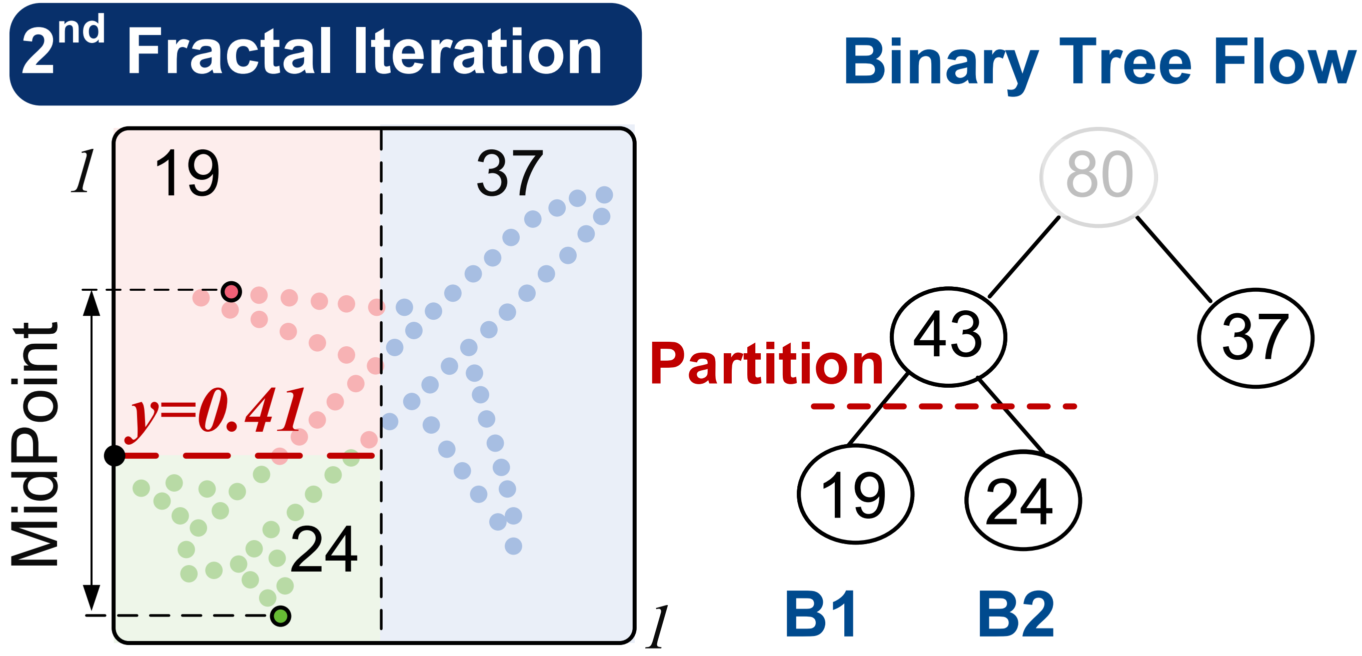
Example for Fractal – 80 Points, threshold 24





Step2: Find midpoint by (min+max)/2

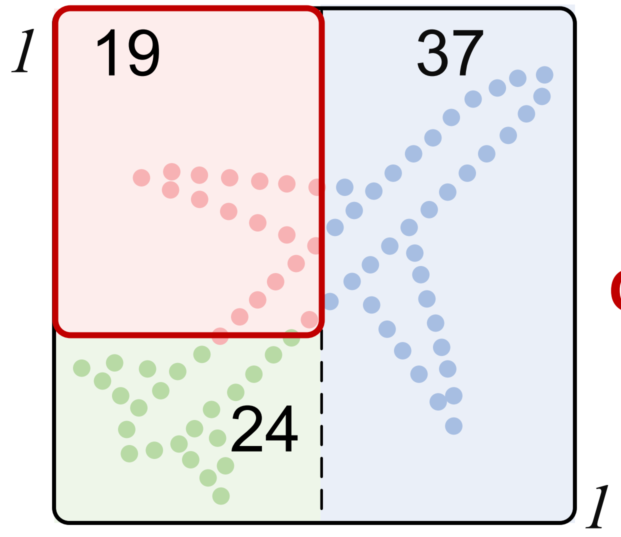
Example for Fractal – 80 Points, threshold 24



Step3: Partition 43 into 19- and 24- point blocks

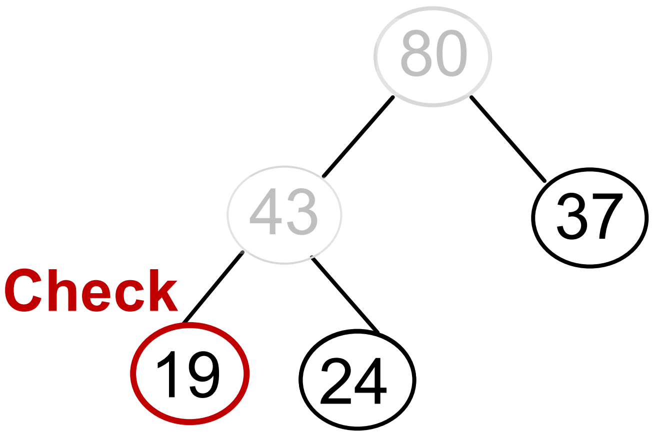
Example for Fractal – 80 Points, threshold 24

Check Threshold



19 < 24, **no Fractal**

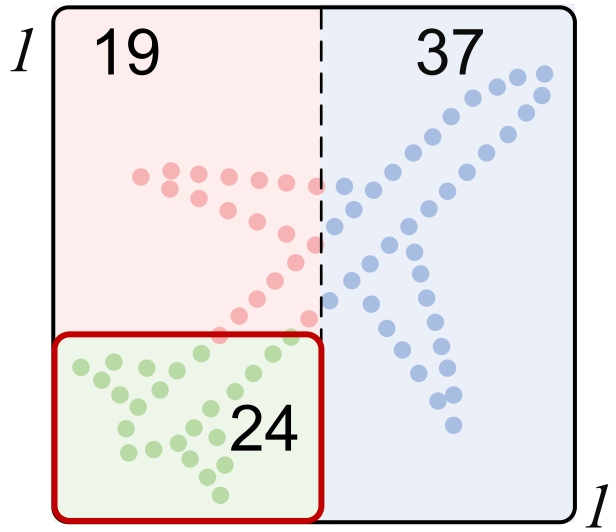
Binary Tree Flow



Duke

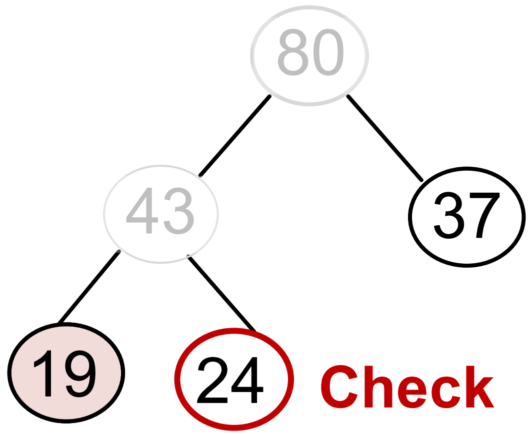
Example for Fractal – 80 Points, threshold 24

Check Threshold



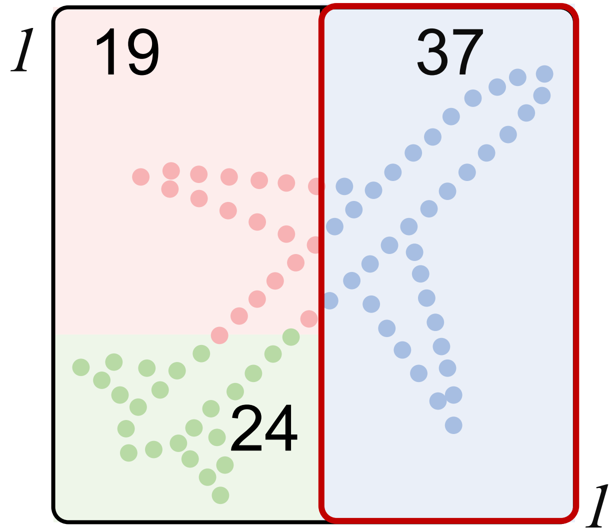
24 == 24, **no Fractal**

Binary Tree Flow

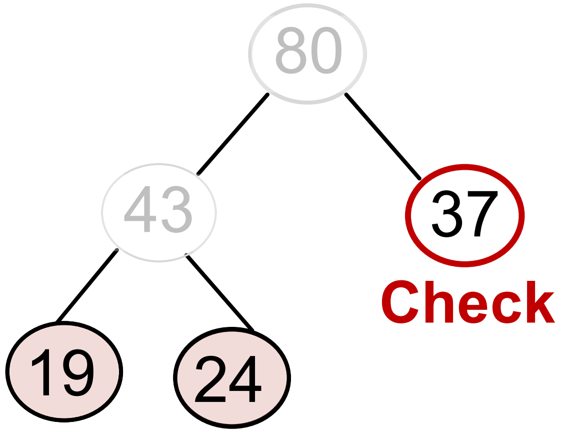


Example for Fractal – 80 Points, threshold 24

Check Threshold



Binary Tree Flow

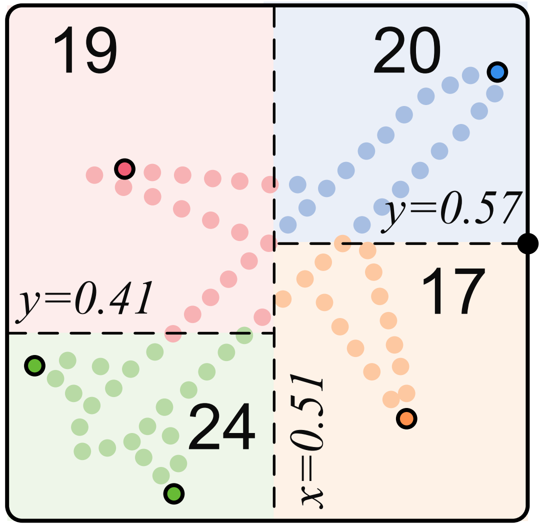


Same flow for all Fractal Iterations

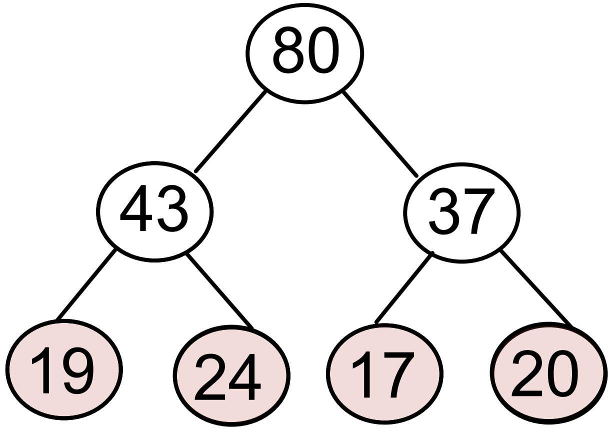
Not sort, **only Linear Traverse**

Example for Fractal – 80 Points, threshold 24

With Fractal

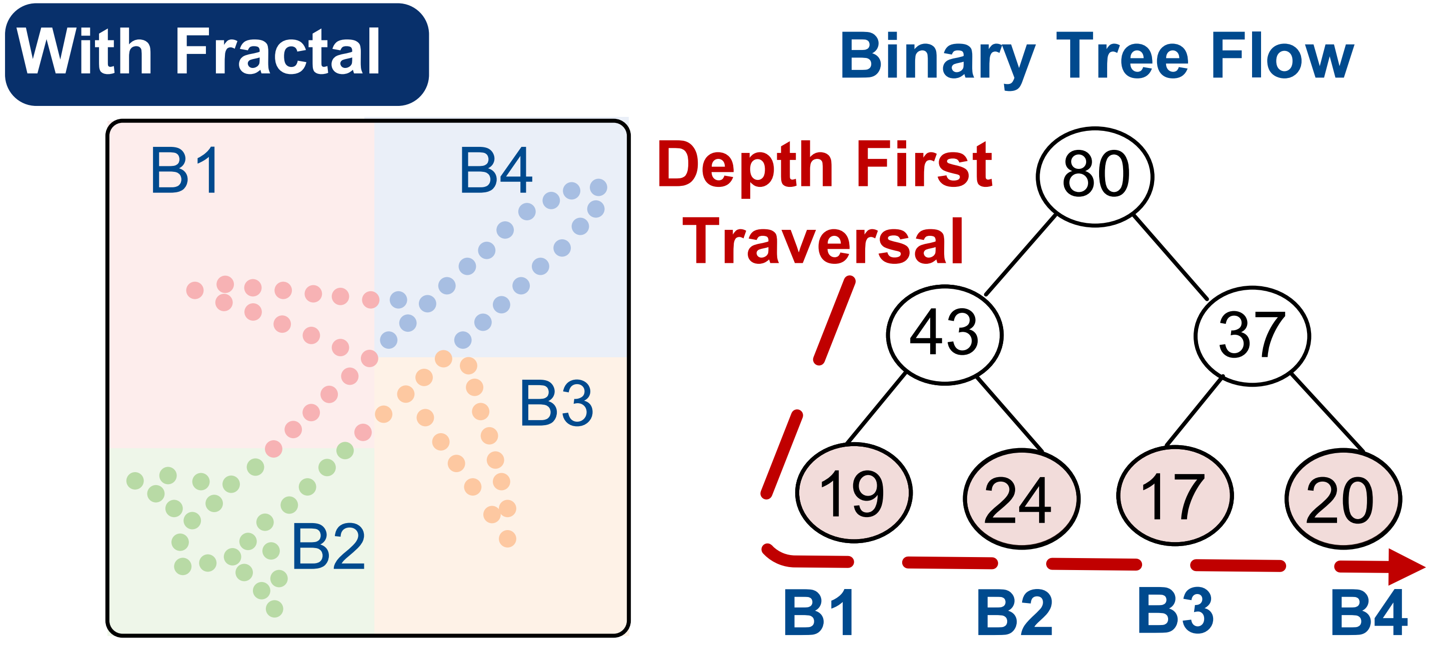


Binary Tree Flow



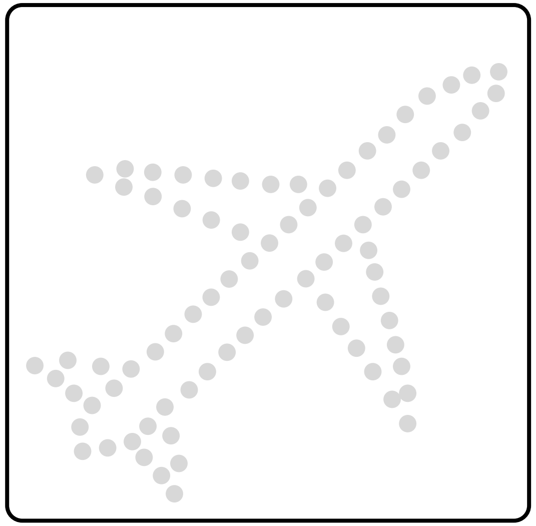
After 3 Fractal Iterations, 4 blocks, all blocks < 24

Example for Fractal – 80 Points, threshold 24



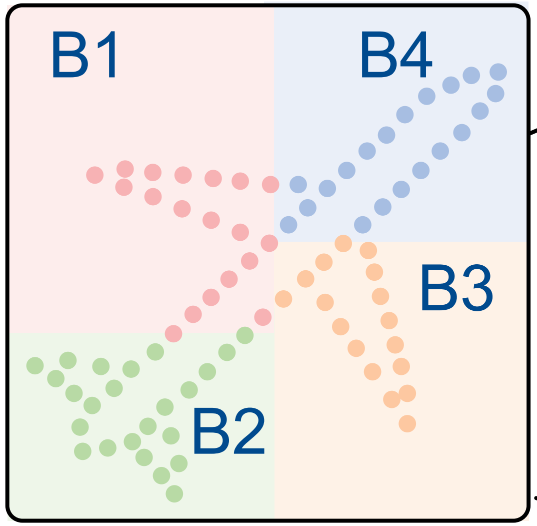
DFT to determine the block order

Example for Fractal – 80 Points, threshold 24



Original Point Cloud

After Fractal



Four Point Blocks

Data Layout in Memory

idx	Coordinates	
1	(x_0,y_0,z_0)	B1
…	……	
20	(x_{32},y_{32},z_{32})	B2
…	……	
44	(x_{40},y_{40},z_{40})	B3
…	……	
80	(x_{56},y_{56},z_{56})	B4

Spatially Orgnized

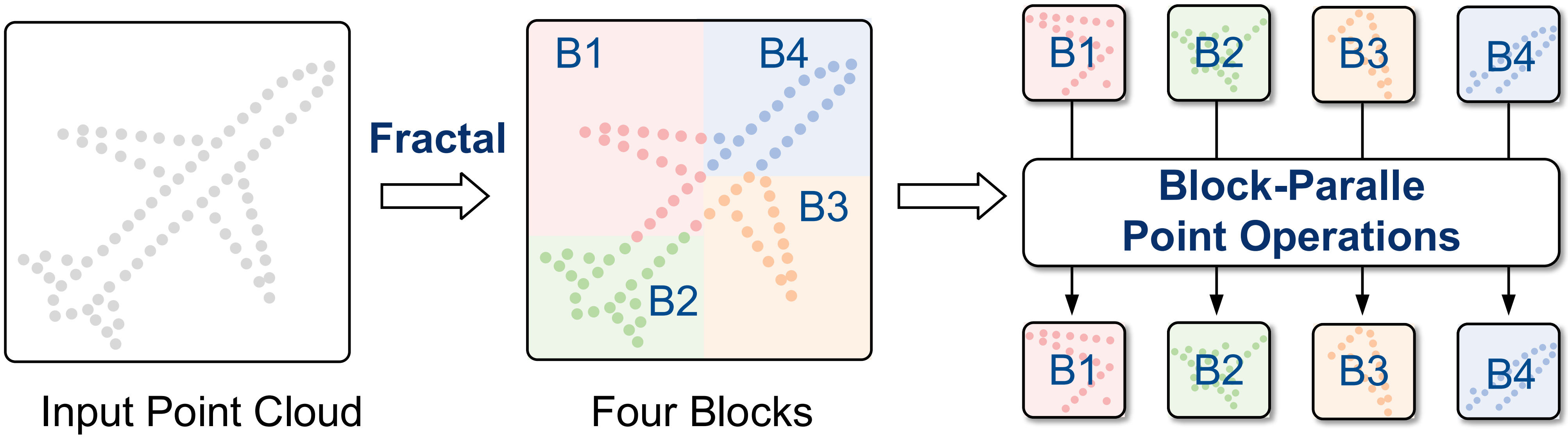
Duke

Extend Fractal from Points to Operations

- **Fractal is cheap and scalable.**
- **Blocks are mutually independent.**

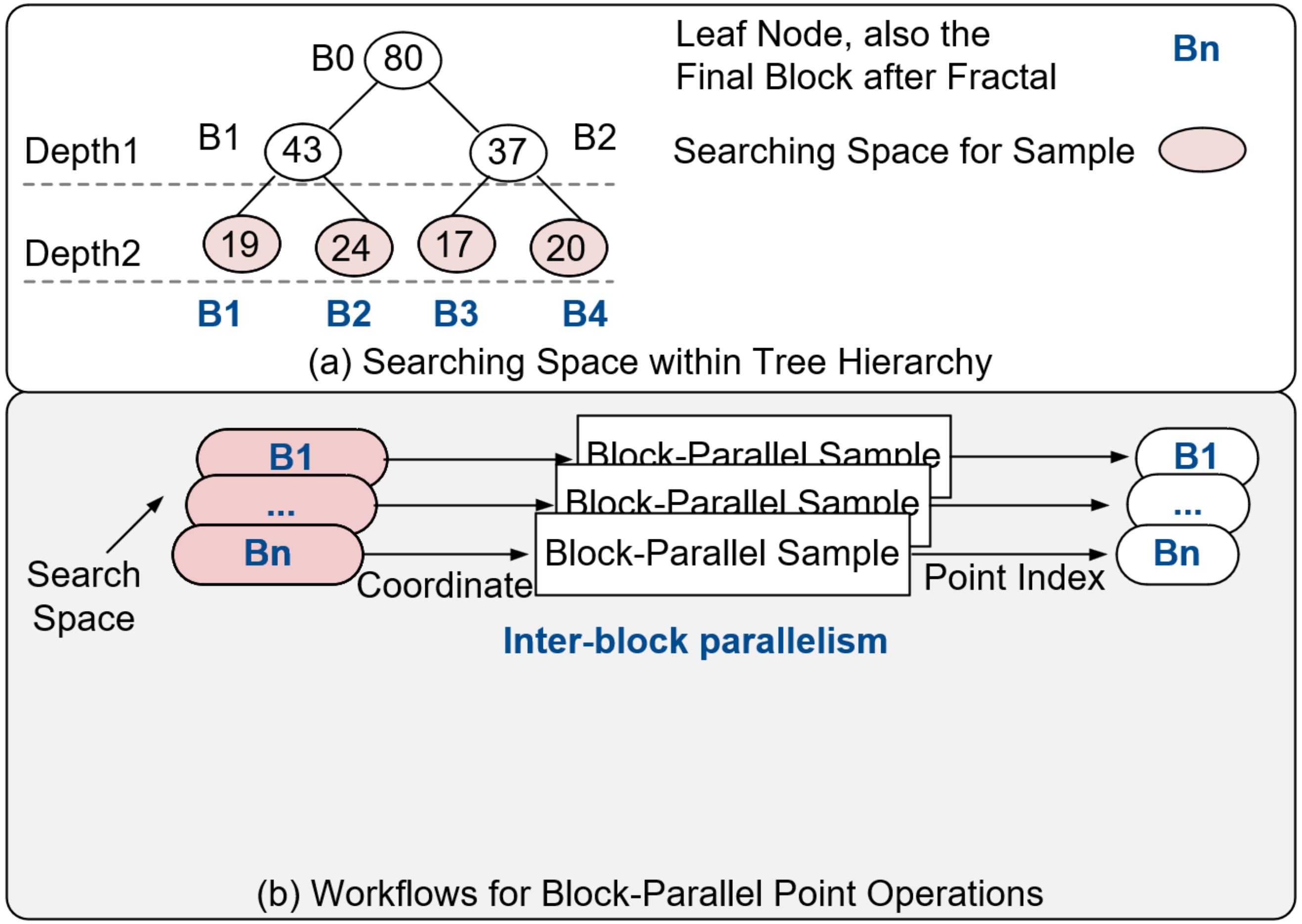
Block-level parallelism

- Local computation and memory access



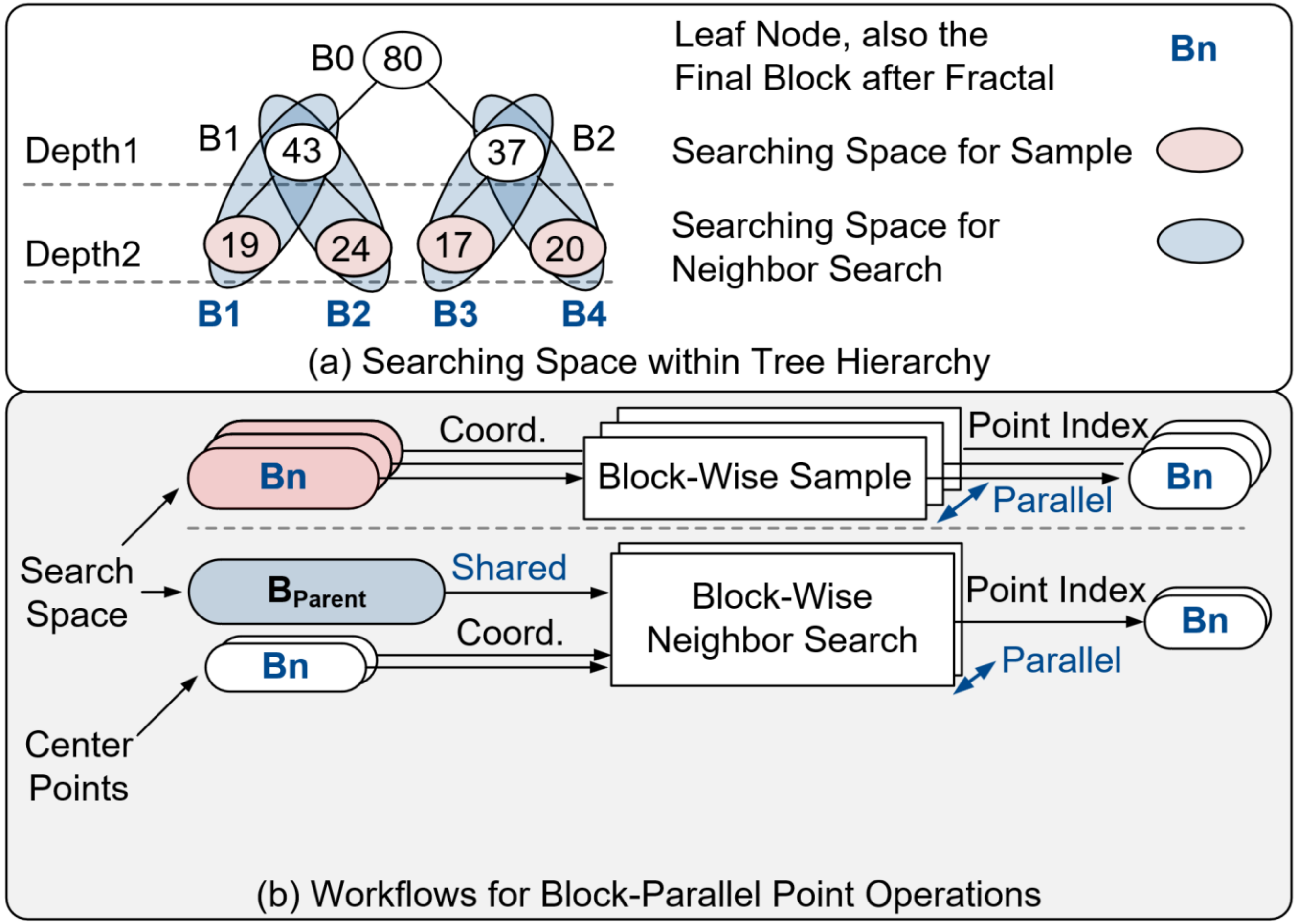
Block-Parallel Point Operations

- **Block-wise Sample**
  - Process within current block
  - Inter-block parallelism



Block-Parallel Point Operations

- **Block-wise Neighbor Search**
  - Expend searching to parent node
  - One parent level is sufficient

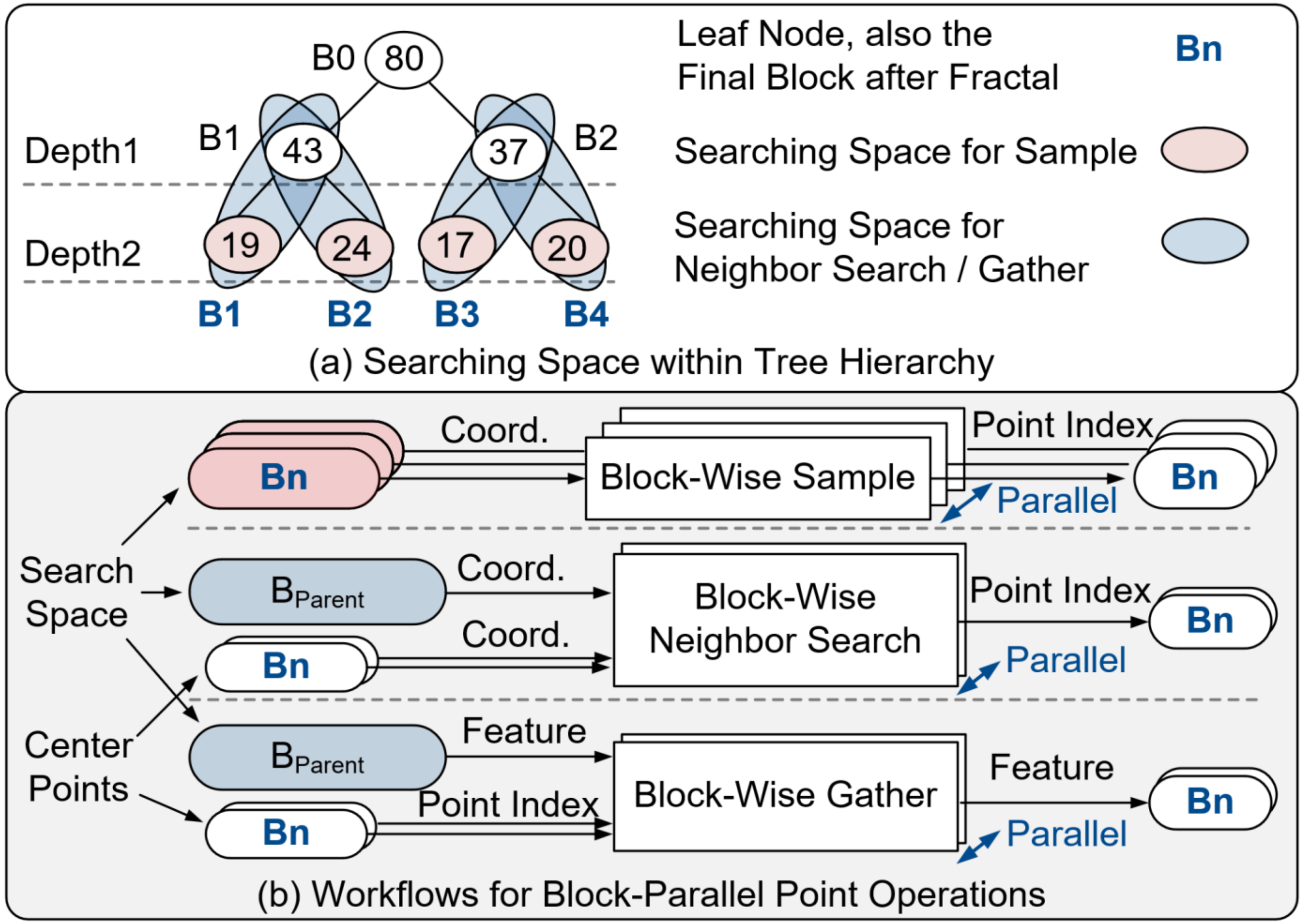


(a) Searching Space within Tree Hierarchy

(b) Workflows for Block-Parallel Point Operations

Block-Parallel Point Operations

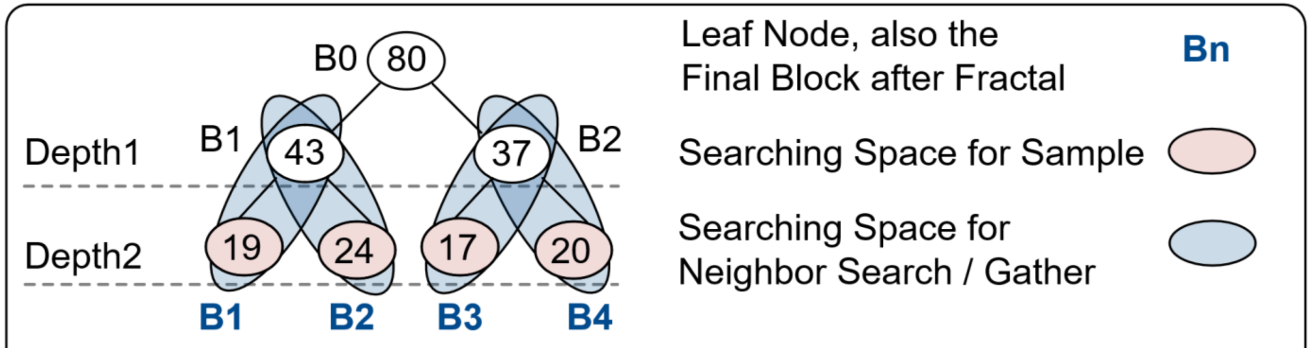
- **Block-wise Gather**
    - Same rules as neighbor search



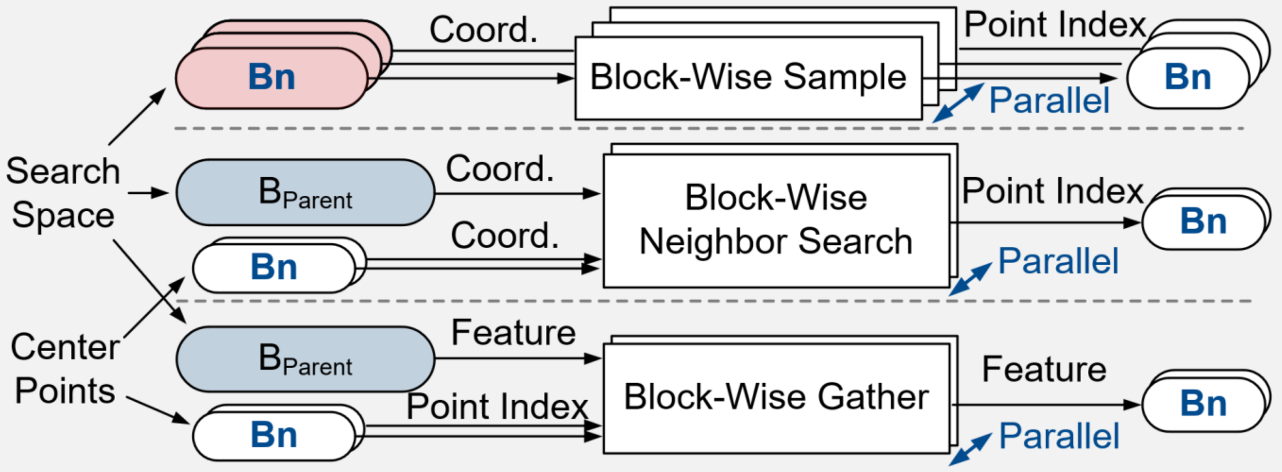
Block-Parallel Point Operations

Block-wise Sample
Block-wise Neighbor Search
Block-wise Gather

- **Eliminate all-to-all computing**
- **Unlock block-level parallelism**
- **On-chip feasible**



(a) Searching Space within Tree Hierarchy

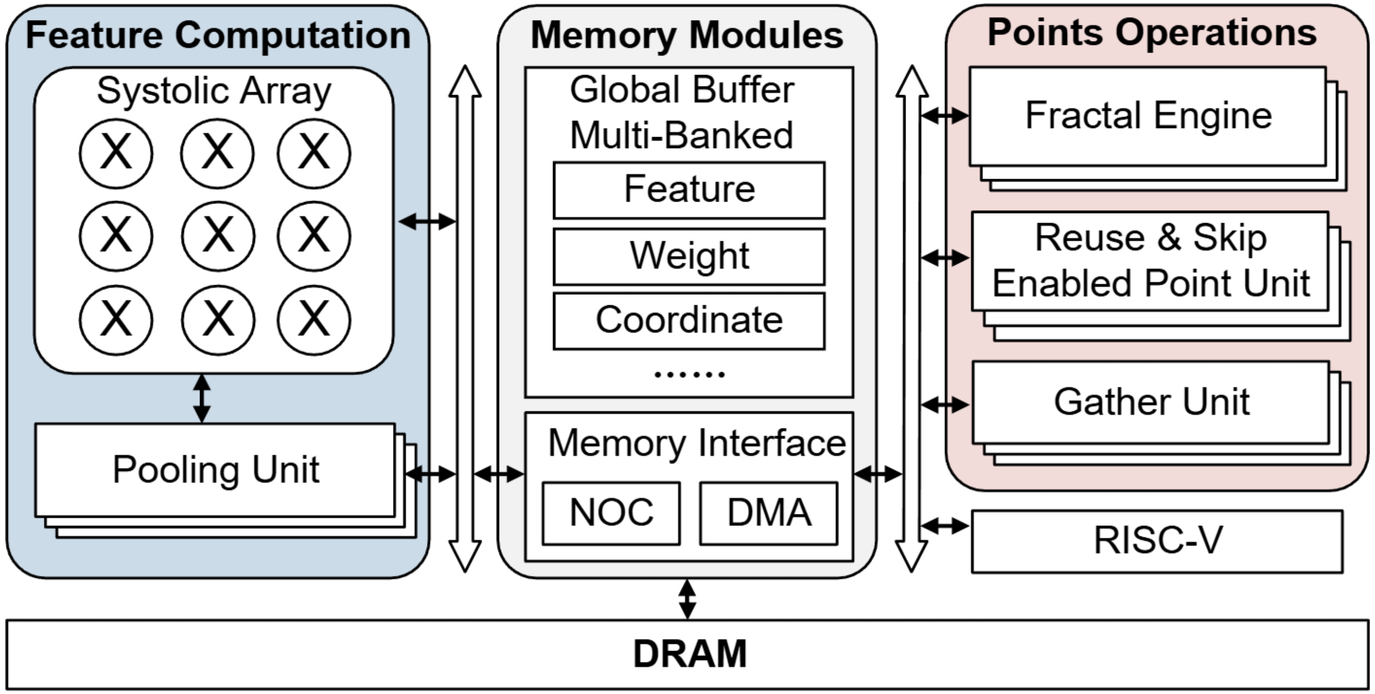


(b) Workflows for Block-Parallel Point Operations

Duke

FractalCloud: Point Cloud Accelerator

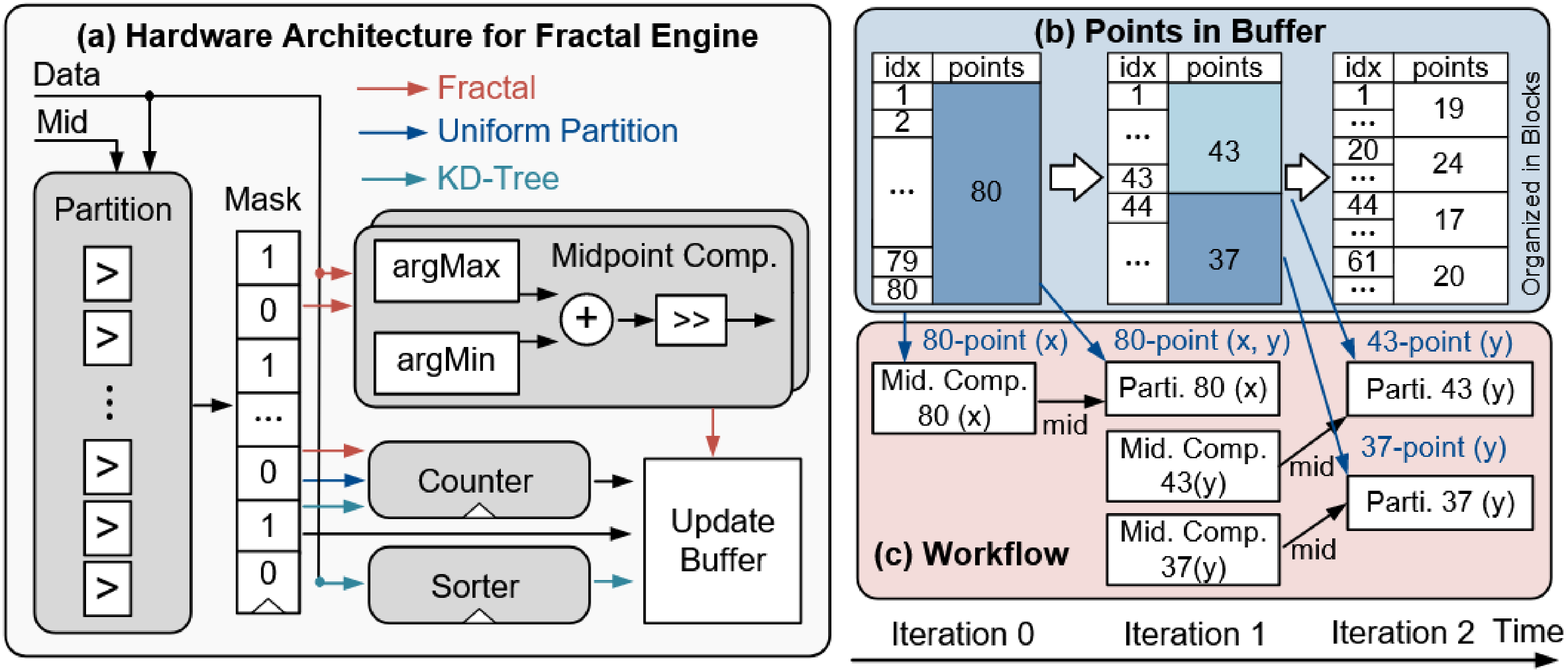
- Systolic Array
- Network on Chip (NOC)
- Direct Memory Access (DMA)
- RISC-V MCU
- SRAM (274KB)
- **Fractal Engine**
- **Reuse-Skip Enabled Point Unit (RSPU)**



Duke

Fractal Engine

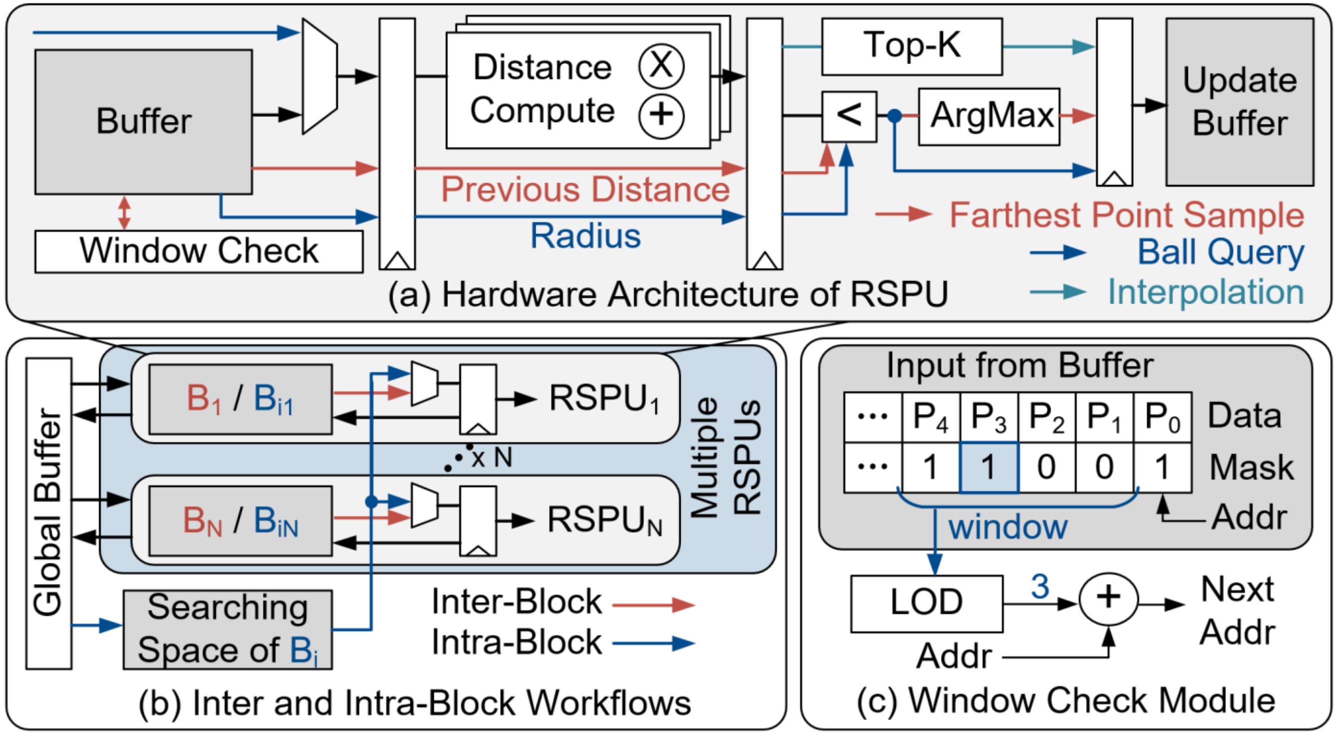
- Reconfigurable structure for multiple partitions:
  - Fractal, KD-Tree, uniform partition.
- Fractal:
  - Simple Hardware
  - Inclusive
  - Fully pipelined



Reuse Skip Enabled Point Unit (RSPU)

- Unified module for all point operations
  - FPS, Ball Query, KNN (Interpolation)
  - Blocks run with DFT order

Single RSPU



Multiple RSPUs

- Flexible Block-Parallel Workflow

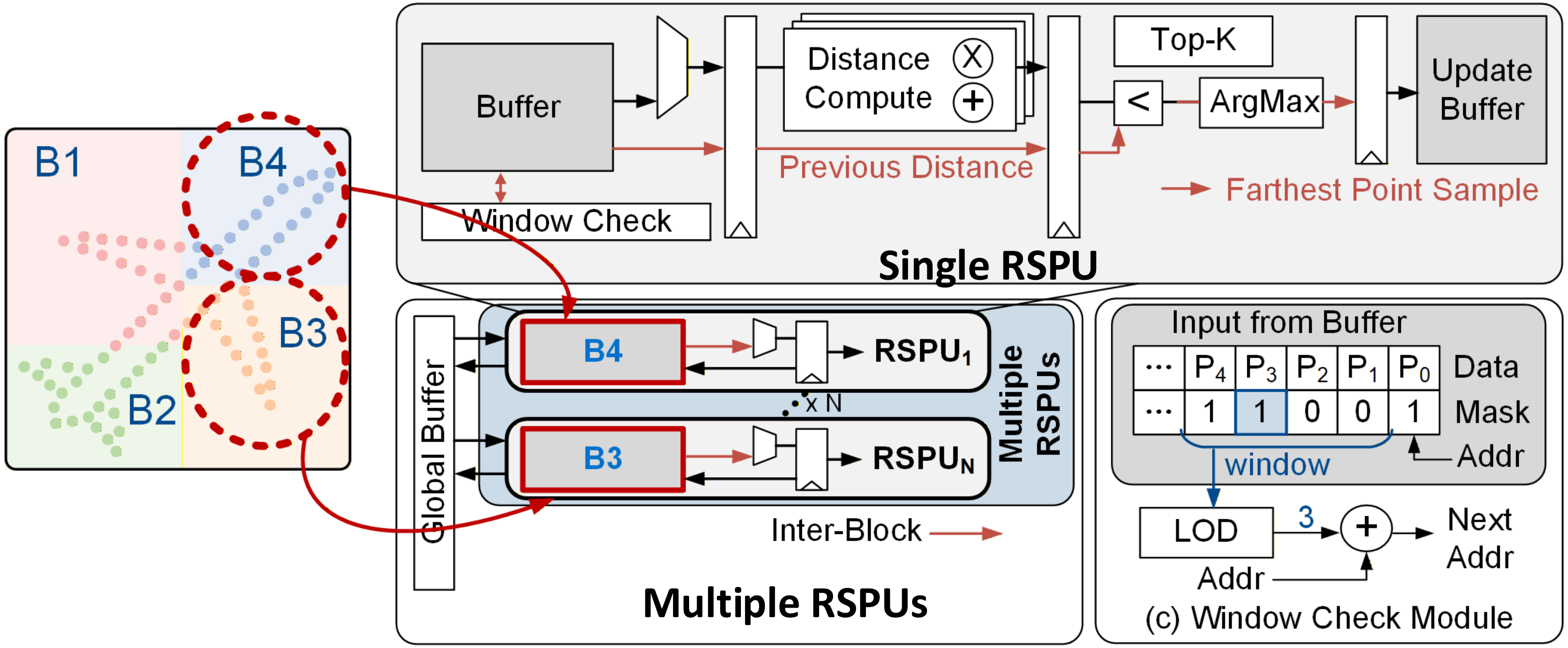
Block-Wise Sample
  - **Inter-block parallelism**

Block-Wise Neighbor Search
  - **Intra-block parallelism**

Duke

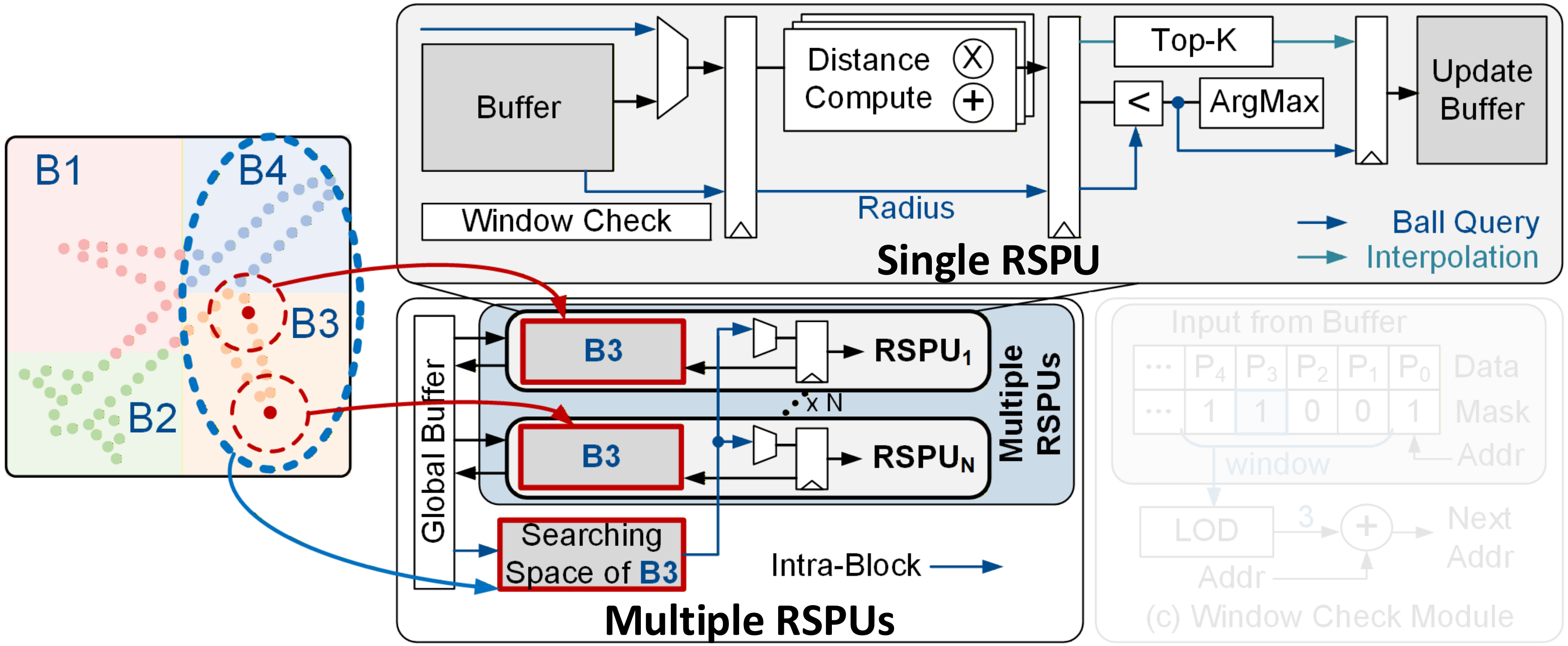
Flexible Block-Parallel for Multiple RSPUs

- **Block-Wise Sample: inter-block parallelism**
  - Each RSPU handles one FPS within one block
- Window check: **Skip redundant computation**



Reuse-and-Skip-enabled Point Unit (RSPU)

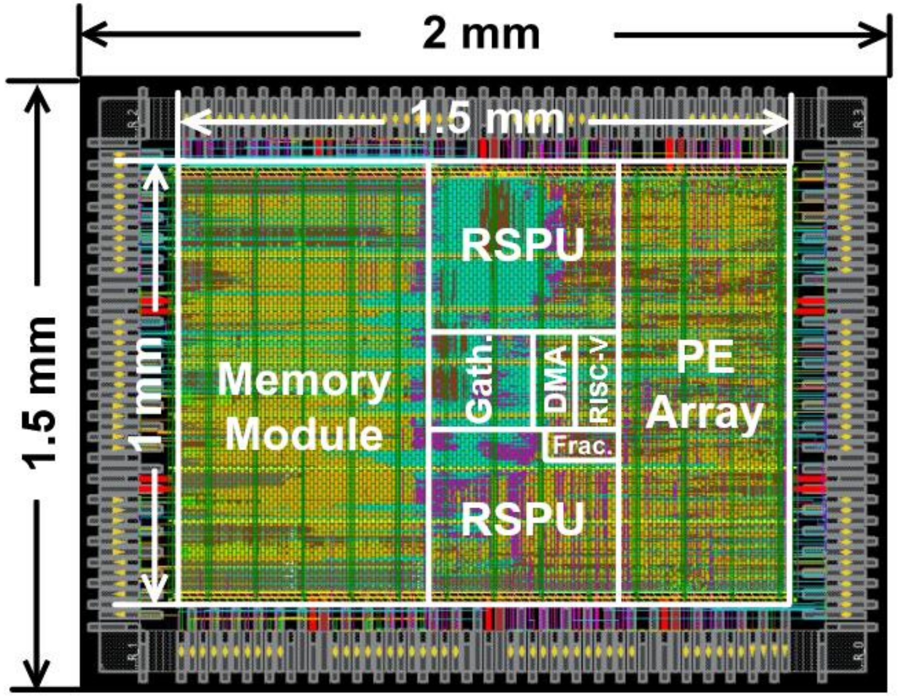
- **Block-Wise Neighbor Search: intra-block parallelism**
  - Each RSPU process different centric points in same block
- **Data reusing from parent node**



HW Implementation

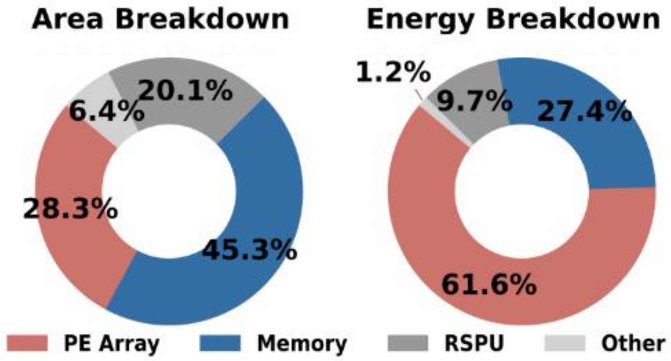
- **Small hardware:**
  - TSMC 28nm
  - Core Area: 1.5 mm^2
  - Power: 0.58 W
  - Frequency: 1 GHz

Chip Layout of Proposed ***FractalCloud***



Detailed Specifications

Technology	28nm
Die Area	3 mm^2
Core Area	1.5 mm^2
SRAM Size	274 KB
Frequency	1 GHz
Ave. Power	0.58 W



Evaluation

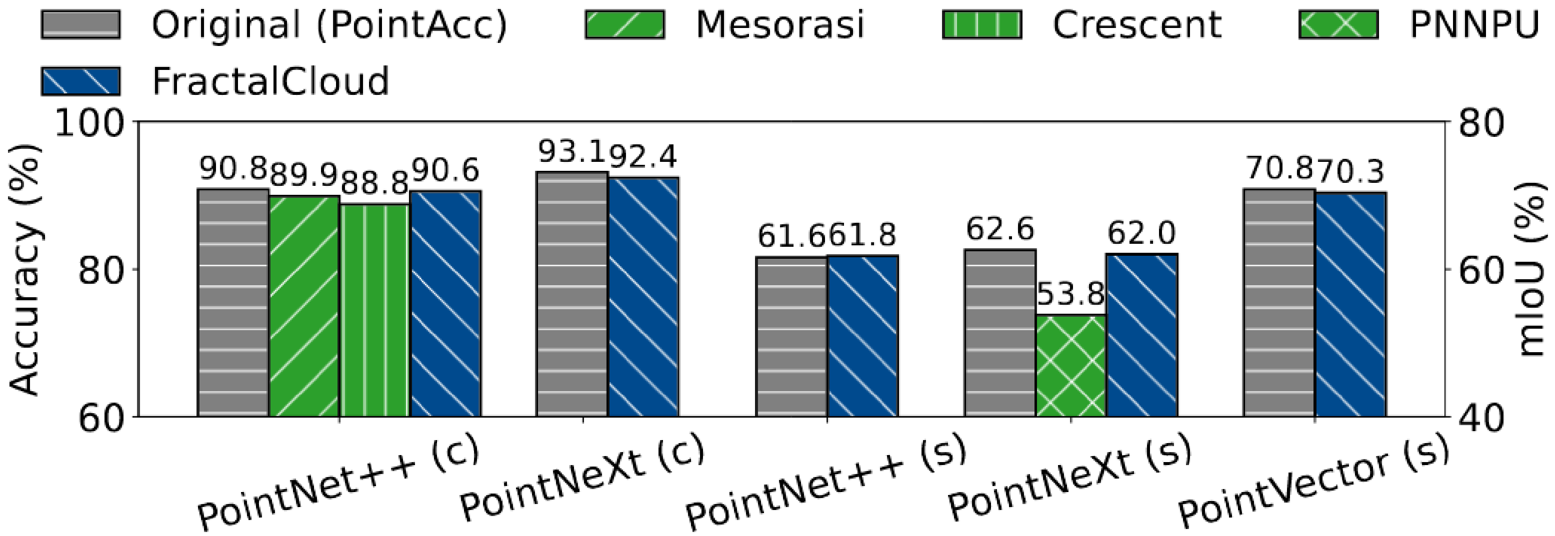
- **Network Benchmarks**
  - Inputs scale from 1K to 289K
  - Three PNNs
  - Three Tasks
  - Three Datasets

- **Hardware Architectures**
  - Same PE cores
  - Fixed Frequency
  - Equal DRAM Bandwidth
  - ……

Model	Notation	Task	Dataset	Scene
PointNet++	PN++ (c)	Classification	ModelNet40	Object
PointNeXt	PNXt (c)			
PointNet++	PN++ (ps)	Part Segmentation	ShapeNet	Object
PointNeXt	PNXt (ps)			
PointNet++	PN++ (s)	Segmentation	S3DIS	Indoor
PointNeXt	PNXt (s)			
PointVector	PVr (s)			

Accelerator	Mesorasi [27]	PointAcc [28]	Crescent [29]	*FractalCloud*
Cores	16x16	16x16	16x16	16x16
SRAM (KB)	1624	274	1622.8	274
Frequency	1GHz	1GHz	1GHz	1GHz
Area (mm2)	4.59	1.91	4.75	1.5
DRAM Bandwidth	DDR4-2133 17GB/s	DDR4-2133 17GB/s	DDR4-2133 17GB/s	DDR4-2133 17GB/s
Technology	28nm	28nm	28nm	28nm
Peak Performance	512 GOPS	512 GOPS	512 GOPS	512 GOPS

Network Accuracy



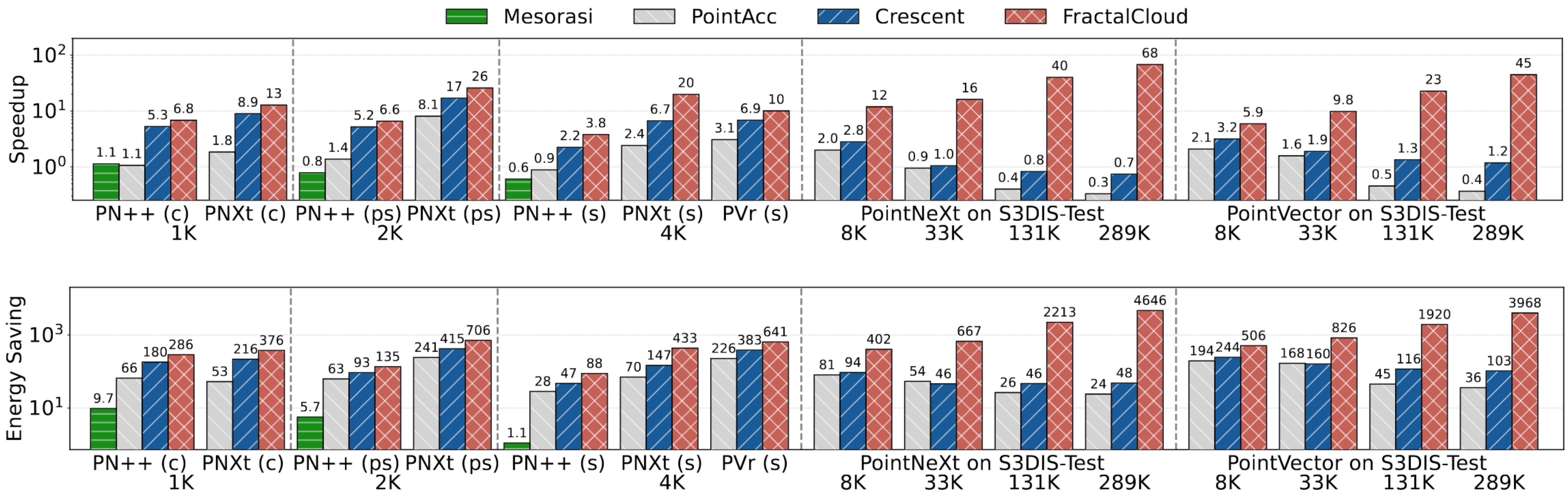
Guaranteed accuracy:

Less than 0.7% accuracy loss for all models

Better performance than SOTA works

Duke

Performance Gain over SOTA accelerators



Huge performance:

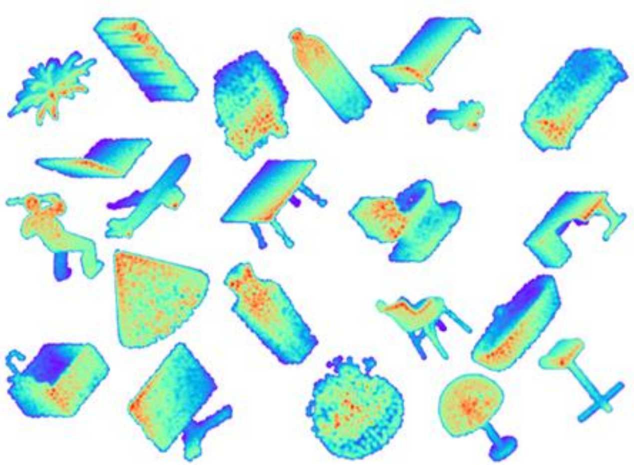
Average 21.7x speedup

Average 27x energy saving

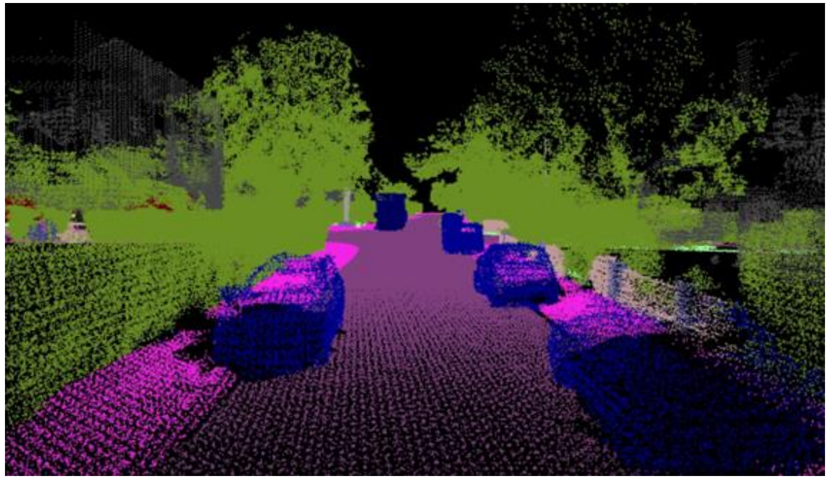
Duke

FractalCloud for Efficient PNN Acceleration

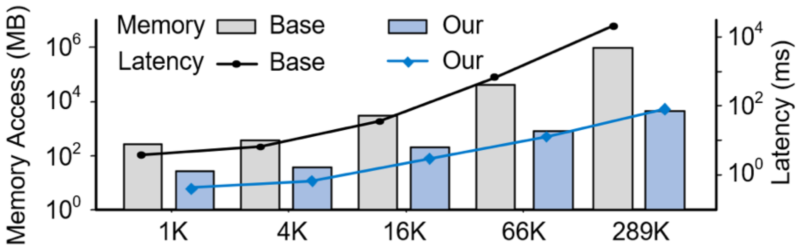
- **Application:** AR/VR, automatic drive, drones, …
- **From small to large input processing**



1K @ 2017 (Simple)
Object Classification



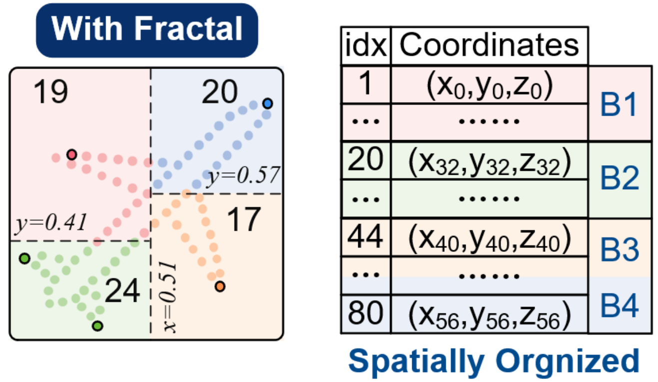
300K @ 2024 (Complex)
Semantic Segmentation



FractalCloud Optimization
21.7x speedup

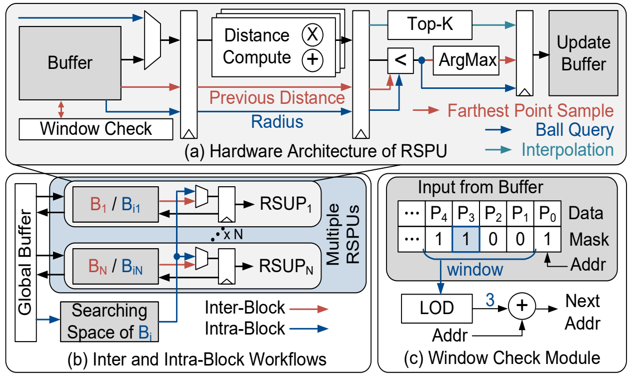
FractalCloud for Efficient PNN Acceleration

Accuracy and Efficiency

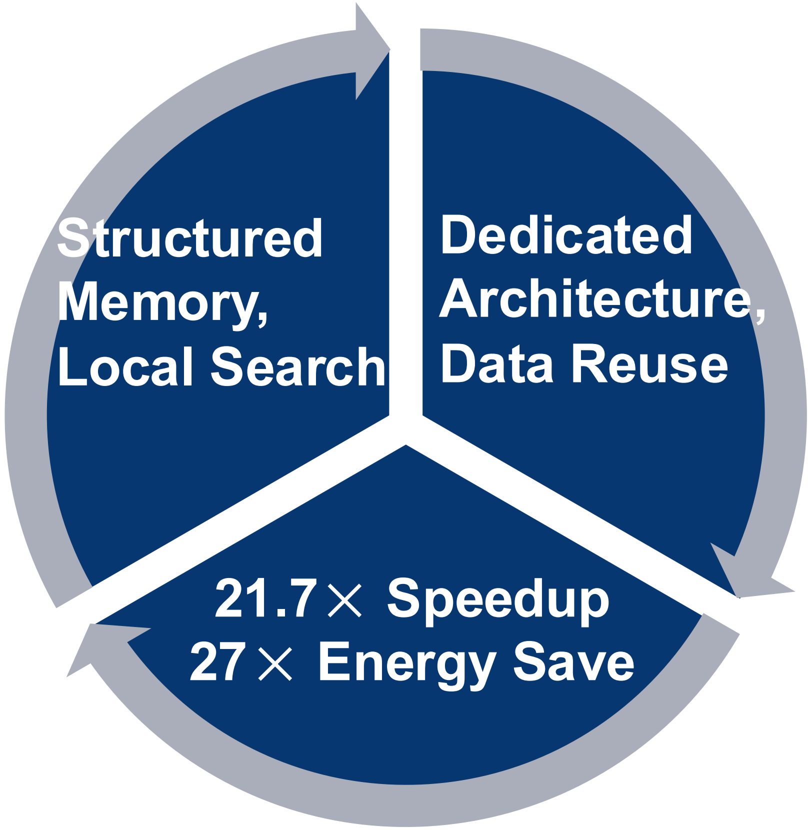


Fractal: Shape-Aware Partition

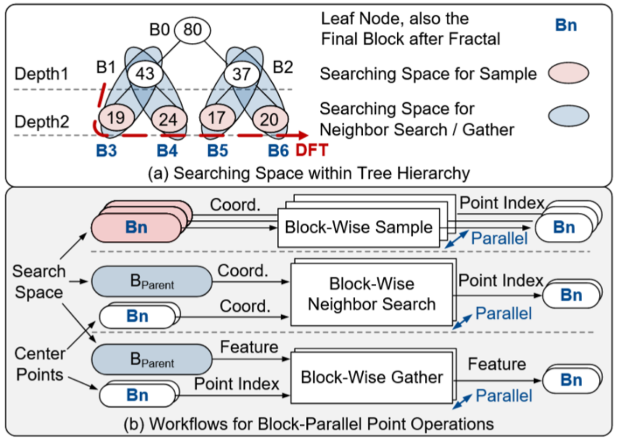
Block-Parallel Hardware



Reuse-and-Skip-Enabled Point Unit

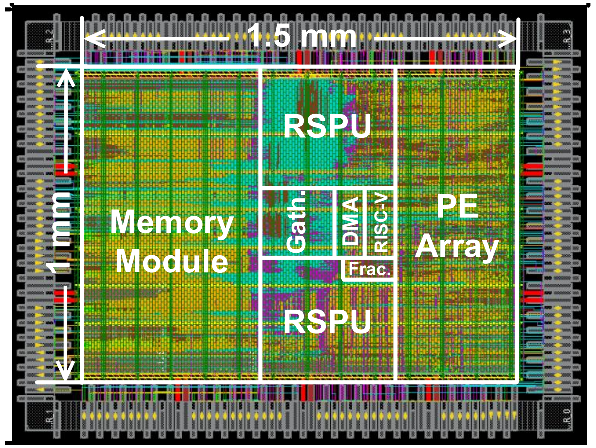


Local Computation



Block-Parallel Point Operation

Low latency & low energy cost

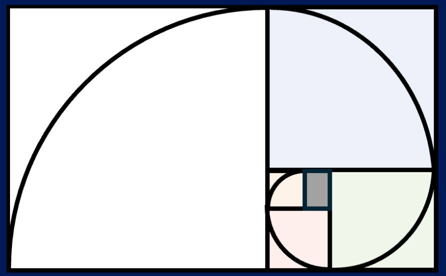


Area: 1.5mm², Power 0.58W

Acknowledgements

Duke | PRATT SCHOOL of ENGINEERING

FractalCloud
HPCA 2026



Thanks for Listening.

Codes are open-sourced at

https://github.com/Yuzhe-Fu/FractalCloud

Authors:

Yuzhe Fu, Changchun Zhou, Hancheng Ye, Bowen Duan, Qiyu Huang, Chiyue Wei, Cong Guo, Hai Li, and Yiran Chen

Duke
Center of Computational Evolutionary Intelligence (CEI)