An Efficient Accelerator for Point-based and Voxel-based Point Cloud Neural Networks

Xinhao Yang[1], Tianyu Fu[1], Guohao Dai[2], Shulin Zeng[1], Kai Zhong[1], Ke Hong[1] and Yu Wang[1]

[1]Dept. of EE, BNRist, Tsinghua University, [2]Shanghai Jiao Tong University

E-mail: yxh21@mails.tsinghua.edu.cn, daiguohao@sjtu.edu.cn, yu-wang@tsinghua.edu.cn

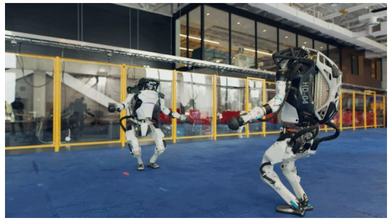
Contents

Contents

Background: Point Cloud NN

- Point Cloud Neural Networks
  - Widely used in autonomous driving, robotics, AR/VR, etc.

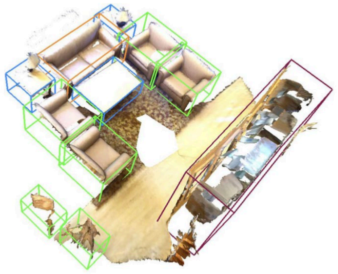
Autonomous driving



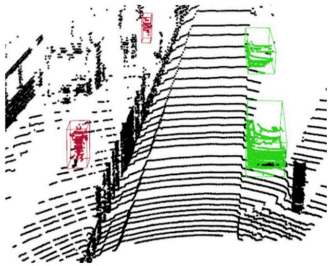
Robotics

AR/VR

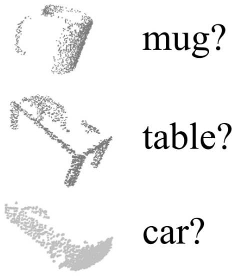
  - Commonly used in object detection, tracking, classification, segmentation and other tasks



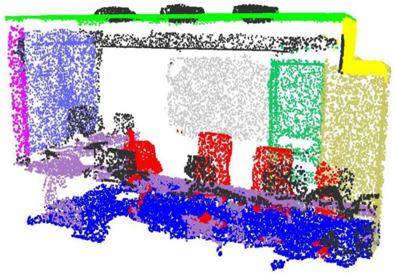
Object Detection



Object Tracking

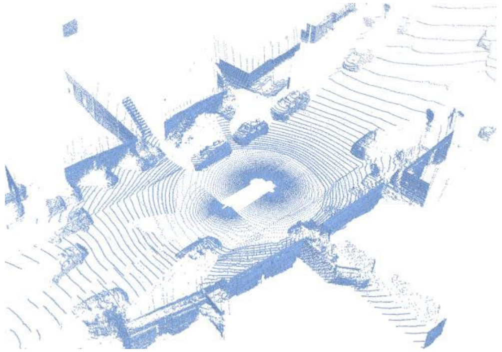


Classification

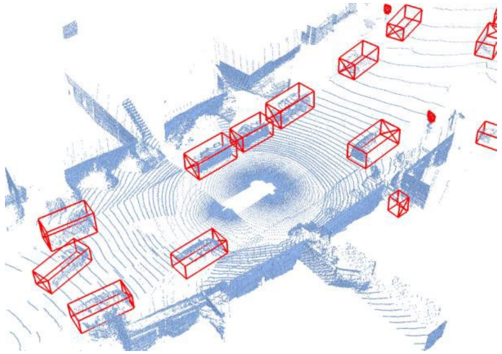


Semantic Segmentation

LiDAR



3D Point Cloud



Object Detection Box

Background: Point Cloud NN

- Point Cloud Neural Networks
  - 3D algorithms have accuracy advantages over CNN and 2D algorithms
  - 3D point cloud data is sparse (0.01%~10%), the sparsity comes from the actual object

Method		Model	Car mAP (%)	
2D	BEV	Complex-YOLO [CVPR'18]	77.4	2D ☹
		PixorNet [CVPR'18]	77.05	
	Range Image	LaserNet [CVPR'19]	73.77	
2.5D	voxel-based	Pointpillars [CVPR'19]	76.86	
3D	voxel-based	SECOND [Sensors'18]	**78.62**	3D ☺
		CIASSD [AAAI'21]	**79.86**	
		Voxel R-CNN [AAAI'21]	**84.52**	
	point-based	PointRCNN [CVPR'19]	**78.63**	
		3D-SSD [CVPR'20]	**79.57**	
	voxel-point	PVRCNN [CVPR'20]	**83.61**	
		SA-SSD [CVPR'20]	**79.91**	
	LiDAR-Image	F-ConvNet [IROS'19]	**76.39**	
		EPNet [CVPR'20]	**79.28**	

3D algorithms has better accuracy compared to 2D algorithms

Background: Point Cloud NN

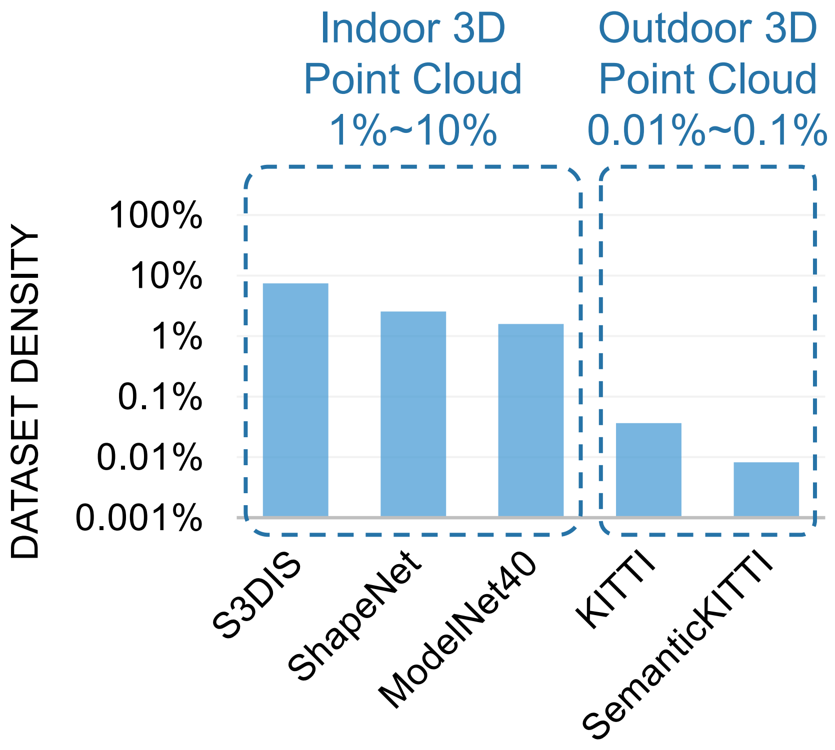
- Point Cloud Neural Networks
  - 3D algorithms have accuracy advantages over CNN and 2D algorithms
  - 3D point cloud data is sparse (0.01%~10%), the sparsity comes from the actual object

Method		Model	Car mAP (%)
2D	BEV	Complex-YOLO [CVPR'18]	77.4
		PixorNet [CVPR'18]	77.05
	Range Image	LaserNet [CVPR'19]	73.77
2.5D	voxel-based	Pointpillars [CVPR'19]	76.86
3D	voxel-based	SECOND [Sensors'18]	**78.62**
		CIASSD [AAAI'21]	**79.86**
		Voxel R-CNN [AAAI'21]	**84.52**
	point-based	PointRCNN [CVPR'19]	**78.63**
		3D-SSD [CVPR'20]	**79.57**
	voxel-point	PVRCNN [CVPR'20]	**83.61**
		SA-SSD [CVPR'20]	**79.91**
	LiDAR-Image	F-ConvNet [IROS'19]	**76.39**
		EPNet [CVPR'20]	**79.28**

2D ☹

3D ☺

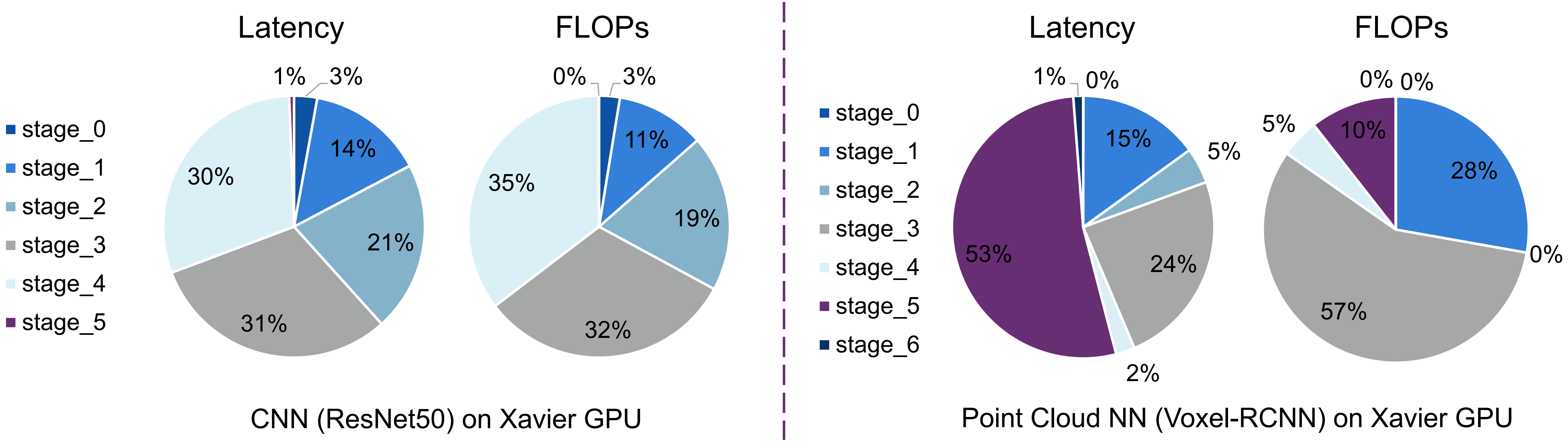
3D algorithms has better accuracy compared to 2D algorithms



3D point clouds are sparse and the sparsity comes from the actual object

Background: Memory Access

- Point cloud neural networks are sparse computing
- Inconsistent ratio of computation to time
  - Additional memory accesses

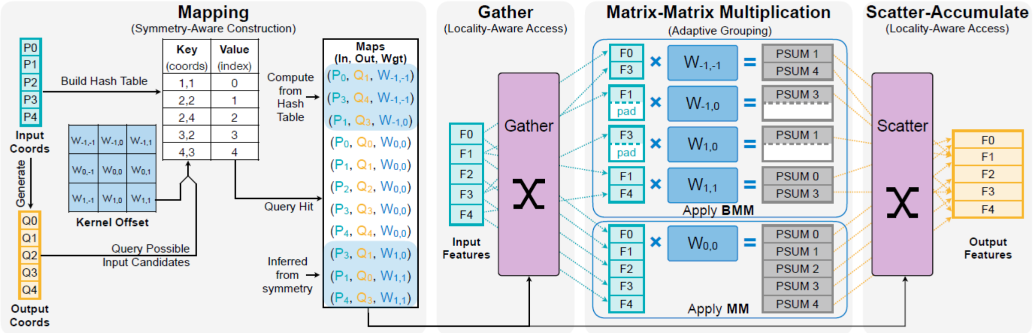


Related Works

- Point Cloud NN Acceleration
  - FPGA: Single operator acceleration
  - GPU: Operator library
  - ASIC: High performance
    - PointAcc: SOTA, Baseline

GPU

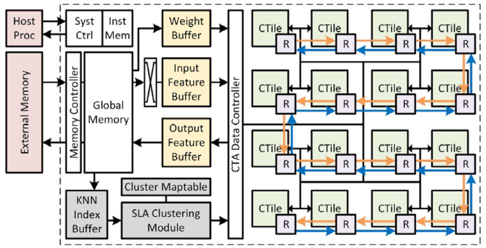
SpConv [Sensors'18], MinkowskiEngine [CVPR'19], TorchSparse [MLSys'22], PCEngine [MLSys'23]



Sparse convolutional acceleration library

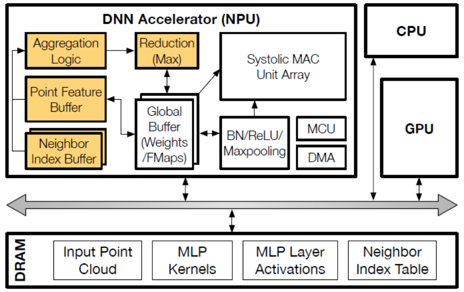
ASIC

Mesorasi [MICRO'20]



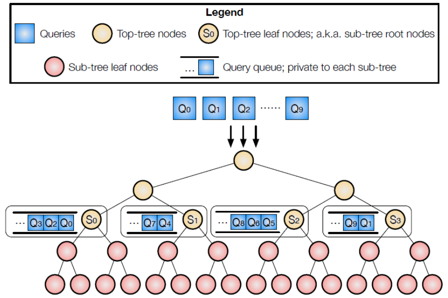
Only supports multiplication with fixed weights

Point-X [MICRO'21]



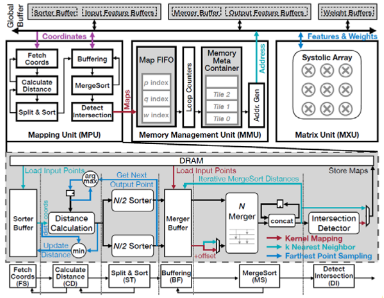
Only supports graph-based network

Crescent [ISCA'22]



Approximate calculation, requires retraining

PointAcc [MICRO'21]



First general point cloud accelerator

Related Works

- Point Cloud NN Acceleration
  - FPGA: Single operator acceleration
  - GPU: Operator library
  - ASIC: High performance
    - PointAcc: SOTA, Baseline

GPU

SpConv [Sensors'18], MinkowskiEngine [CVPR'19], TorchSparse [MLSys'22], PCEngine [MLSys'23]

Sparse convolutional acceleration library

ASIC

Mesorasi [MICRO'20]

Only supports multiplication with fixed weights

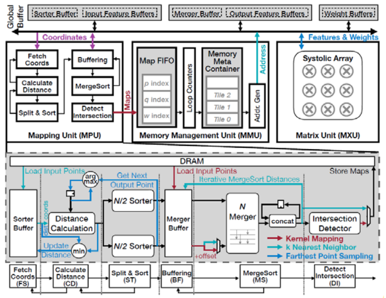
Point-X [MICRO'21]

Only supports graph-based network

Crescent [ISCA'22]

Approximate calculation, requires retraining

PointAcc [MICRO'21]

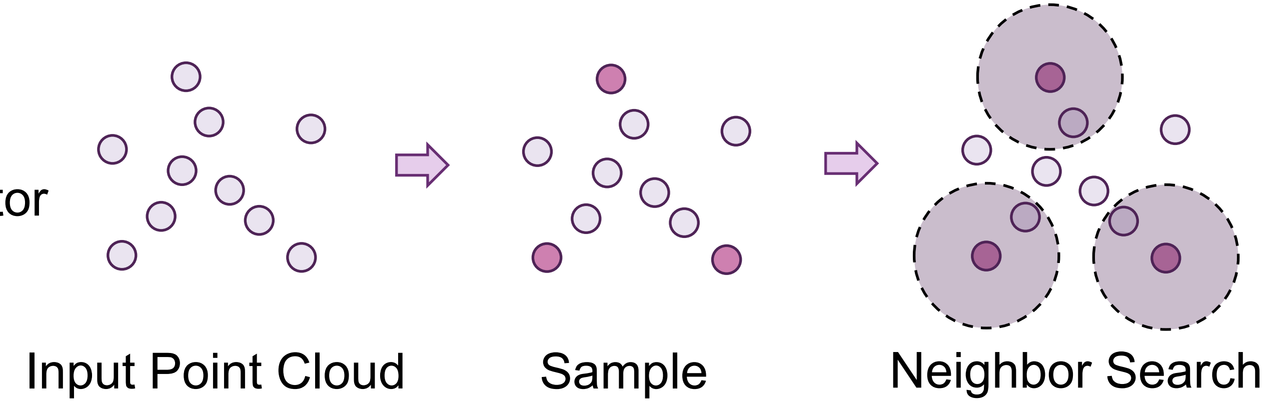


First general point cloud accelerator

Challenges: Memory Access

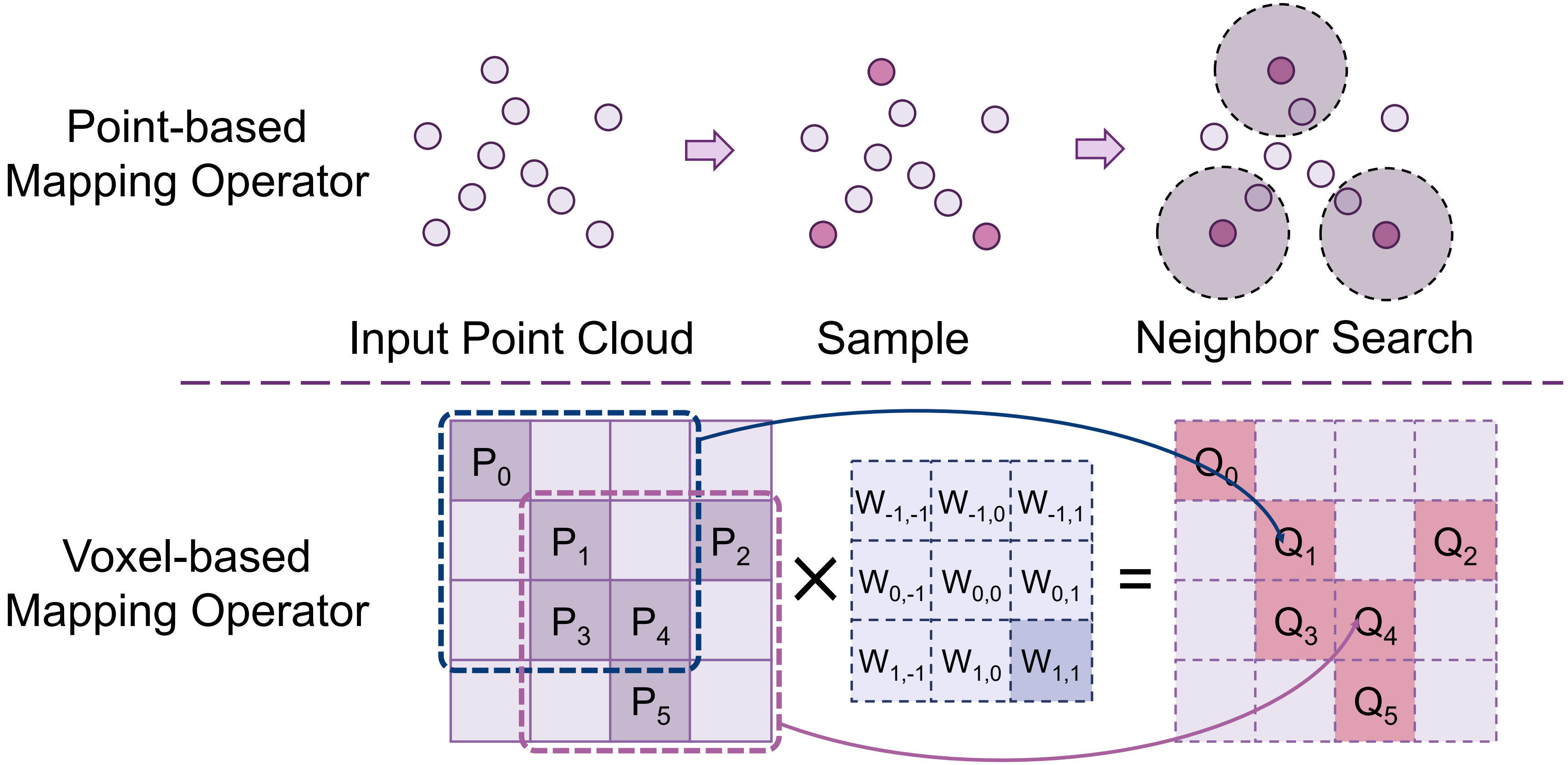
- Sparse mapping operation:
  - Redundant repetitive off-chip accesses to features **(6.5~26.3x)**

Point-based Mapping Operator



Challenges: Memory Access

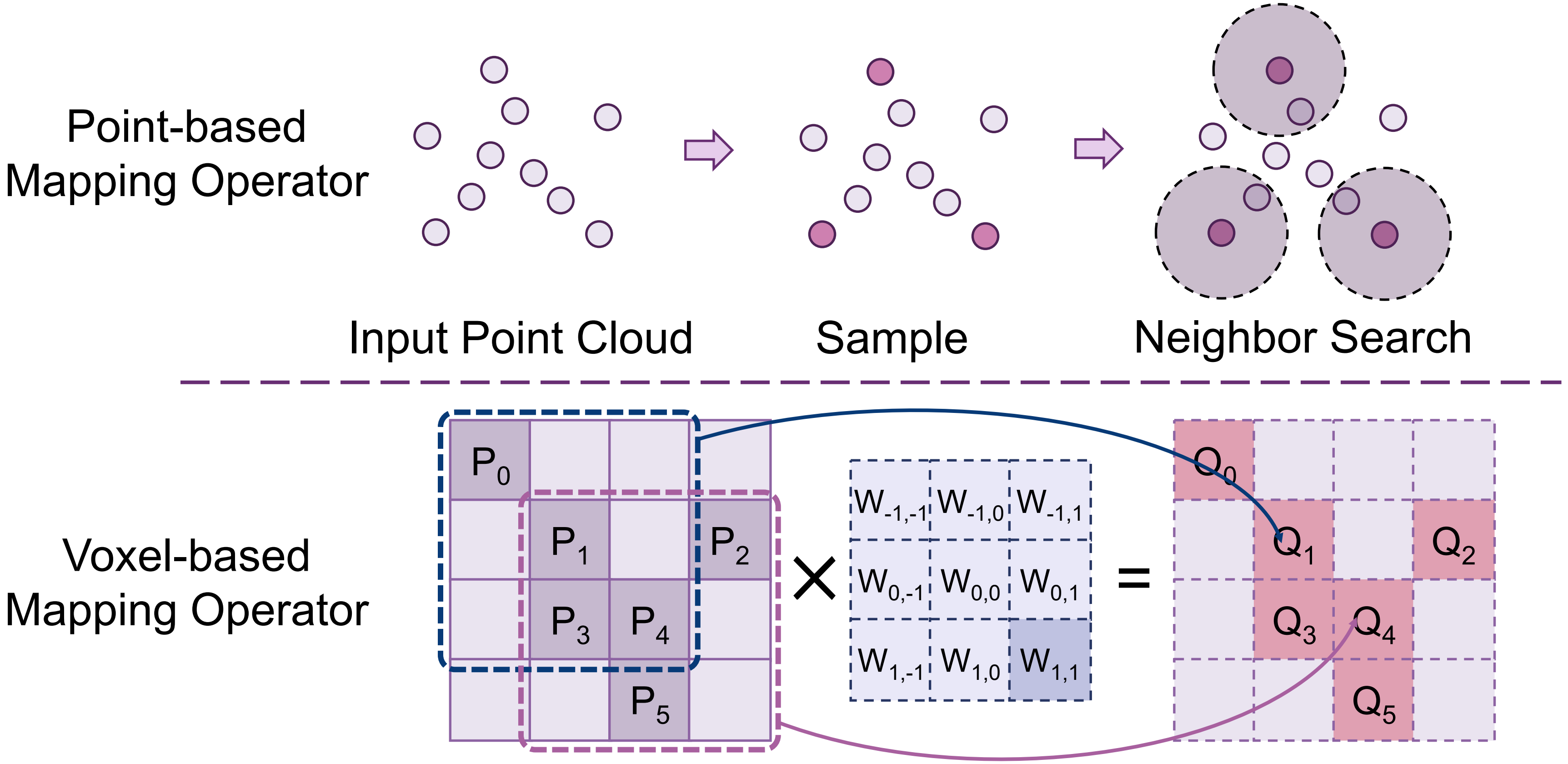
- Sparse mapping operation:
  - Redundant repetitive off-chip accesses to features **(6.5~26.3x)**



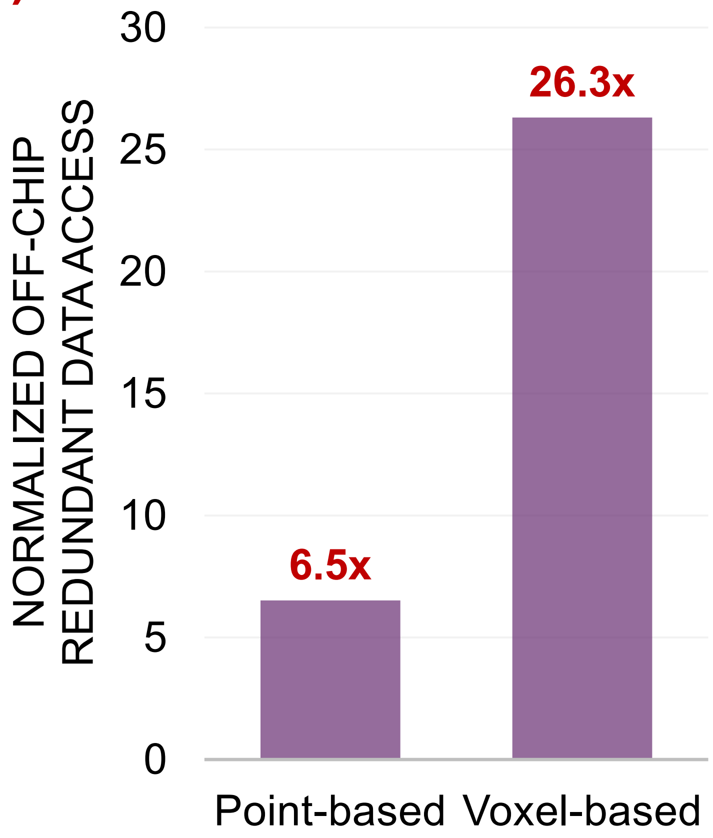
Access to all nonzero voxels when calculating each weight offset

Challenges: Memory Access

- Sparse mapping operation:
  - Redundant repetitive off-chip accesses to features **(6.5~26.3x)**



Access to all nonzero voxels when calculating each weight offset



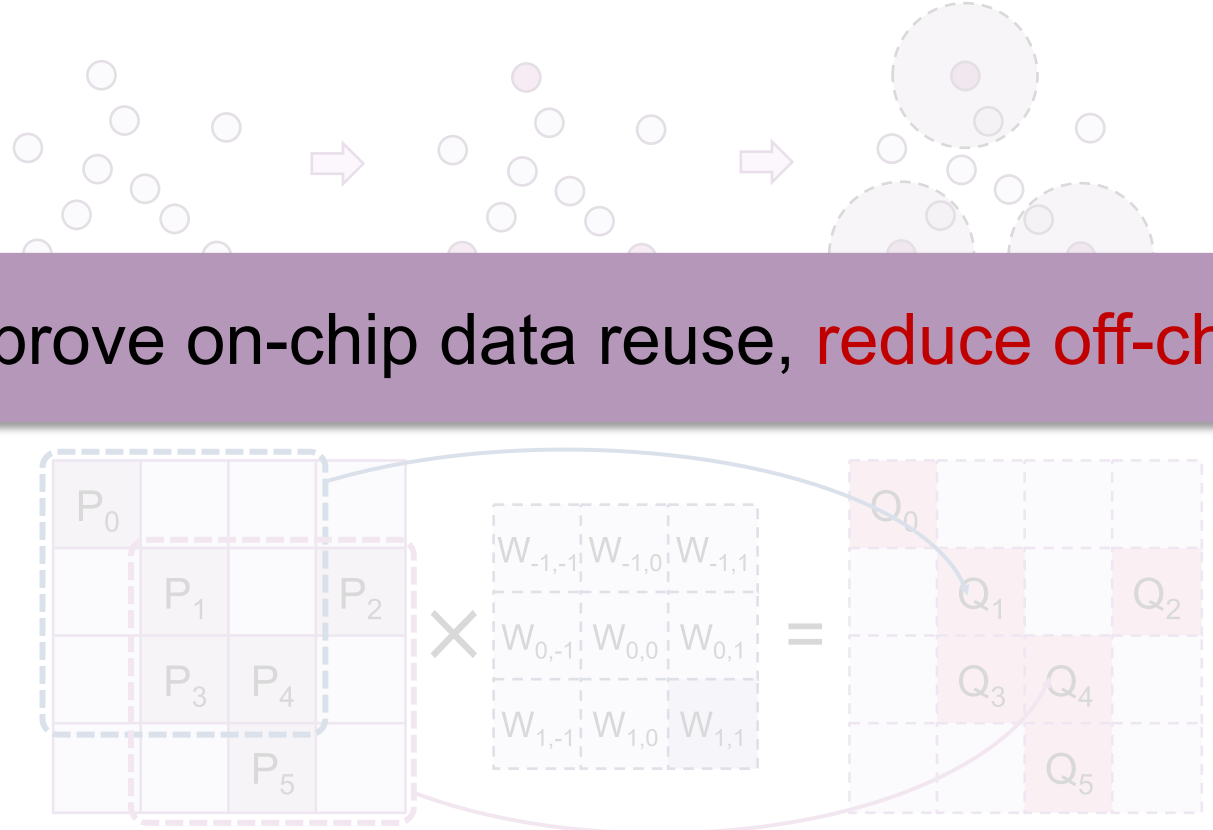
Redundant memory accesses

Challenges: Memory Access

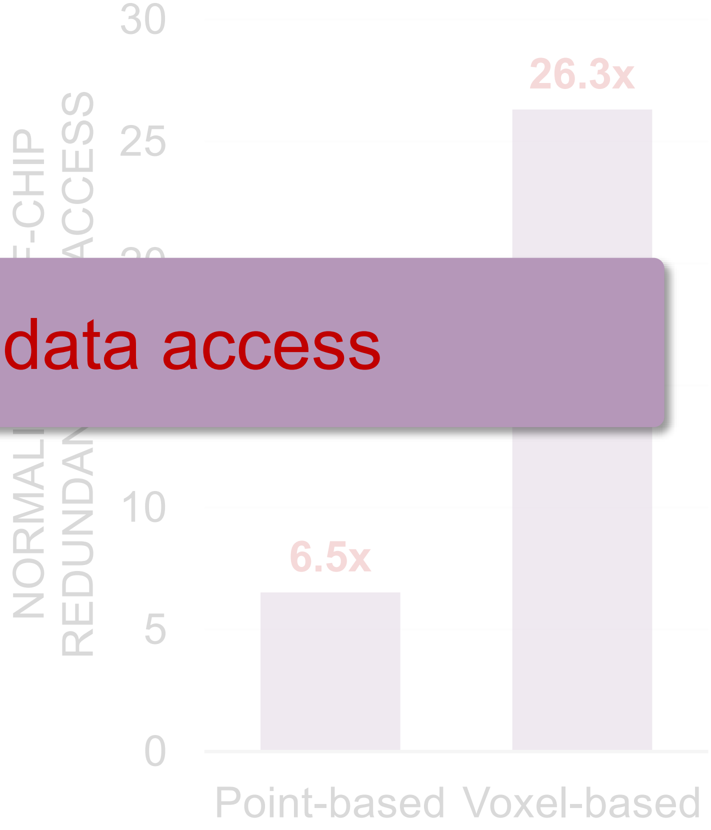
- Sparse mapping operation:
  - Redundant repetitive off-chip accesses to features **(6.5~26.3x)**

Point-based Mapping Operator

Voxel-based Mapping Operator



Access to all nonzero voxels when calculating each weight offset



Redundant memory accesses

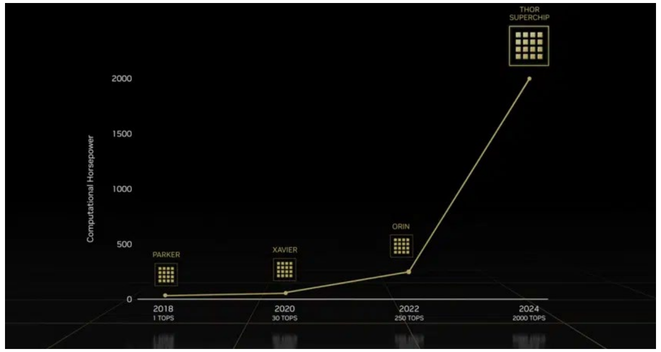
Improve on-chip data reuse, reduce off-chip data access

Challenges: Utilization

- Computing unit:
  - Increasing computing capacity of autonomous driving chips
  - Significant deterioration in computing unit utilization
    - GPU: **45.7%**@30TOPS → **27.7%**@275TOPS
    - ASIC: **40.2%**@8TOPS → **16.4%**@32TOPS

Challenges: Utilization

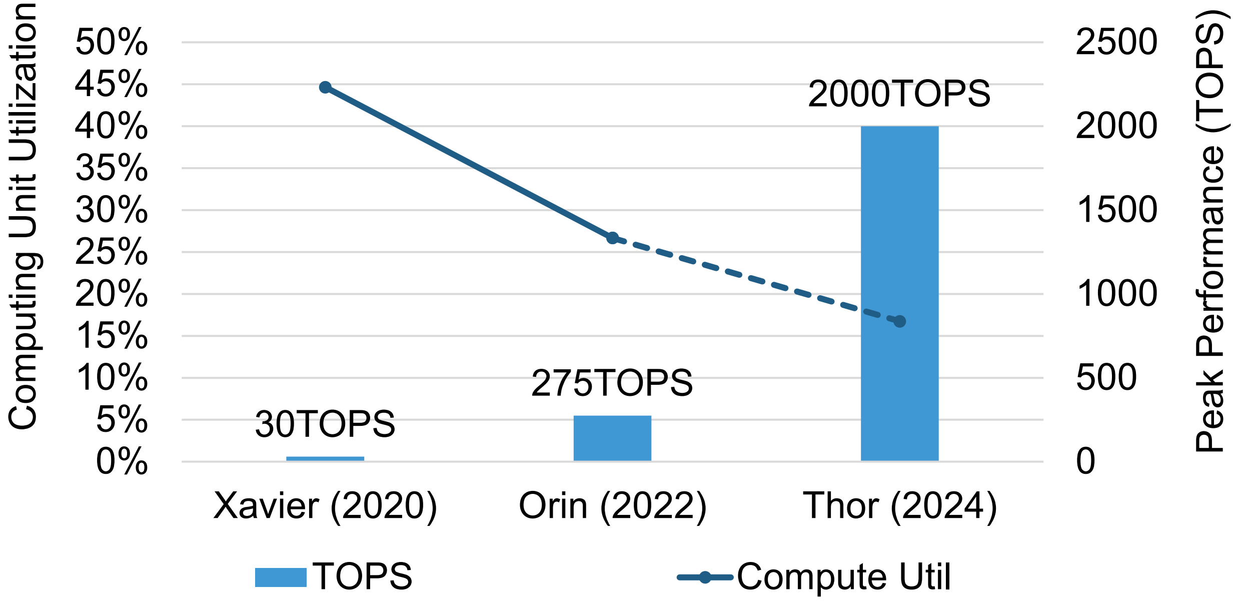
- Computing unit:
  - Increasing computing capacity of autonomous driving chips
  - Significant deterioration in computing unit utilization
    - GPU: **45.7%**@30TOPS → **27.7%**@275TOPS
    - ASIC: **40.2%**@8TOPS → **16.4%**@32TOPS



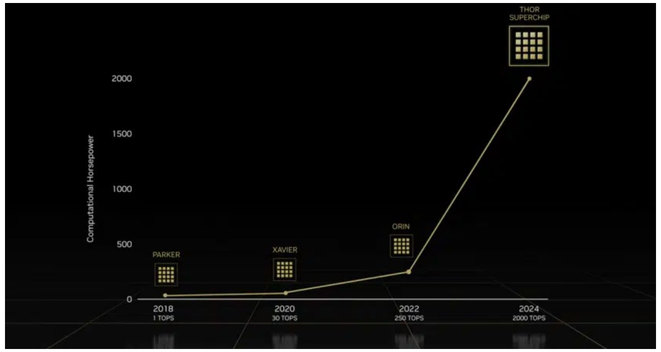
NVIDIA Roadmap: Thor chip with 2000 TOPS to be launched in 2024

Challenges: Utilization

- Computing unit:
  - Increasing computing capacity of autonomous driving chips
  - Significant deterioration in computing unit utilization
    - GPU: **45.7%**@30TOPS → **27.7%**@275TOPS
    - ASIC: **40.2%**@8TOPS → **16.4%**@32TOPS



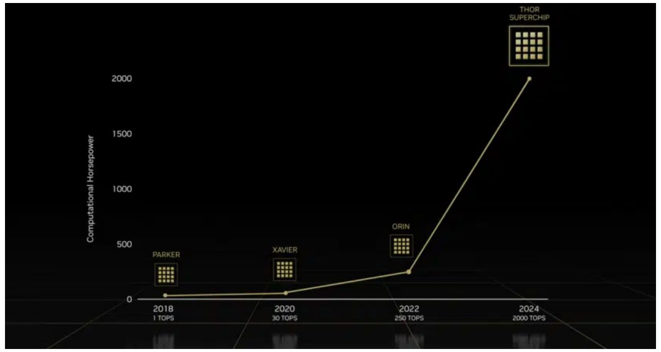
NVIDIA autonomous driving GPU



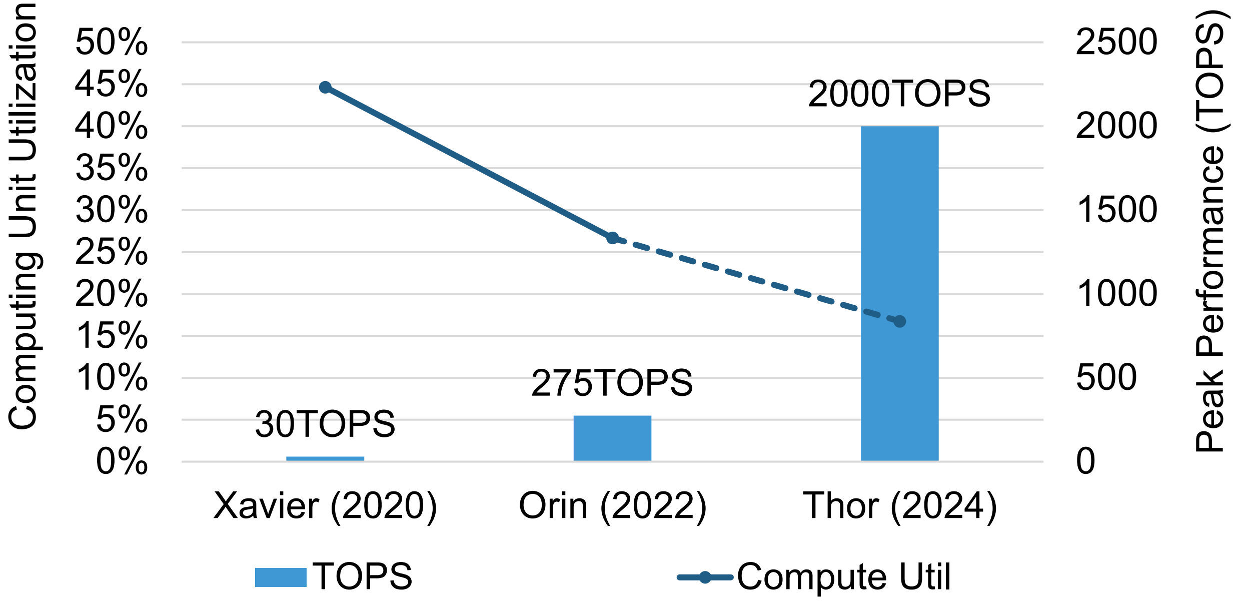
NVIDIA Roadmap: Thor chip with 2000 TOPS to be launched in 2024

Challenges: Utilization

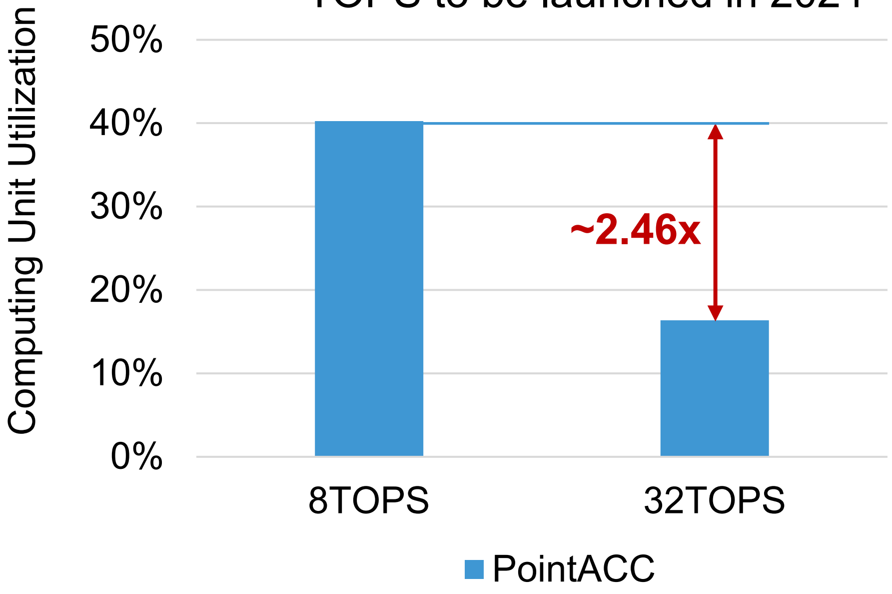
- Computing unit:
  - Increasing computing capacity of autonomous driving chips
  - Significant deterioration in computing unit utilization
    - GPU: **45.7%**@30TOPS → **27.7%**@275TOPS
    - ASIC: **40.2%**@8TOPS → **16.4%**@32TOPS



NVIDIA Roadmap: Thor chip with 2000 TOPS to be launched in 2024



NVIDIA autonomous driving GPU



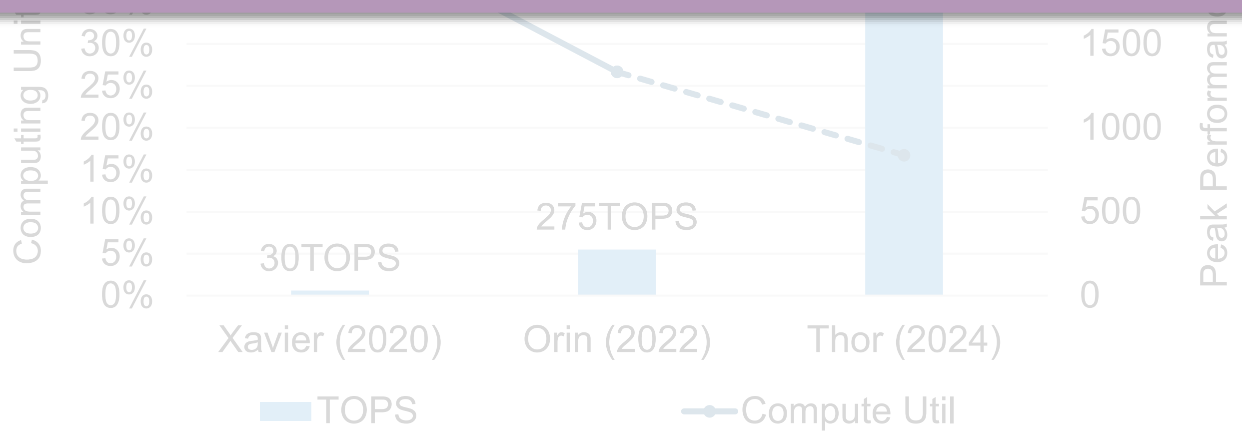
ASIC accelerator PointAcc [MICRO’21]

Challenges: Utilization

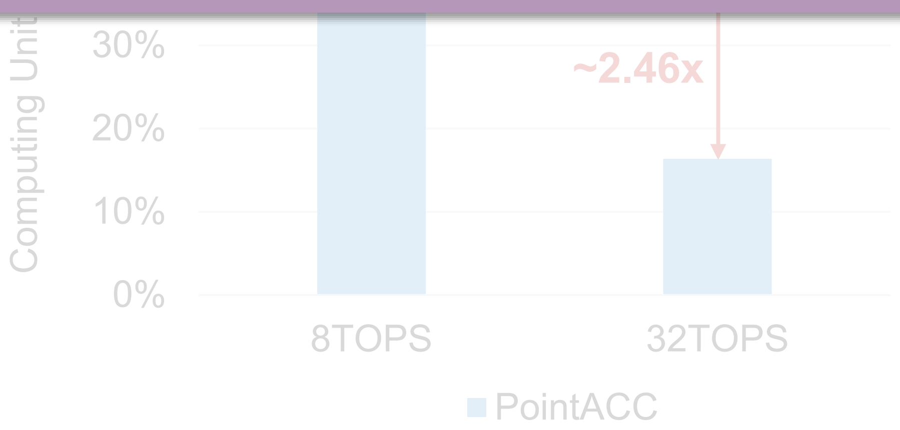
- Computing unit:
  - Increasing computing capacity of autonomous driving chips
  - Significant deterioration in computing unit utilization
    - GPU: **45.7%**@30TOPS → **27.7%**@275TOPS
    - ASIC: **40.2%**@8TOPS → **16.4%**@32TOPS

NVIDIA Roadmap: Thor chip with 2000 TOPS to be launched in 2024

Design scalable hardware architecture with high utilization

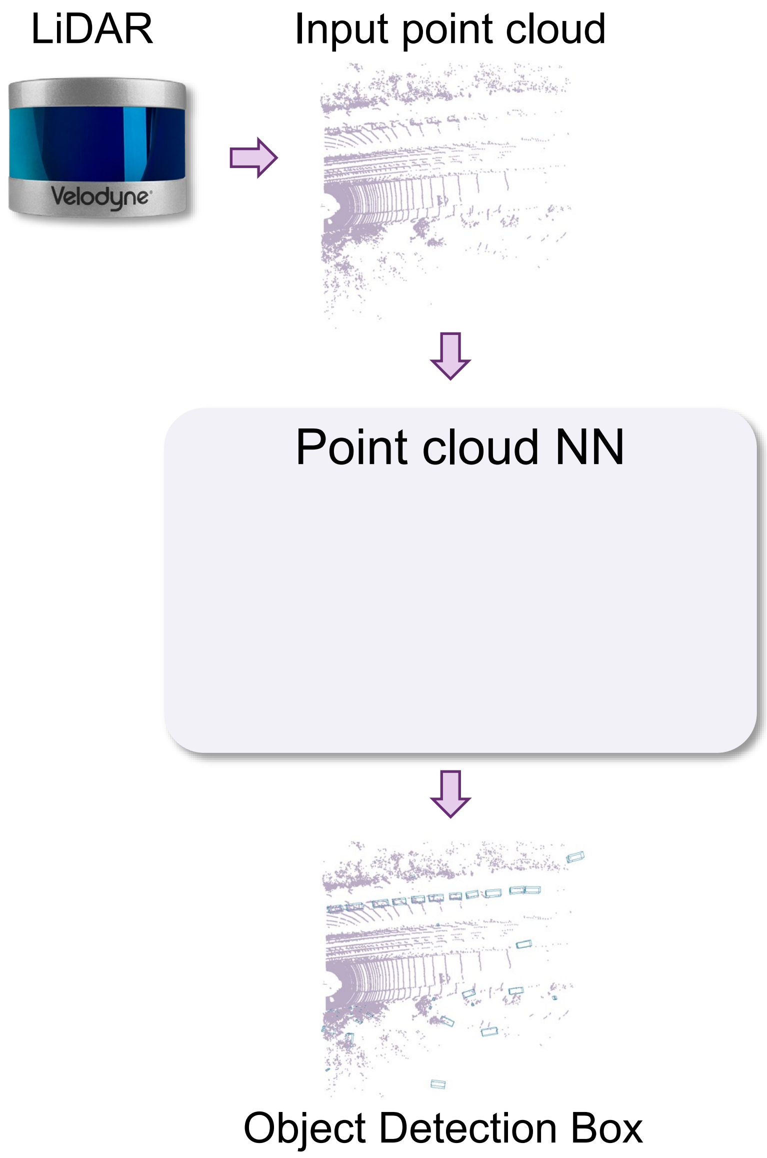


NVIDIA autonomous driving GPU



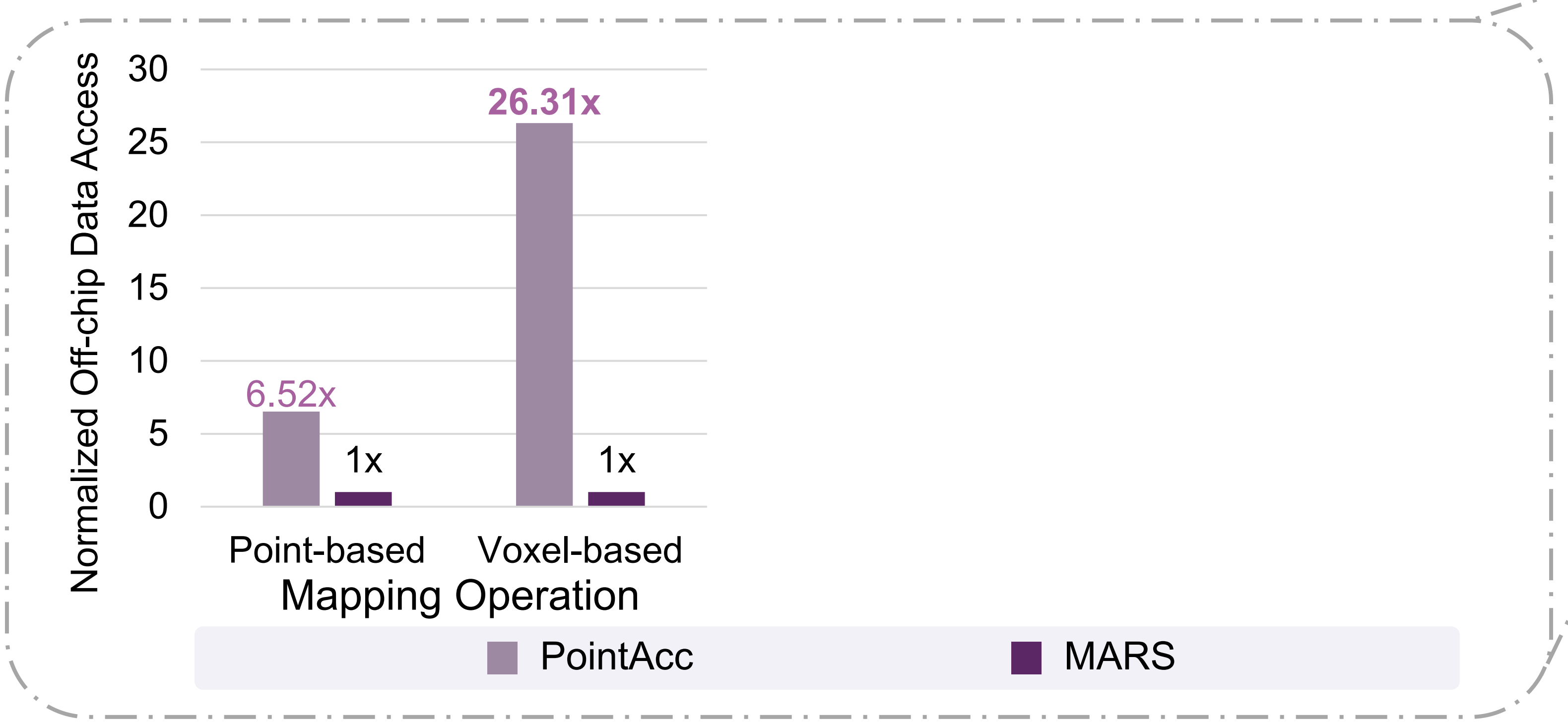
ASIC accelerator PointAcc [MICRO'21]

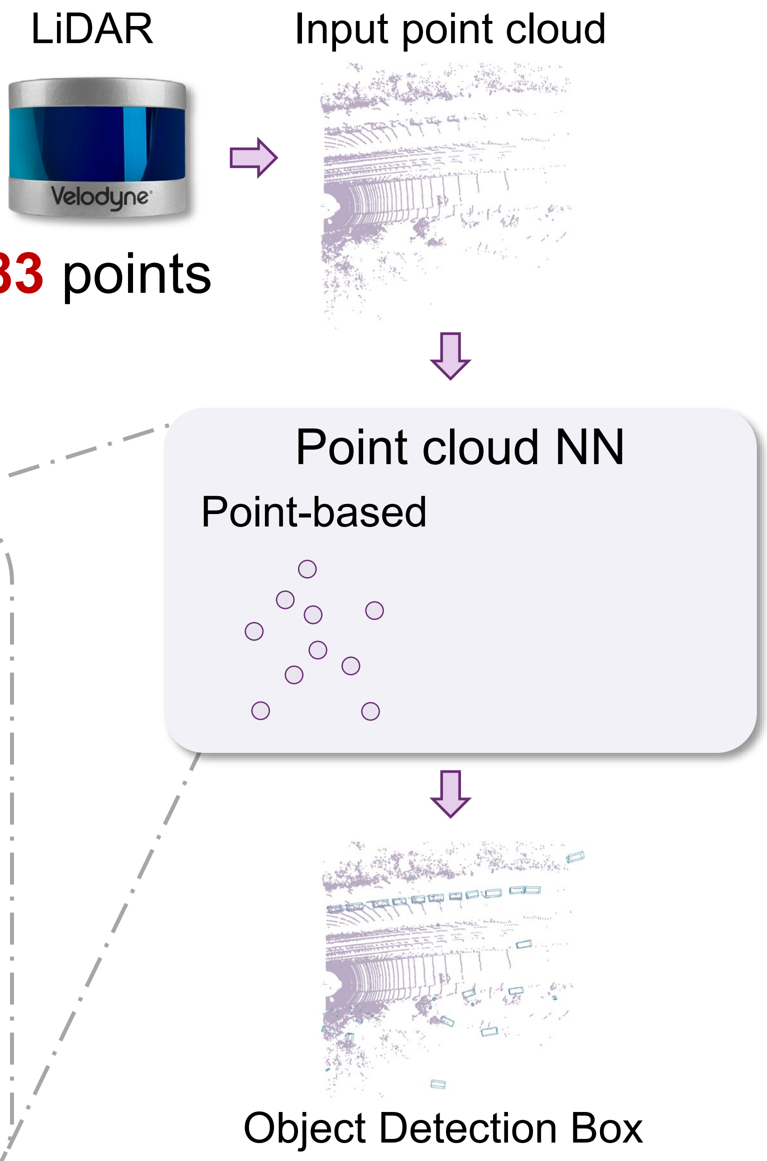
Challenge: Summary



Challenge: Summary

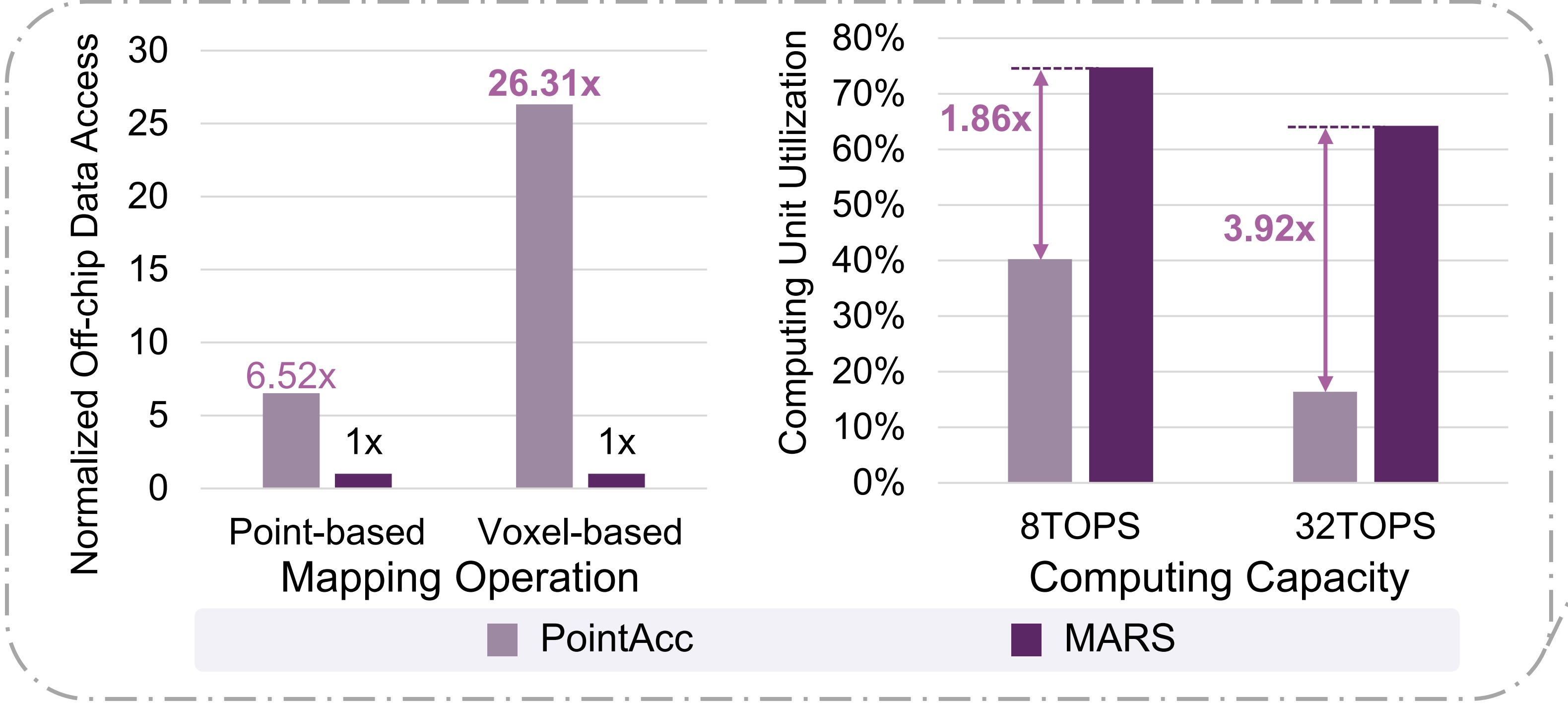
- Mapping: Large off-chip memory access
  - E.g., in PointNet++, each sampled point requires visiting **~983** points

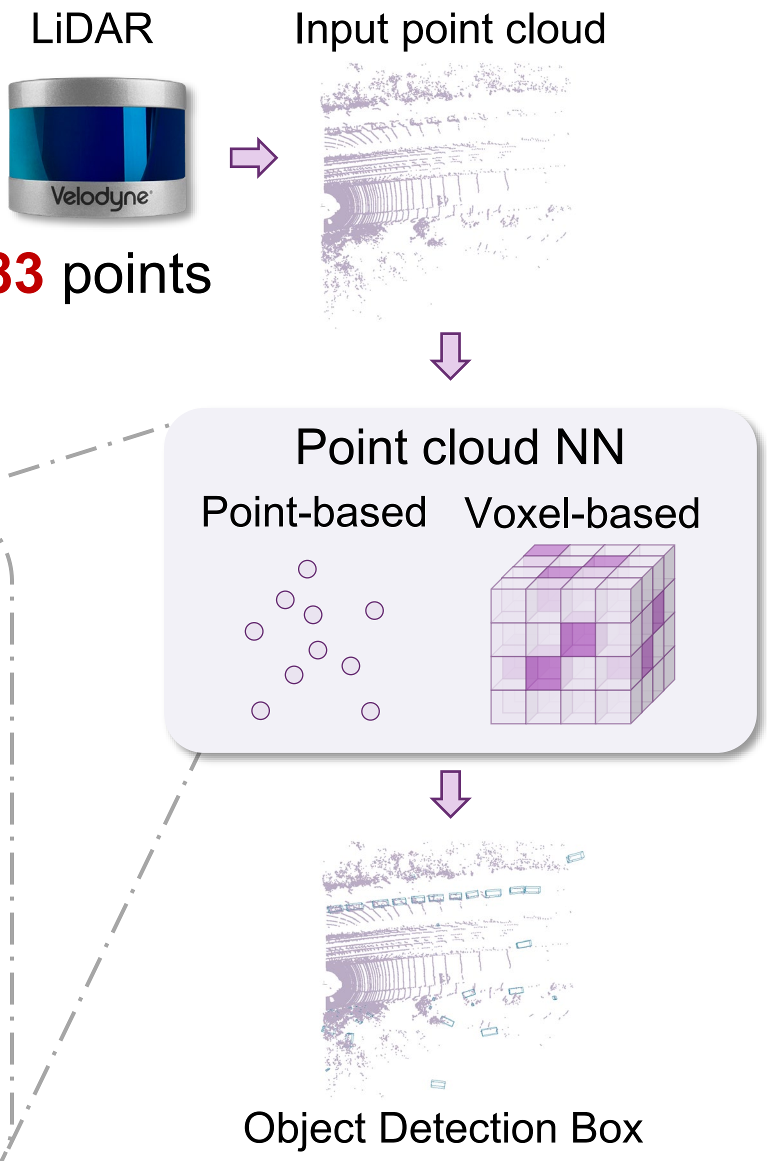




Challenge: Summary

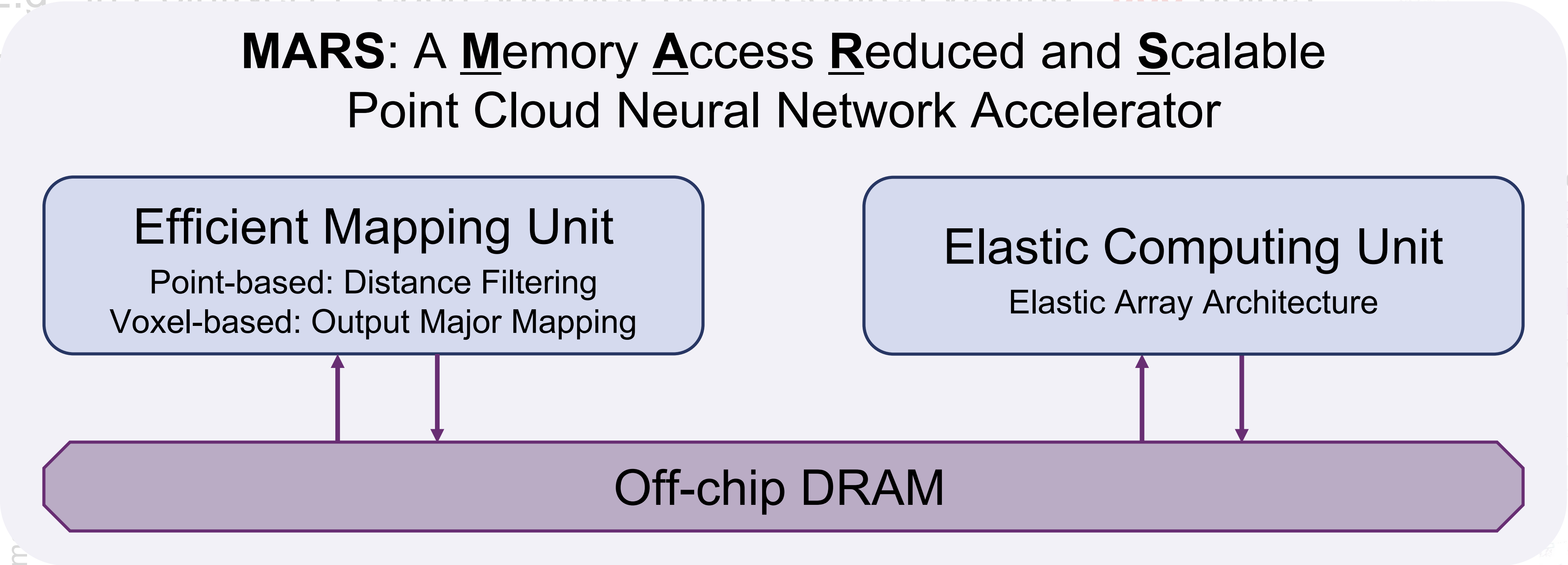
- Mapping: Large off-chip memory access
  - E.g., in PointNet++, each sampled point requires visiting **~983** points
- Computing: Low computing unit utilization
  - E.g., **40.25%** on 64x64 array → **16.37%** on 128x128 array





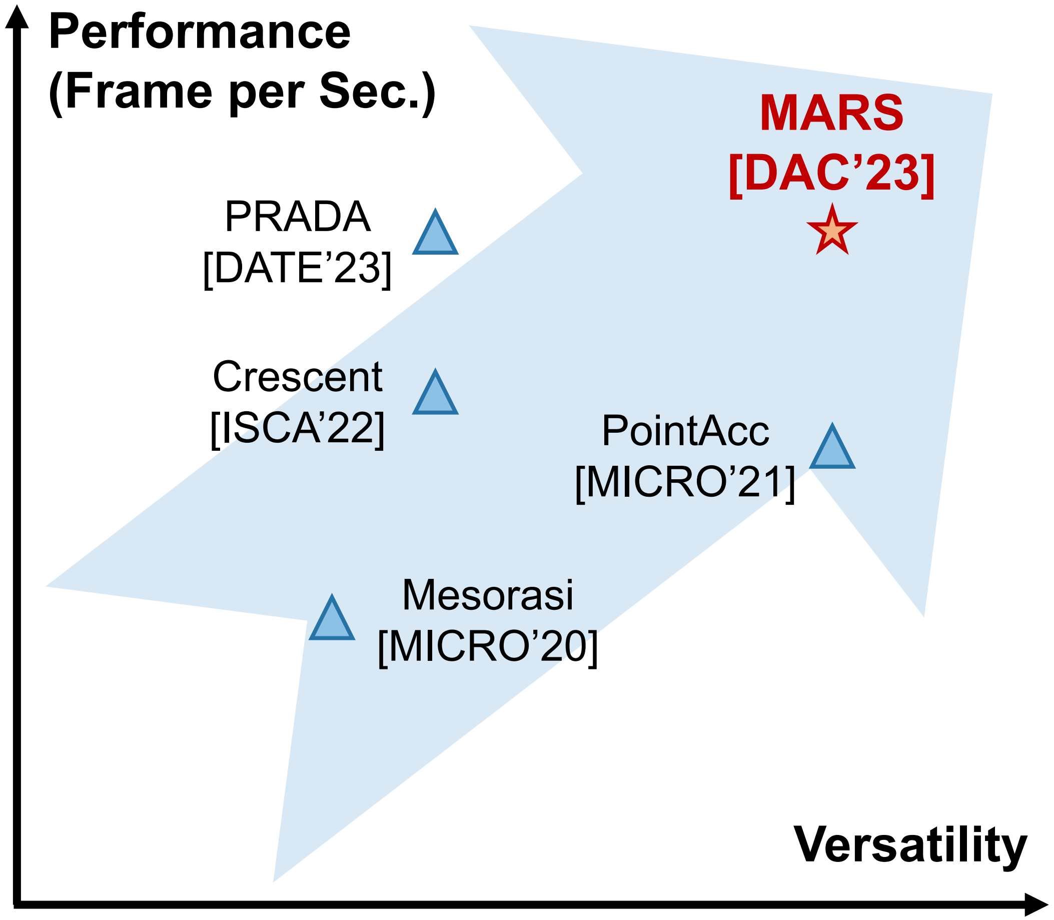
Challenge: Summary

- Mapping: Large off-chip memory access
  - E.g., in PointNet++, each sampled point requires visiting ~983 points



Comparison with existing works

- The comparison of existing ASIC-based point cloud accelerators

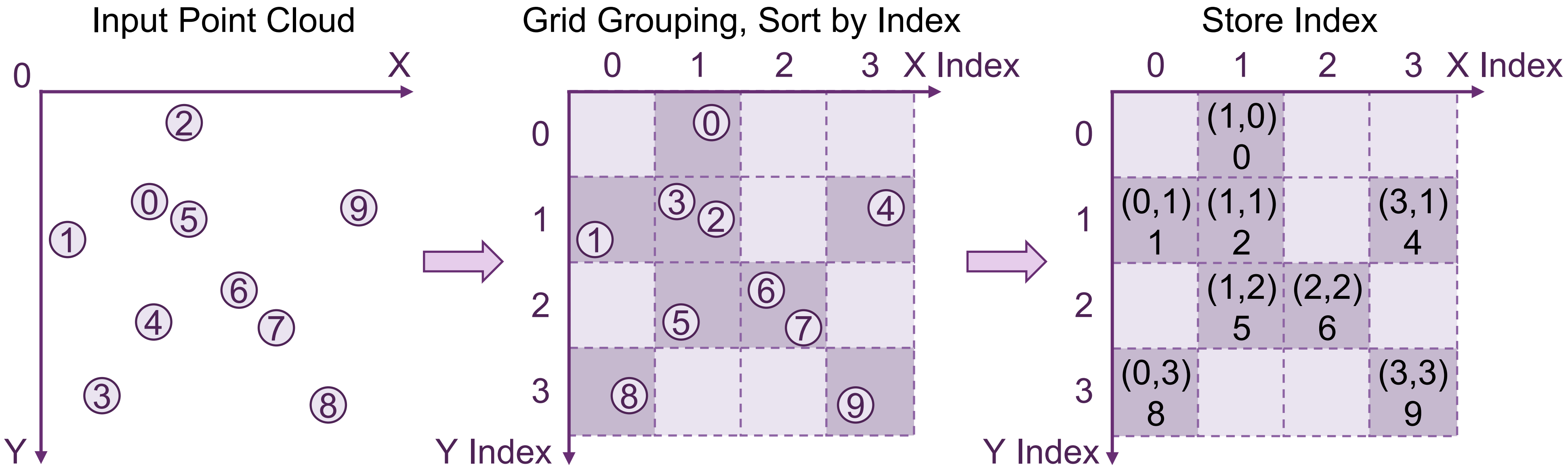


Point Cloud Accelerator	Versatility	HW/SW Co-opt.	No Acc. Loss	Memory Reduced
Mesorasi [MICRO'20]	✗	✓	✗	✓
PointAcc [MICRO'21]	✓	✓	✓	✗
Crescent [ISCA'22]	✗	✓	✗	✓
PRADA [DATE'23]	✗	✓	✗	✓
MARS [DAC'23]	✓	✓	✓	✓

Contents

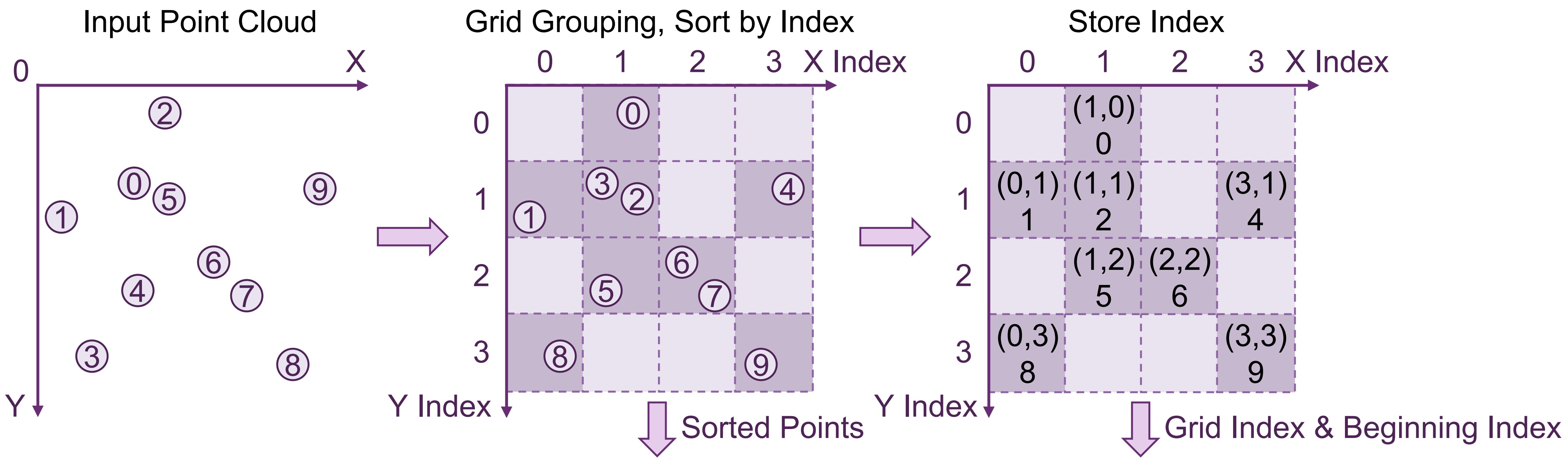
Efficient Mapping Unit: Point-based

- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Preprocessing stage: Partition the points into grids



Efficient Mapping Unit: Point-based

- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Preprocessing stage: Partition the points into grids

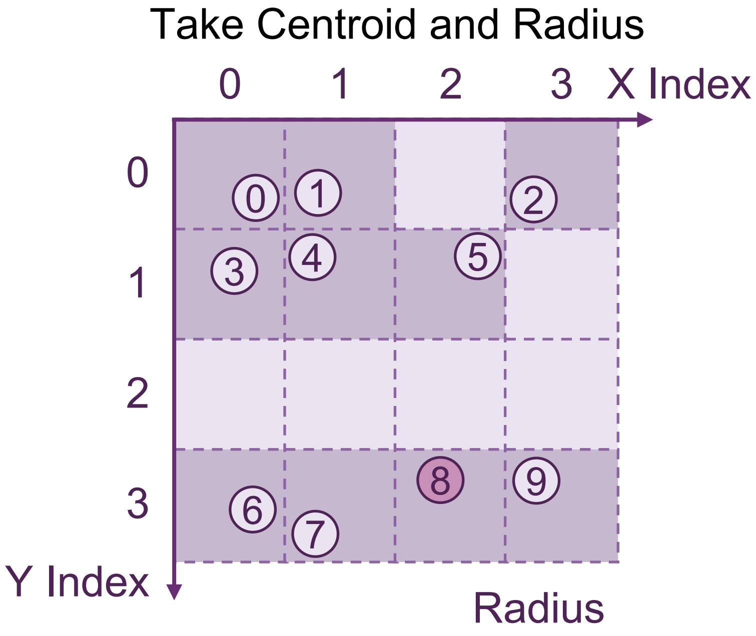


Data Layout

Grid Index	(1,0)	(0,1)	(1,1)	(3,1)	(1,2)	(2,2)	(0,3)	(3,3)	On-chip Buffer
Beginning Index	0	1	2	4	5	6	8	9	
Sorted Points	⓪	①	②③	④	⑤	⑥⑦	⑧	⑨	Off-chip DRAM

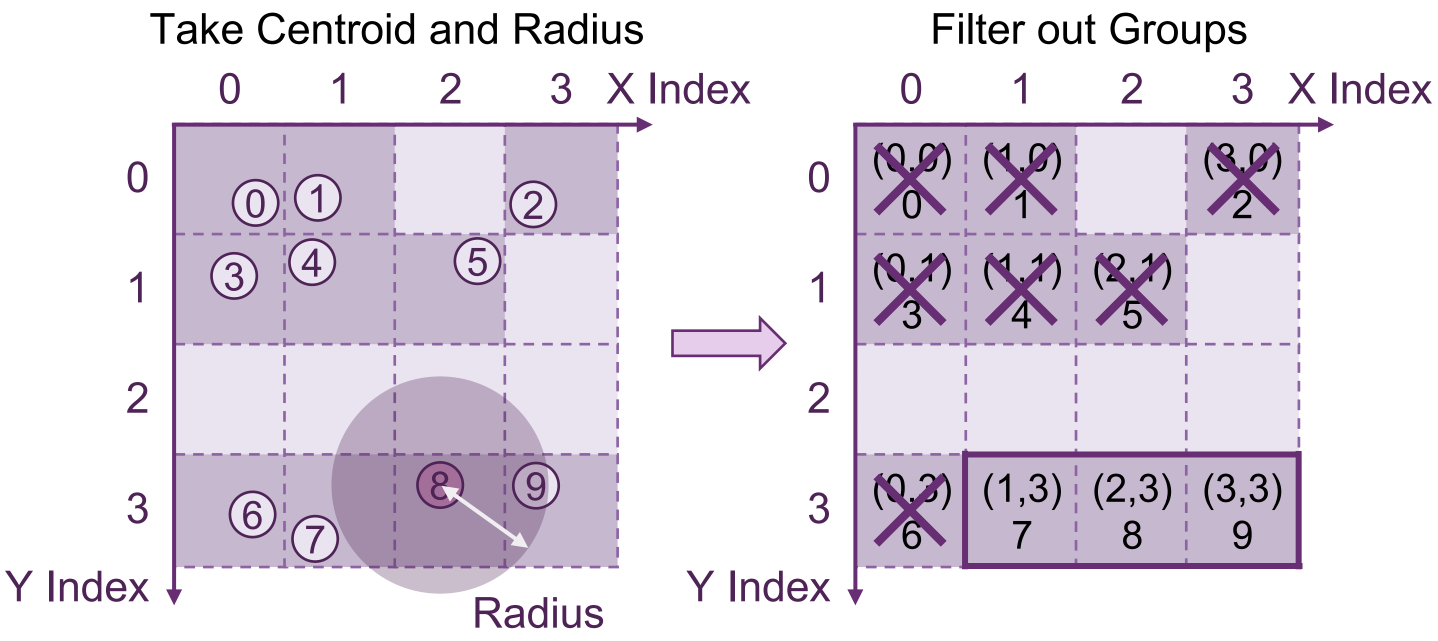
Efficient Mapping Unit: Point-based

- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Runtime stage: Exclude grids with distances outside the radius



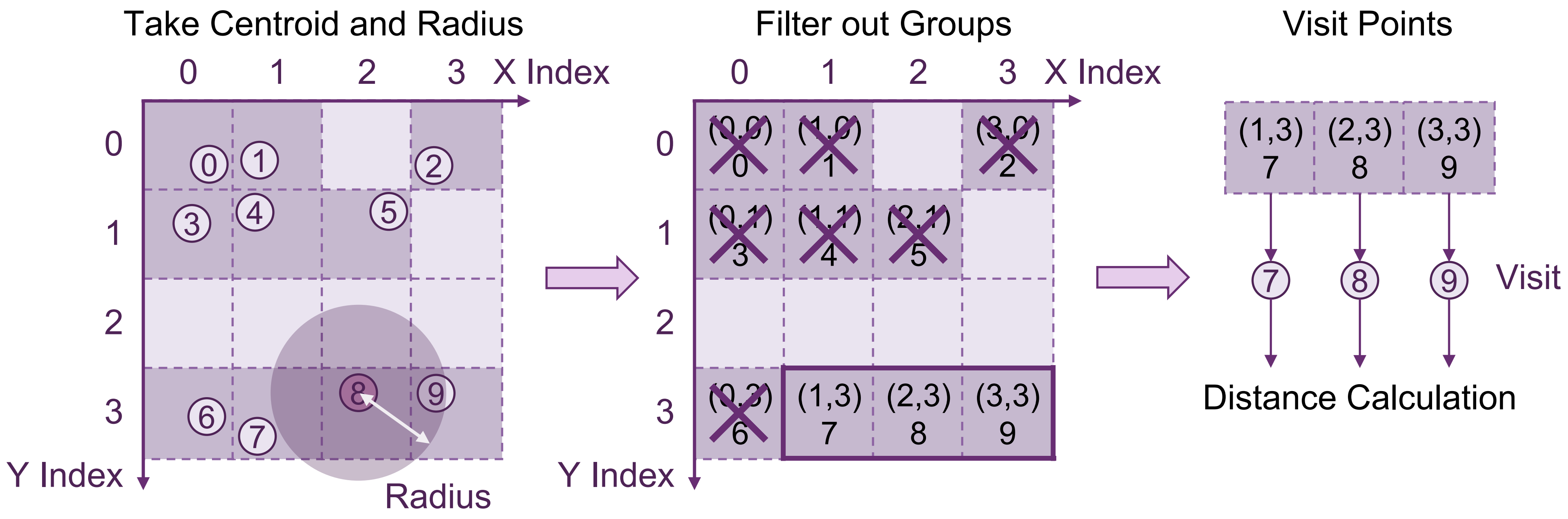
Efficient Mapping Unit: Point-based

- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Runtime stage: Exclude grids with distances outside the radius



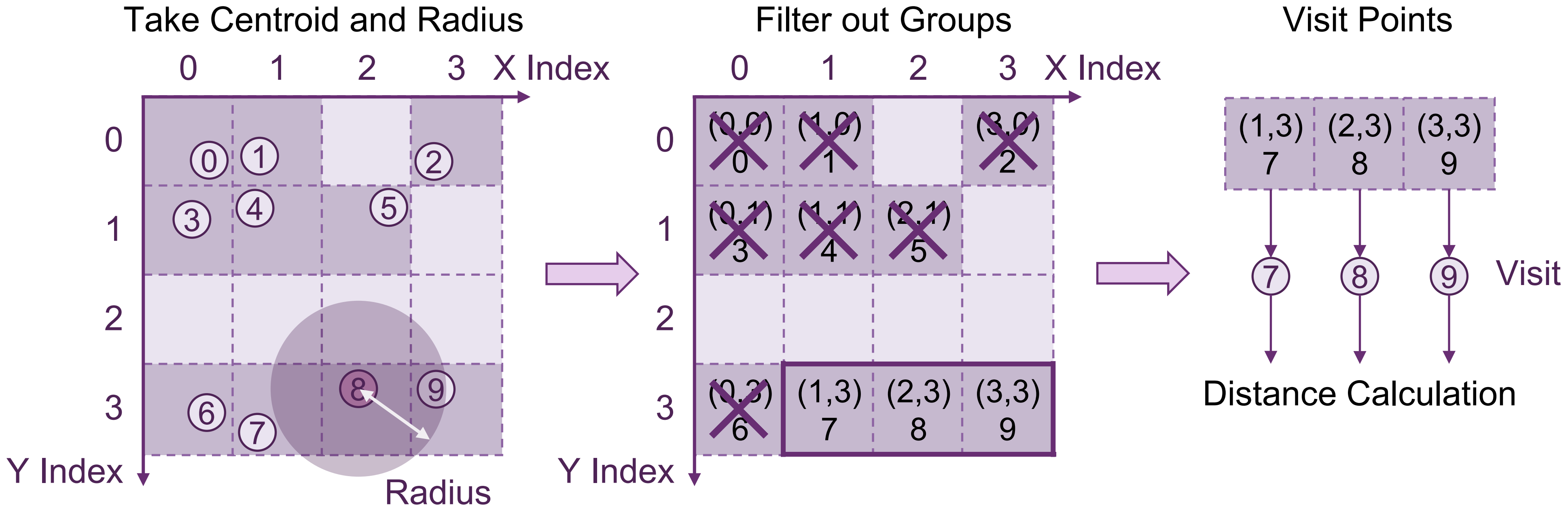
Efficient Mapping Unit: Point-based

- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Runtime stage: Exclude grids with distances outside the radius



Efficient Mapping Unit: Point-based

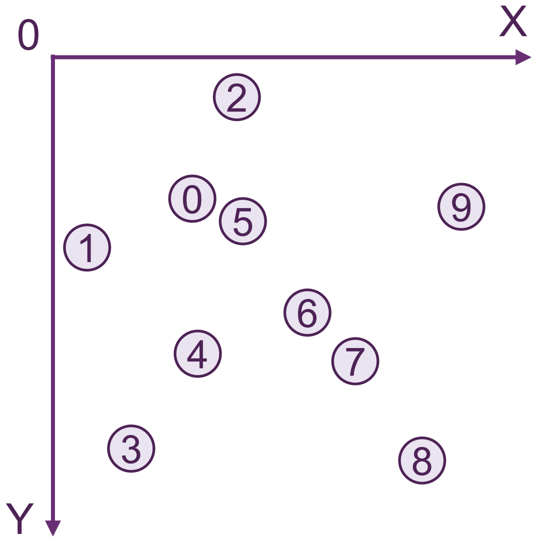
- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Runtime stage: Exclude grids with distances outside the radius



$Radius = ?\, in\, FPS$

Efficient Mapping Unit: Point-based

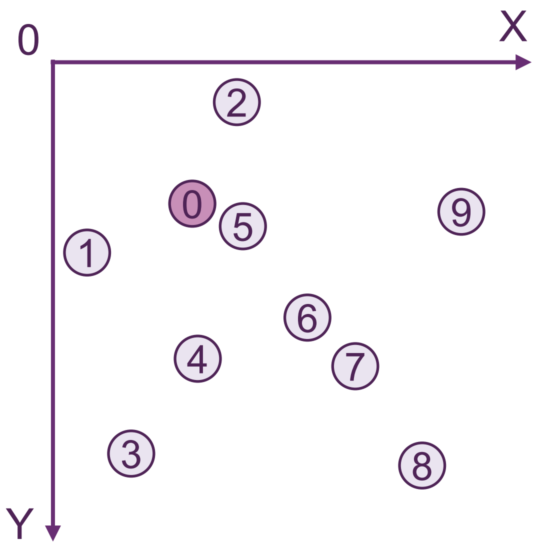
- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Runtime stage: Exclude grids with distances outside the radius
  - FPS: Maintain a table of minimum distances from all points to the sample centroid set



Iteration 0:

Efficient Mapping Unit: Point-based

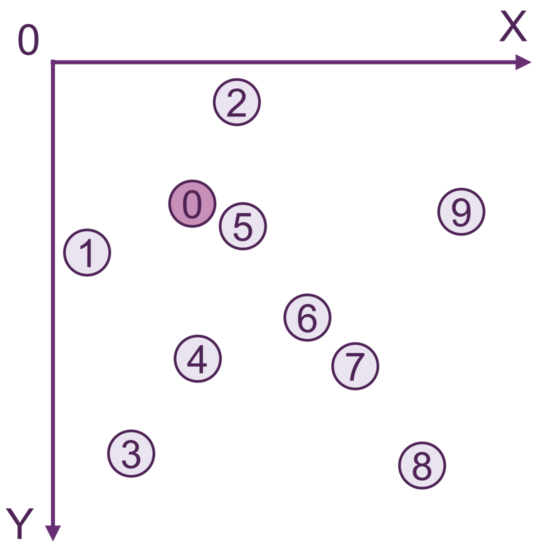
- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Runtime stage: Exclude grids with distances outside the radius
  - FPS: Maintain a table of minimum distances from all points to the sample centroid set



Iteration 0: Choose 0

Efficient Mapping Unit: Point-based

- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Runtime stage: Exclude grids with distances outside the radius
  - FPS: Maintain a table of minimum distances from all points to the sample centroid set

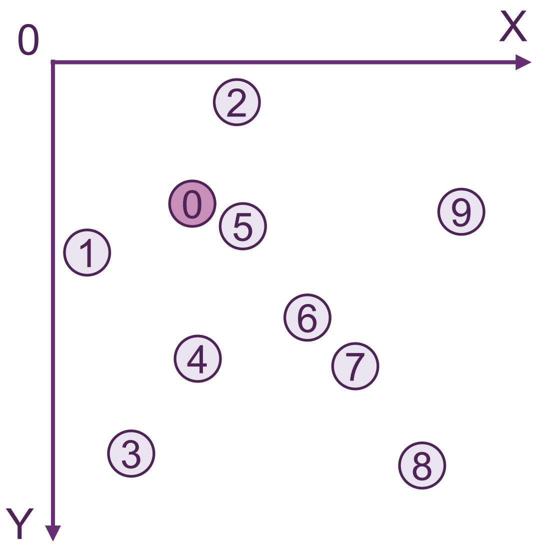


Point	Min Dist
0	0
1	2.5
2	2.4
3	4.8
4	3.1
5	0.5
6	3.3
7	4.0
8	**5.5**
9	4.9

Iteration 0: Choose 0, Initiate Table

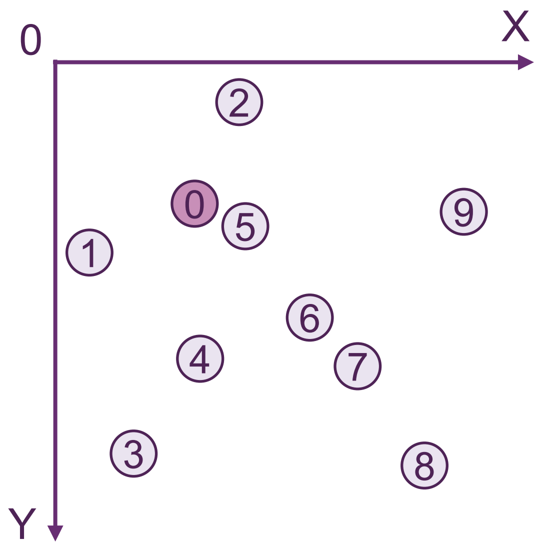
Efficient Mapping Unit: Point-based

- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Runtime stage: Exclude grids with distances outside the radius
  - FPS: Maintain a table of minimum distances from all points to the sample centroid set



Point	Min Dist
0	0
1	2.5
2	2.4
3	4.8
4	3.1
5	0.5
6	3.3
7	4.0
8	**5.5**
9	4.9

Iteration 0: Choose 0, Initiate Table

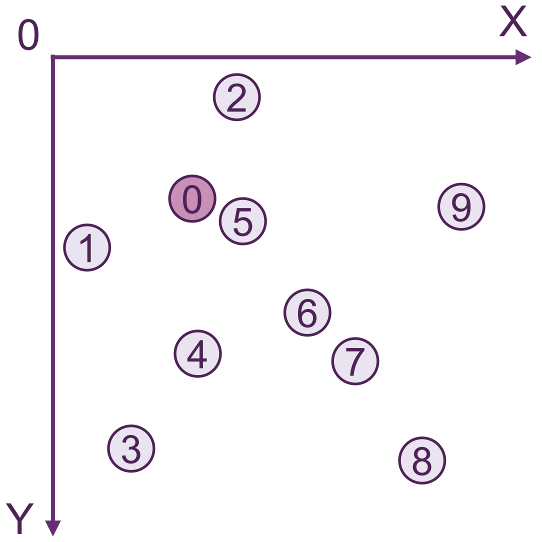


Point	Min Dist
0	0
1	2.5
2	2.4
3	4.8
4	3.1
5	0.5
6	3.3
7	4.0
8	**5.5**
9	4.9

Iteration 1:

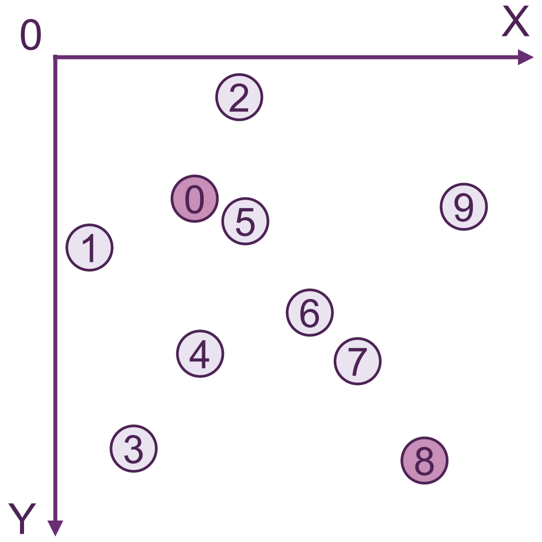
Efficient Mapping Unit: Point-based

- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Runtime stage: Exclude grids with distances outside the radius
  - FPS: Maintain a table of minimum distances from all points to the sample centroid set



Point	Min Dist
0	0
1	2.5
2	2.4
3	4.8
4	3.1
5	0.5
6	3.3
7	4.0
8	**5.5**
9	4.9

Iteration 0: Choose 0, Initiate Table

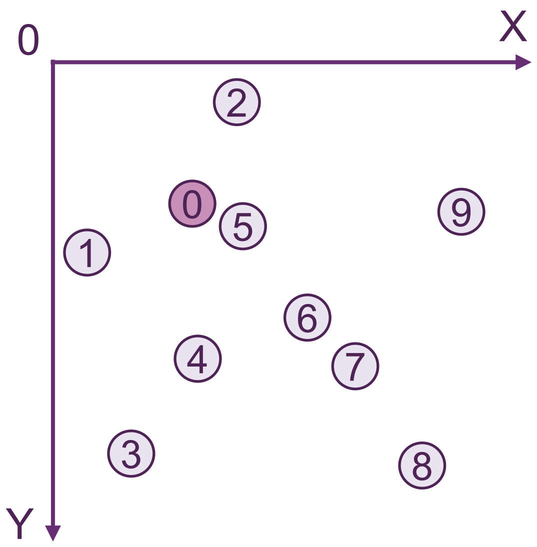


Point	Min Dist
0	0
1	2.5
2	2.4
3	4.8
4	3.1
5	0.5
6	3.3
7	4.0
8	**5.5**
9	4.9

Iteration 1: Choose 8

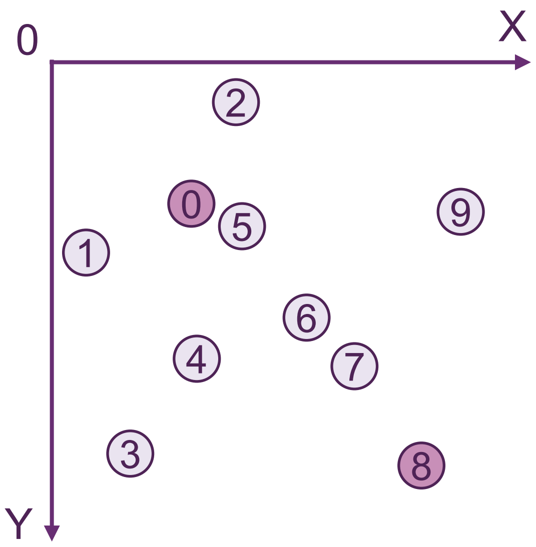
Efficient Mapping Unit: Point-based

- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Runtime stage: Exclude grids with distances outside the radius
  - FPS: Maintain a table of minimum distances from all points to the sample centroid set



Point	Min Dist
0	0
1	2.5
2	2.4
3	4.8
4	3.1
5	0.5
6	3.3
7	4.0
8	**5.5**
9	4.9

Iteration 0: Choose ⓪, Initiate Table

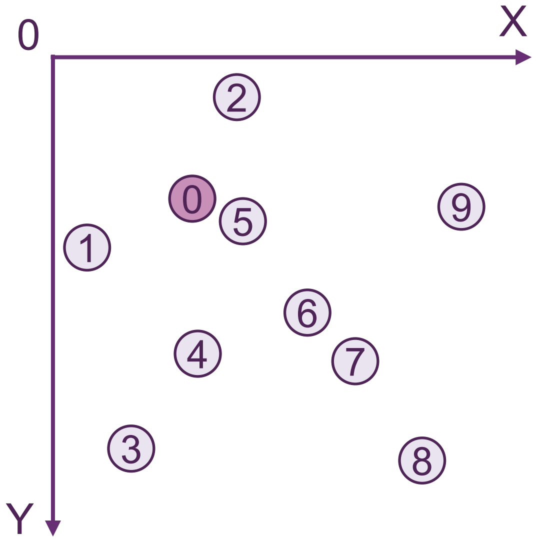


Point	Min Dist
0	0
1	2.5
2	2.4
3	4.8
4	3.1
5	0.5
6	3.3
7	2.7
8	0
9	4.7

Iteration 1: Choose ⑧, Update Table

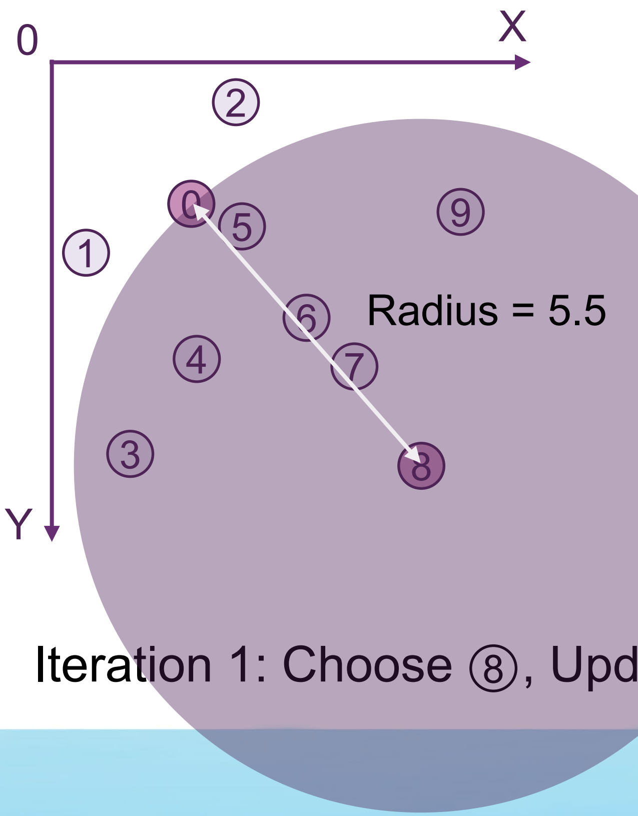
Efficient Mapping Unit: Point-based

- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Runtime stage: Exclude grids with distances outside the radius
  - FPS: Maintain a table of minimum distances from all points to the sample centroid set



Point	Min Dist
0	0
1	2.5
2	2.4
3	4.8
4	3.1
5	0.5
6	3.3
7	4.0
8	**5.5**
9	4.9

Iteration 0: Choose 0, Initiate Table

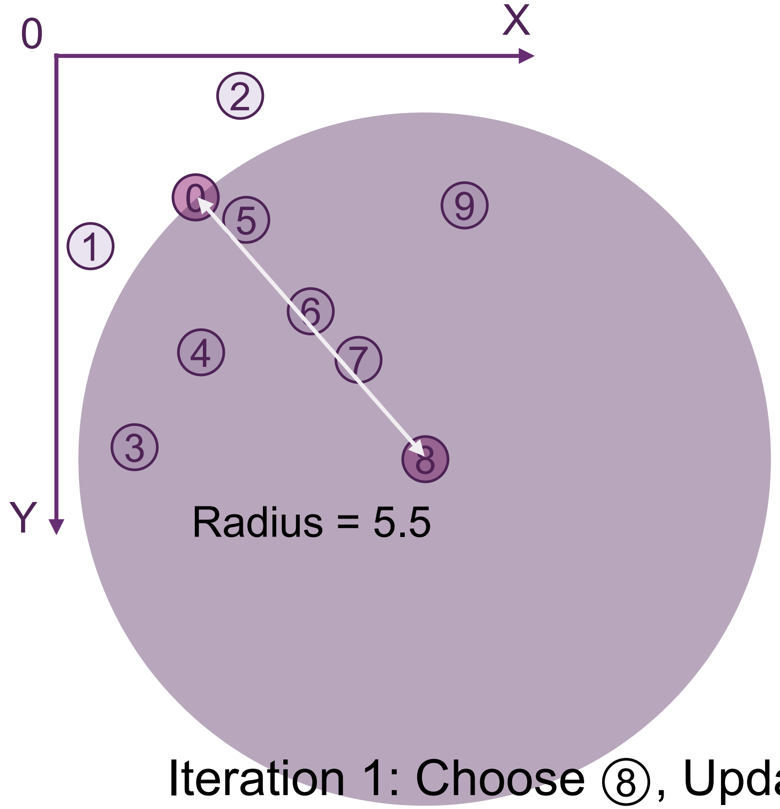


Point	Min Dist
0	0
1	2.5
2	2.4
3	4.8
4	3.1
5	0.5
6	3.3
7	2.7
8	0
9	4.7

Iteration 1: Choose 8, Update Table

Efficient Mapping Unit: Point-based

- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Runtime stage: Exclude grids with distances outside the radius
  - FPS: Maintain a table of minimum distances from all points to the sample centroid set

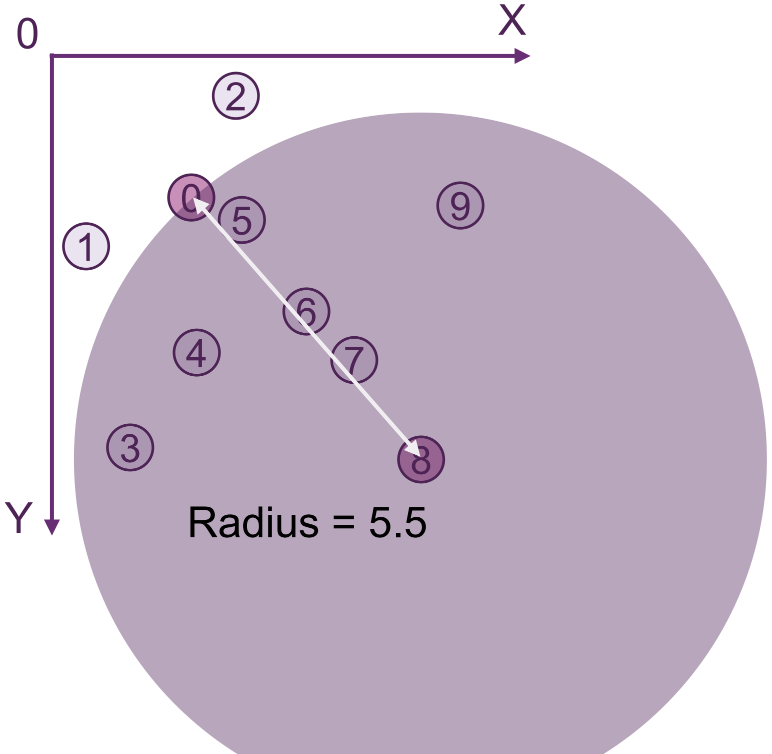


Point	Min Dist
0	0
1	2.5
2	2.4
3	4.8
4	3.1
5	0.5
6	3.3
7	4.0
8	**5.5**
9	4.9

Iteration 1: Choose 8, Update Table

Efficient Mapping Unit: Point-based

- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Runtime stage: Exclude grids with distances outside the radius
  - FPS: Maintain a table of minimum distances from all points to the sample centroid set



Iteration 1: Choose ⑧, Update Table

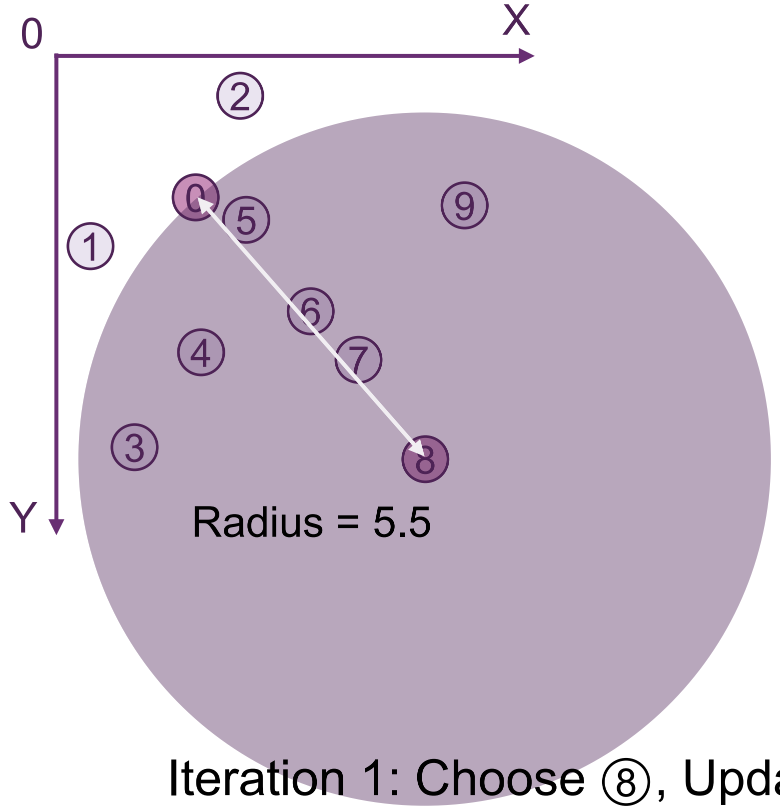
Point	Min Dist
0	0
1	2.5
2	2.4
3	4.8
4	3.1
5	0.5
6	3.3
7	4.0
8	**5.5**
9	4.9

Sufficient and unnecessary conditions for **no** distance updating:

$$\Rightarrow d(point, centroid) \geq \max(min\ dist)$$

Efficient Mapping Unit: Point-based

- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Runtime stage: Exclude grids with distances outside the radius
  - FPS: Maintain a table of minimum distances from all points to the sample centroid set



Point	Min Dist
0	0
1	2.5
2	2.4
3	4.8
4	3.1
5	0.5
6	3.3
7	4.0
8	**5.5**
9	4.9

Iteration 1: Choose ⑧, Update Table

Sufficient and unnecessary conditions for **no** distance updating:

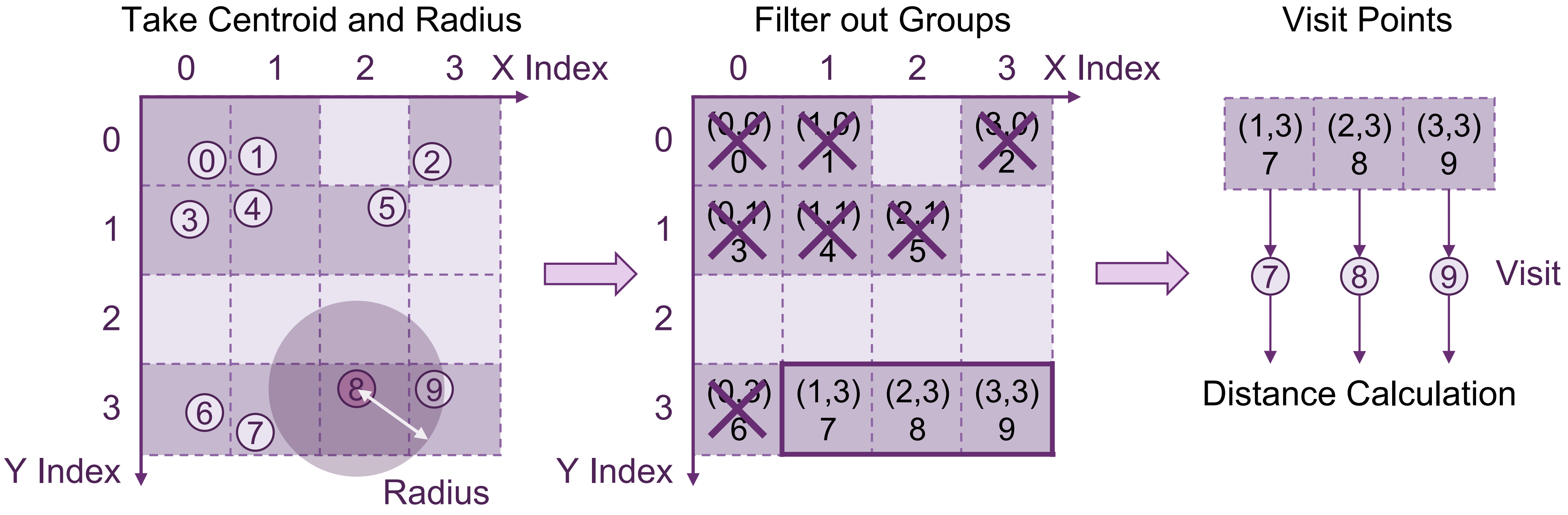
$$d(point, centroid) \geq \max(min\ dist)$$

$$Radius = \max(min\ dist)$$

No additional calculations required !

Efficient Mapping Unit: Point-based

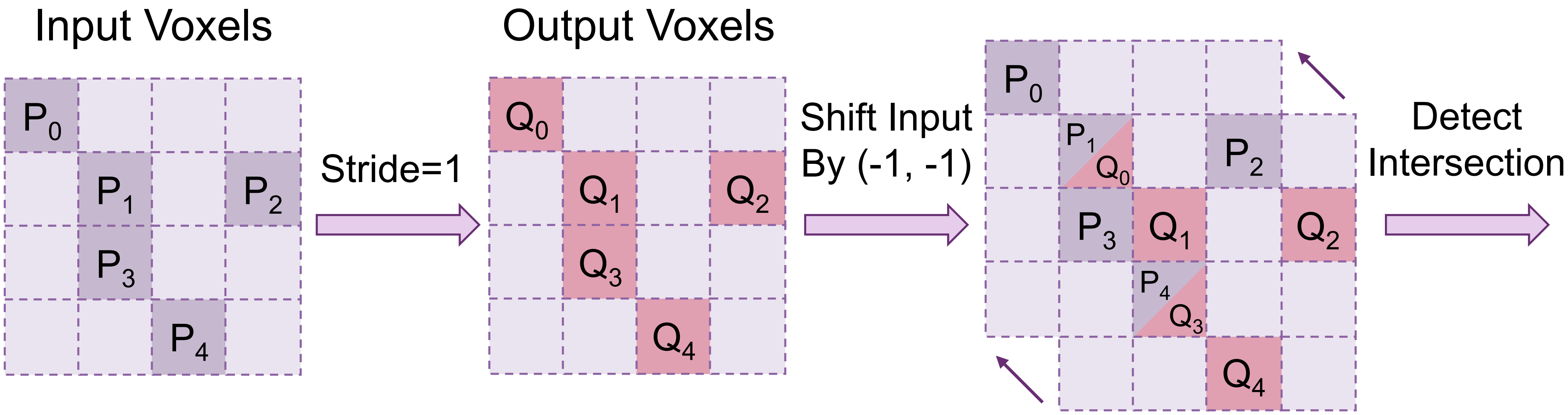
- Point-based Mapping Operation (FPS, Ball Query, kNN): Distance Filtering
  - Runtime stage: Exclude grids with distances outside the radius
  - FPS: Maintain a table of minimum distances from all points to the sample centroid set



$$Radius = \max(min\ dist)$$

Efficient Mapping Unit: Voxel-based

- Voxel-based Mapping Operation (Kernel mapping): Output-Major Mapping
  - Existing implementation: calculate the mapping of one weight offset at a time



(In, Out, Wgt)
(P_1, Q_0, $W_{1,1}$)
(P_4, Q_3, $W_{1,1}$)

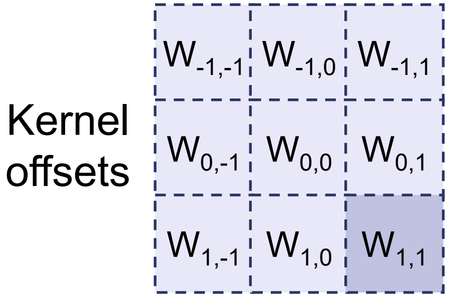
$W_{-1,-1}$	$W_{-1,0}$	$W_{-1,1}$
$W_{0,-1}$	$W_{0,0}$	$W_{0,1}$
$W_{1,-1}$	$W_{1,0}$	$W_{1,1}$

Mapping for **one** kernel offset at a time

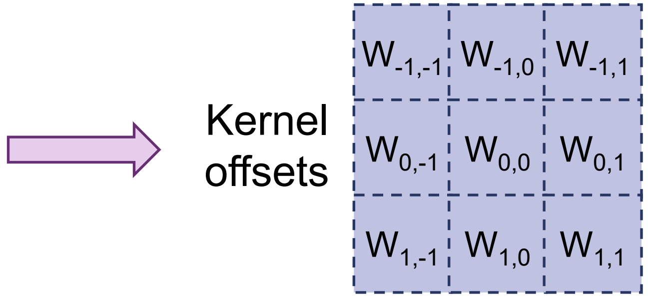
Efficient Mapping Unit: Voxel-based

- Voxel-based Mapping Operation (Kernel mapping): Output-Major Mapping
  - Our design: Calculate the mapping of all offsets of the weights at once

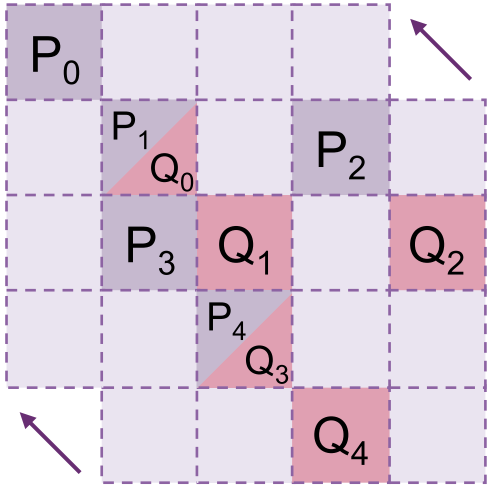
Weight: One at a time



Weight: All at a time

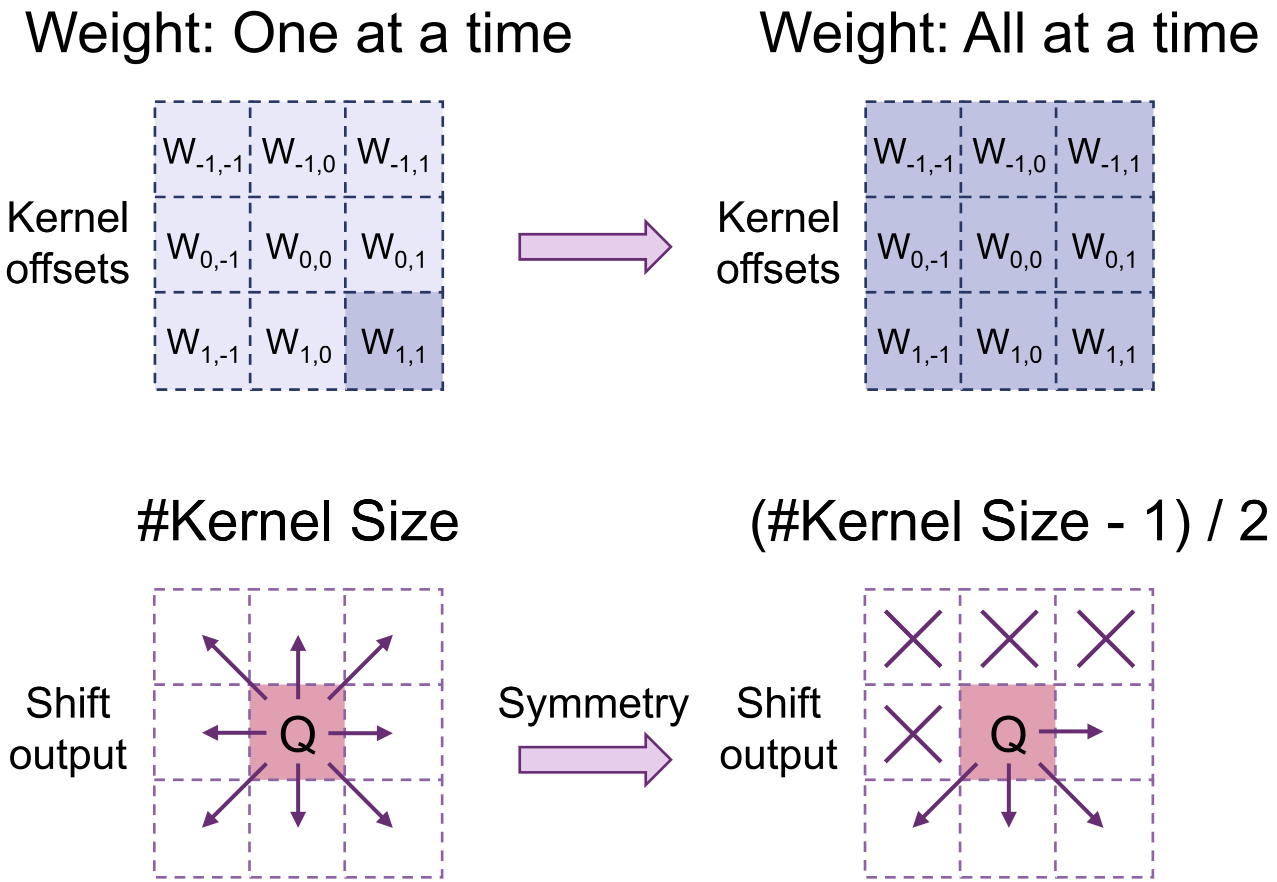


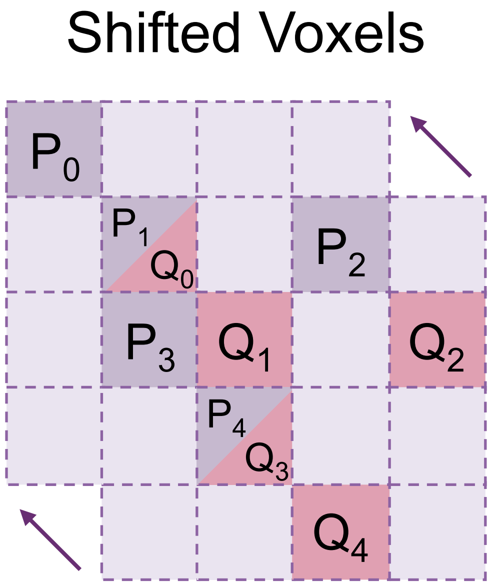
Shifted Voxels



Efficient Mapping Unit: Voxel-based

- Voxel-based Mapping Operation (Kernel mapping): Output-Major Mapping
  - Our design: Calculate the mapping of all offsets of the weights at once





Mapping

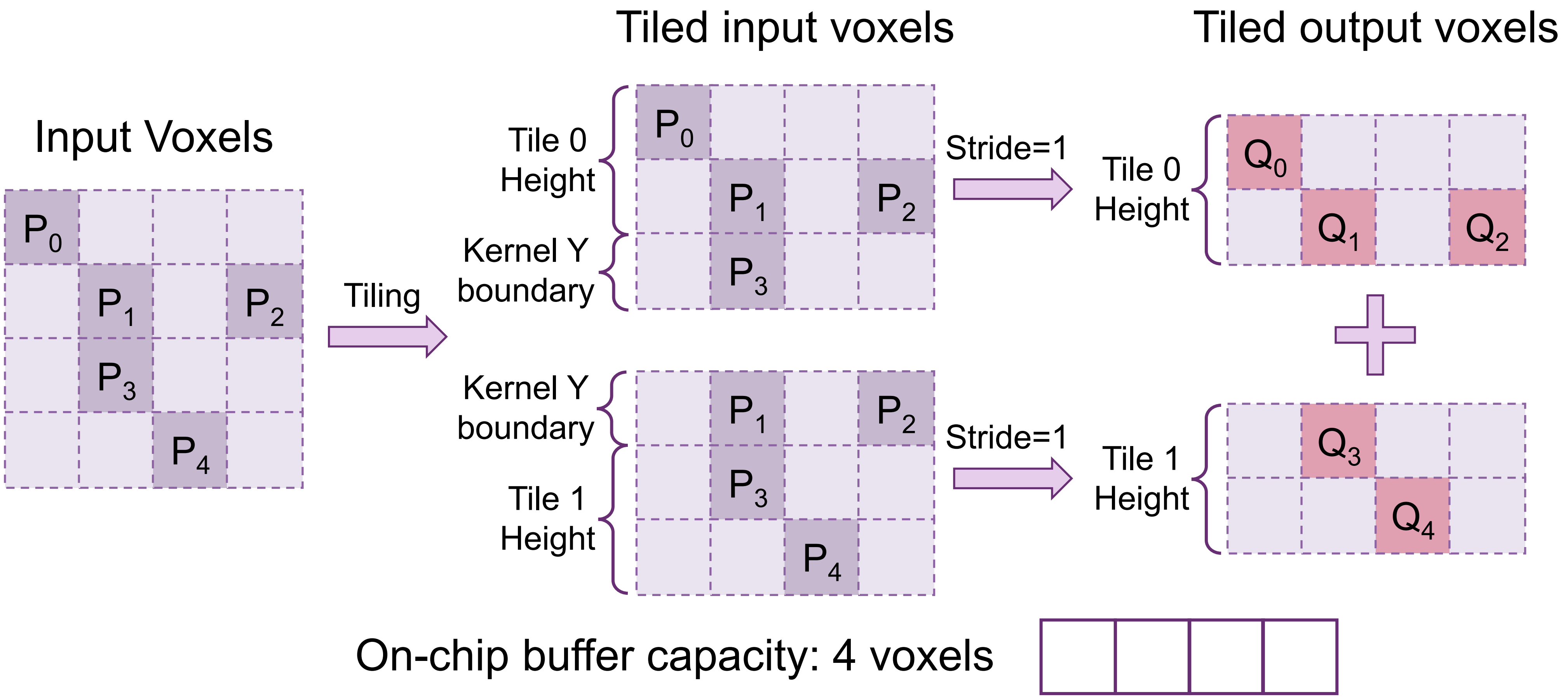
(In, Out, Wgt)
(P_i, Q_j, $W_{k,l}$)

Reverse

(In, Out, Wgt)
(P_j, Q_i, $W_{-k,-l}$)

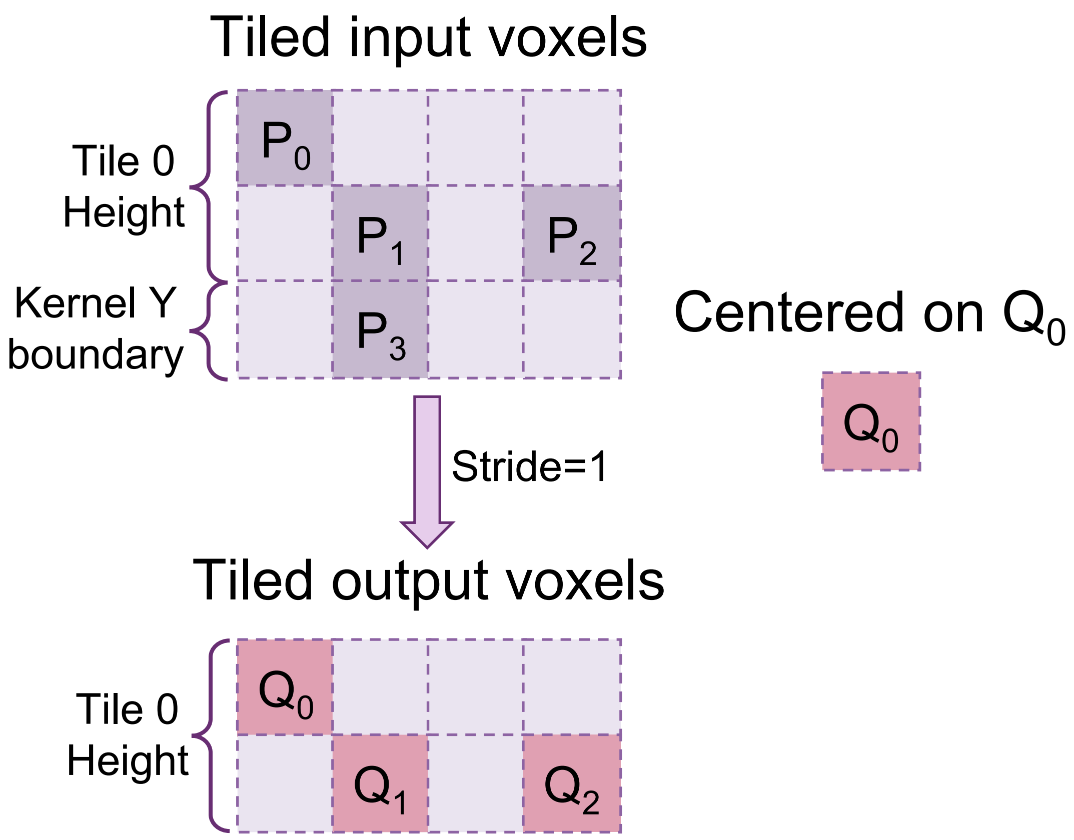
Efficient Mapping Unit: Voxel-based

- Voxel-based Mapping Operation (Kernel mapping): Output-Major Mapping
  - Our design: Calculate the mapping of all offsets of the weights at once



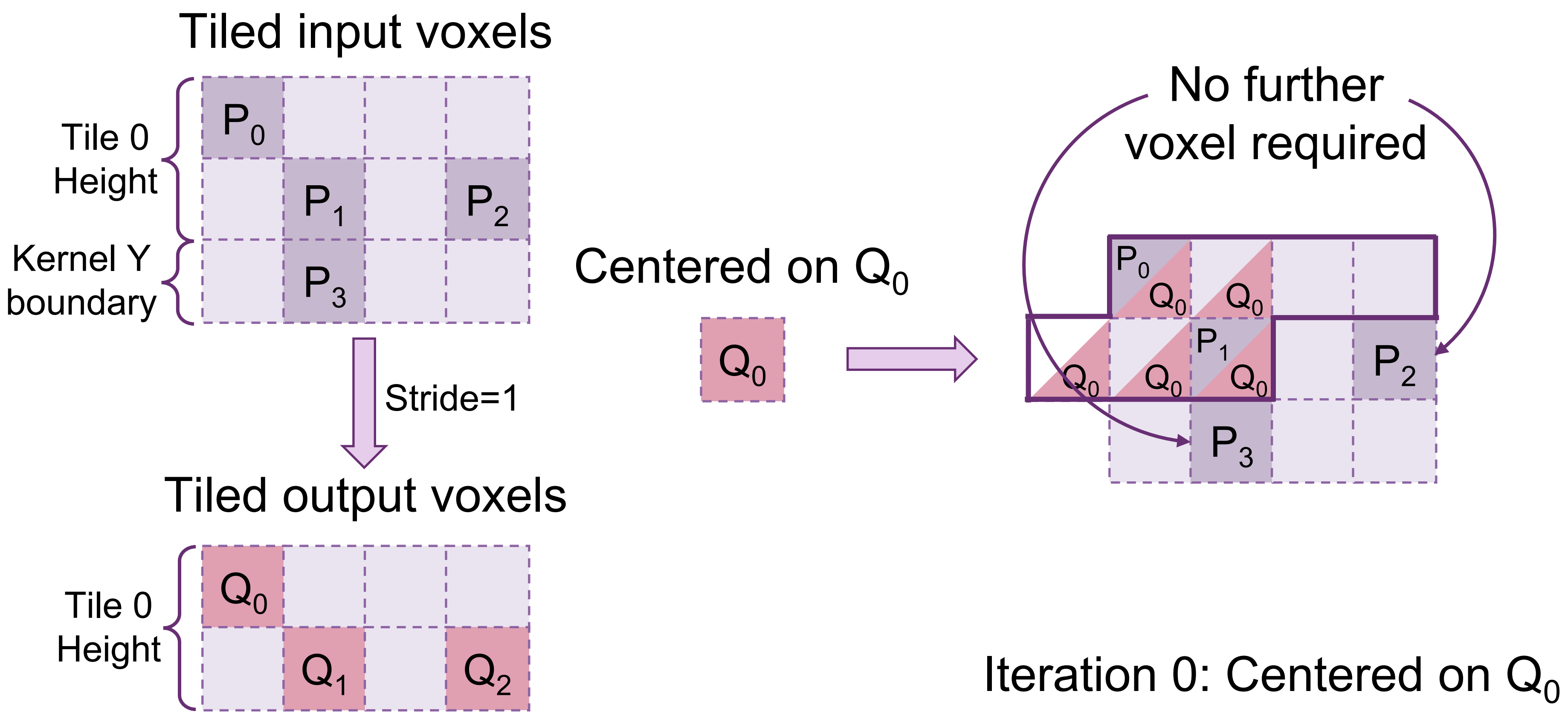
Efficient Mapping Unit: Voxel-based

- Voxel-based Mapping Operation (Kernel mapping): Output-Major Mapping
  - Our design: Calculate the mapping of all offsets of the weights at once



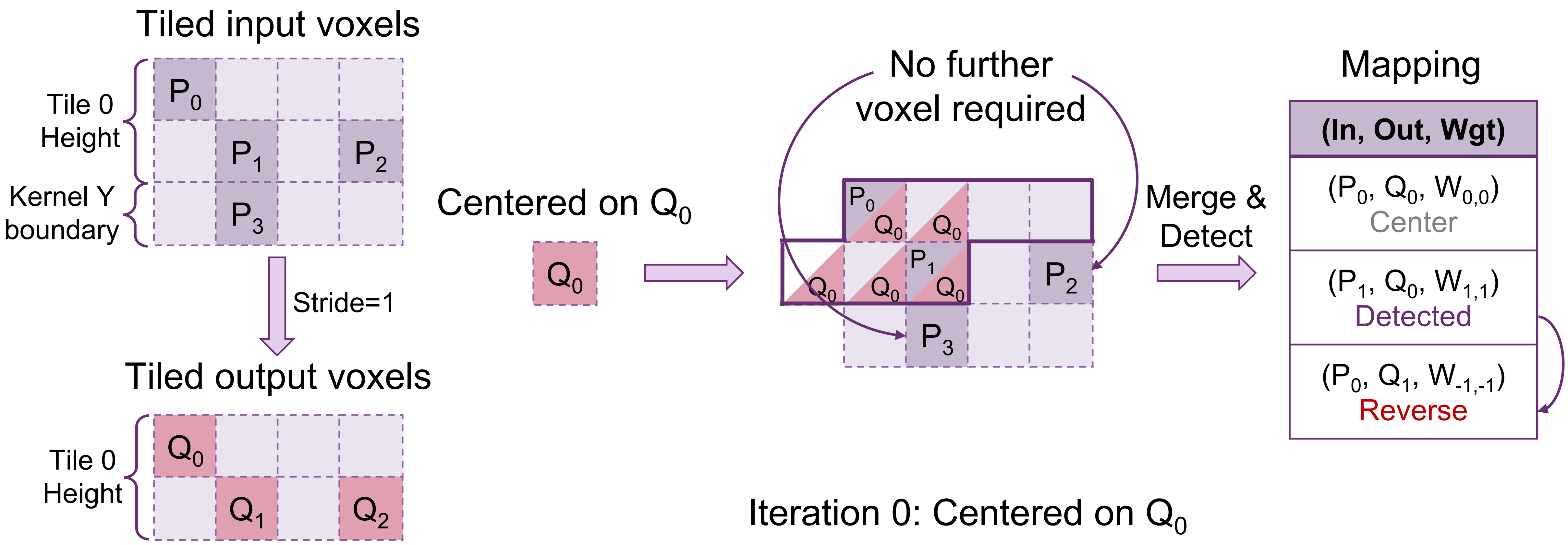
Efficient Mapping Unit: Voxel-based

- Voxel-based Mapping Operation (Kernel mapping): Output-Major Mapping
  - Our design: Calculate the mapping of all offsets of the weights at once



Efficient Mapping Unit: Voxel-based

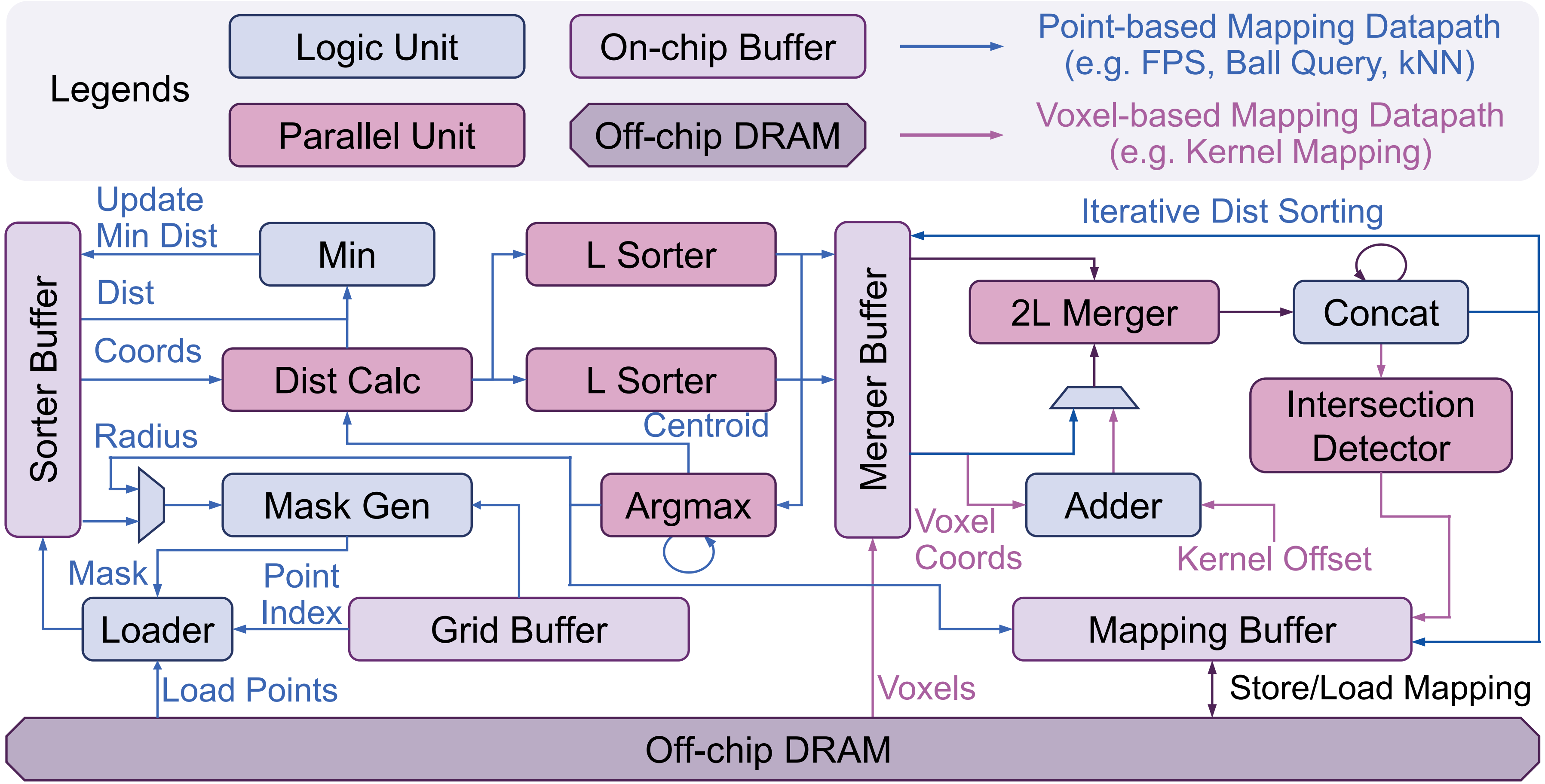
- Voxel-based Mapping Operation (Kernel mapping): Output-Major Mapping
  - Our design: Calculate the mapping of all offsets of the weights at once



Iteration 0: Centered on Q_0

Efficient Mapping Unit: Hardware

- Unified hardware architecture supporting distance filtering (point-based) and output-major mapping (voxel-based)

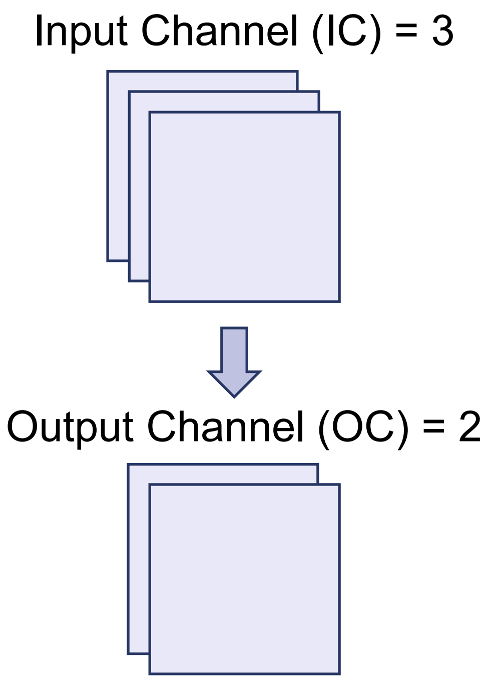


Contents

Elastic Computing Unit

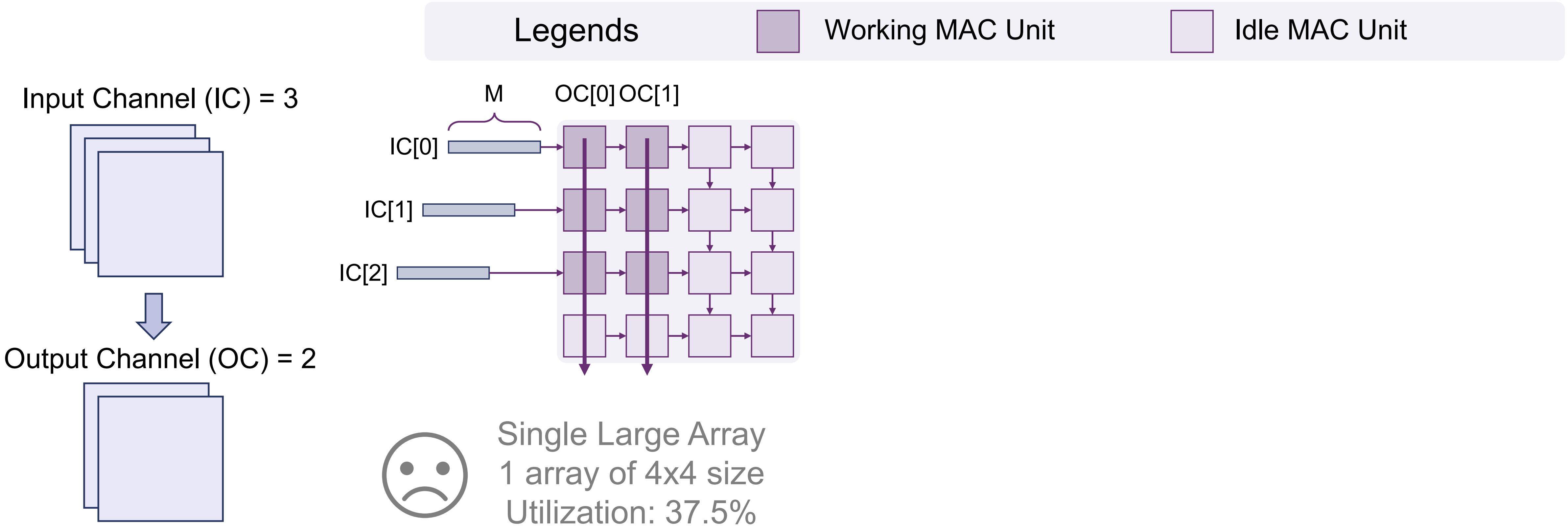
- Elastic Computing Unit
  - Two-level sub-array structure supporting dynamic splitting
  - Tiling features based on on-chip buffer size

Legends | Working MAC Unit | Idle MAC Unit



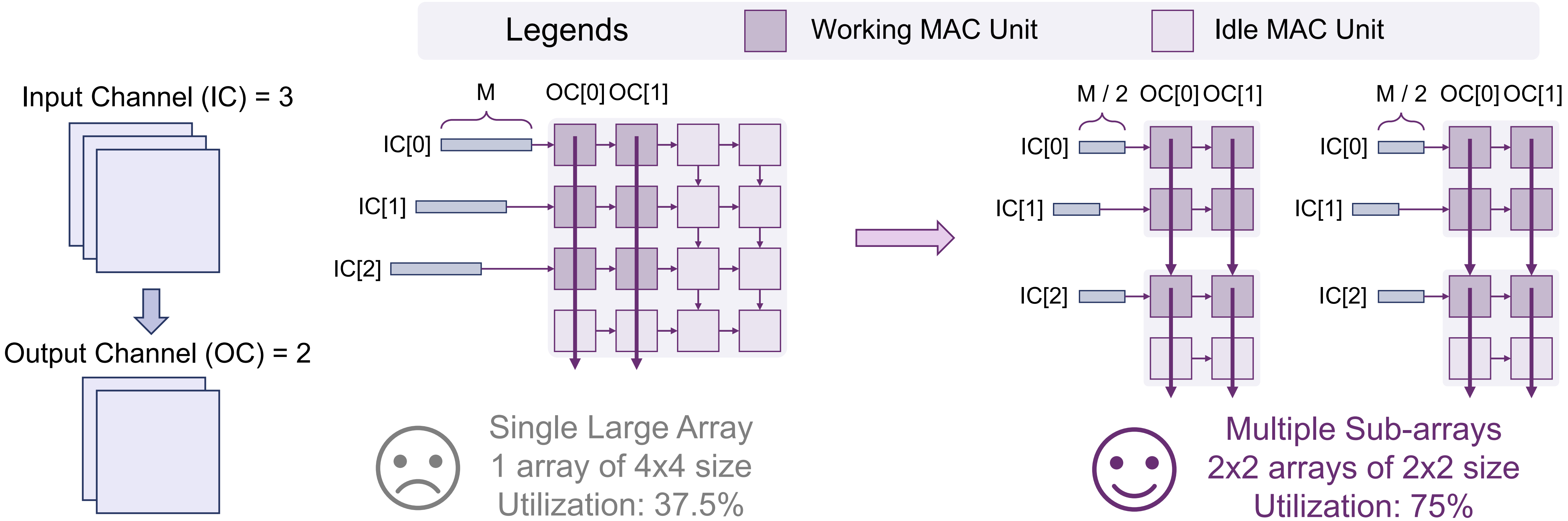
Elastic Computing Unit

- Elastic Computing Unit
  - Two-level sub-array structure supporting dynamic splitting
  - Tiling features based on on-chip buffer size



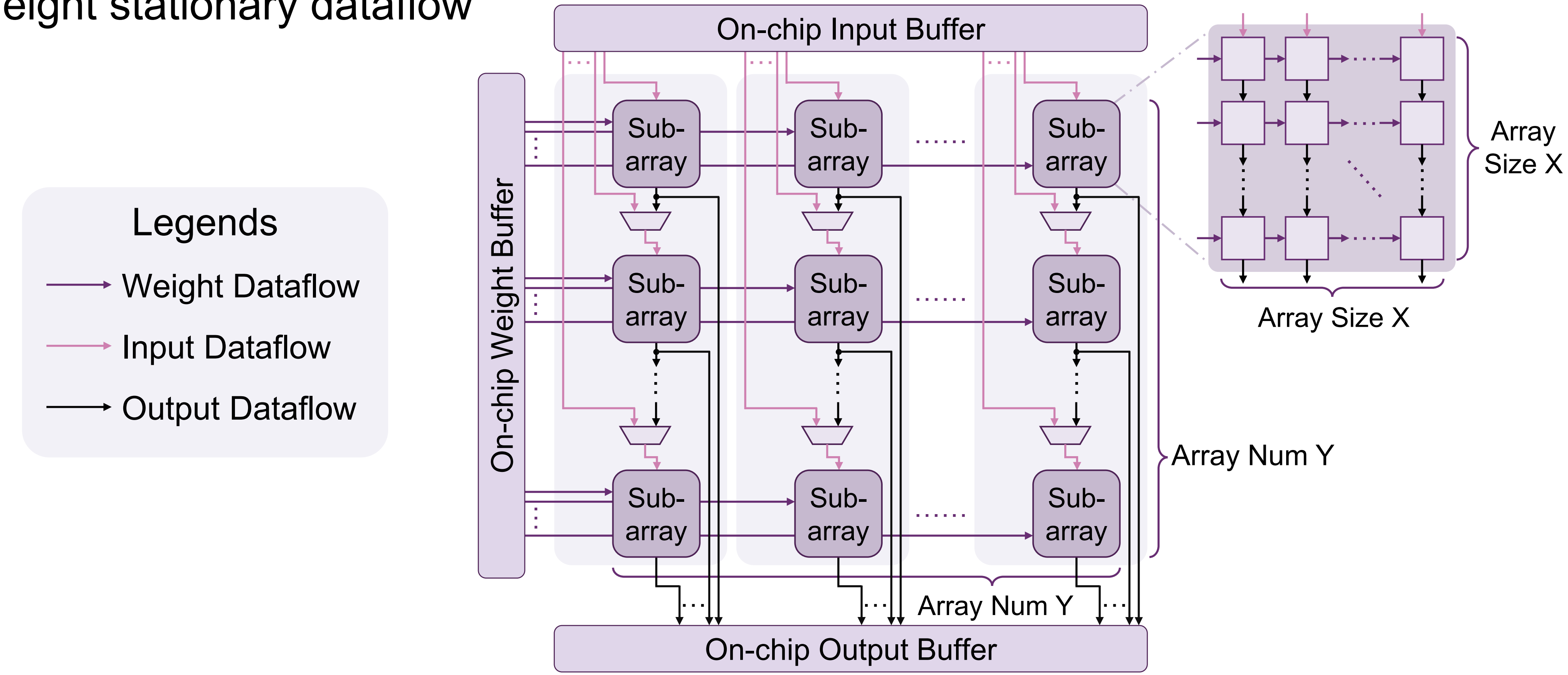
Elastic Computing Unit

- Elastic Computing Unit
  - Two-level sub-array structure supporting dynamic splitting
  - Tiling features based on on-chip buffer size



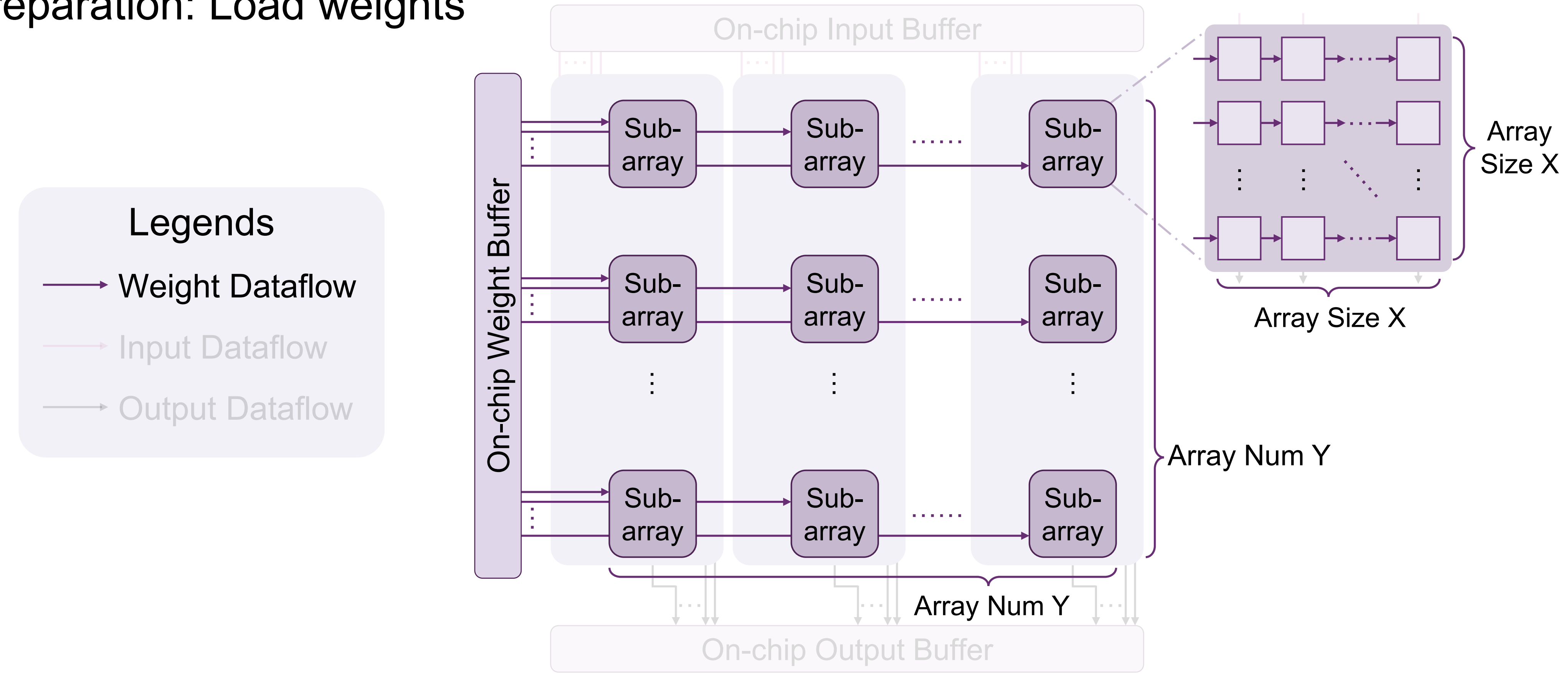
Elastic Computing Unit

- Elastic Computing Unit
  - Two-level sub-array structure supporting dynamic splitting
  - Weight stationary dataflow



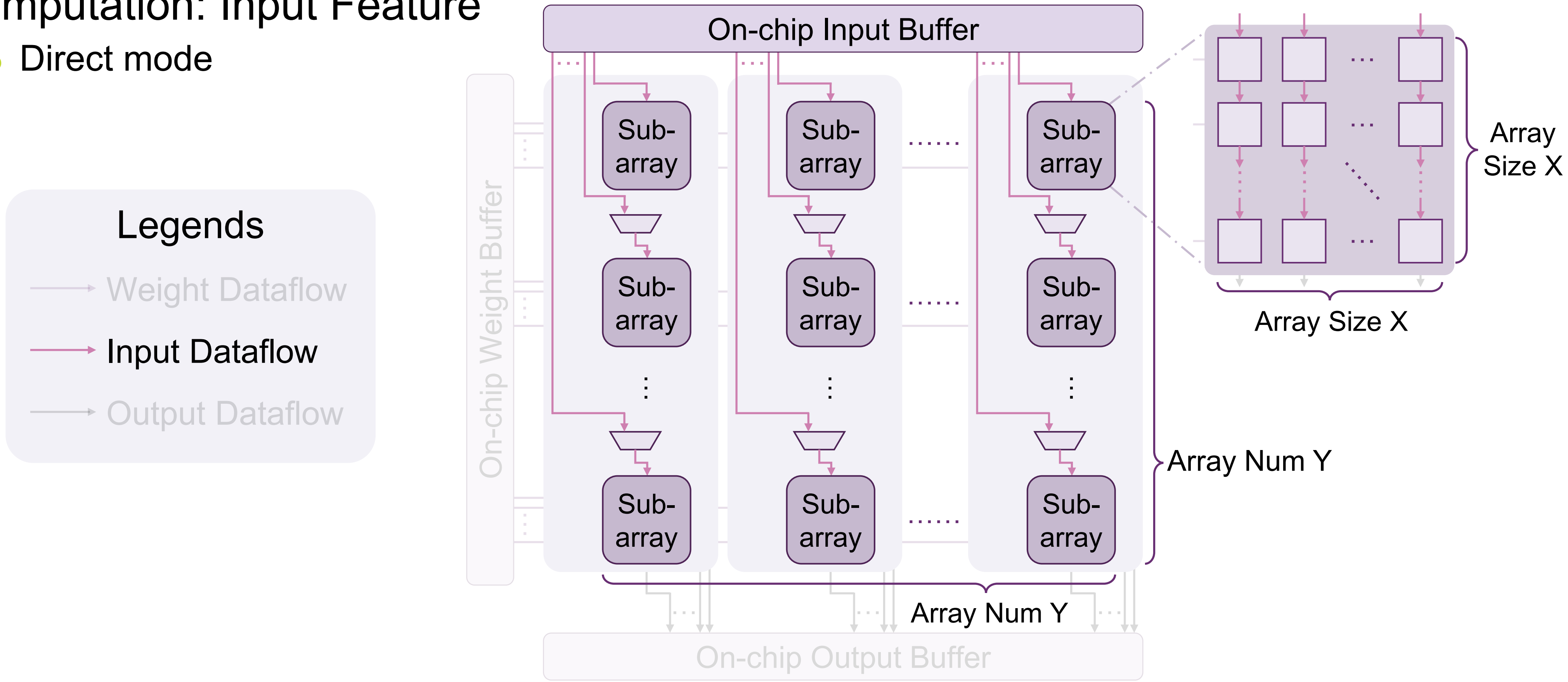
Elastic Computing Unit

- Elastic Computing Unit
  - Two-level sub-array structure supporting dynamic splitting
  - Preparation: Load weights



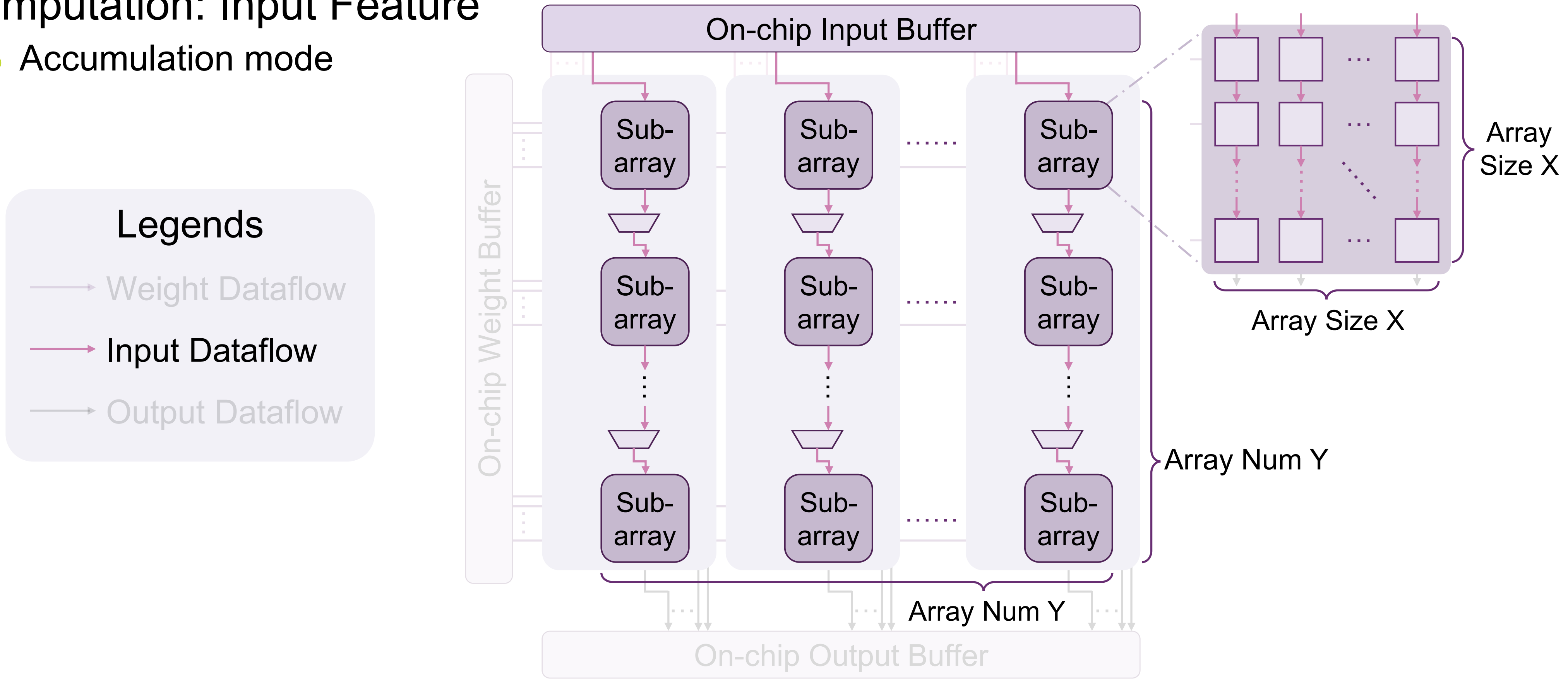
Elastic Computing Unit

- Elastic Computing Unit
  - Two-level sub-array structure supporting dynamic splitting
  - Computation: Input Feature
    - Direct mode



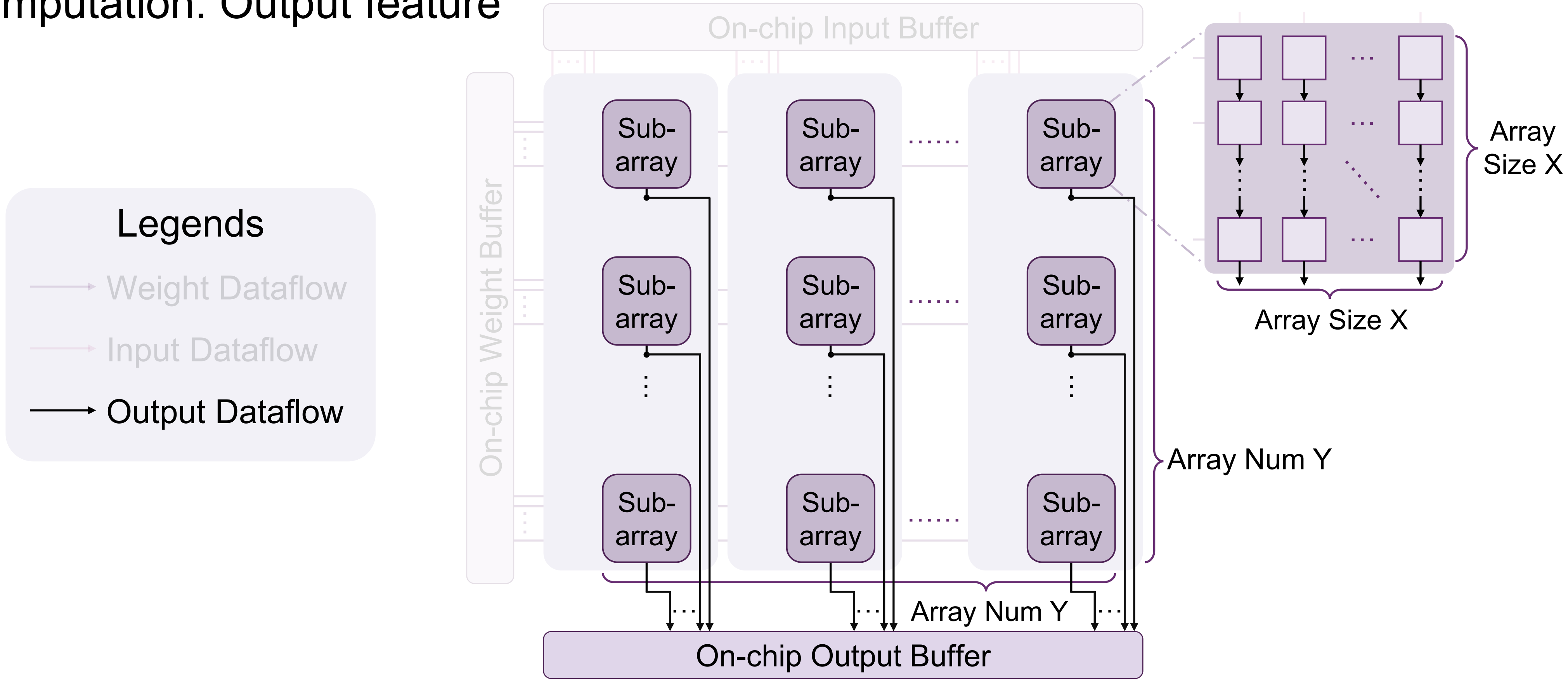
Elastic Computing Unit

- Elastic Computing Unit
  - Two-level sub-array structure supporting dynamic splitting
  - Computation: Input Feature
    - Accumulation mode



Elastic Computing Unit

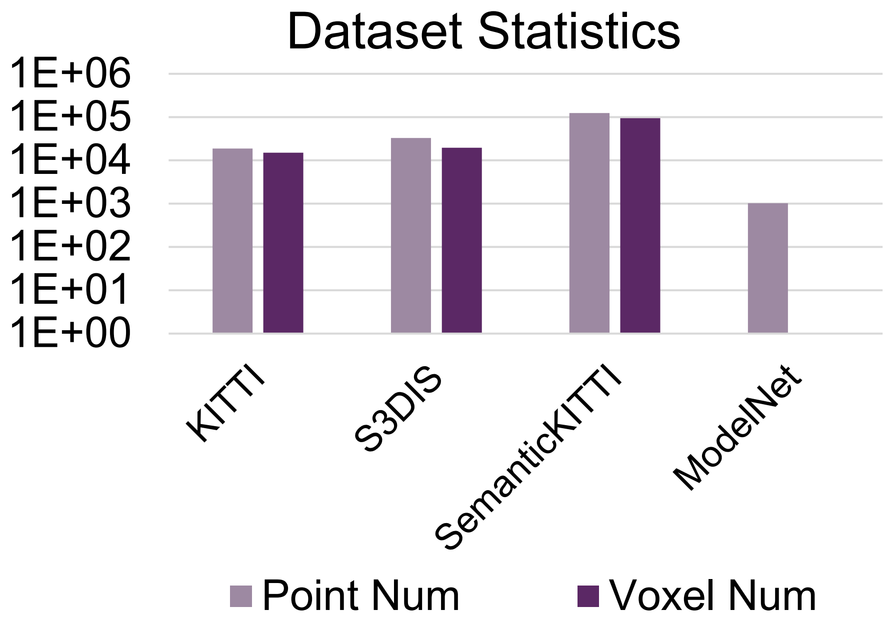
- Elastic Computing Unit
  - Two-level sub-array structure supporting dynamic splitting
  - Computation: Output feature



Contents

Evaluations

- Evaluation Setup
  - Baseline: PointAcc
  - Performance: Simulator
  - Power: TSMC 65nm
  - Point Num: **1k~124k**
  - Voxel Num: **15k~94k**

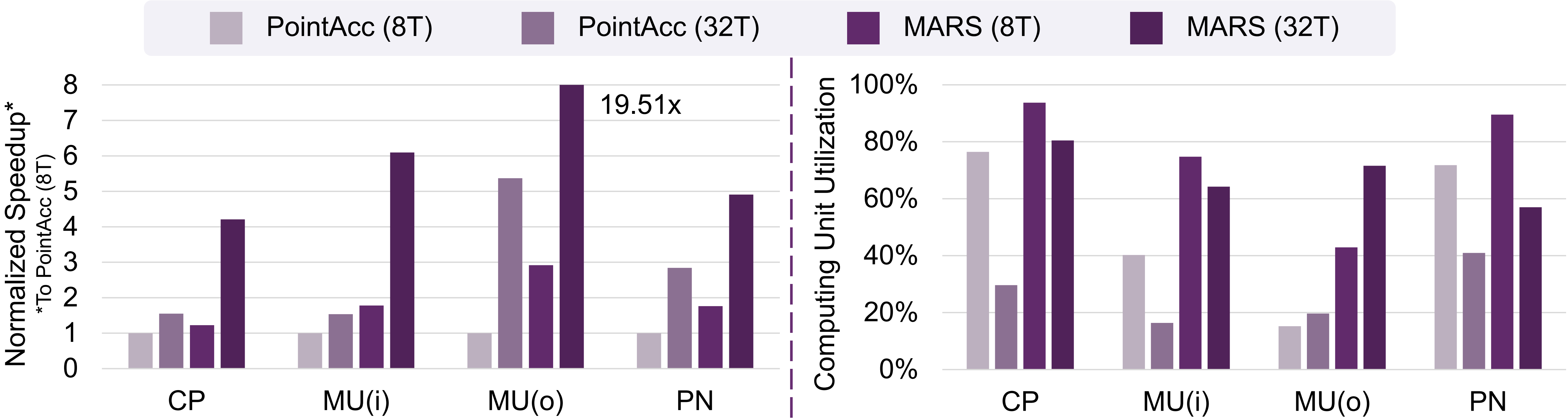


Application	Dataset	Model	Method	Notation
Classification	ModelNet40	PointNet++	Point	PN
Detection	KITTI	CenterPoint	Voxel	CP
Segmentation	S3DIS	MinkowskiUNet	Voxel	MU(i)
	SemanticKITTI	MinkowskiUNet	Voxel	MU(o)

Hardware Configs	PointAcc (8T)	MARS (8T)	PointAcc (32T)	MARS (32T)
Array Size	64x64	16x16	128x128	16x16
Array Num	1x1	4x4	1x1	8x8
SRAM (KB)	776	776	3107	3107
DRAM	HBM2	HBM2	HBM2	HBM2
Bandwidth	256GB/s	256GB/s	256GB/s	256GB/s
Peak Perf.	8TOPS	8TOPS	32TOPS	32TOPS

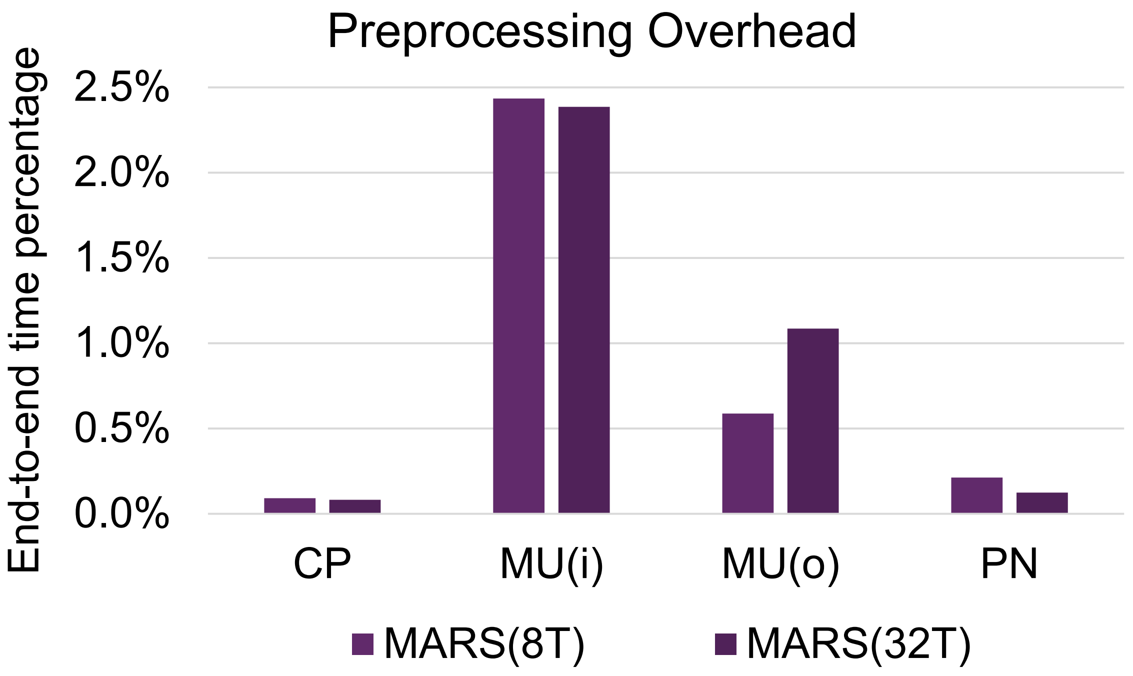
Evaluations

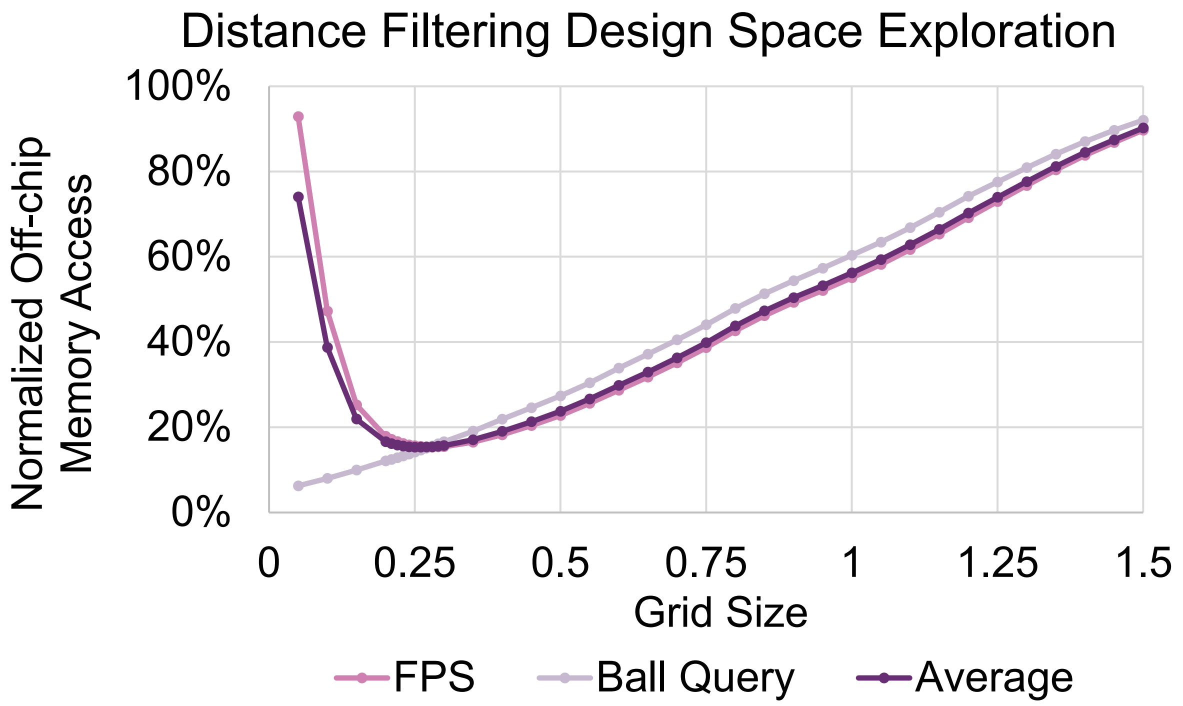
- End-to-end acceleration
  - Point-based network (aligned computing capacity): up to **1.76x**
  - Voxel-base network (aligned computing capacity): up to **3.97x**
  - Average computing unit utilization (32TOPS): **26.67% → 68.32%**
  - Worst computing unit utilization (32TOPS): **16.37% → 57.01%**



Evaluations

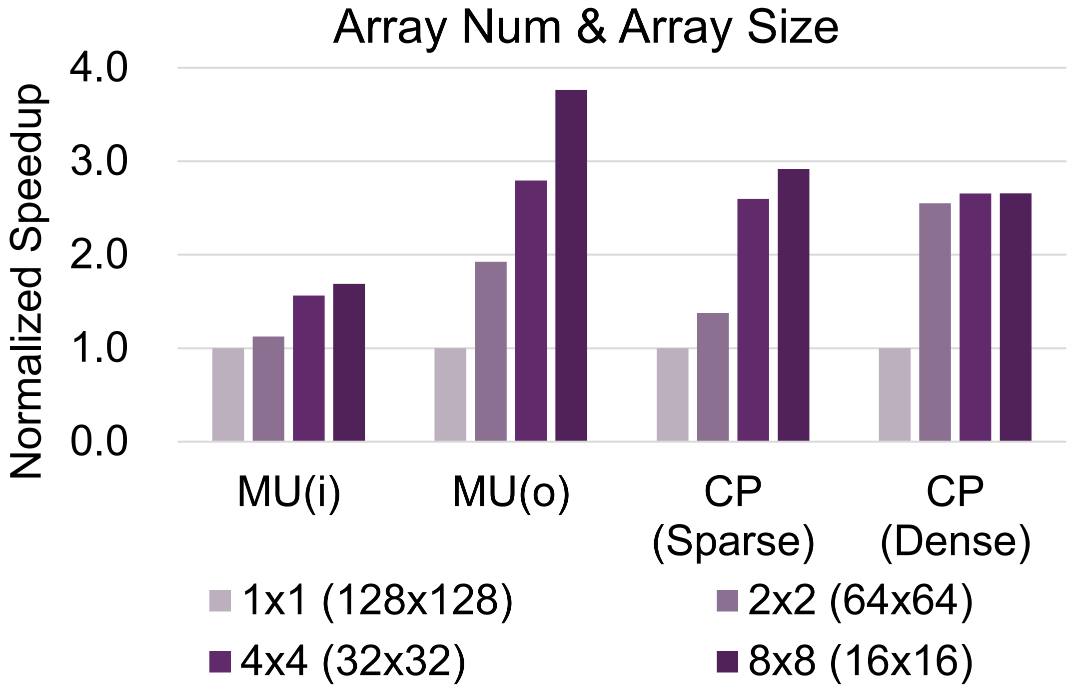
- Preprocessing Overhead
  - Preprocessing: Sorting input points / voxels
  - Overhead: **<2.43%**
  - Distance Filtering Grid Size: **U-shaped** curve
    - Memory access: **-84.68%**

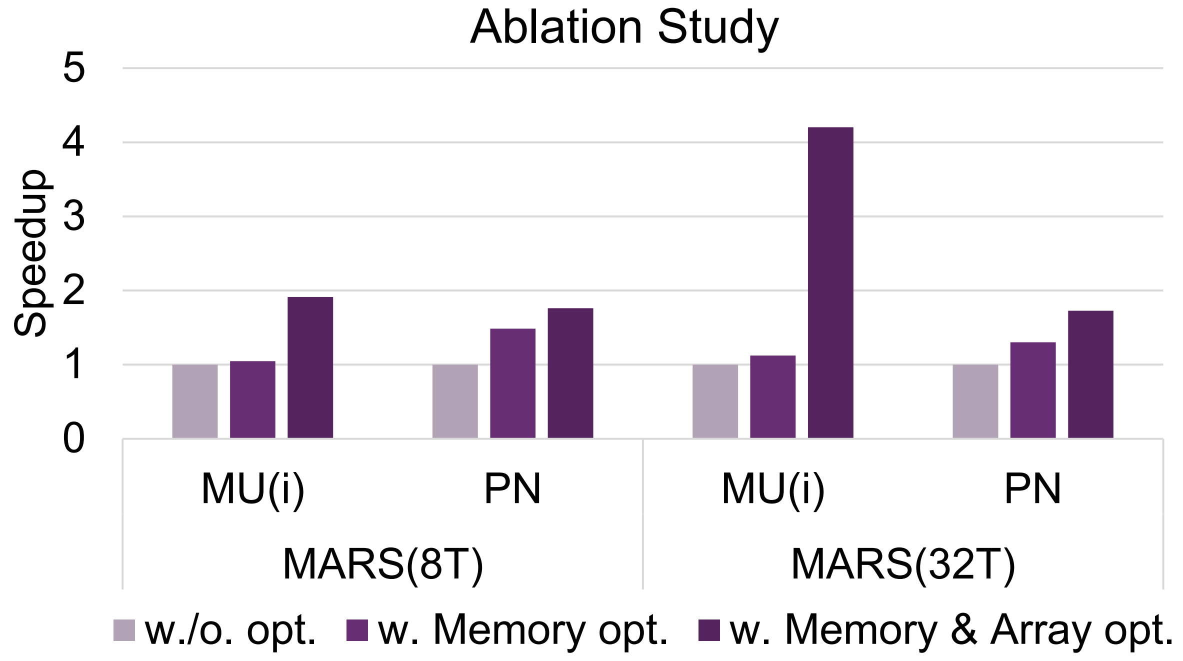




Evaluations

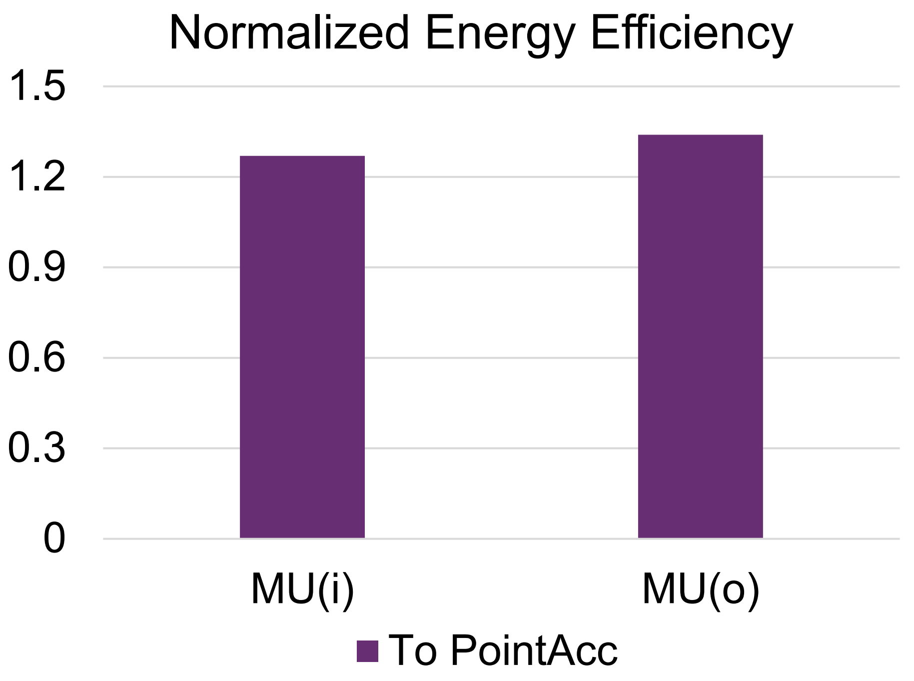
- Ablation Study
  - Utilization and hardware overhead increases with the number of arrays
  - Ablation Study:
    - Point-based: **Memory access** optimization is more important
    - Voxel-based: **Utilization** improvement is more important

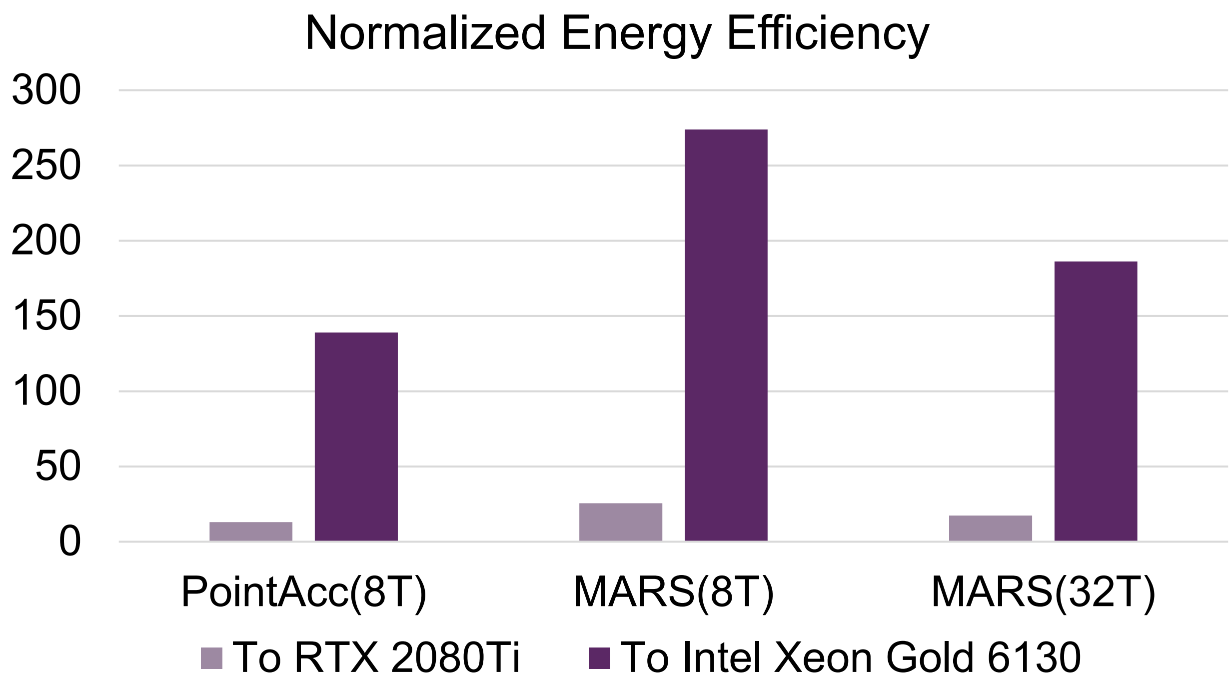




Evaluations

- Energy Efficiency
  - To PointAcc: **~1.30x** Under TSMC 65nm
  - To GPU & CPU: Converted from PointAcc paper:
    - MARS(8T) to GPU: **25.62x**, MARS(8T) to CPU: **273.89x**
    - MARS(32T) to GPU: **17.42x**, MARS(32T) to CPU: **186.23x**





Thanks and Q&A

Xinhao Yang[1], Tianyu Fu[1], Guohao Dai[2], Shulin Zeng[1],
Kai Zhong[1], Ke Hong[1] and Yu Wang[1]

[1]Dept. of EE, BNRist, Tsinghua University, [2]Shanghai Jiao Tong University

E-mail: yxh21@mails.tsinghua.edu.cn, daiguohao@sjtu.edu.cn, yu-wang@tsinghua.edu.cn