



Exploiting Voxel Data Structural Properties for Efficient Sparse Convolution in Point Cloud Networks

[Dionysios Adamopoulos](#),

Anastasia Poulopoulou, Georgios Goumas, Christina Giannoula

MLSys 2026

Bellevue, USA, May 22nd, 2026



MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS



Executive Summary

Motivation: Sparse Convolution (SpC) is a *key operator in point cloud networks* widely used in autonomous driving, robotics, AR/VR...

Problem: SpC execution on GPUs has a mapping step and a feature computation step; prior works incur significant **pre-** and **post-processing overheads** in the mapping step of SpC

Opportunities: We identify that SpC operates on voxel data that have inherent structural *properties*, which existing works *do not exploit* them

Spira: The *first* SpC engine for point cloud networks on GPUs that

- ✓ *intelligently leverages* structural properties of voxel data
- ✓ *effectively mitigates* pre- and post-processing costs in the SpC mapping step

Key Results: Spira improves inference performance over existing state-of-the-art frameworks by **1.68×** averaged across **six** GPU architectures



Outline

Background

Prior Works

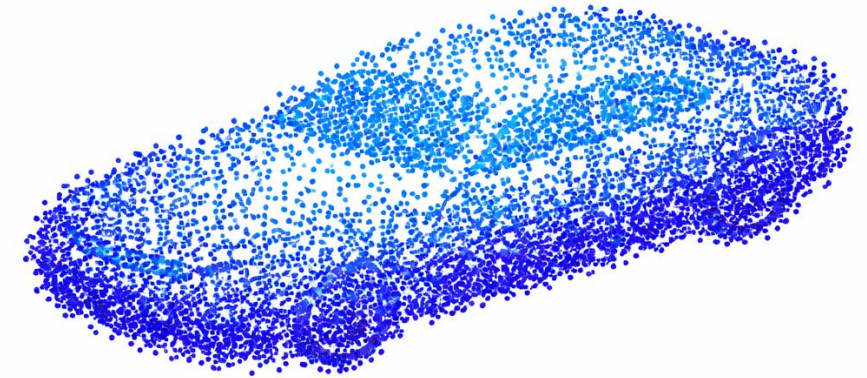
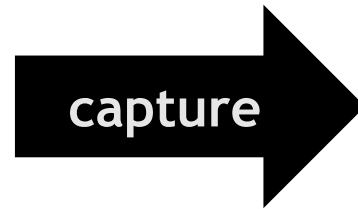
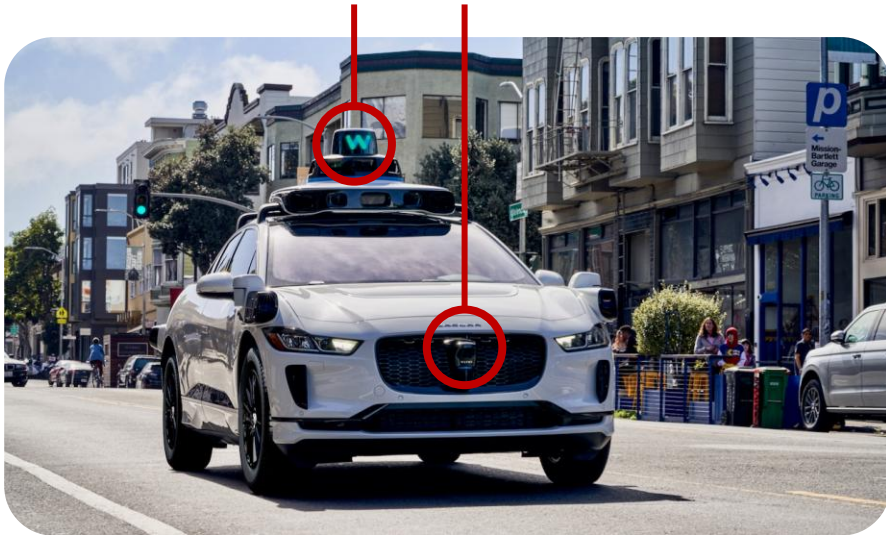
Voxel Data Properties

Spira Design

Evaluation

LiDAR Scanners Collect 3D Point Clouds

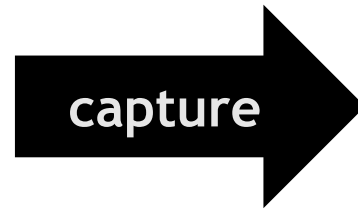
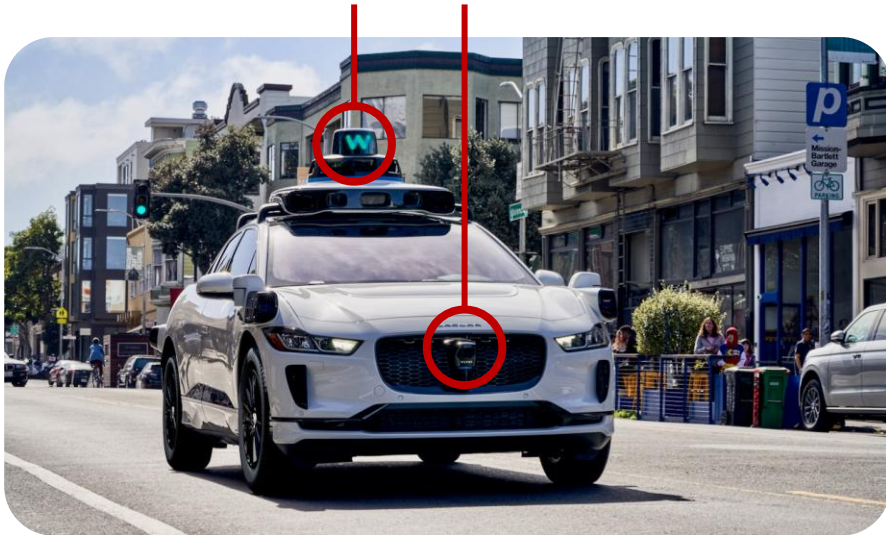
LiDAR Scanners



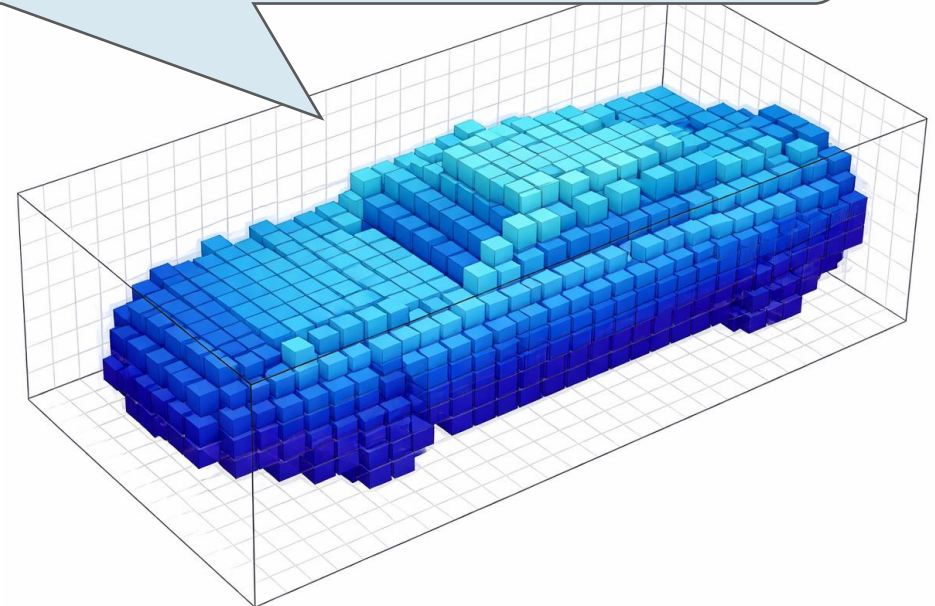
3D Point Clouds

LiDAR Scanners Collect 3D Point Clouds

LiDAR Scanners



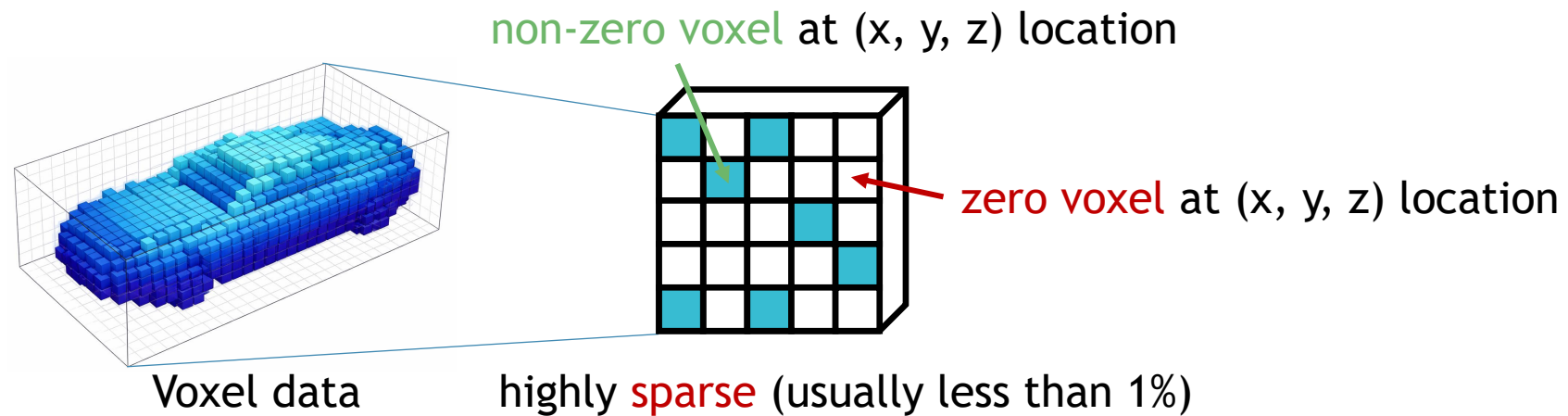
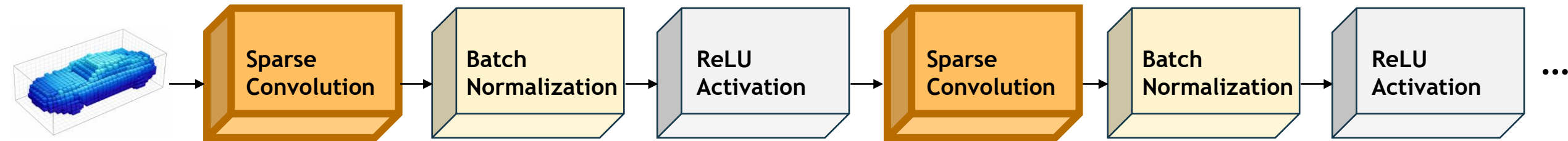
Transformed into 3D discrete cubes (voxels) that are grid-aligned



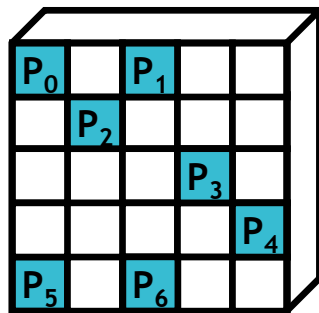
Voxelized Point Cloud
(abbr. **voxel data**)

Point Cloud Networks Rely on Sparse Convolution (SpC)

- Point cloud networks efficiently process voxel data using **Sparse Convolution**

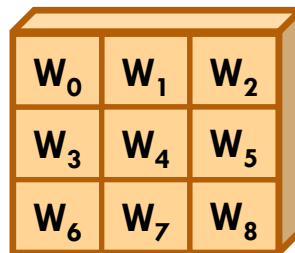


SpC Preserves the Sparsity in Output



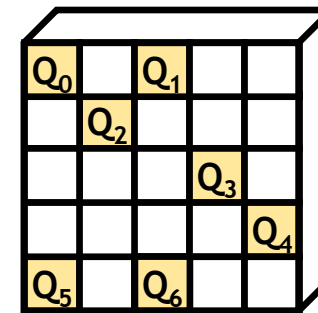
P: Input Non-Zero Coord.

×



W: Weights

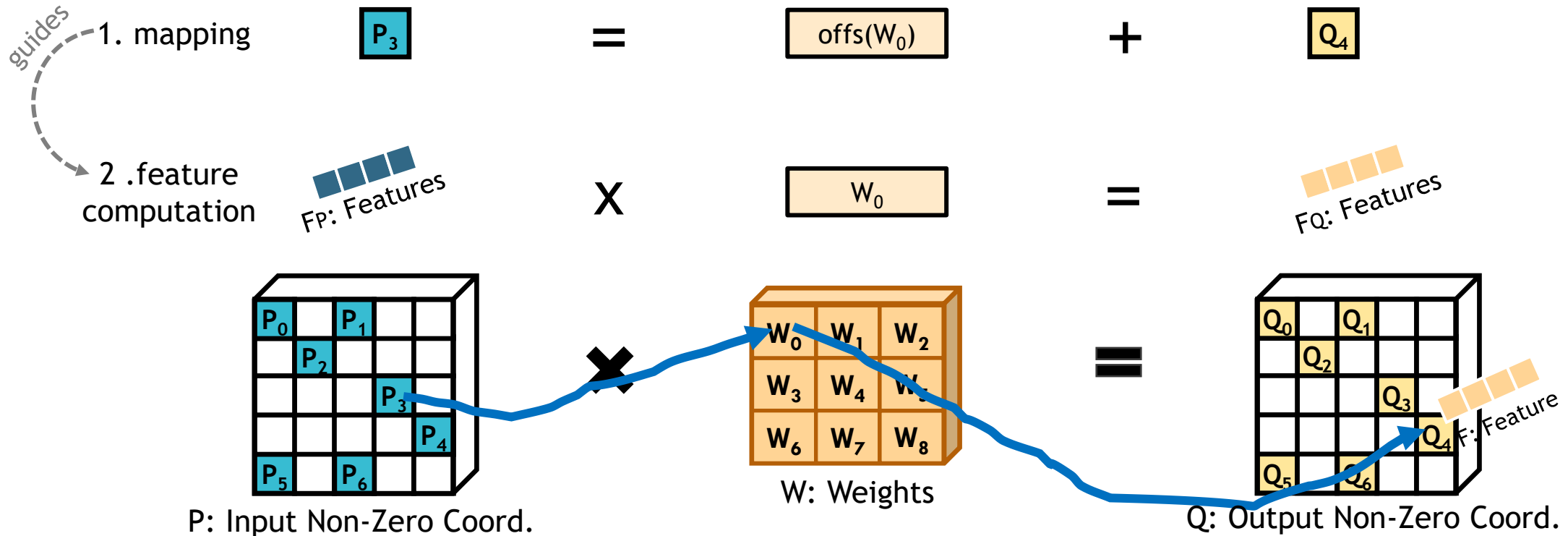
=



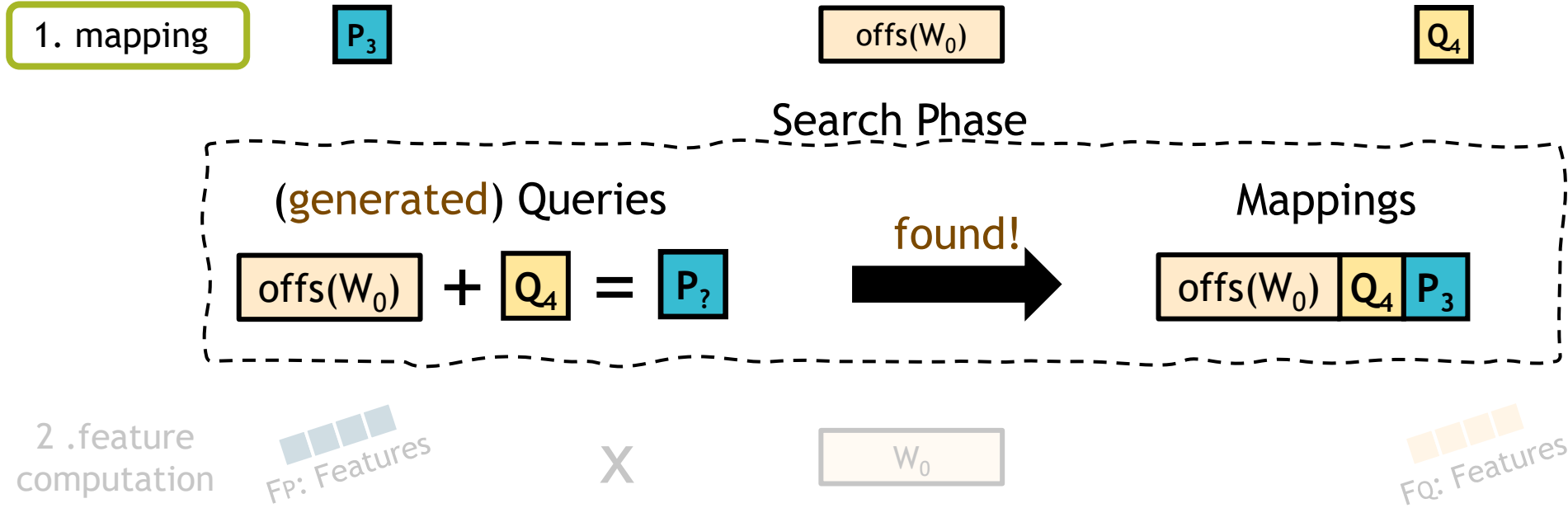
Q: Output Non-Zero Coord.

The 2 Steps of SpC

1. The Mapping Step
2. The Feature Computation Step

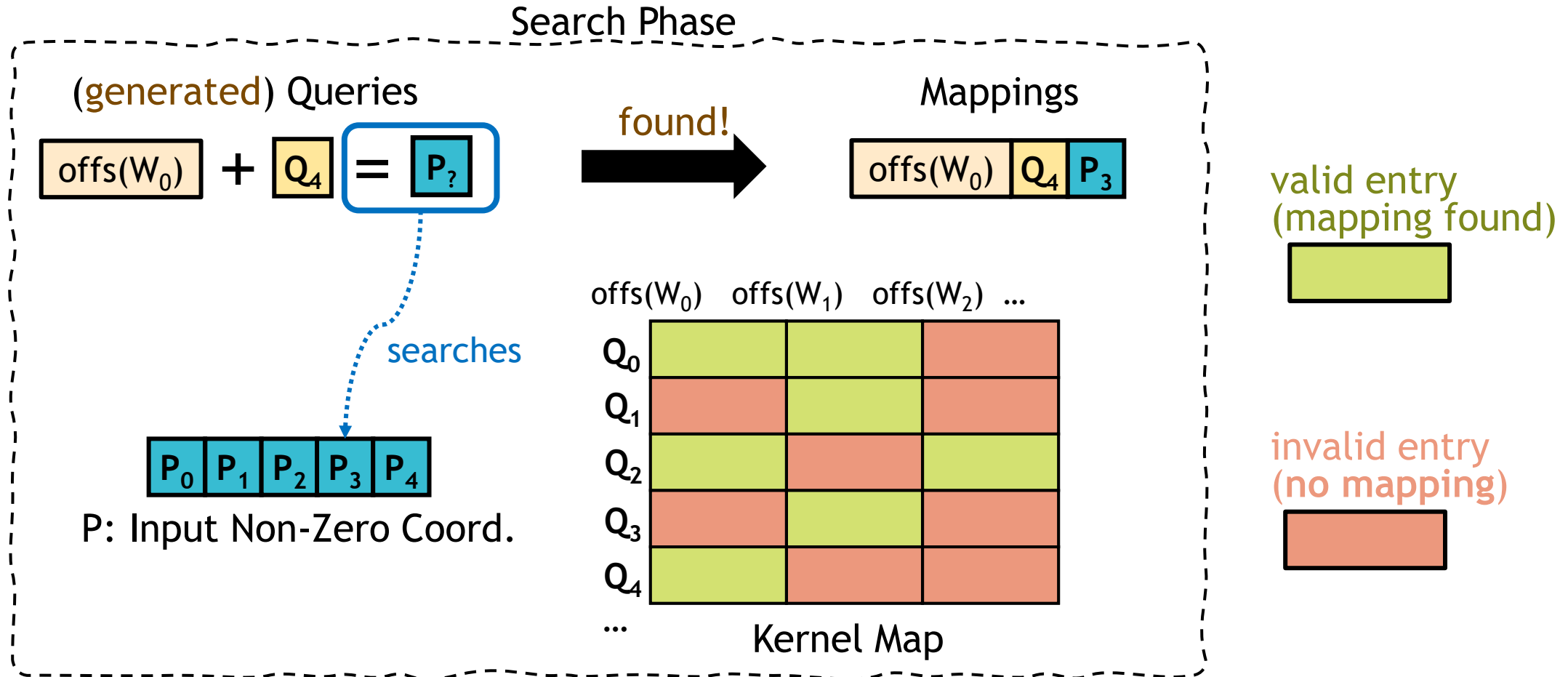


The 2 Steps of SpC

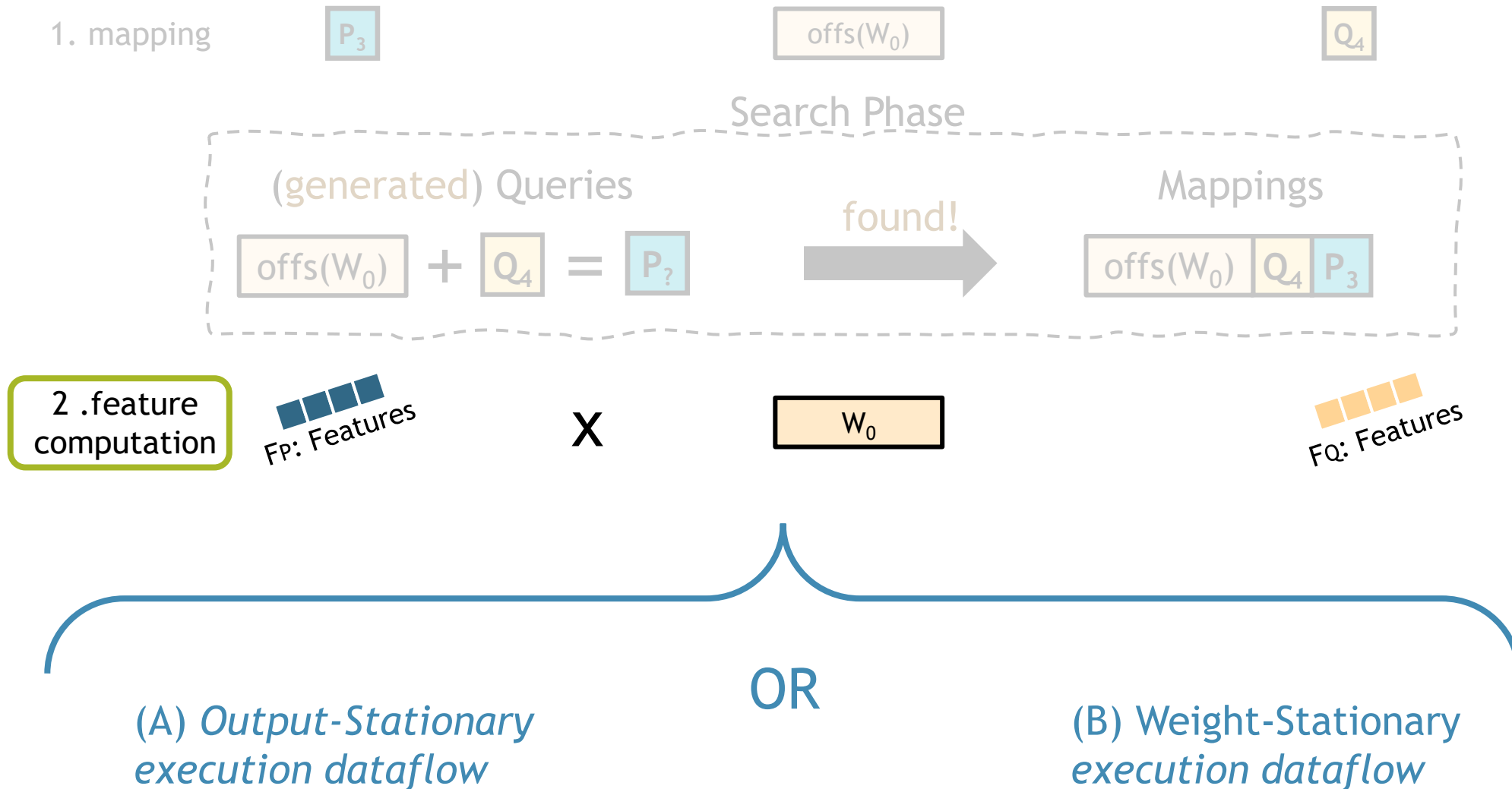


The Search Phase of the Mapping Step

- Creates the mappings between the input, weights and output non-zero voxels



The 2 Steps of SpC



Two Dataflows for the Feature Computation Step

- Two different execution dataflows about how to process the kernel map and produce the output features

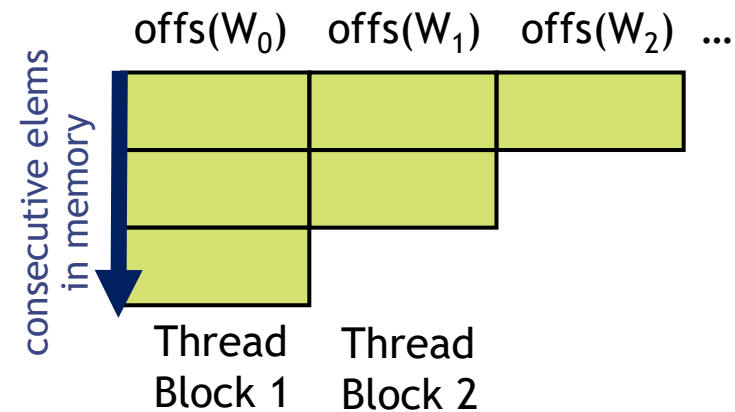
(A) *Output-Stationary execution dataflow*



Kernel Map
(**row-major**)

→ performs better when having **small** #invalid entries (**dense-friendly**)

(B) *Weight-Stationary execution dataflow*



Kernel Map
(**column-major + filtered**)

→ performs better when having **large** #invalid entries (**sparse-friendly**)

Outline

Background

Prior Works

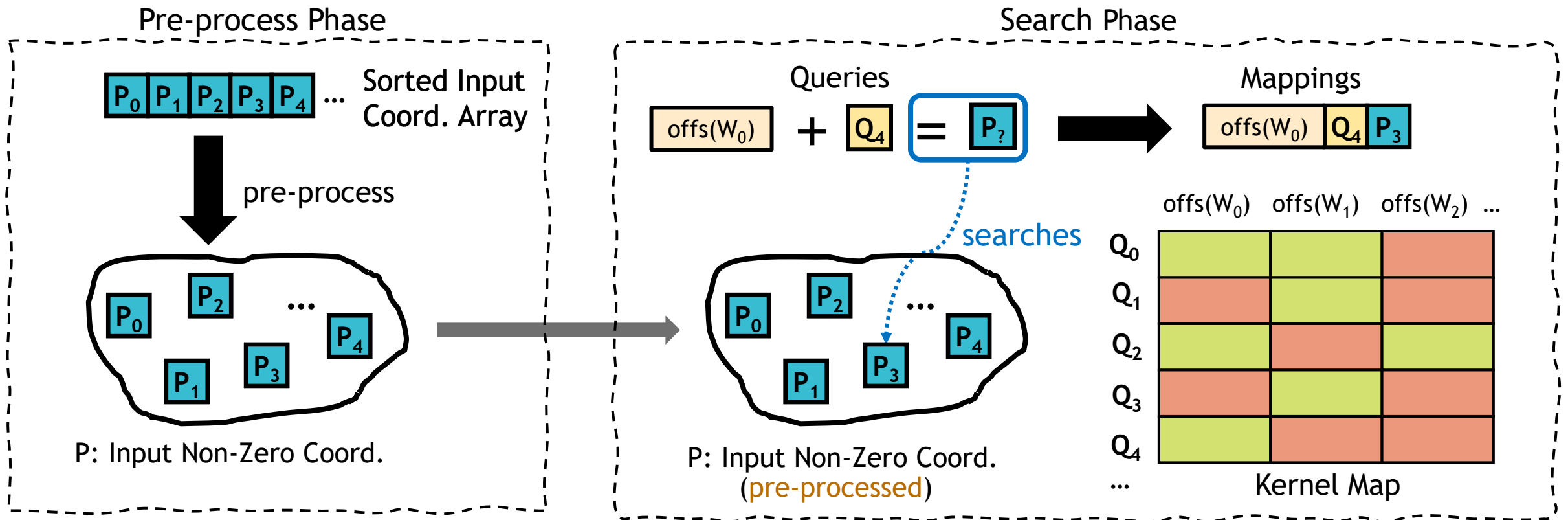
Voxel Data Properties

Spira Design

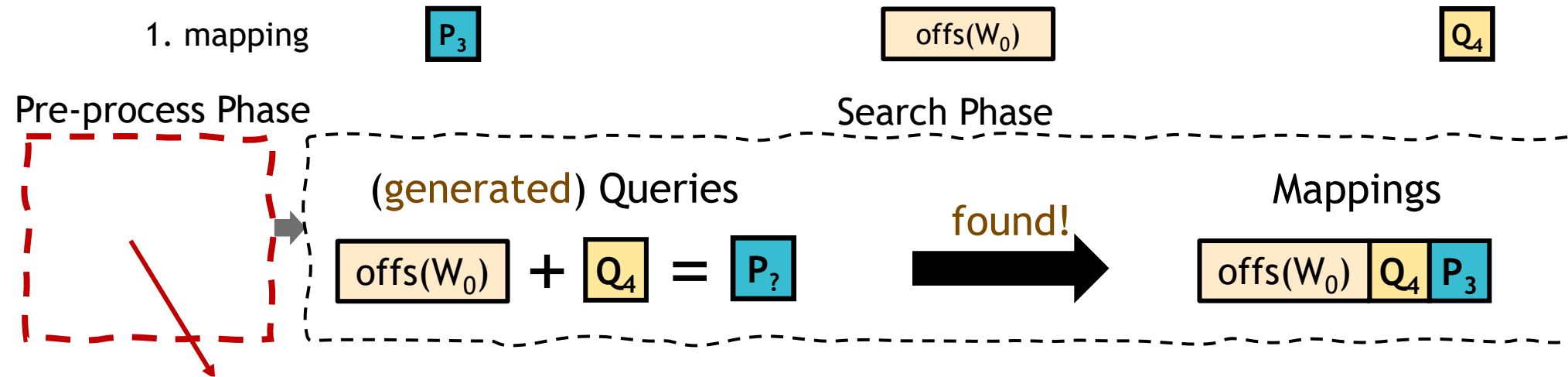
Evaluation

Prior Works Have a Pre-Process Phase in the Mapping Step

- Prior works introduce pre-processing the input coordinates before the search phase



Prior Works Incur Large Pre-Processing Costs



e.g., Minuet [Yang et al., EuroSys'24] → pre-processing time is almost equal to search time

FQ: Features

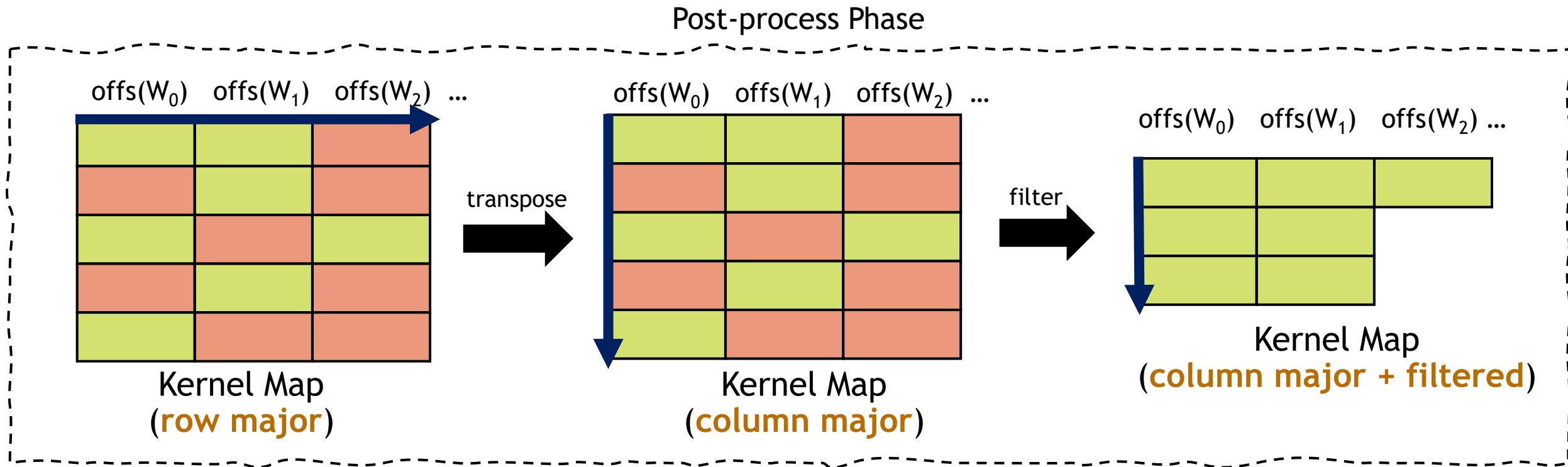
(A) Output-Stationary
execution dataflow

OR

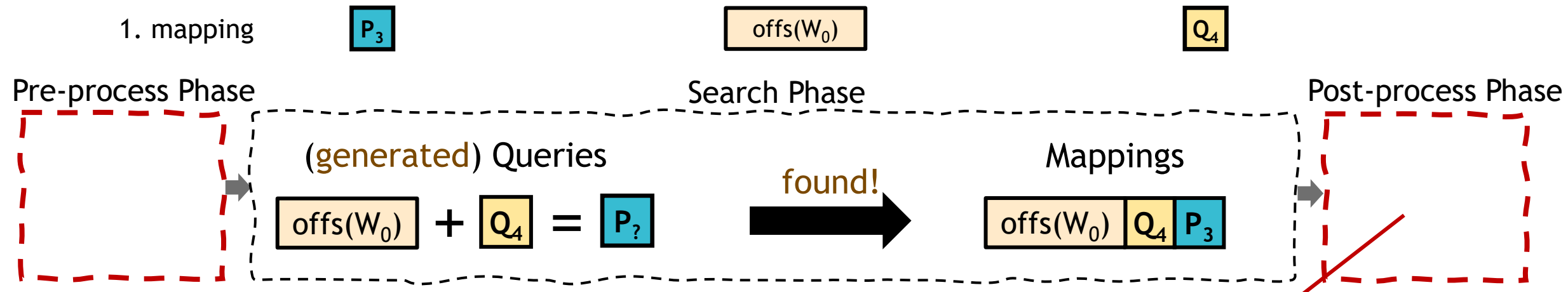
(B) Weight-Stationary
execution dataflow

Prior Works Have a Post-Process Phase in the Mapping Step

- Prior works need to perform post-processing on the kernel map to support **both** output- and weight-stationary dataflows



Prior Works Incur Large Post-Processing Costs



2. feature computation

Fp: Features

X

W_0

features

- Minuet [Yang et al., EuroSys'24] → **only** weight-stationary dataflow
- TorchSparse++ [Tang et al., MICRO'23] → post-processing costs up to **~40%** of a layer in the weight-stationary dataflow

(A) Output-Stationary execution dataflow

OR

(B) Weight-Stationary execution dataflow

Outline

Background

Prior Works

Voxel Data Properties

Spira Design

Evaluation

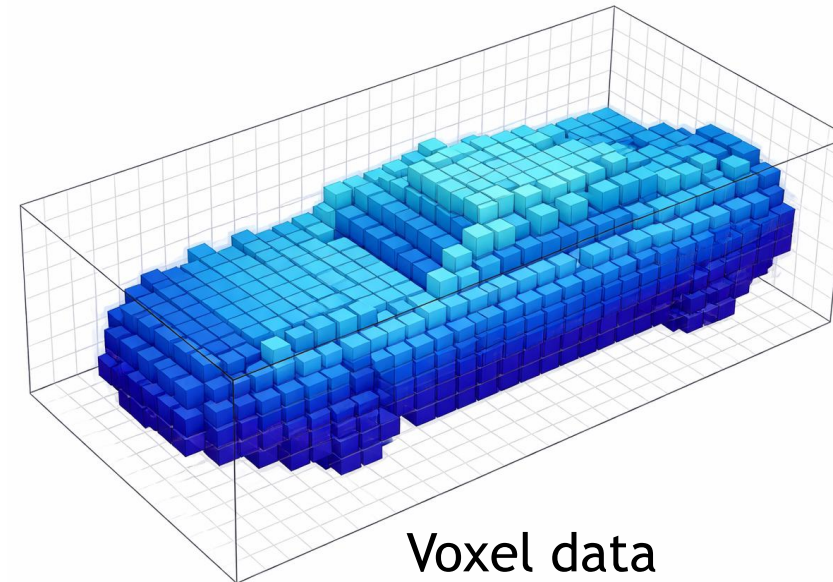
Identified Voxel Structural Properties

- We identify three properties regarding real-world voxel data:

Integer Property

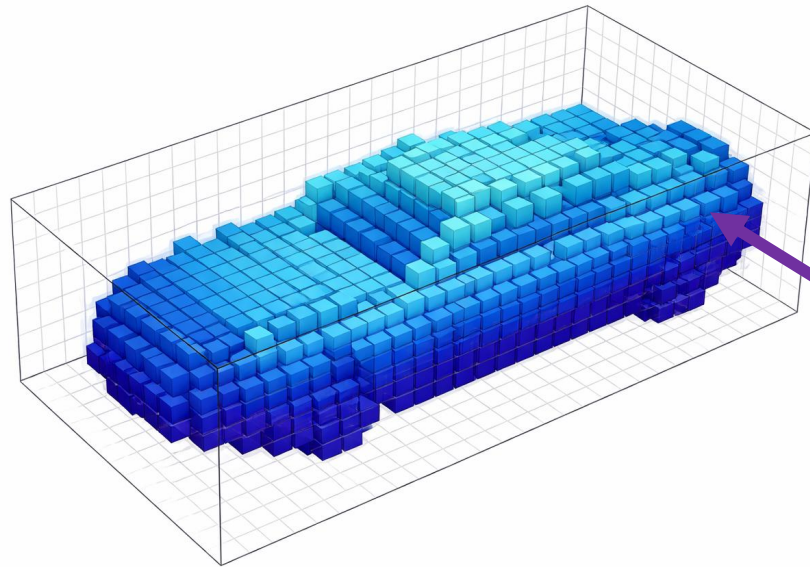
Bounded Property

Neighboring Property



Integer Property

- Voxel coordinates are *integer-valued*: a triplet of values represents a *discrete* location in a 3D grid

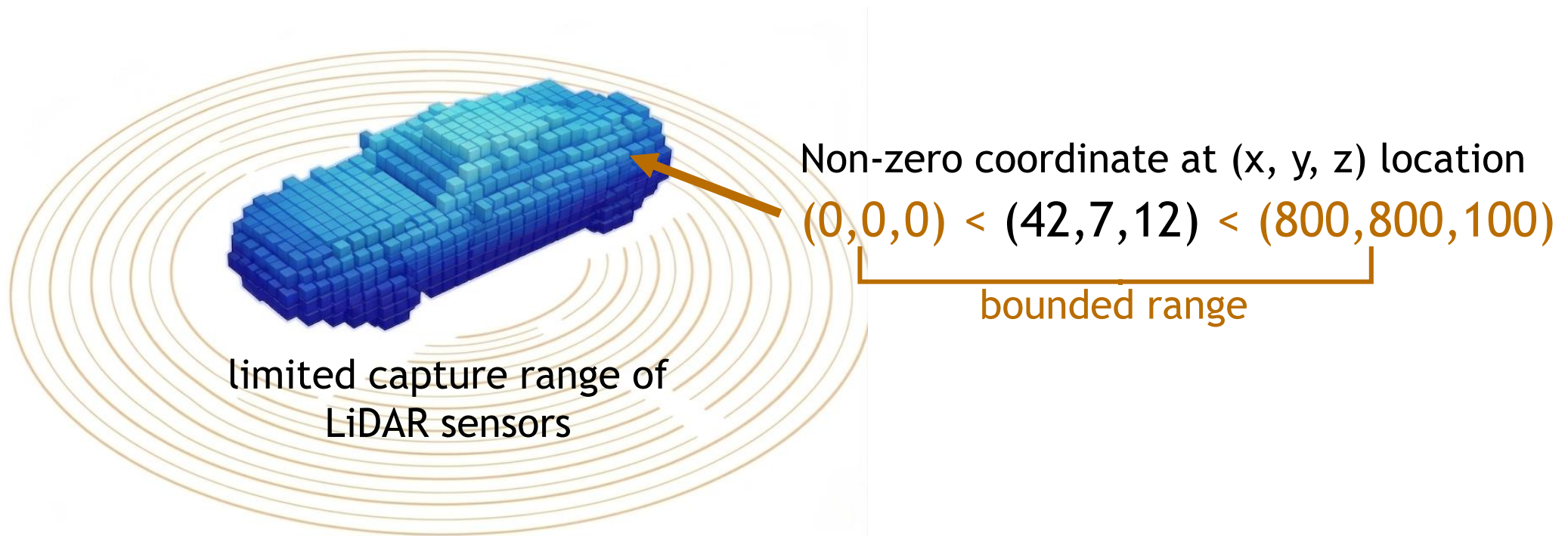


Non-zero coordinate at (x, y, z) location

(42,7,12)
integer values

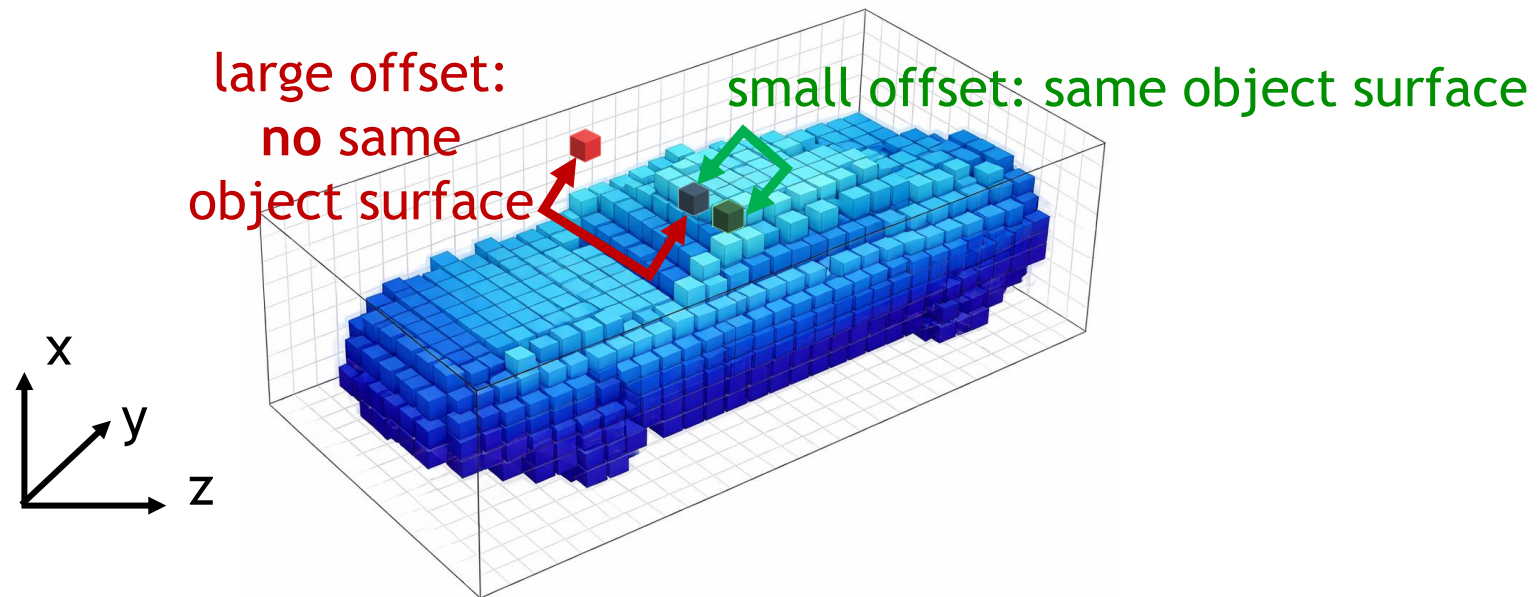
Bounded Property

- Voxel coordinates are *spatially constrained* due to technological sensor limits



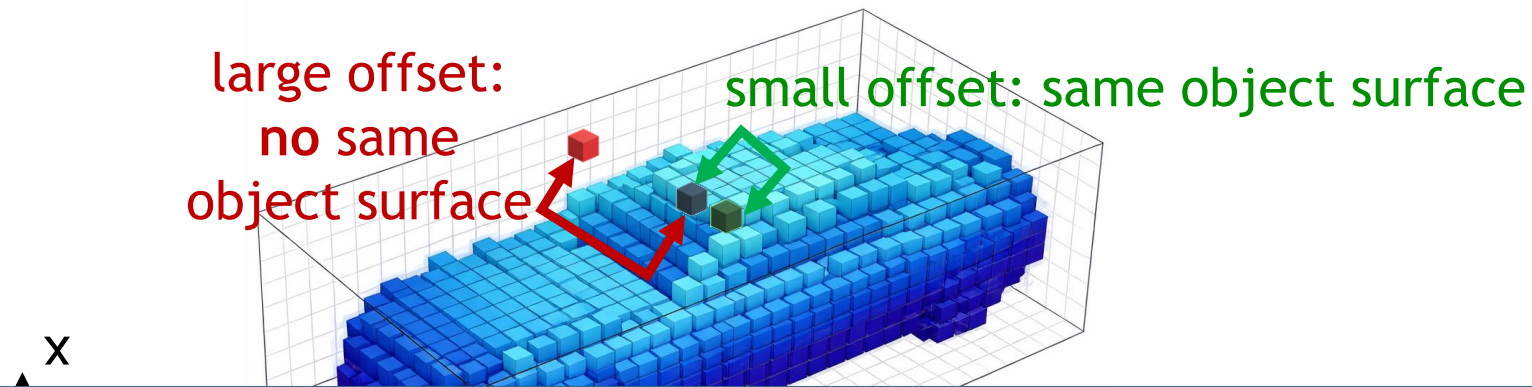
Neighboring Property

- Neighboring voxel coordinates of the same object surface *are likely* to exist in *small* offset displacements



Neighboring Property

- Neighboring voxel coordinates of the same object surface *are likely* to exist in *small* offset displacements



Existing SpC engines [e.g., TorchSparse++'23, Minuet'24]
do not leverage structural properties of voxel data

Outline

Background

Prior Works

Voxel Data Properties

Spira Design

Evaluation

Designing Spira

- The **first voxel-property-aware** SpC engine
- Spira incorporates **4 key components**:

One-Shot Z-Delta Search Mapping

Packed-Native Mapping Step

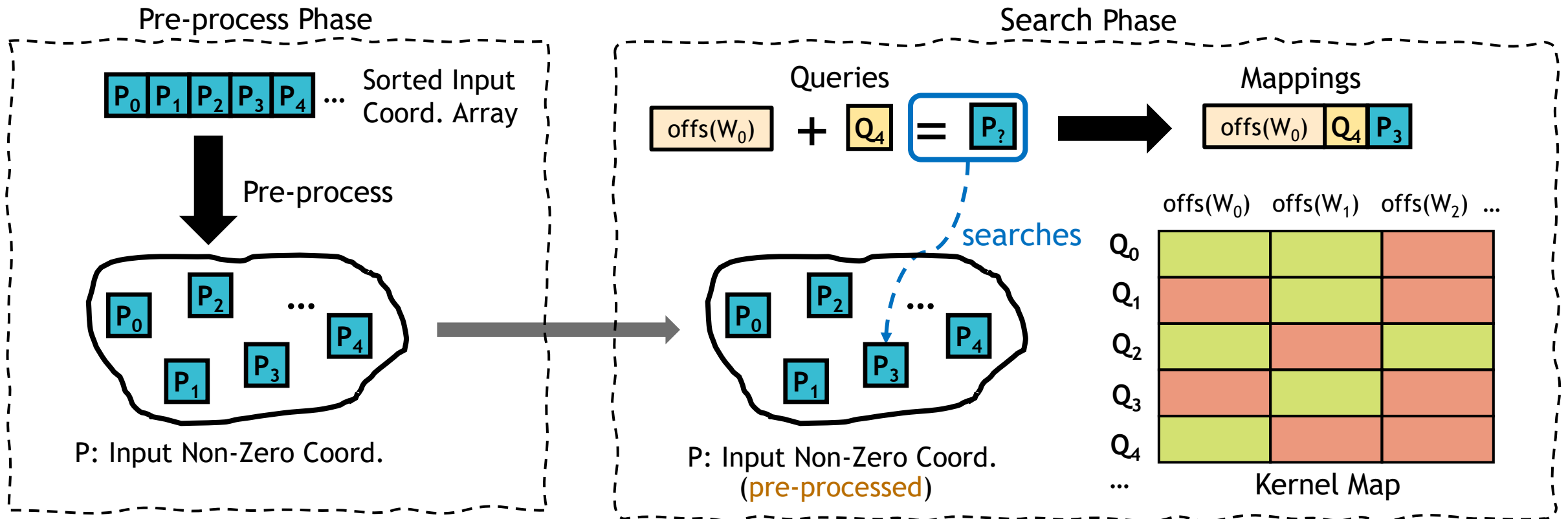
Adaptive Hybrid Dataflow Execution

Network-Wide Mapping

One-Shot Search in the Mapping Step

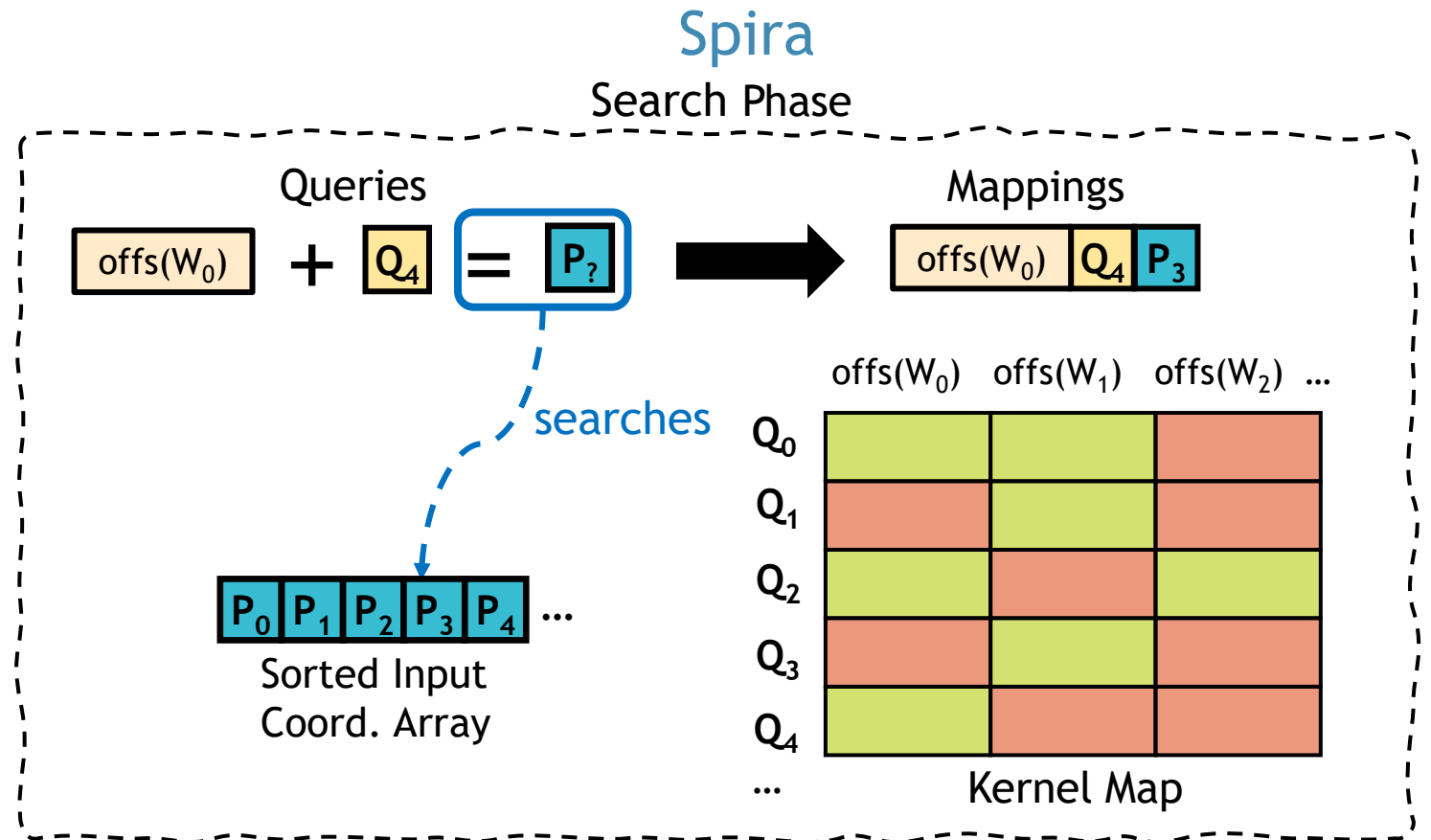
- Spira eliminates pre-processing by **directly** searching on the sorted input coordinates as given

Existing Works



One-Shot Search in the Mapping Step

- Spira eliminates pre-processing by **directly** searching on the sorted input coordinates as given

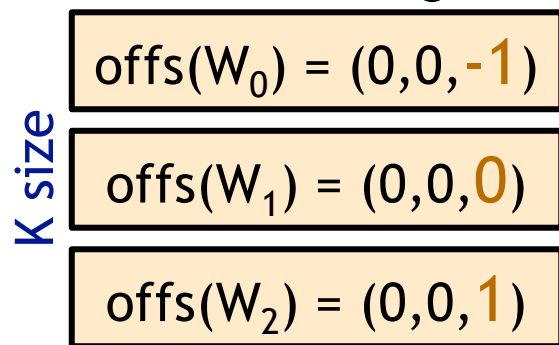


Key Observation on Consecutive Weight Offsets

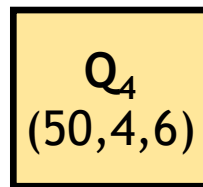


Integer Property

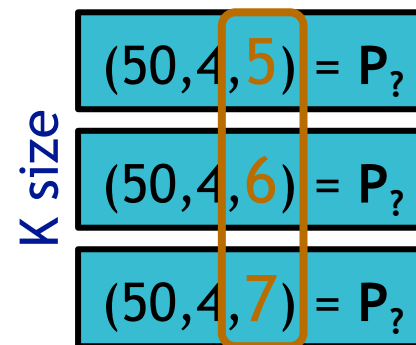
Consecutive Weight Offsets



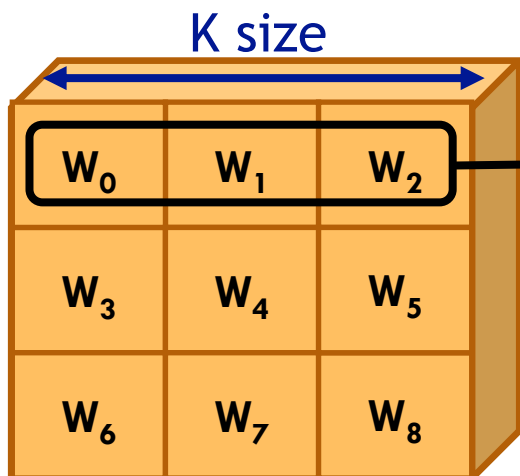
Output Coordinate



Generated Consecutive Queries



identical x, y values + consecutive z values



$\text{offs}(W_0), \text{offs}(W_1), \text{offs}(W_2)$: identical x, y values + consecutive z values

e.g., $(0, 0, -1)$ $(0, 0, 0)$ $(0, 0, 1)$

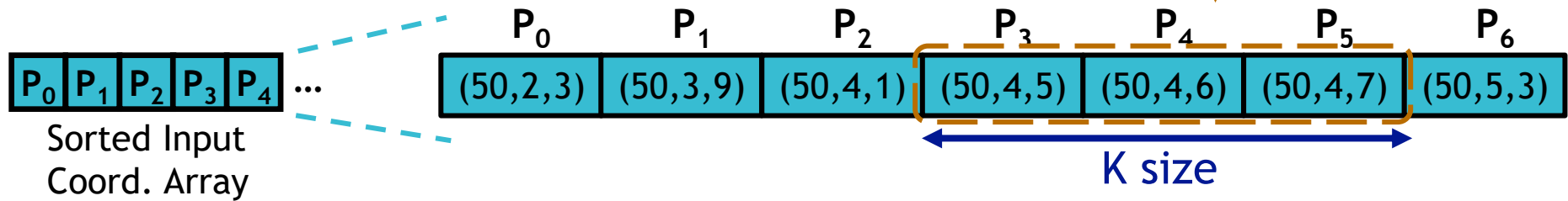
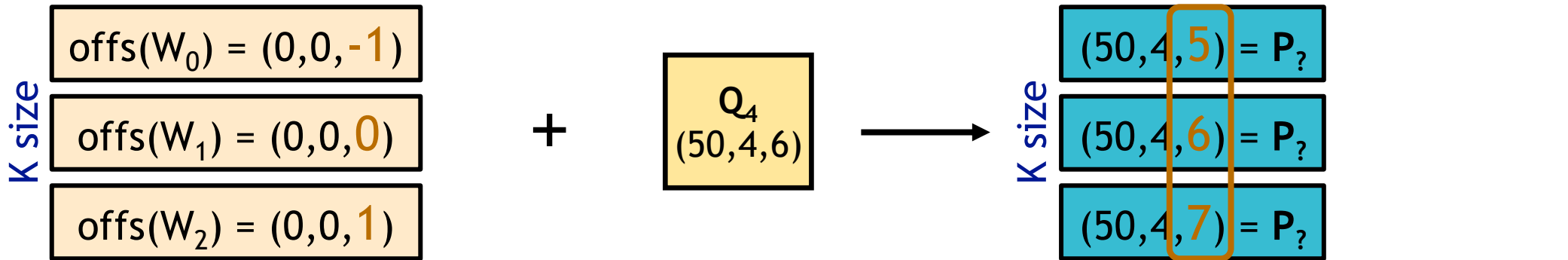
Consecutive weight offsets in z-axis produce consecutive queries

W: Weights

One-Shot Z-Delta Search Mapping



Consecutive Weight Offsets Output Coordinate Generated **Consecutive** Queries

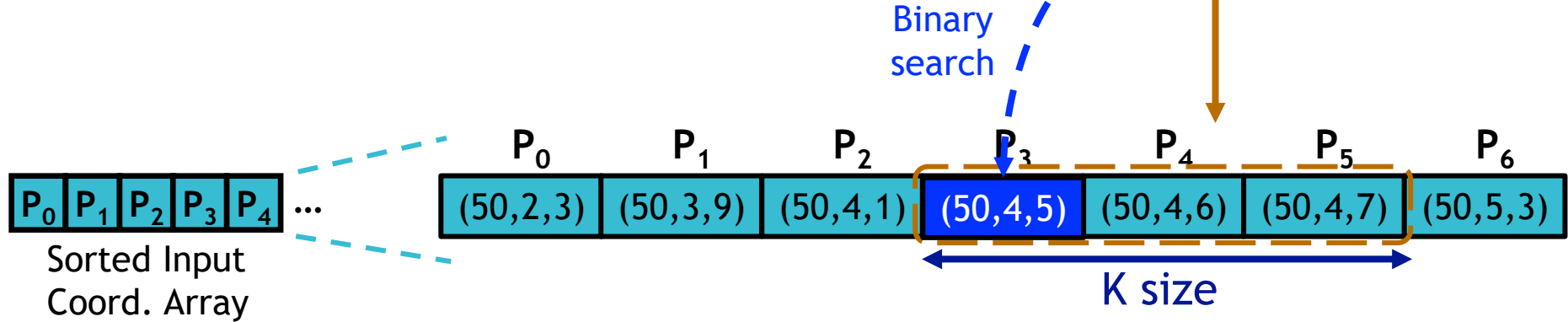
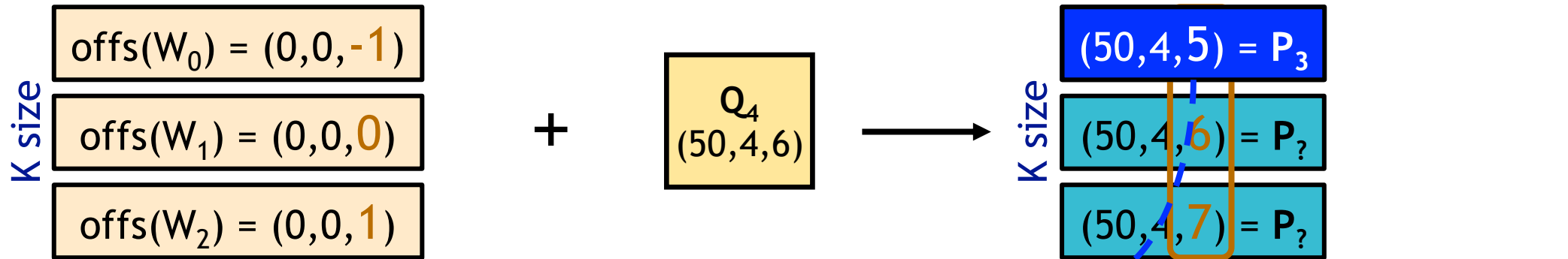


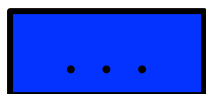
K consecutive queries can only map to up to **K consecutive** elements in the sorted input coordinates


One-Shot Z-Delta Search Mapping



Consecutive Weight Offsets Output Coordinate Generated **Consecutive** Queries



 → binary search

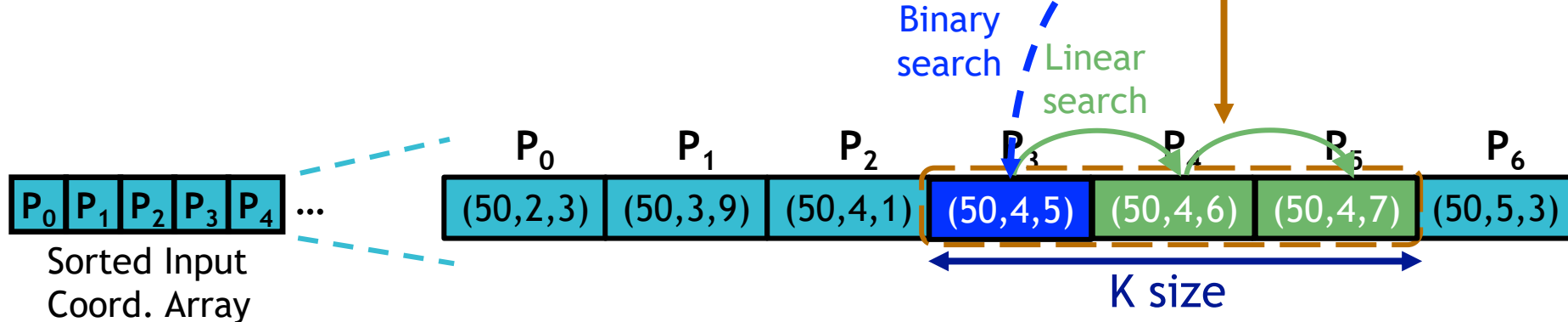
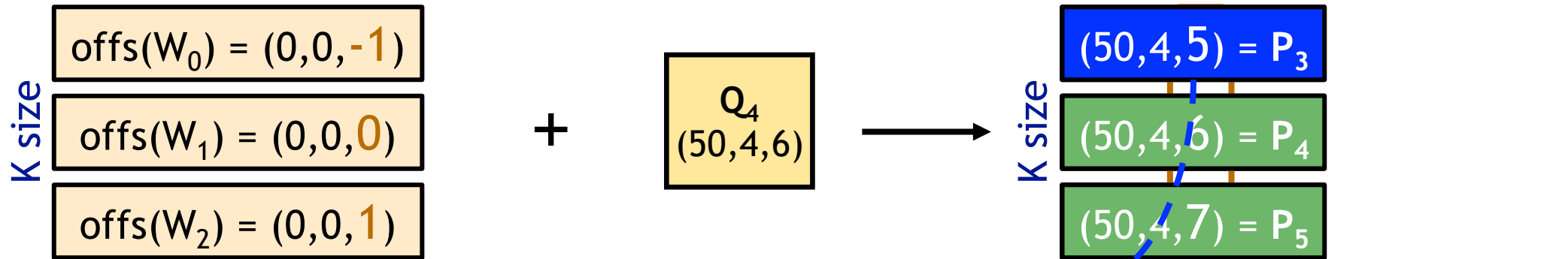
 → linear search

K consecutive queries can only map to up to K consecutive elements in the sorted input coordinates

One-Shot Z-Delta Search Mapping



Consecutive Weight Offsets Output Coordinate Generated **Consecutive** Queries



... → binary search
... → linear search

K consecutive queries can only map to up to K consecutive elements in the sorted input coordinates

Designing Spira

- The **first voxel-property-aware** SpC engine
- Spira incorporates the following **4 key components**:

One-Shot Z-Delta Search Mapping

- ✓ No pre-processing costs
- ✓ High data locality
- ✓ $\sim K\times$ less binary searches

Packed-Native Mapping Step

Adaptive Hybrid Dataflow Execution

Network-Wide Mapping



Packed-Native Mapping Step

- Key Idea: pack all coordinates into a **single** integer number (32-bit or 64-bit)
- Pack **only once** at the network **start**; perform mapping steps **as packed-native**

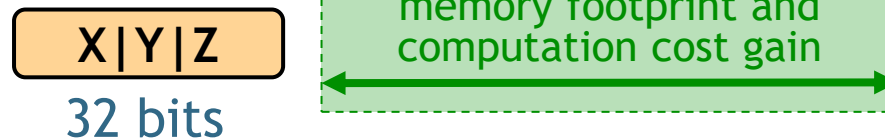
Prior Works

three coordinates



Spira

one **packed** coordinates



Voxel Coordinates require **less bits** in the x-, y-, z- axis

Designing Spira

- The **first voxel-property-aware** SpC engine
- Spira incorporates the following **4 key ideas**:

One-Shot Z-Delta Search Mapping

- ✓ No pre-processing costs
- ✓ High data locality
- ✓ $\sim K\times$ less binary searches

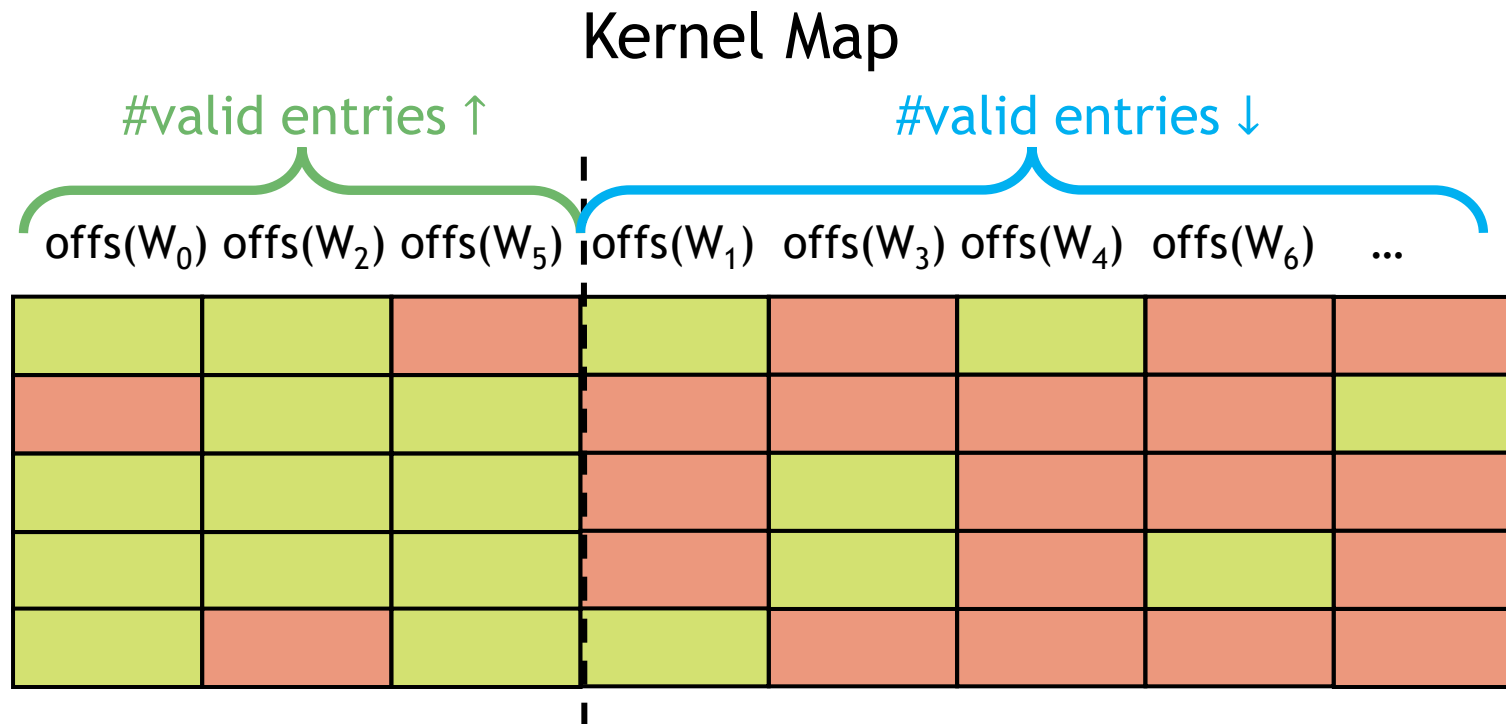
Packed-Native Mapping Step

- ✓ $\sim 3\times$ less #comparisons
- ✓ $\sim 3\times$ less #memory accesses

Adaptive Hybrid Dataflow Execution

Network-Wide Mapping

Adaptive Hybrid Dataflow Execution

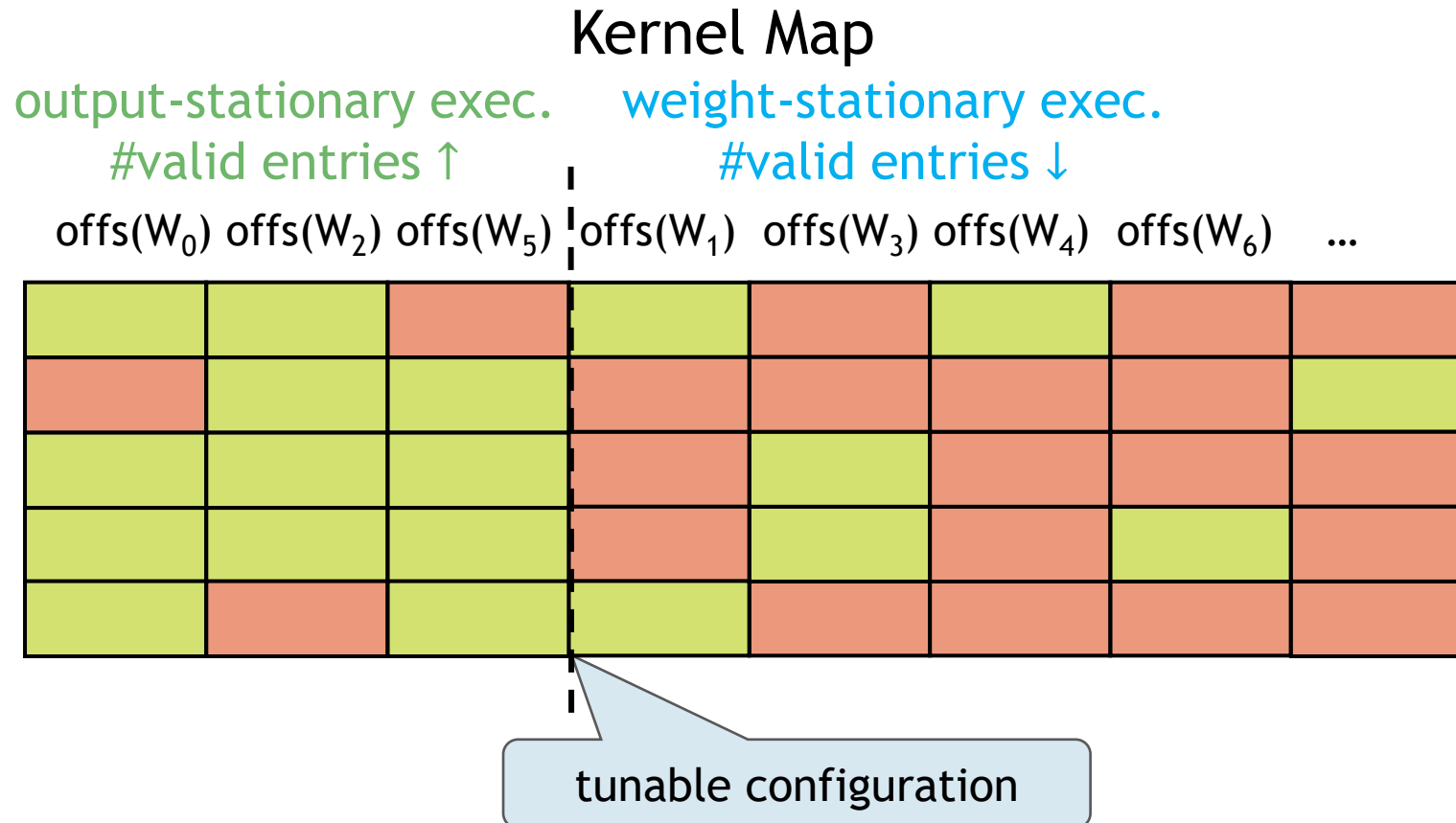


Columns associated to smaller offset displacements have larger #valid entries in the Kernel Map

Adaptive Hybrid Dataflow Execution



- Different weight offsets in kernel map can be processed with either output- or weight-stationary dataflow execution



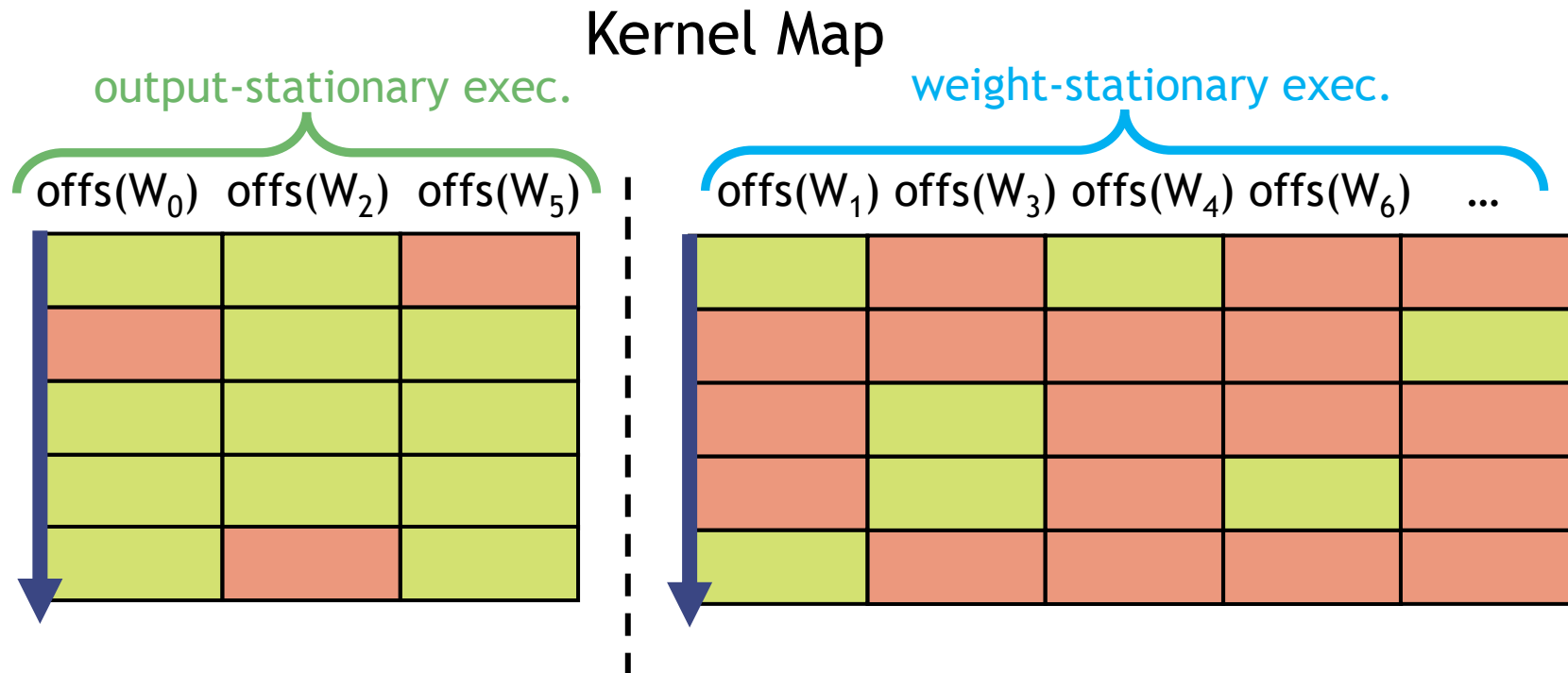
Dataflow-Aware Kernel Map Post-Processing

Three possible dataflow scenarios:

Dataflow-Aware Kernel Map Post-Processing

Three possible dataflow scenarios:

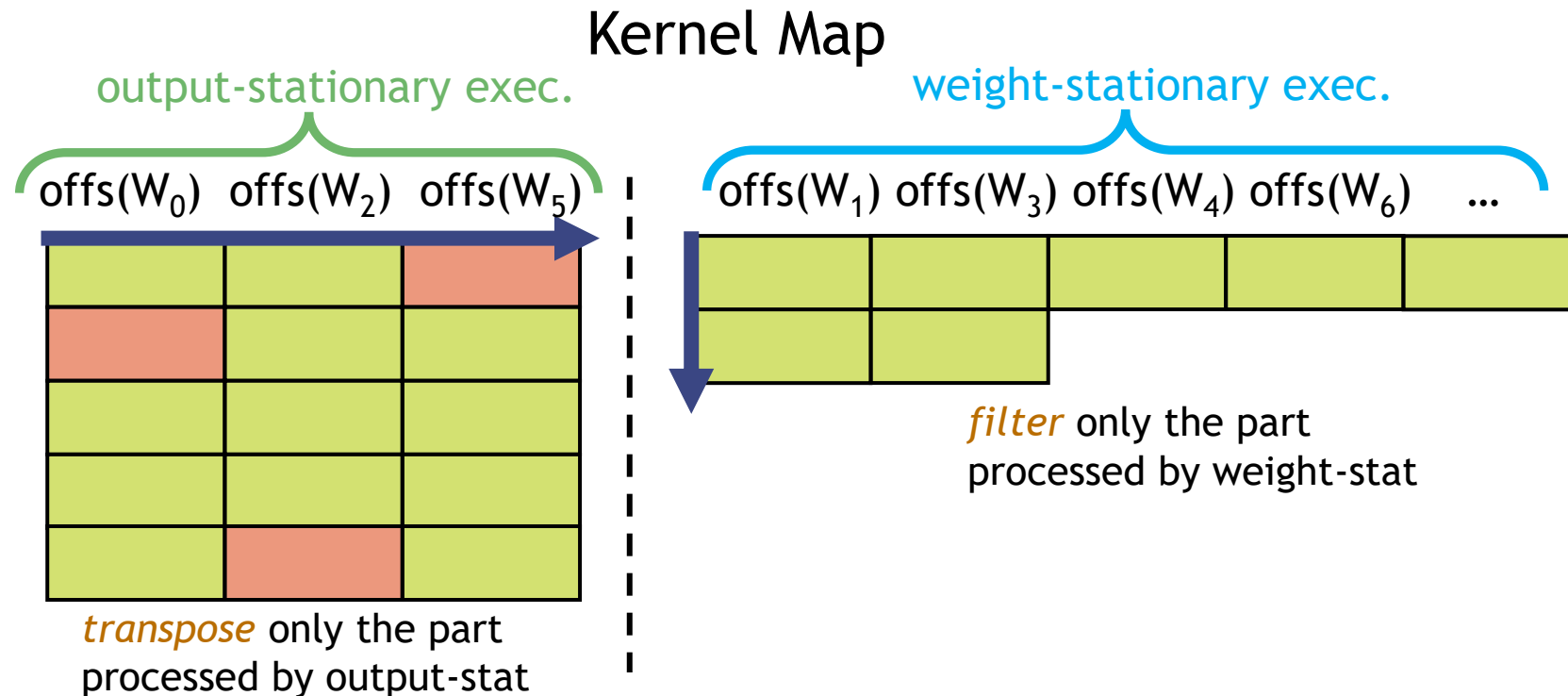
3. **Hybrid-Dual**: directly materialized as **column-major**



Dataflow-Aware Kernel Map Post-Processing

Three possible dataflow scenarios:

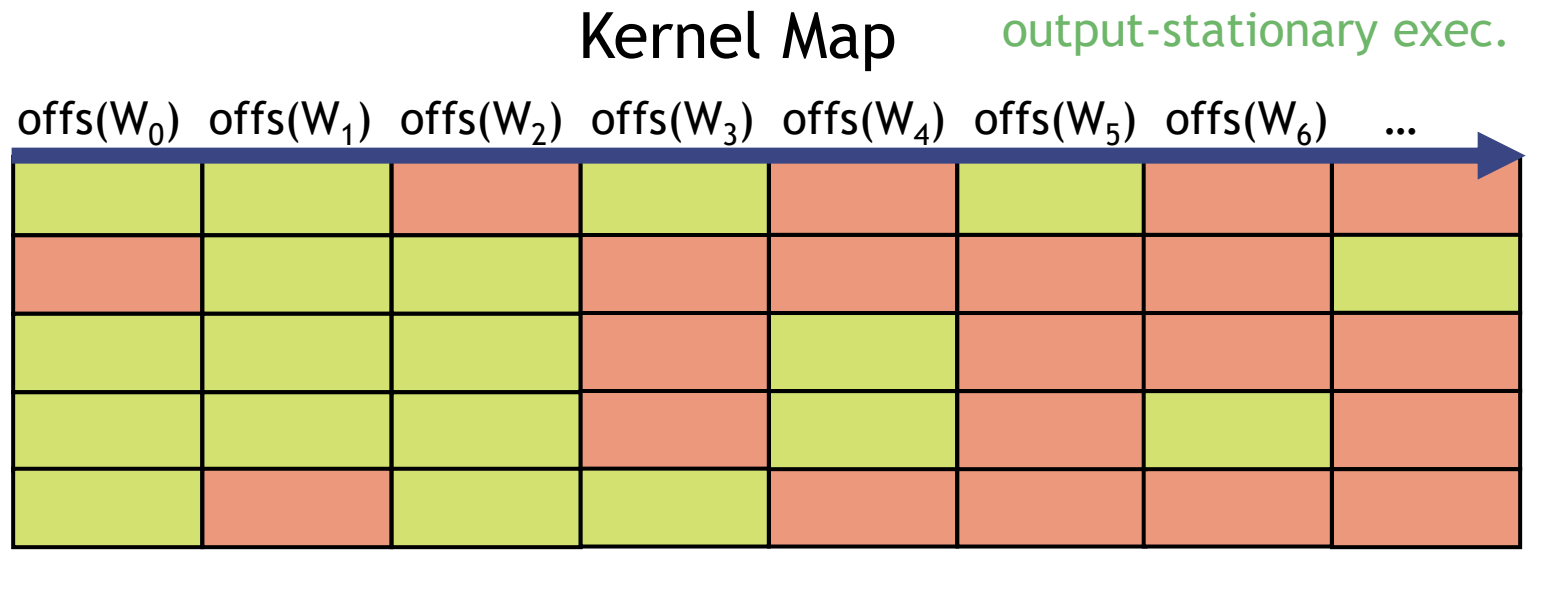
3. **Hybrid-Dual**: directly materialized as **column-major**



Dataflow-Aware Kernel Map Post-Processing

Three possible dataflow scenarios:

1. **Pure Output-Stat**: directly materialized as **row-major** → **no** post-processing
3. Hybrid-Dual: directly materialized as **column-major**



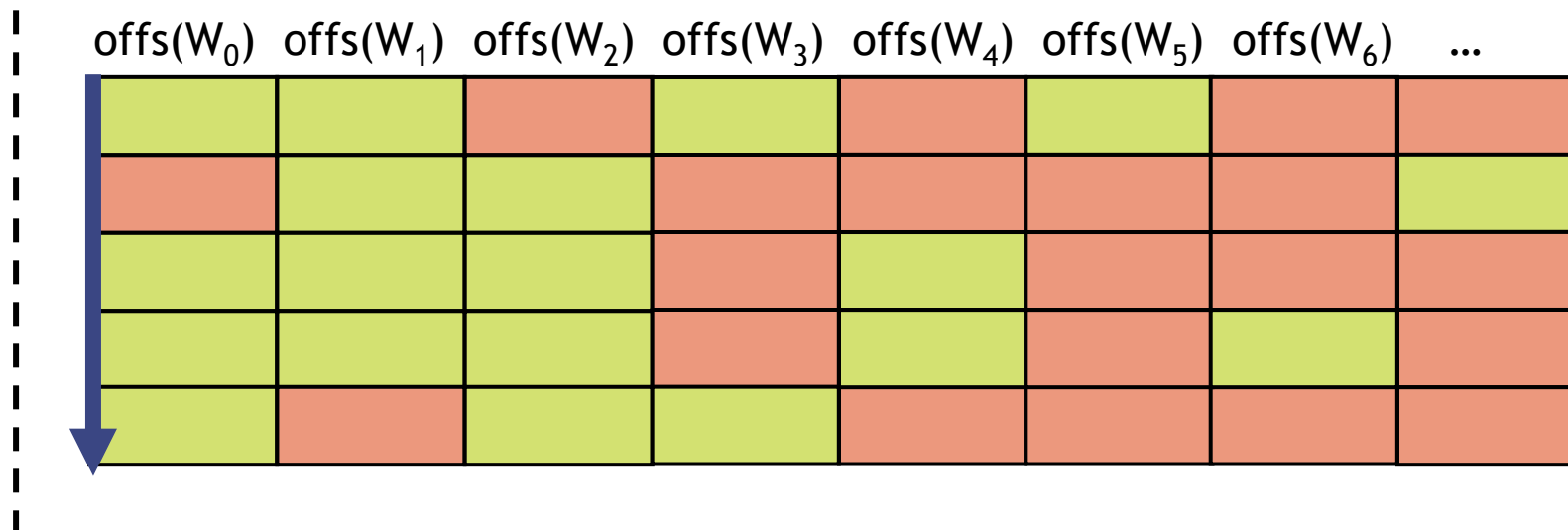
Dataflow-Aware Kernel Map Post-Processing

Three possible dataflow scenarios:

1. Pure Output-Stat: directly materialized as row-major → no post-processing
2. **Pure Weight-Stat**: directly materialized as column-major →
3. Hybrid-Dual: directly materialized as column-major

weight-stationary exec.

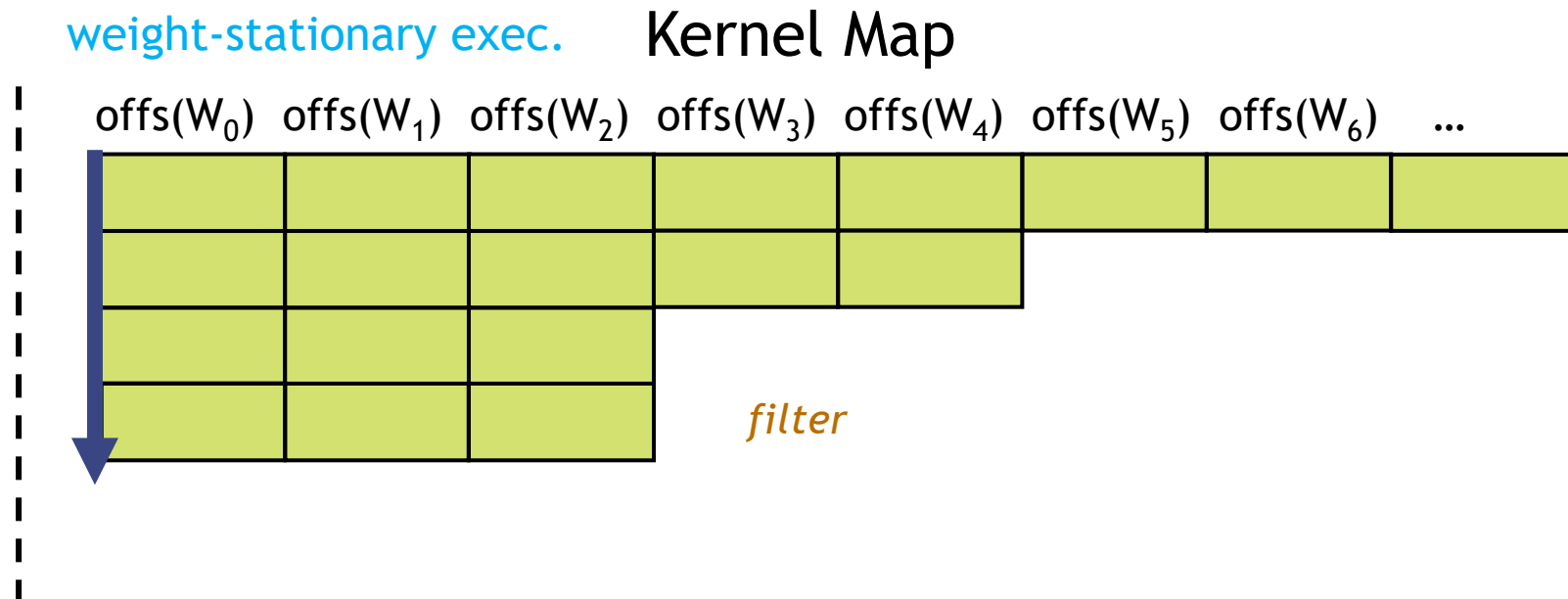
Kernel Map



Dataflow-Aware Kernel Map Post-Processing

Three possible dataflow scenarios:

1. Pure Output-Stat: directly materialized as row-major → no post-processing
2. **Pure Weight-Stat**: directly materialized as column-major → **only** filtering
3. Hybrid-Dual: directly materialized as column-major



Designing Spira

- The **first voxel-property-aware** SpC engine
- Spira incorporates the following **4 key ideas**:

One-Shot Z-Delta Search Mapping

- ✓ No pre-processing costs
- ✓ High data locality
- ✓ $\sim K\times$ less binary searches

Packed-Native Mapping Step

- ✓ $\sim 3\times$ less #comparisons
- ✓ $\sim 3\times$ less #memory accesses

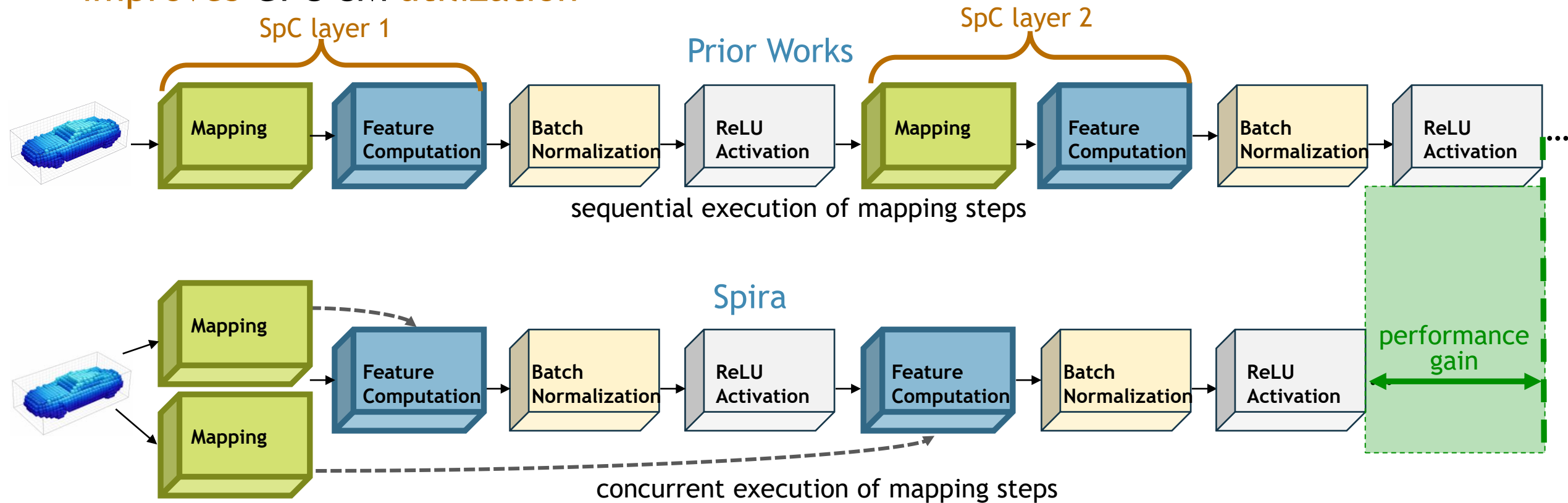
Adaptive Hybrid Dataflow Execution

- ✓ Minimal post-processing costs in all dataflows
- ✓ Adapting execution schema based on data and layer characteristics

Network-Wide Mapping

Network-Wide Mapping

- Key Observation: *Mapping steps have no true dependencies neither with feature computation steps nor between them*
- Spira executes all mapping steps concurrently **at network start** and significantly **improves GPU SM utilization**



Designing Spira

- The **first voxel-property-aware** SpC engine
- Spira incorporates the following **4 key ideas**:

One-Shot Z-Delta Search Mapping

- ✓ No pre-processing costs
- ✓ High data locality
- ✓ $\sim K\times$ less binary searches

Packed-Native Mapping Step

- ✓ $\sim 3\times$ less #comparisons
- ✓ $\sim 3\times$ less #memory accesses

Adaptive Hybrid Dataflow Execution

- ✓ Minimal post-processing costs in all dataflows
- ✓ Adapting execution schema based on data and layer characteristics

Network-Wide Mapping

- ✓ Higher execution parallelism
- ✓ Higher GPU utilization

Outline

Background

Prior Works

Voxel Data Properties

Spira Design

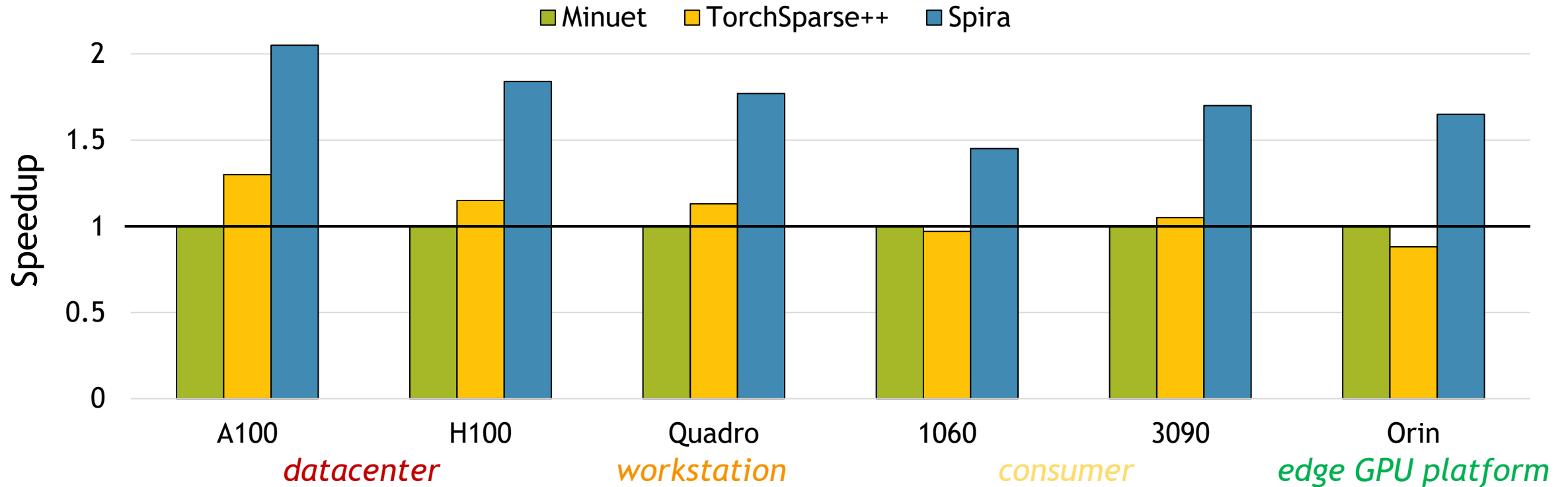
Evaluation

Evaluation Methodology

- 3x Point Cloud Networks: ResNet, Large ResNet, UNet
- 3x Datasets: Waymo, SemanticKITTI, ScanNet
- 6x GPUs:
 - A100, H100: *datacenter*
 - Quadro RTX 5000: *workstation*
 - GTX 1060, RTX 3090: *consumer*
 - Jetson Orin AGX: *edge GPU platform*
- Comparison points:
 - **TorchSparse++** [MICRO'23] - supports both output- and weight-stationary dataflow
 - **Minuet** [EuroSys'24] - supports only weight-stationary dataflow

Inference Performance across Multiple GPUs

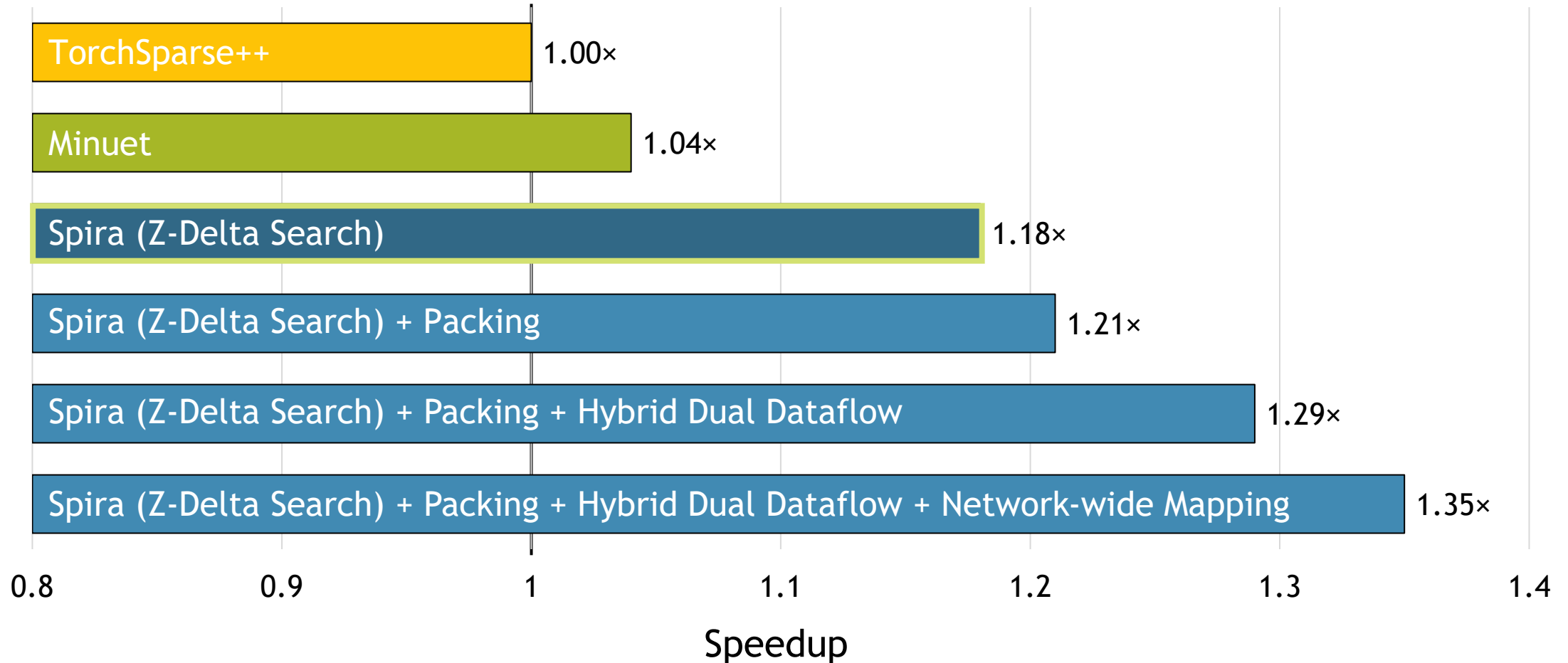
Averaged inference results across 3 networks and 3 datasets



Spira significantly outperforms prior SpC engines by **1.68×** averaged across **six** GPUs

Breakdown of Spira's Key Ideas

Breakdown in UNet



More in the Paper

- Detailed inference analysis for various datasets and networks
- Layer-wise analysis
- Hybrid dataflow sensitivity analysis
- Mapping step analysis
- Effect of network-wide mapping
- Input density ablation study
- Scalability study
- Memory footprint study



<https://arxiv.org/abs/2511.20834>

Spira is Open Source



SPIN-Research-Group / Spira Public

Notifications Fork 0 Star 1

<> Code Issues Pull requests Actions Projects Security and quality Insights

main 1 Branch 0 Tags

Go to file Code

dion-adam Update README.md 0f4bdd4 · 16 minutes ago 6 Commits

| | | |
|-------------------------|------------------|----------------|
| assets | Pushing Code | 24 minutes ago |
| automate | Pushing Code | 24 minutes ago |
| datasets | Pushing Code | 24 minutes ago |
| figures | Pushing Code | 24 minutes ago |
| results | Pushing Code | 24 minutes ago |
| scripts | Pushing Code | 24 minutes ago |
| source | Pushing Code | 24 minutes ago |
| Dockerfile | Pushing Code | 24 minutes ago |
| LICENSE | Pushing Code | 24 minutes ago |
| README.md | Update README.md | 16 minutes ago |
| THIRD_PARTY_LICENSES.md | Pushing Code | 24 minutes ago |

About

[MLSys'26] Spira: Exploiting Voxel Data Structural Properties for Efficient Sparse Convolution in Point Cloud Networks

Readme

Apache-2.0 license

Activity

Custom properties

1 star

0 watching

0 forks

Report repository

Releases

No releases published

Packages

No packages published

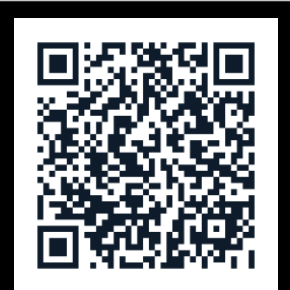
Conclusion

Spira is the first SpC engine that *leverages structural properties* of voxel data and significantly *improves* performance of *point cloud networks* on GPUs

Key Ideas & Benefits:

- ✓ *completely eliminates pre-processing* costs in the mapping step
- ✓ enables *high data locality* and *low computational cost* in the search phase of the mapping step via an intelligent search algorithm
- ✓ *proposes* a *packing format* that *reduces memory accesses* and *computation cost* on voxel coordinate processing
- ✓ *supports* a *tunable* adaptive dataflow schema with *low post-processing* costs
- ✓ *executes* all mapping steps *concurrently* to *improve GPU utilization*

Key Results: improves inference performance by on average **1.68×** over existing state-of-the-art SpC engines across **six** GPU architectures, spanning from *datacenter* to *edge* GPU devices



GitHub



Paper



Exploiting Voxel Data Structural Properties for Efficient Sparse Convolution in Point Cloud Networks

[Dionysios Adamopoulos](#),

Anastasia Pouloupoulou, Georgios Goumas, Christina Giannoula

MLSys 2026

Bellevue, USA, May 22nd, 2026



MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

