Spira

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

SPIN

MAX PLANCK INSTITUTE FOR SOFTWARE SYSTEMS

National Technical University of Athens CSLab

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

Spira is open-source: GitHub

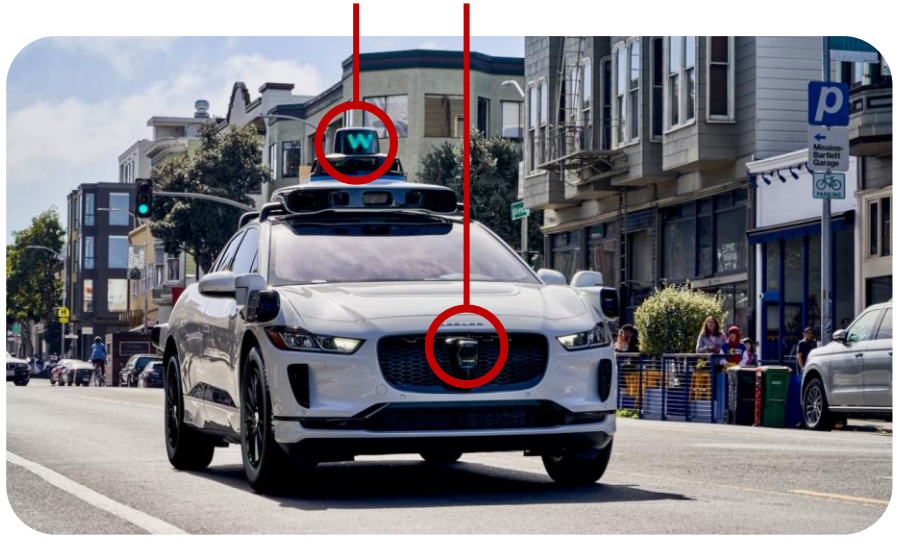
SPIN

Outline

- Background
- Prior Works
- Voxel Data Properties
- Spira Design
- Evaluation

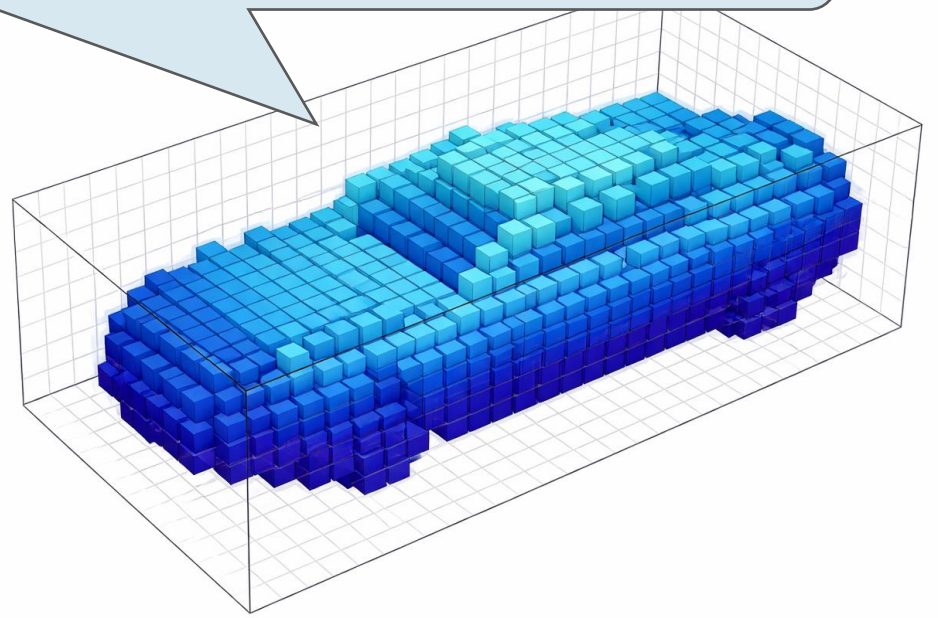
LiDAR Scanners Collect 3D Point Clouds

LiDAR Scanners



capture

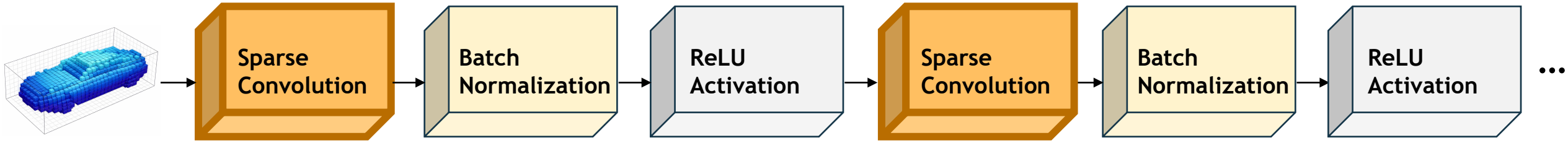
Transformed into 3D discrete cubes (**voxels**) that are grid-aligned

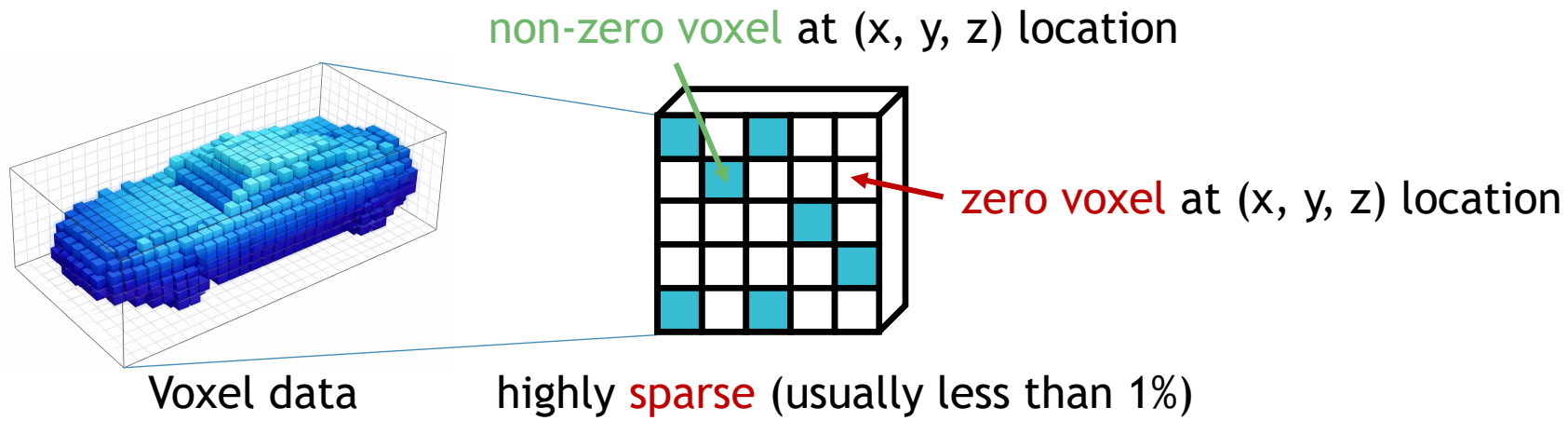


Voxelized Point Cloud
(abbr. voxel data)

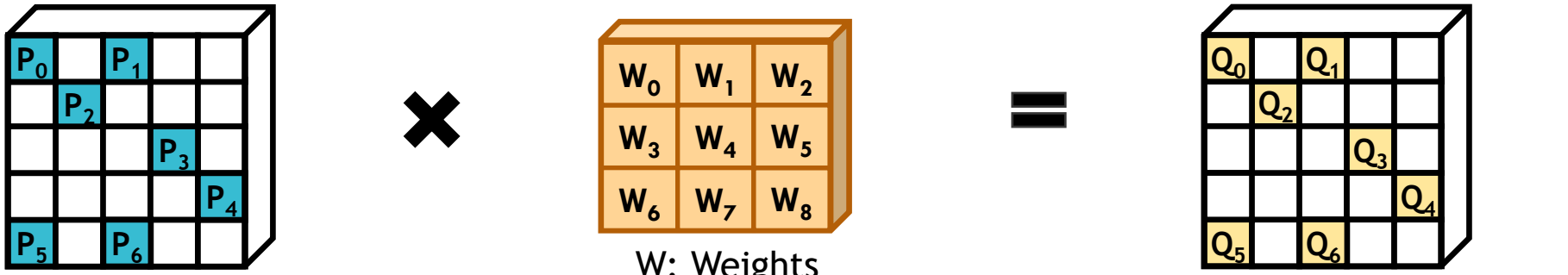
Point Cloud Networks Rely on Sparse Convolution (SpC)

- Point cloud networks efficiently process voxel data using Sparse Convolution





SpC Preserves the Sparsity in Output



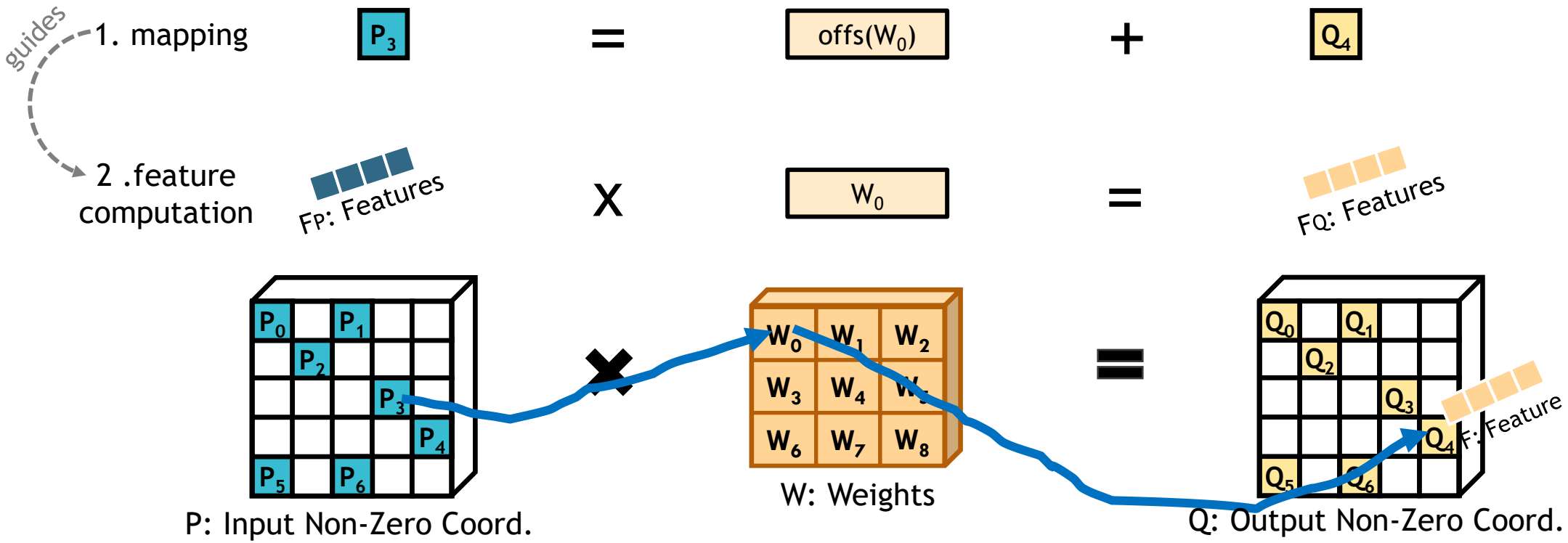
P: Input Non-Zero Coord.

W: Weights

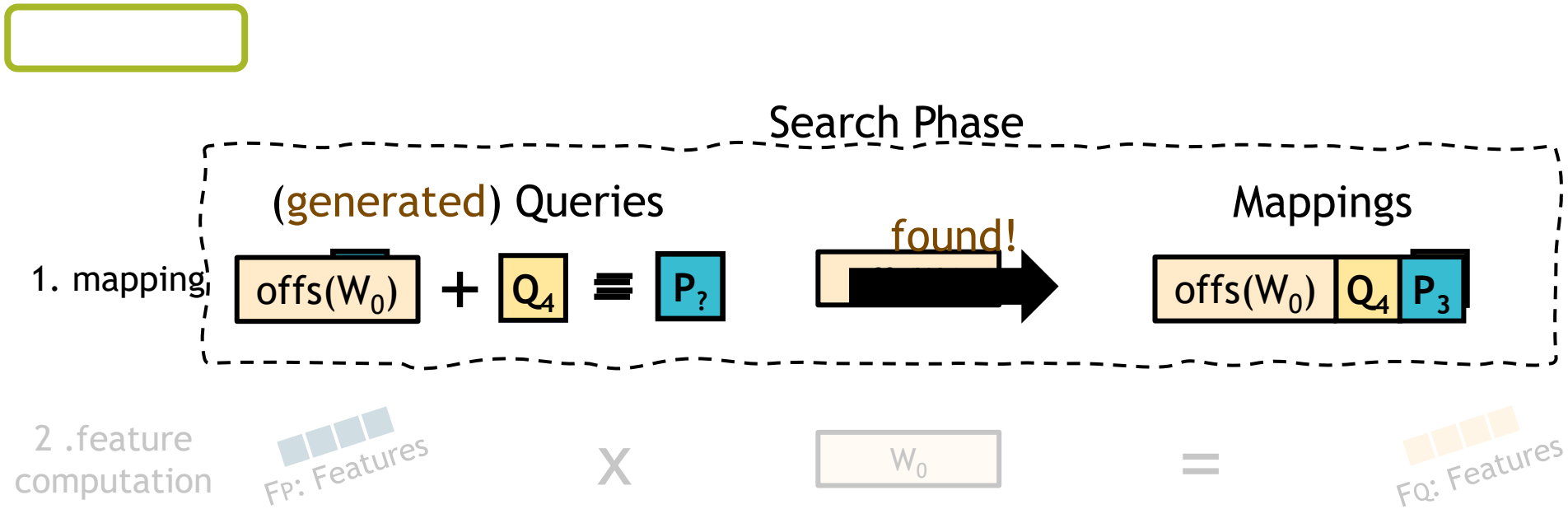
Q: Output Non-Zero Coord.

The 2 Steps of SpC

1. The Mapping Step
2. The Feature Computation Step

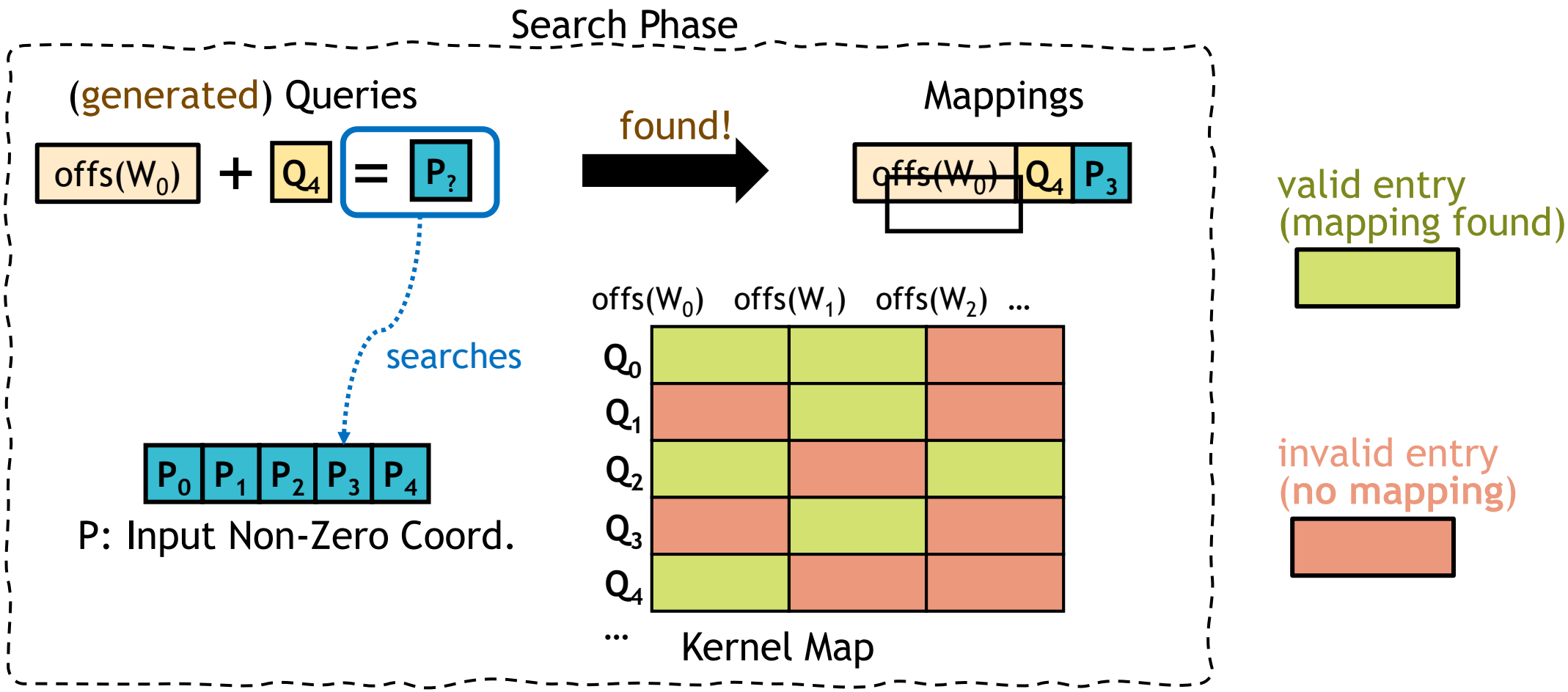


The 2 Steps of SpC

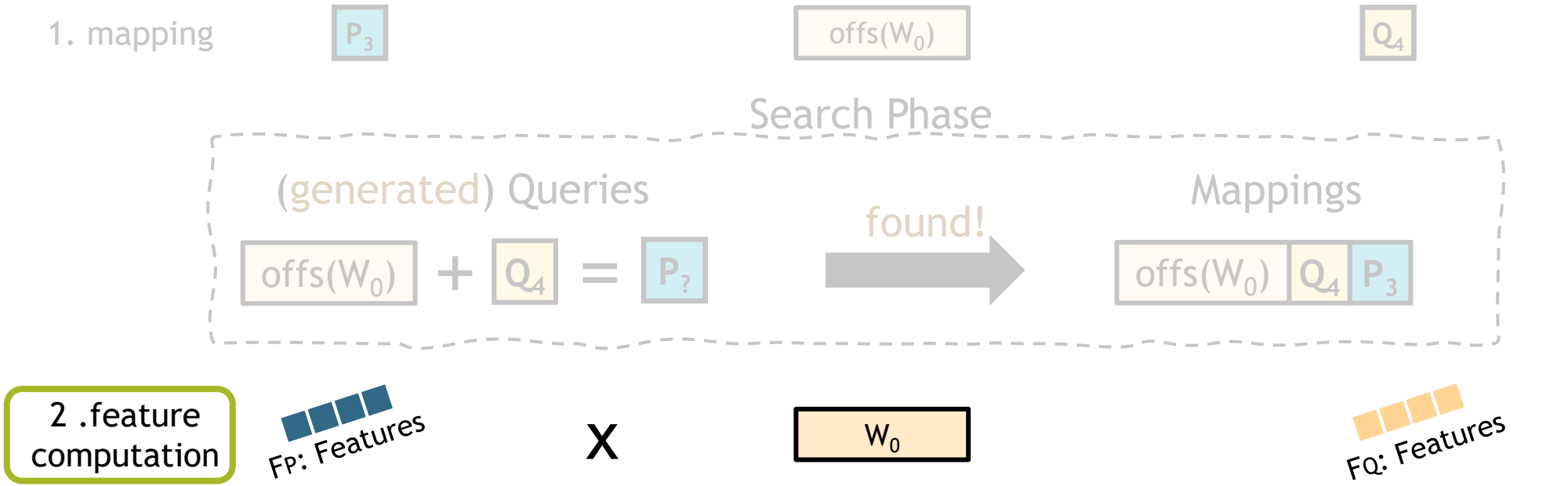


The Search Phase of the Mapping Step

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



The 2 Steps of SpC



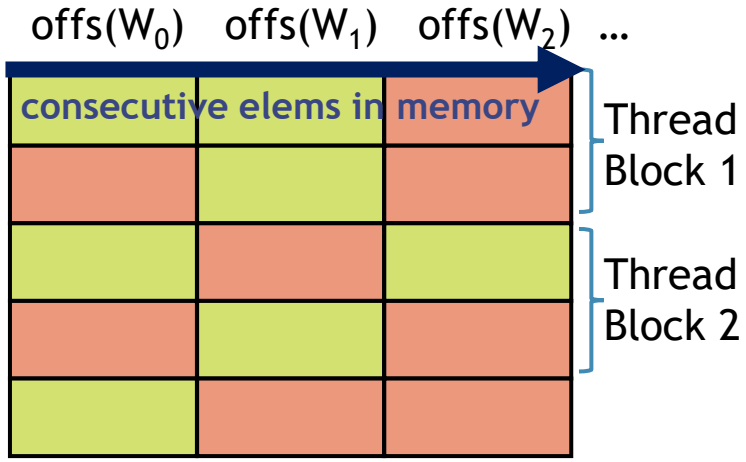
OR

(A) *Output-Stationary execution dataflow*

(B) Weight-Stationary *execution dataflow*

Two Dataflows for the Feature Computation Step

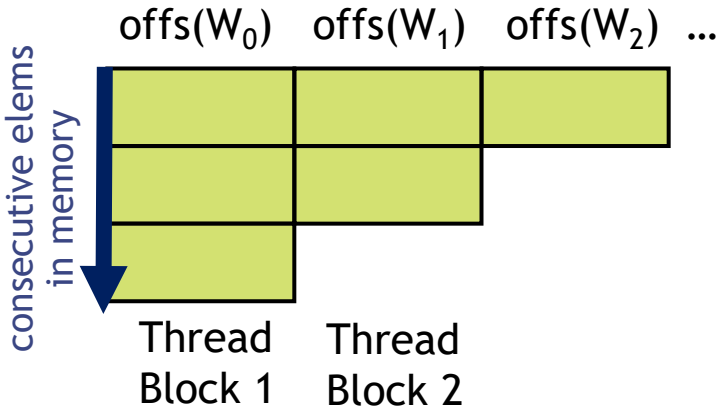
- Two different execution dataflows about how to process the kernel map and produce the output features



Kernel Map
(**row-major**)

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

(A) Output-Stationary execution dataflow



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

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

(B) Weight-Stationary execution dataflow

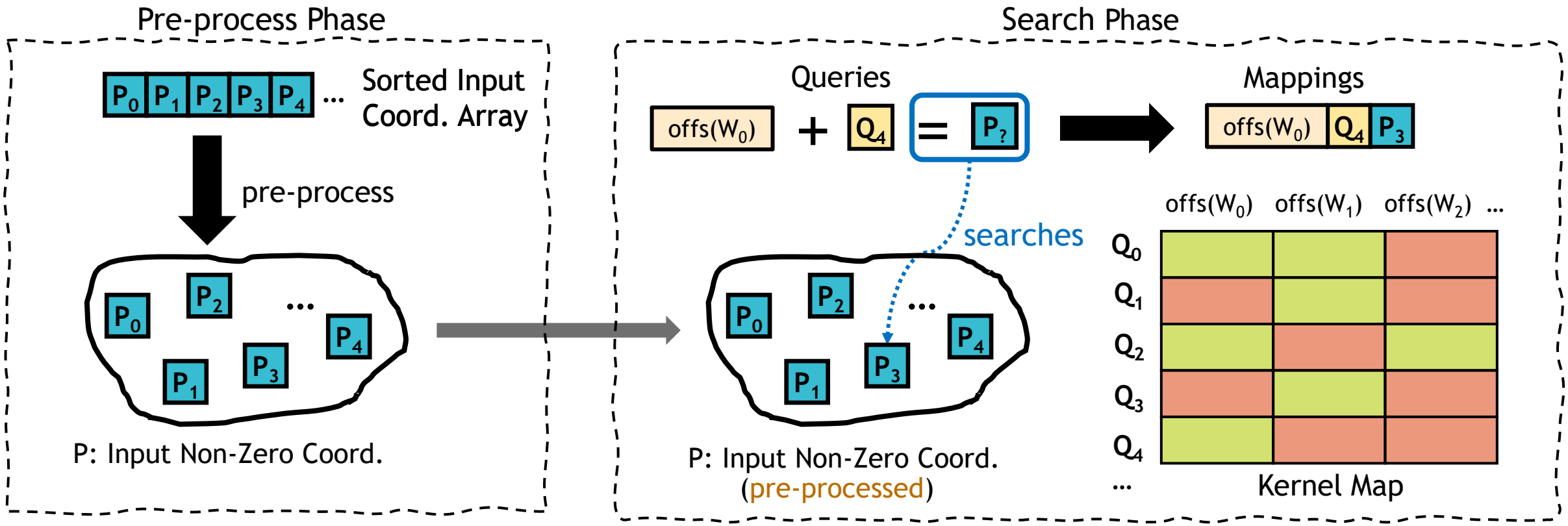
Either dataflow performs best depending on data and layer characteristics

Outline

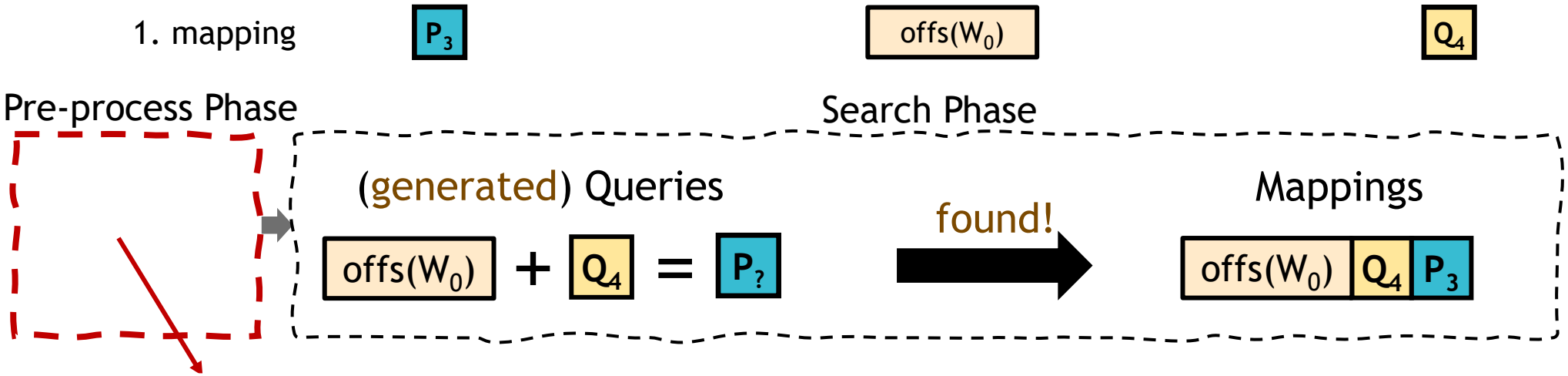
- Background
- **Prior Works**
- Voxel Data Properties
- Spira Design
- Evaluation

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

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



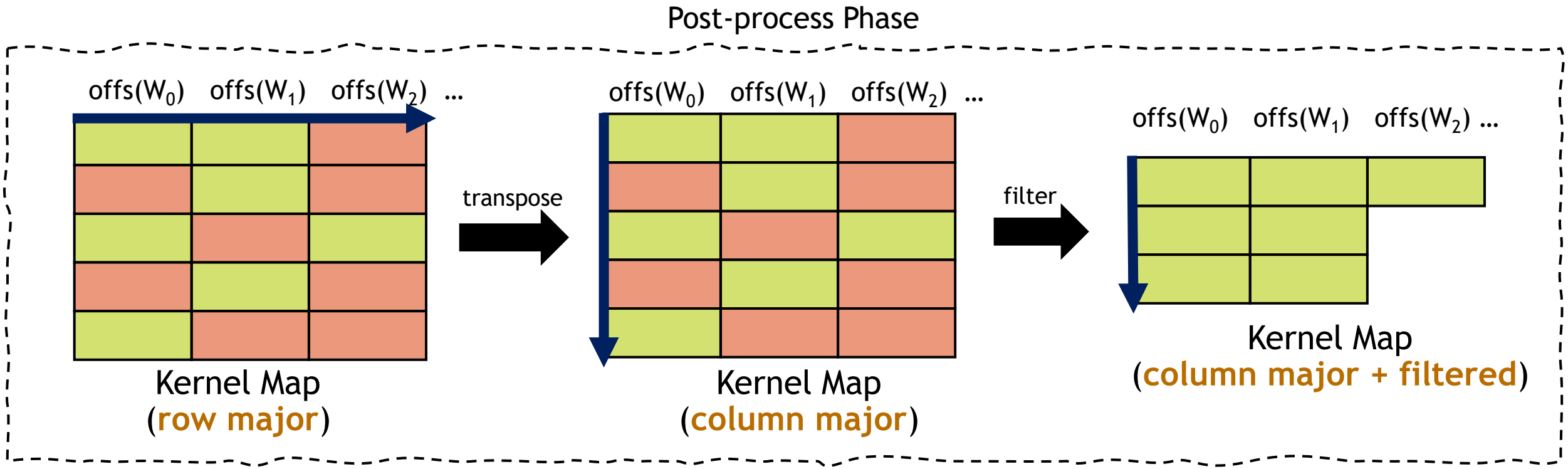
Prior Works Have Large Pre-Processing Costs



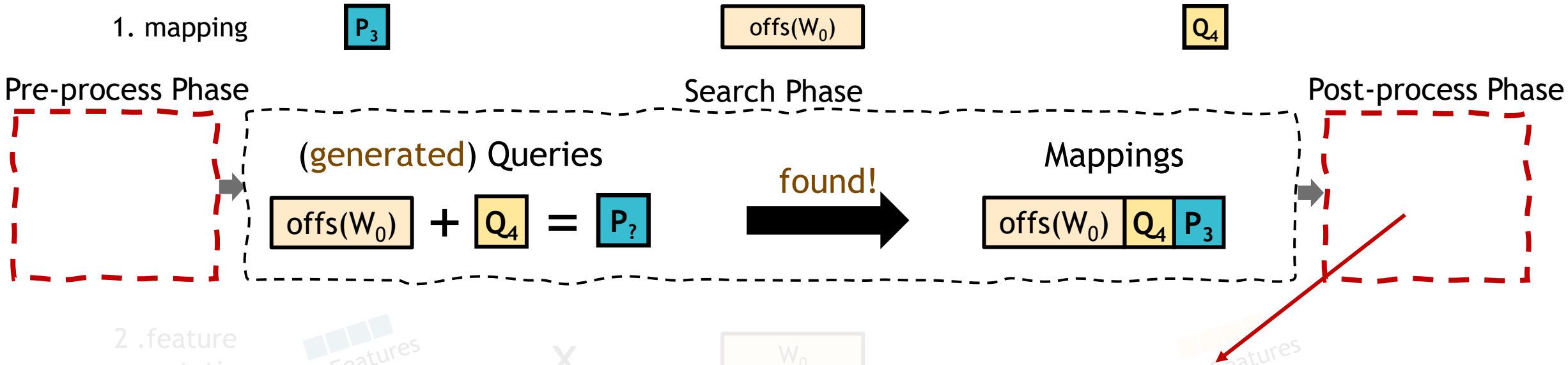
e.g., Minuet [Yang et al., EuroSys'24] → pre-processing time is almost equal to search time

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 Have Large Post-Processing Costs



- 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

Outline

Background

Prior Works

Voxel Data Properties

Spira Design

Evaluation

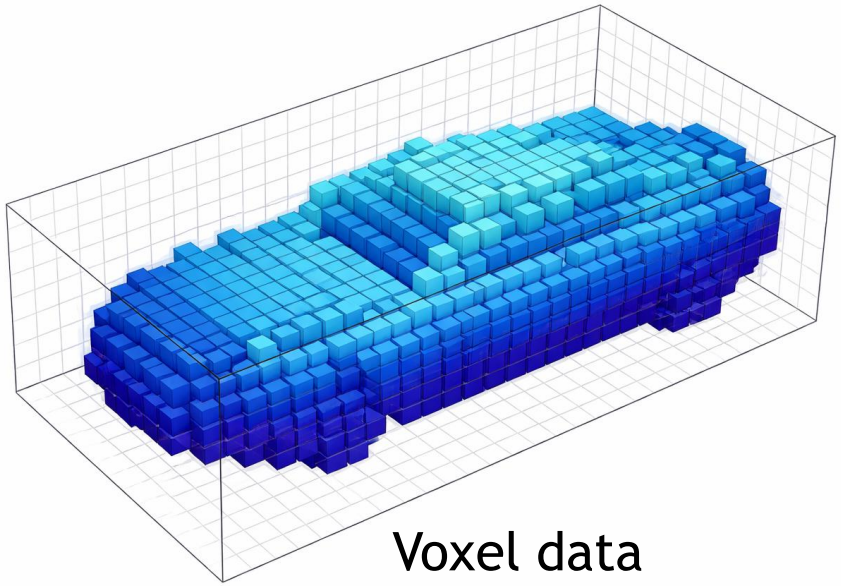
Identified Voxel Structural Properties

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

Integer Property

Bounded Property

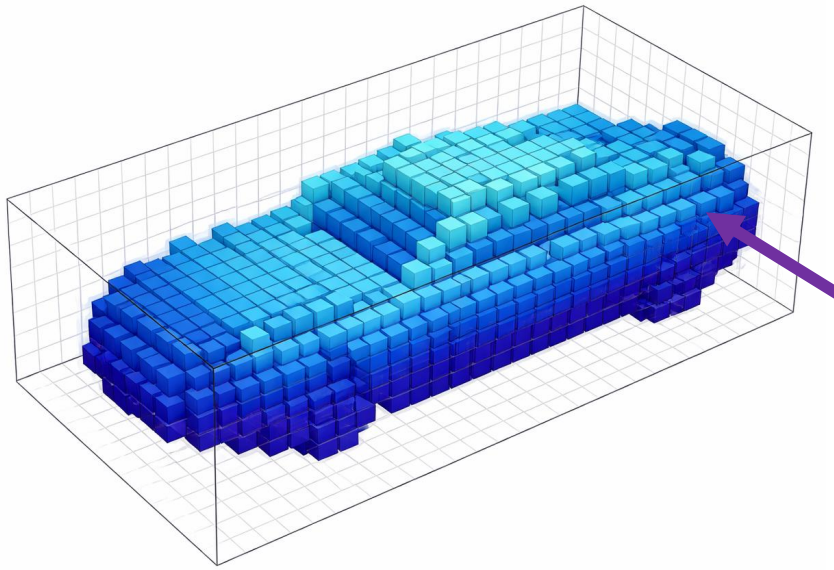
Neighboring Property



Voxel data

Integer Property

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



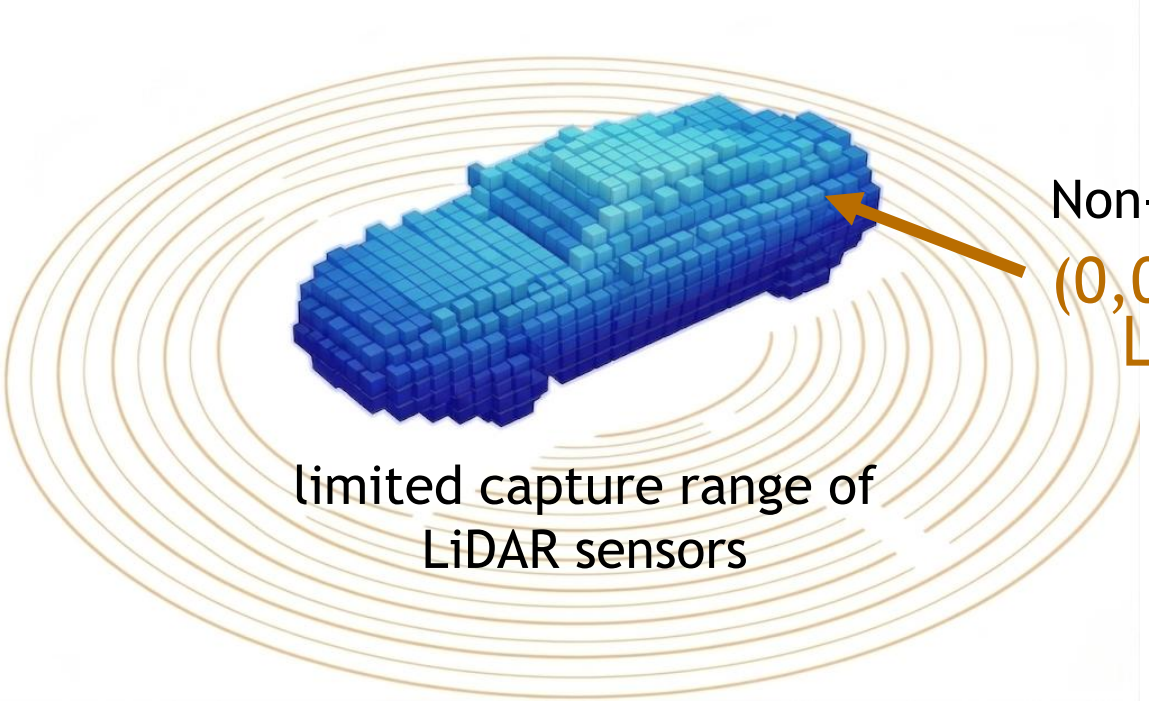
Non-zero coordinate at (x, y, z) location

(42,7,12)

integer values

Bounded Property

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



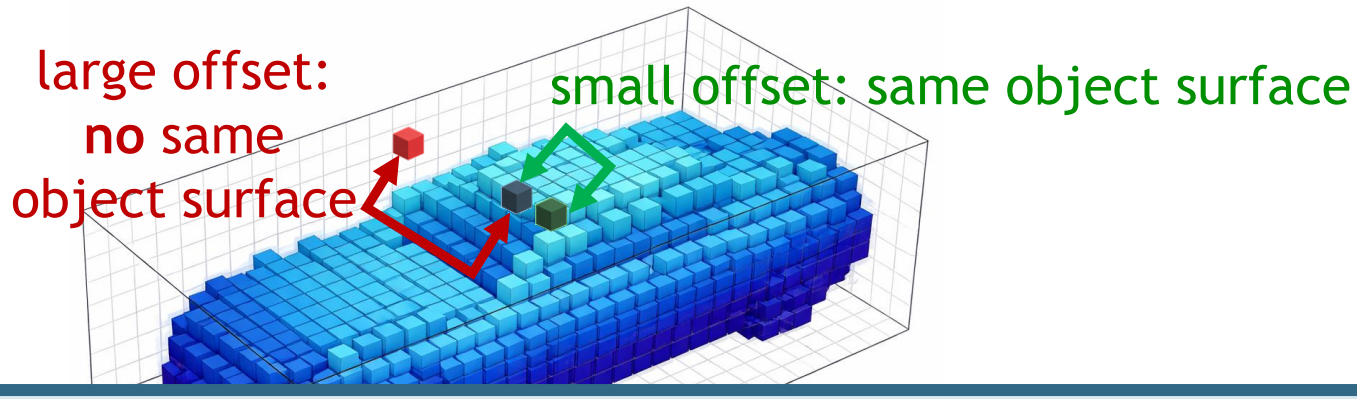
Non-zero coordinate at (x, y, z) location

(0,0,0) < (42,7,12) < (800,800,100)

bounded range

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

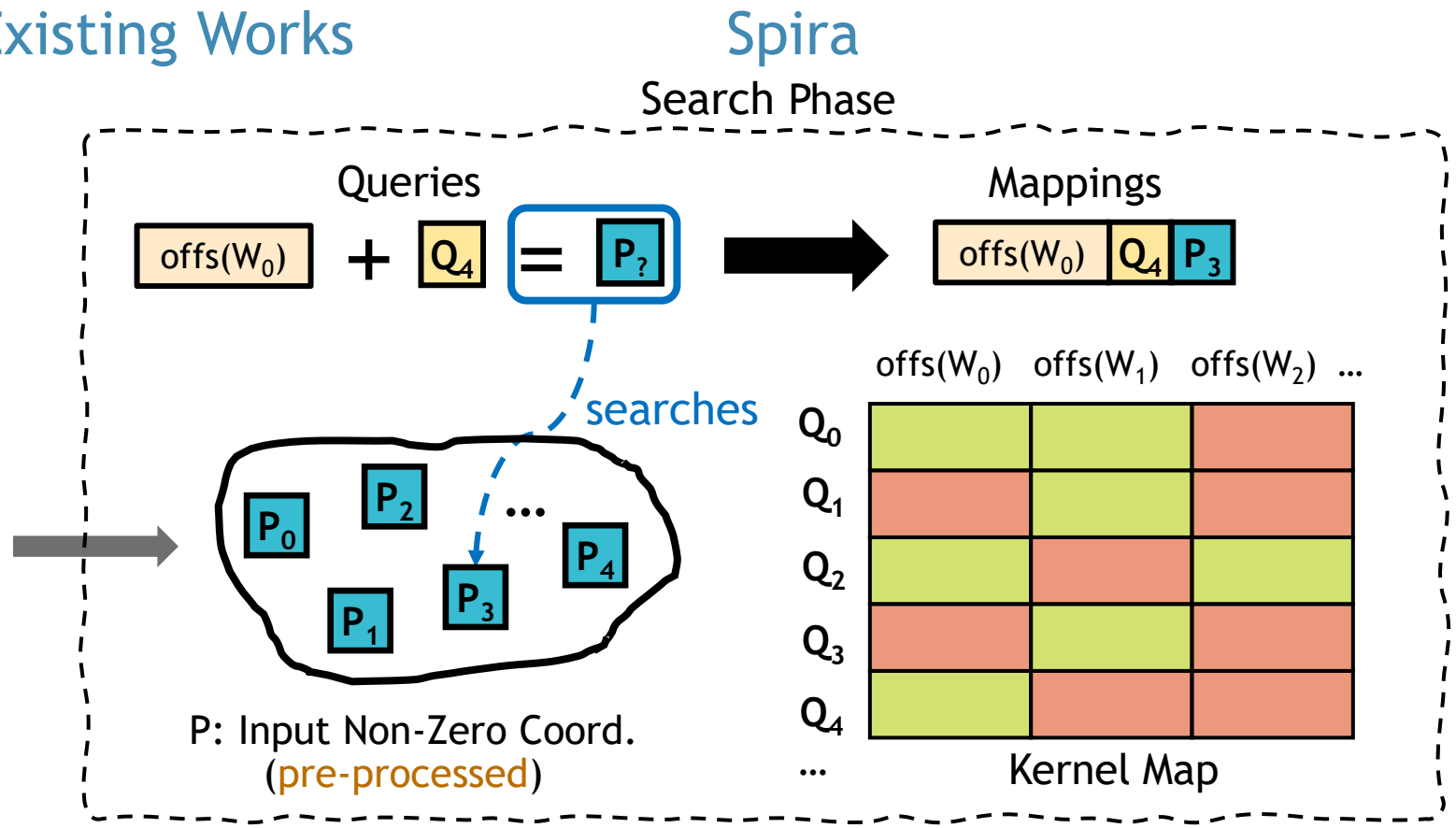
Packed-Native Mapping Step

Adaptive Hybrid Dataflow Execution

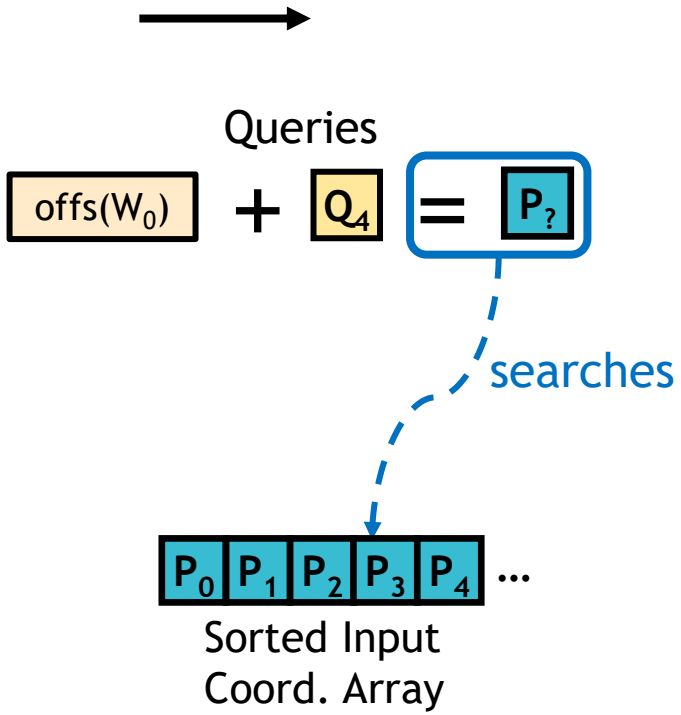
Network-Wide Mapping

One-Shot Search in the Mapping Step

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



Key Observation on Consecutive Weight Offsets



Key Observation on Consecutive Weight Offsets

Integer Property

Consecutive Weight Offsets

K size

- offs(W_0) = (0,0,-1)
- offs(W_1) = (0,0,0)
- offs(W_2) = (0,0,1)

+

Output Coordinate

Q_4 (50,4,6)

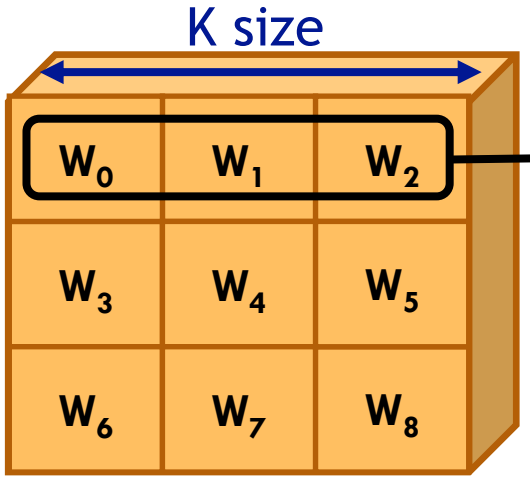
→

Generated **Consecutive** Queries

K size

- (50,4,5) = $P_?$
- (50,4,6) = $P_?$
- (50,4,7) = $P_?$

identical x, y values + **consecutive** z values



W: Weights

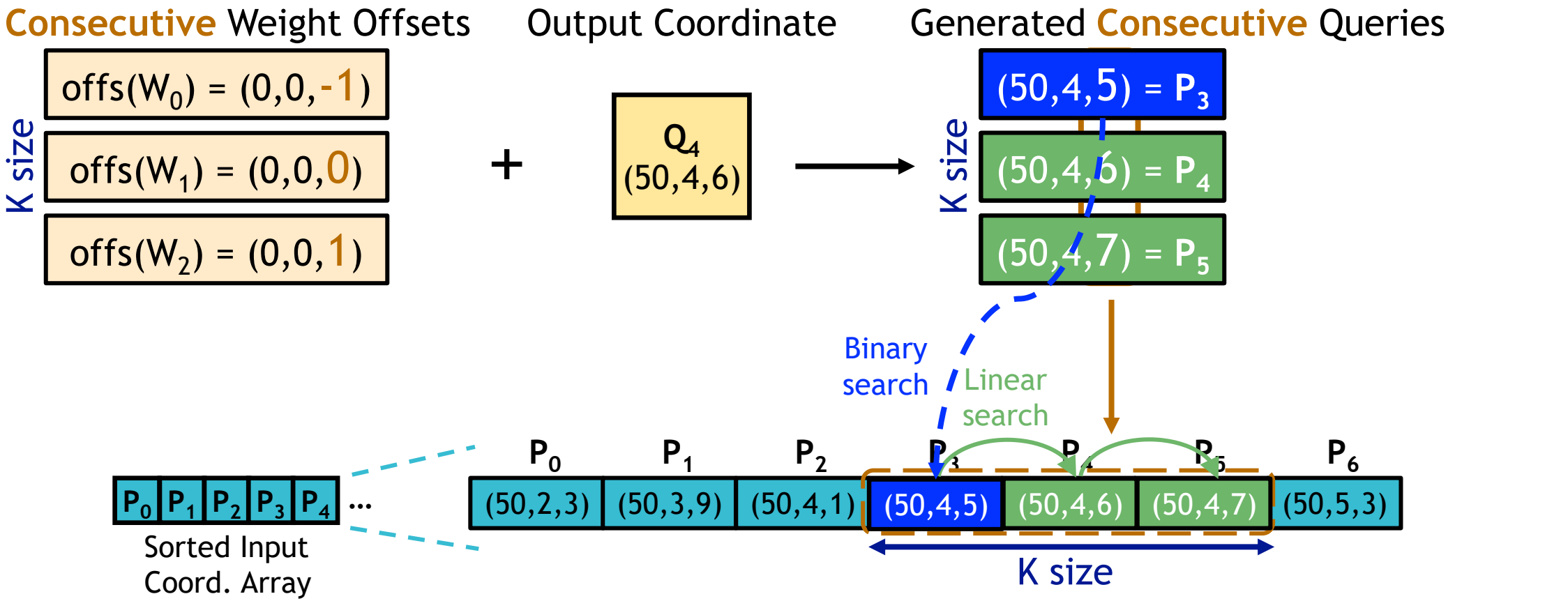
offs(W_0), offs(W_1), offs(W_2): identical x, y values + ***consecutive*** z values

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

Consecutive weight offsets in z-axis produce **consecutive** queries

One-Shot Z-Delta Search Mapping

Integer Property



. . . → binary search

. . . → linear search

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

SPIN

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
- ✓ ~K× less binary searches

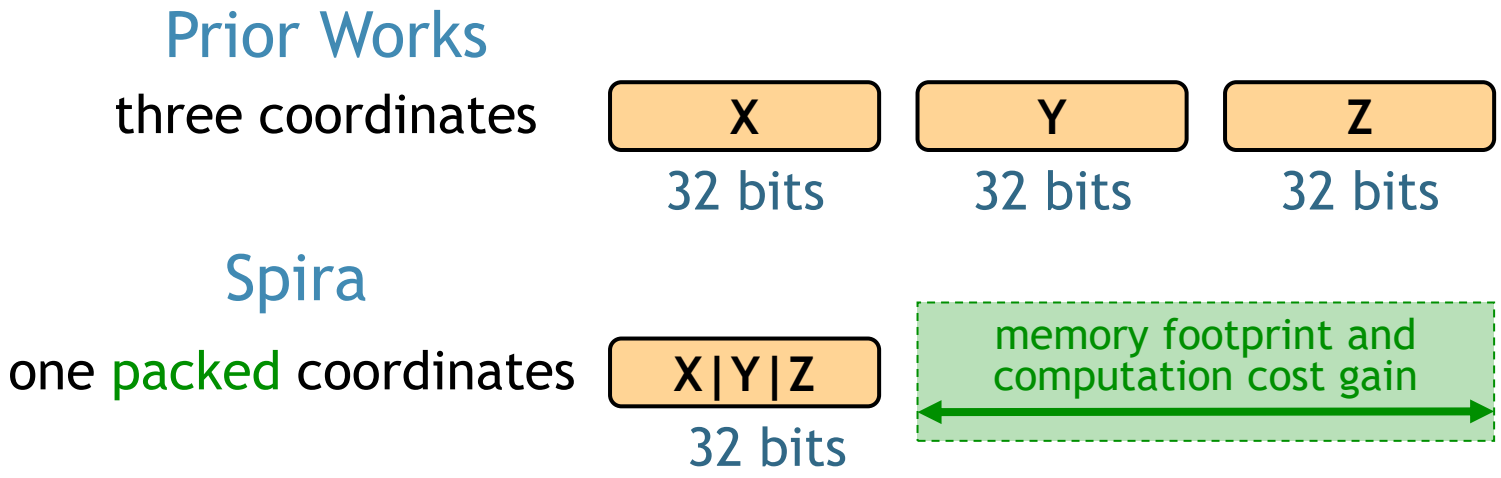
Packed-Native Mapping Step

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



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
- ✓ ~K× less binary searches

Packed-Native Mapping Step

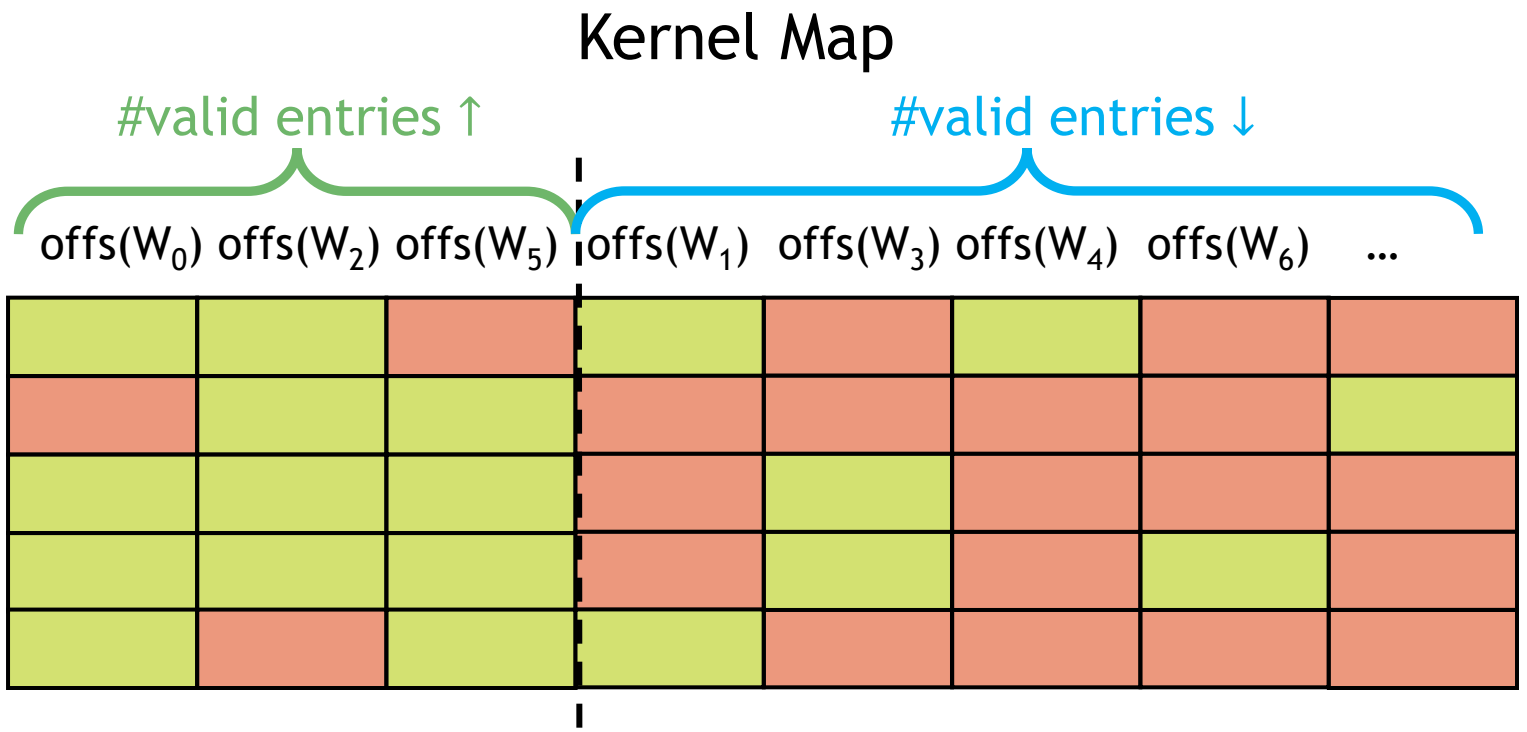
- ✓ ~3× less #comparisons
- ✓ ~3× less #memory accesses

Adaptive Hybrid Dataflow Execution

Network-Wide Mapping

Adaptive Hybrid Dataflow Execution

Neighboring Property

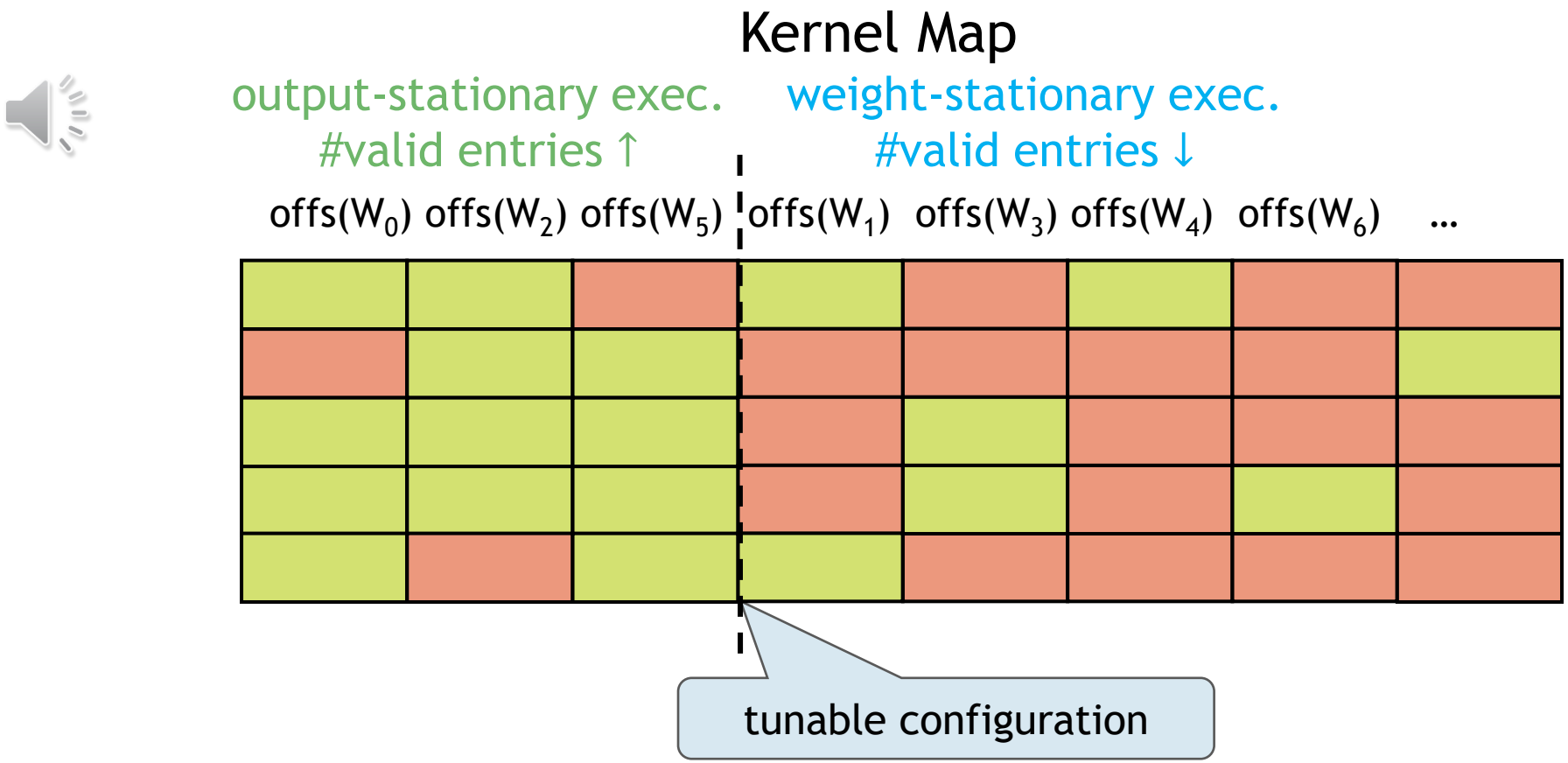


Columns associated to smaller offset displacements have larger #valid entries in the Kernel Map

Adaptive Hybrid Dataflow Execution

Neighboring Property

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



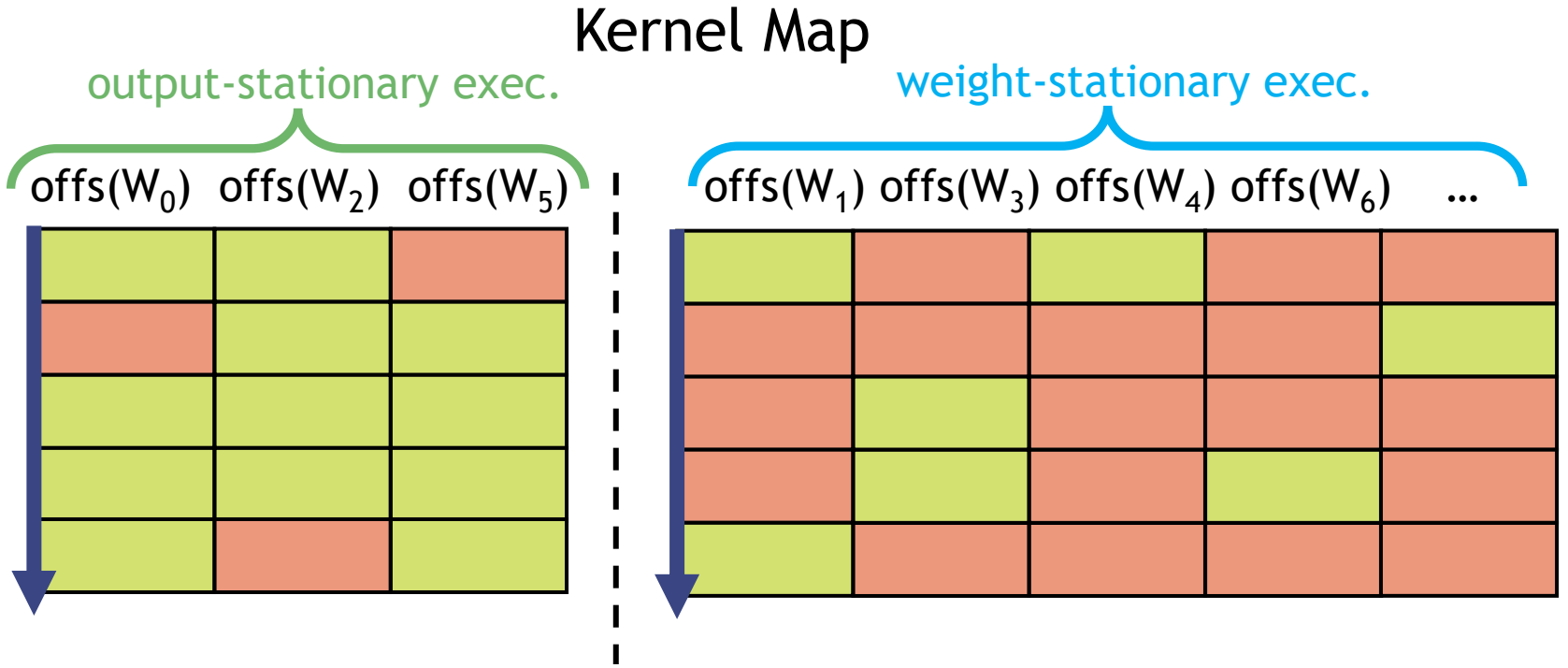
Dataflow-aware Kernel Map Post-Processing

Three possible dataflow scenarios:

Dataflow-aware Kernel Map Post-Processing

Three possible dataflow scenarios:

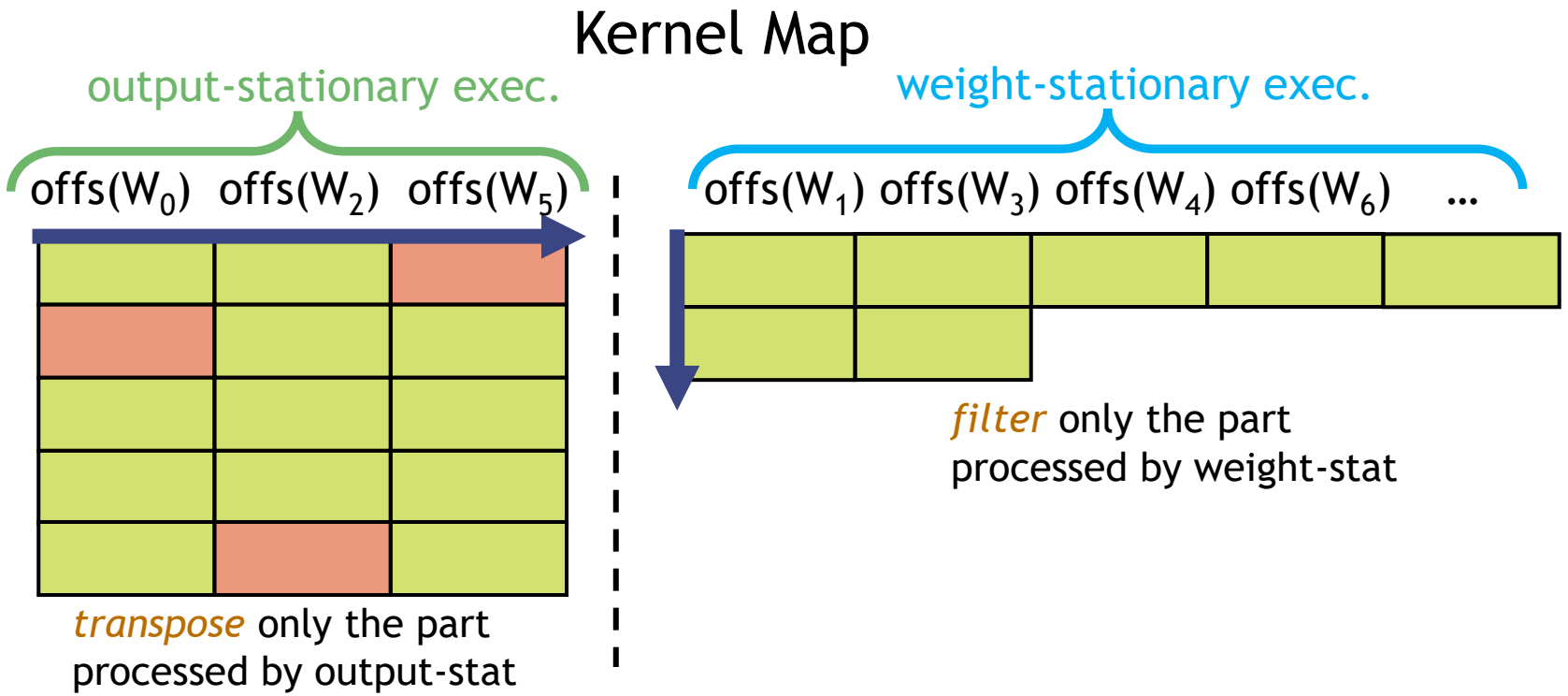
3. Hybrid-Dual: directly materialized as column-major



Dataflow-aware Kernel Map Post-Processing

Three possible dataflow scenarios:

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

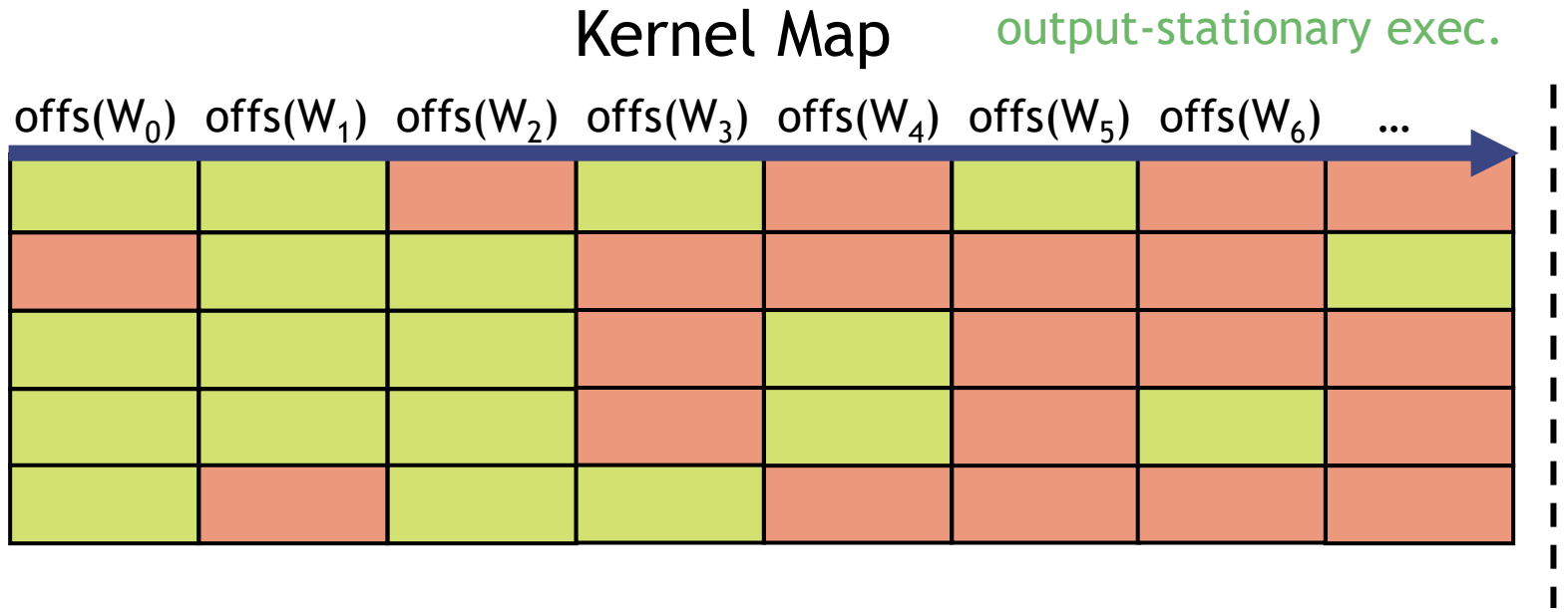


Dataflow-aware Kernel Map Post-Processing

Three possible dataflow scenarios:

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

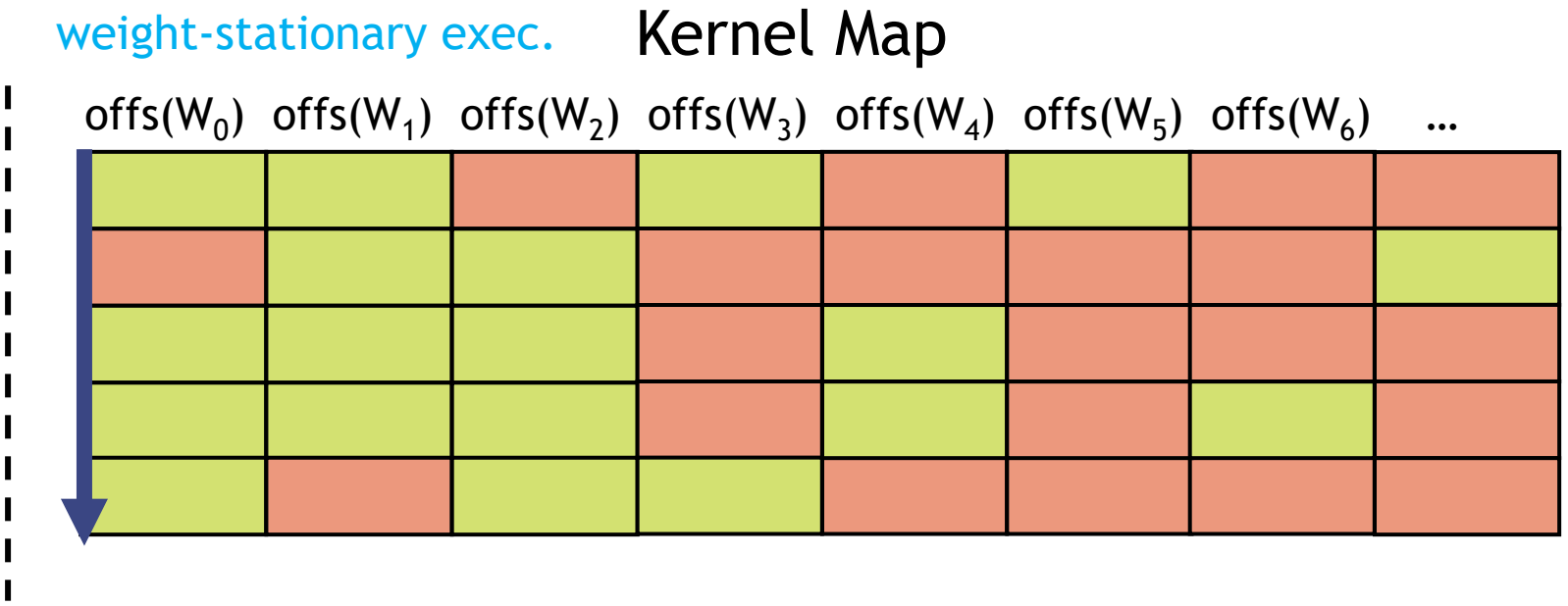
3. Hybrid-Dual: directly materialized as column-major



Dataflow-aware Kernel Map Post-Processing

Three possible dataflow scenarios:

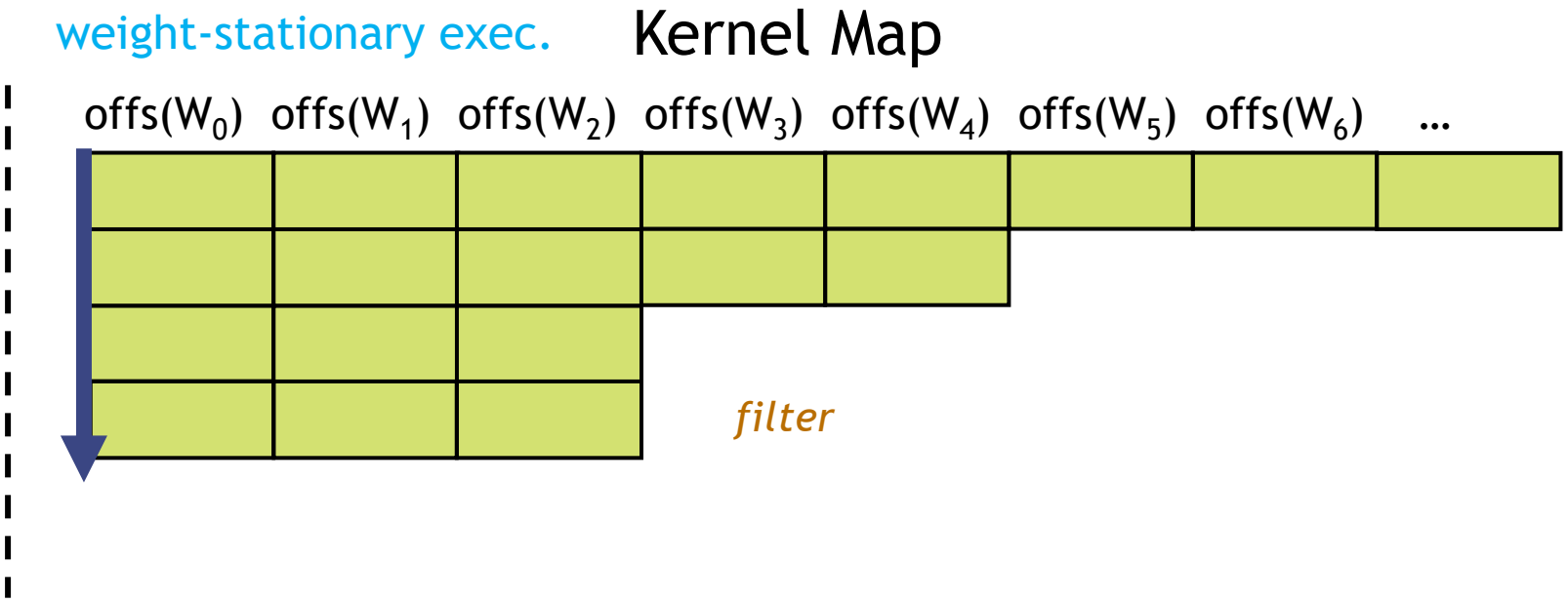
1. Pure Output-Stat: directly materialized as row-major → no post-processing
2. Pure Weight-Stat: directly materialized as column-major →
3. Hybrid-Dual: directly materialized as column-major



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
- ✓ ~K× less binary searches

Packed-Native Mapping Step

- ✓ ~3× less #comparisons
- ✓ ~3× less #memory accesses

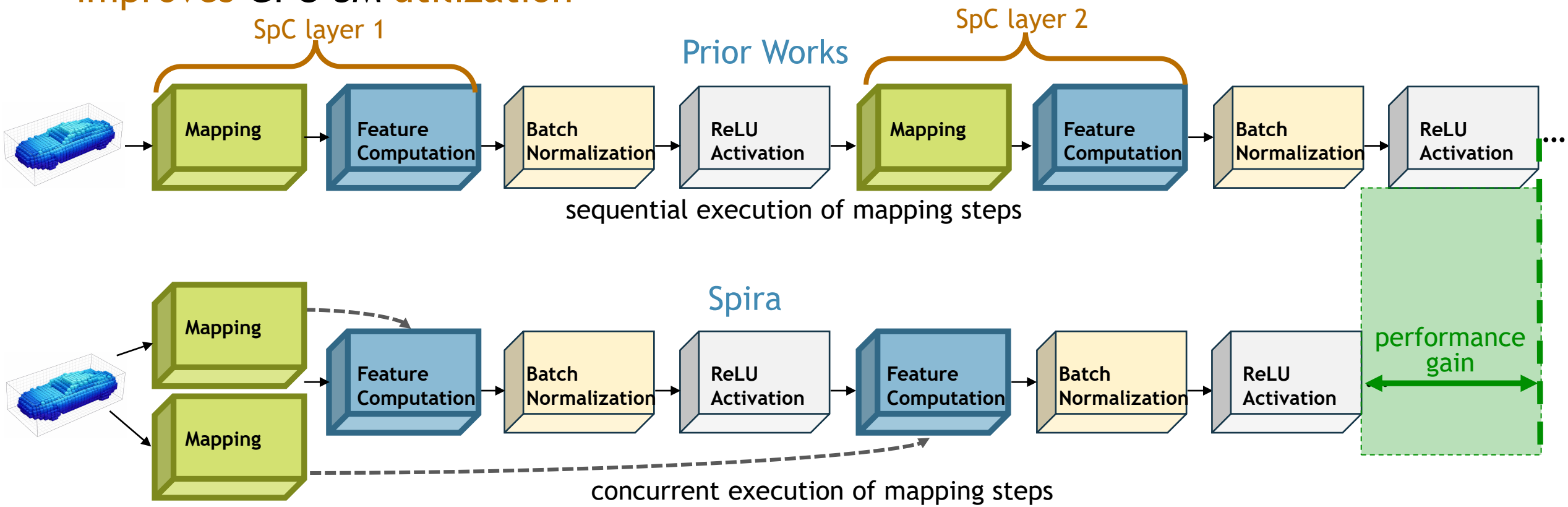
Adaptive Hybrid Dataflow Execution

- ✓ 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 ✓ ~K× less binary searches
Packed-Native Mapping Step	✓ ~3× less #comparisons ✓ ~3× 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

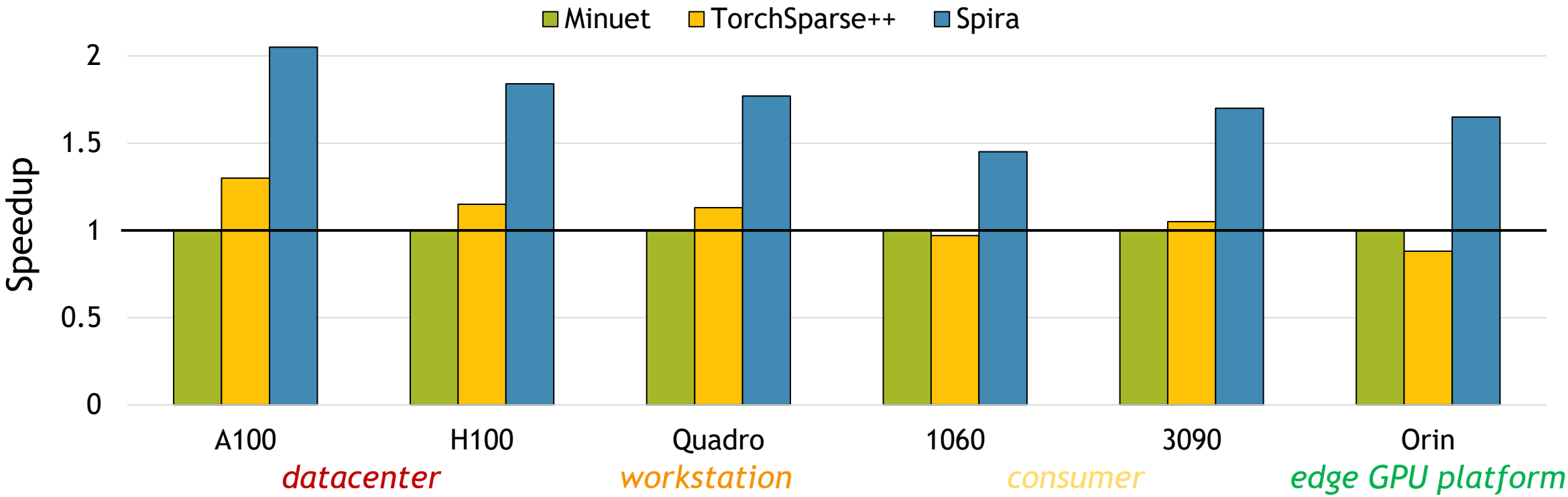
- Background
- Prior Works
- Voxel Data Properties
- Spira Design
- **Evaluation**

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

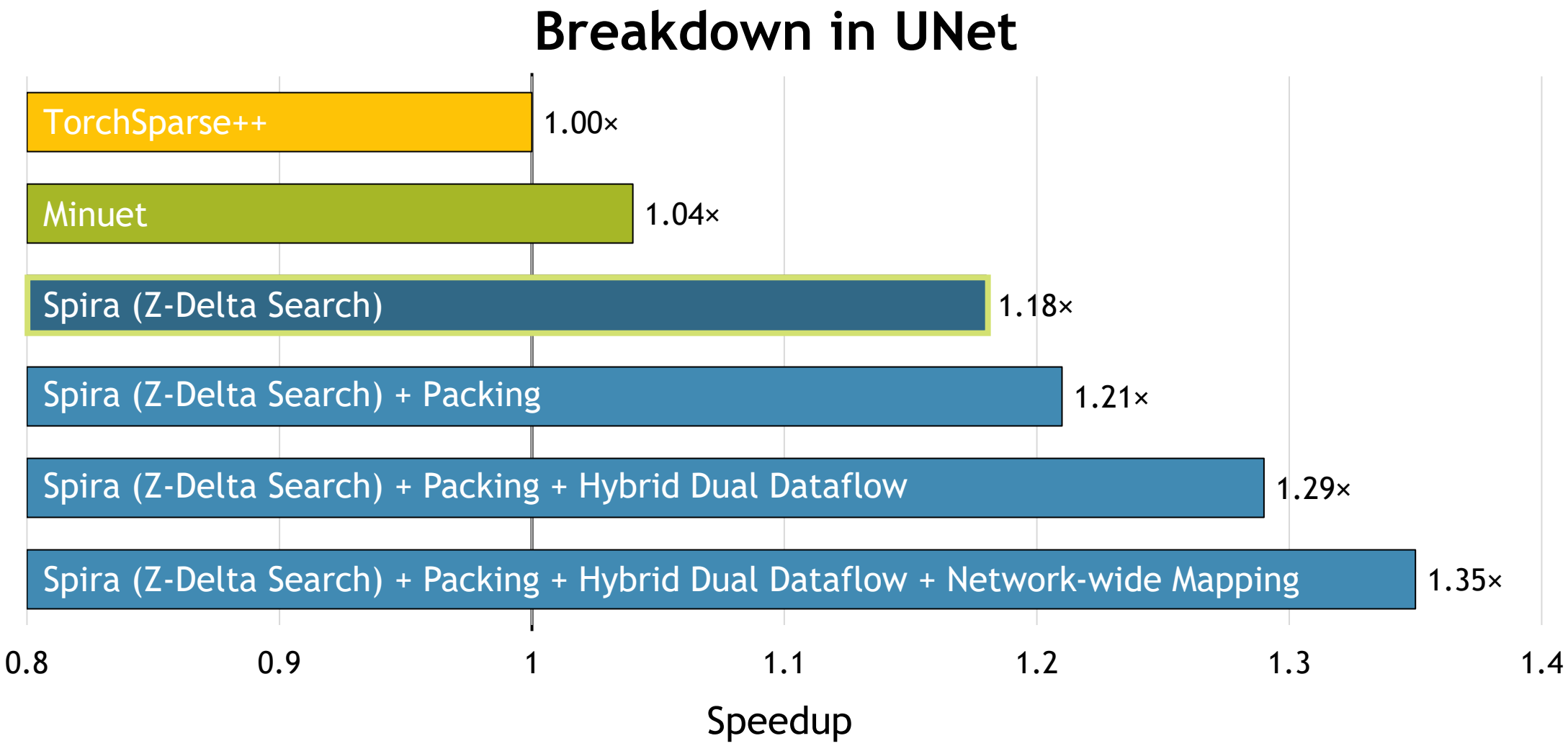
Inference Performance across Multiple GPUs

Averaged inference results across 3 networks and 3 datasets



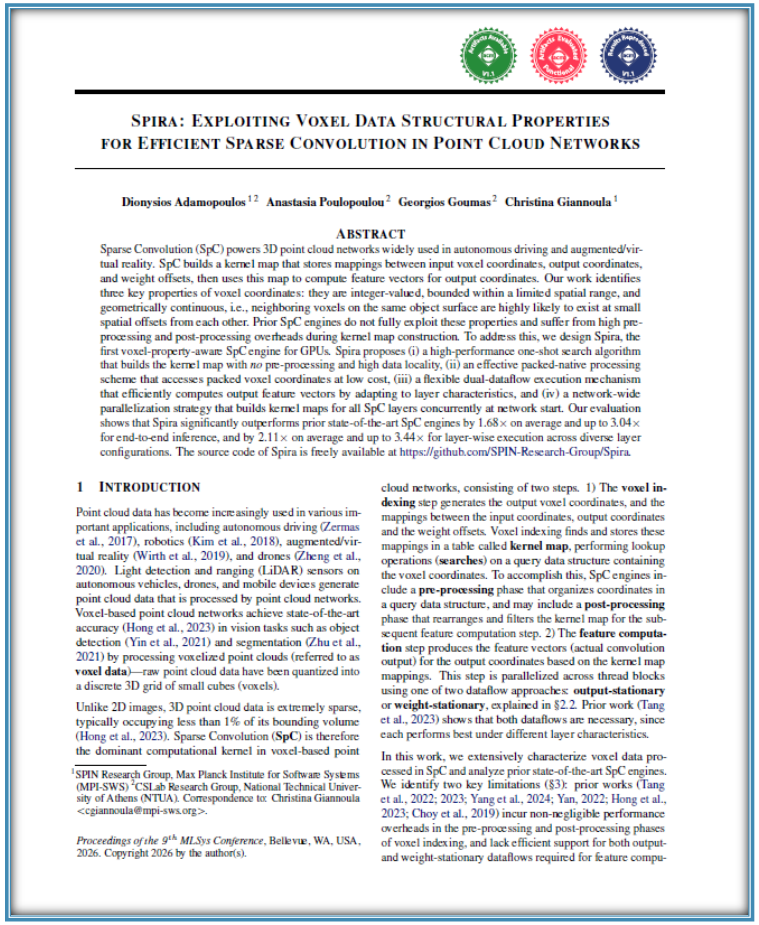
Spira significantly outperforms prior SpC engines by 1.68× averaged across six GPUs

Breakdown of Spira's Key Ideas



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



SPIRA: EXPLOITING VOXEL DATA STRUCTURAL PROPERTIES FOR EFFICIENT SPARSE CONVOLUTION IN POINT CLOUD NETWORKS

Dionysios Adamopoulos [1 2] Anastasia Poulopoulou [2] Georgios Goumas [2] Christina Giannoula [1]

ABSTRACT

Sparse Convolution (SpC) powers 3D point cloud networks widely used in autonomous driving and augmented/virtual reality. SpC builds a kernel map that stores mappings between input voxel coordinates, output coordinates, and weight offsets, then uses this map to compute feature vectors for output coordinates. Our work identifies three key properties of voxel coordinates: they are integer-valued, bounded within a limited spatial range, and geometrically continuous, i.e., neighboring voxels on the same object surface are highly likely to exist at small spatial offsets from each other. Prior SpC engines do not fully exploit these properties and suffer from high pre-processing and post-processing overheads during kernel map construction. To address this, we design Spira, the first voxel-property-aware SpC engine for GPUs. Spira proposes (i) a high-performance one-shot search algorithm that builds the kernel map with *no* pre-processing and high data locality, (ii) an effective packed-native processing scheme that accesses packed voxel coordinates at low cost, (iii) a flexible dual-dataflow execution mechanism that efficiently computes output feature vectors by adapting to layer characteristics, and (iv) a network-wide parallelization strategy that builds kernel maps for all SpC layers concurrently at network start. Our evaluation shows that Spira significantly outperforms prior state-of-the-art SpC engines by 1.68× on average and up to 3.04× for end-to-end inference, and by 2.11× on average and up to 3.44× for layer-wise execution across diverse layer configurations. The source code of Spira is freely available at https://github.com/SPIN-Research-Group/Spira.

1 INTRODUCTION

Point cloud data has become increasingly used in various important applications, including autonomous driving (Zermas et al., 2017), robotics (Kim et al., 2018), augmented/virtual reality (Wirth et al., 2019), and drones (Zheng et al., 2020). Light detection and ranging (LiDAR) sensors on autonomous vehicles, drones, and mobile devices generate point cloud data that is processed by point cloud networks. Voxel-based point cloud networks achieve state-of-the-art accuracy (Hong et al., 2023) in vision tasks such as object detection (Yin et al., 2021) and segmentation (Zhu et al., 2021) by processing voxelized point clouds (referred to as **voxel data**)—raw point cloud data have been quantized into a discrete 3D grid of small cubes (voxels).

Unlike 2D images, 3D point cloud data is extremely sparse, typically occupying less than 1% of its bounding volume (Hong et al., 2023). Sparse Convolution (**SpC**) is therefore the dominant computational kernel in voxel-based point cloud networks, consisting of two steps. 1) The **voxel indexing** step generates the output voxel coordinates, and the mappings between the input coordinates, output coordinates and the weight offsets. Voxel indexing finds and stores these mappings in a table called **kernel map**, performing lookup operations (**searches**) on a query data structure containing the voxel coordinates. To accomplish this, SpC engines include a **pre-processing** phase that organizes coordinates in a query data structure, and may include a **post-processing** phase that rearranges and filters the kernel map for the subsequent feature computation step. 2) The **feature computation** step produces the feature vectors (actual convolution output) for the output coordinates based on the kernel map mappings. This step is parallelized across thread blocks using one of two dataflow approaches: **output-stationary** or **weight-stationary**, explained in §2.2. Prior work (Tang et al., 2023) shows that both dataflows are necessary, since each performs best under different layer characteristics.

In this work, we extensively characterize voxel data processed in SpC and analyze prior state-of-the-art SpC engines. We identify two key limitations (§3): prior works (Tang et al., 2022; 2023; Yang et al., 2024; Yan, 2022; Hong et al., 2023; Choy et al., 2019) incur non-negligible performance overheads in the pre-processing and post-processing phases of voxel indexing, and lack efficient support for both output- and weight-stationary dataflows required for feature compu-

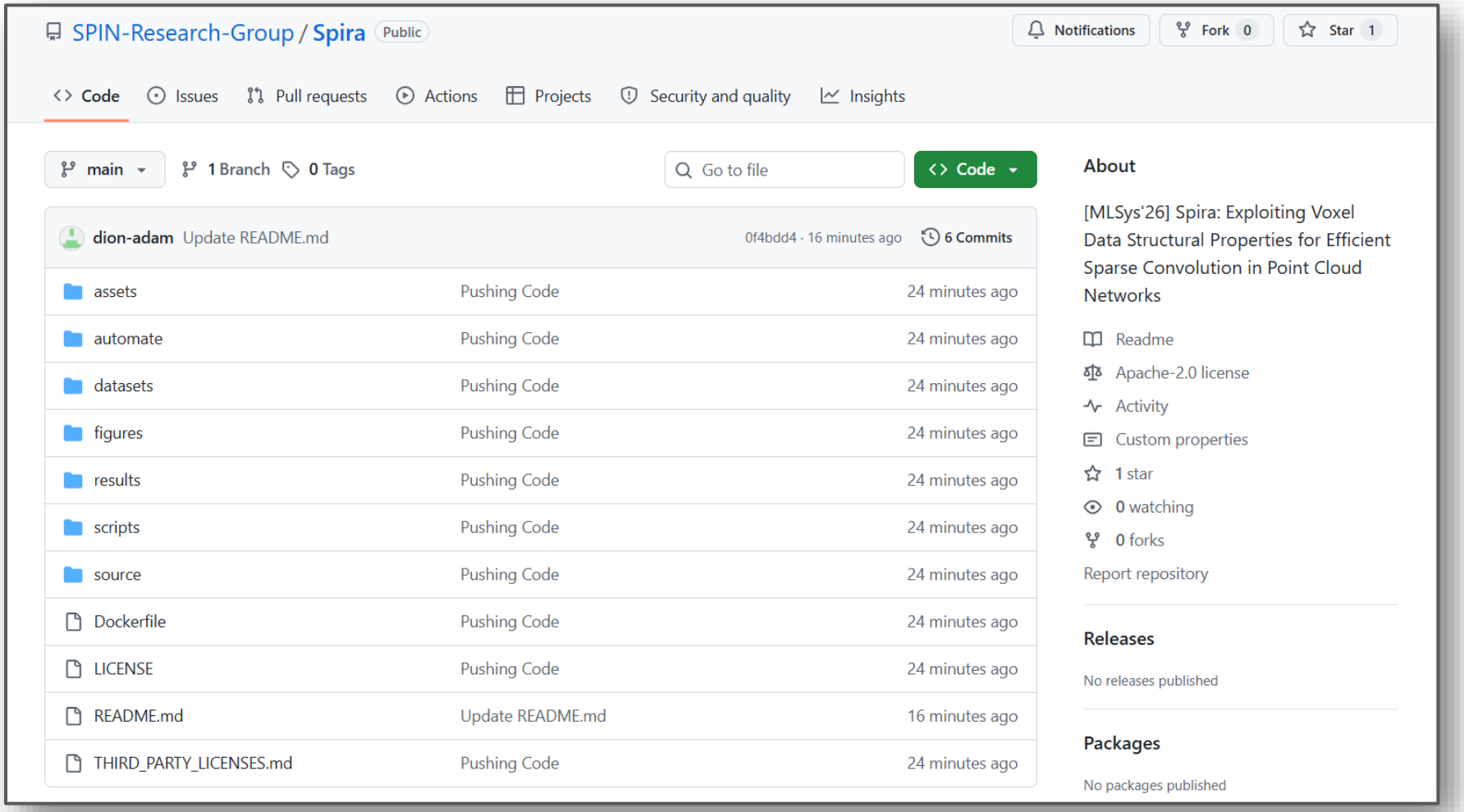
[1]SPIN Research Group, Max Planck Institute for Software Systems (MPI-SWS) [2]CSLab Research Group, National Technical University of Athens (NTUA). Correspondence to: Christina Giannoula <cgiannoula@mpi-sws.org>.

Proceedings of the 9th MLSys Conference, Bellevue, WA, USA, 2026. Copyright 2026 by the author(s).

https://arxiv.org/abs/2511.20834

Spira is Open Source

Spira



https://github.com/SPIN-Research-Group/Spira

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

Spira

GitHub

Paper

Spira

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

SPIN

MAX PLANCK INSTITUTE FOR SOFTWARE SYSTEMS

National Technical University of Athens CSLab