30th Asia and South Pacific Design Automation Conference (ASPDAC '25)

Tokyo, Japan

APTO: Accelerating Serialization-Based Point Cloud Transformers with Position-Aware Pruning

Qichu Sun, Rui Meng, Haishuang Fan, Fangqiang Ding, Linxi Lu, Jingya Wu, Xiaowei Li, Guihai Yan

1. State Key Laboratory of Processors, Institute of Computing Technology, Chinese Academy of Sciences
2. University of Chinese Academy of Sciences 3. University of Edinburgh 4. YUSUR Technology Co., Ltd.

sunqichu22z@ict.ac.cn

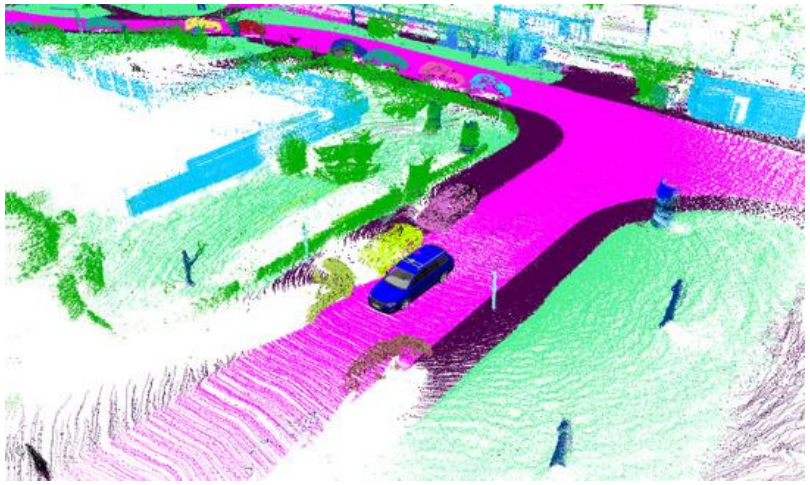
Jan. 21, 2025

SKLP 处理器芯片全国重点实验室 STATE KEY LAB OF PROCESSORS, ICT, CAS | ICT 中国科学院计算技术研究所 INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES | 中国科学院大学 University of Chinese Academy of Sciences | THE UNIVERSITY of EDINBURGH | 中科驭数 YUSUR

Point Cloud Processing

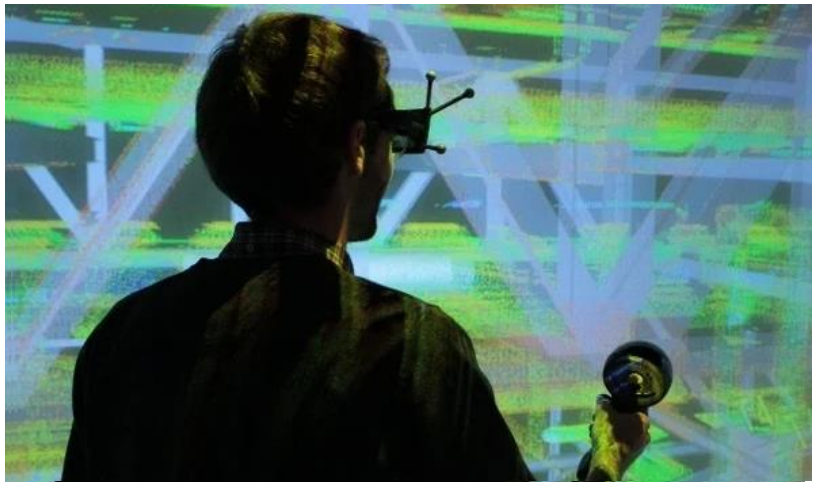
Point cloud is a representation of 3D data, containing valuable **geometry** & **color** information

Autonomous Driving



Robotic Perception

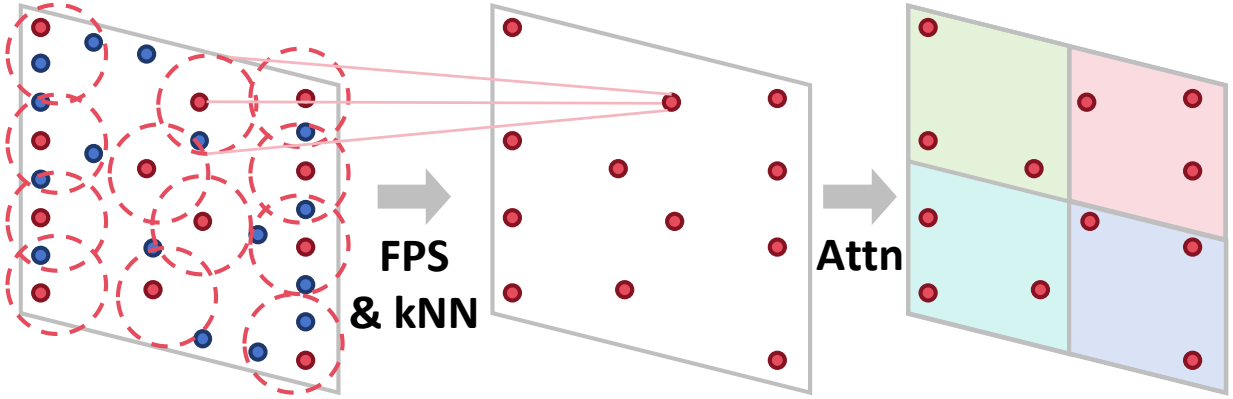
AR / VR



Accurate, real-time and energy-efficient point cloud processing is crucial

Point Cloud Transformers

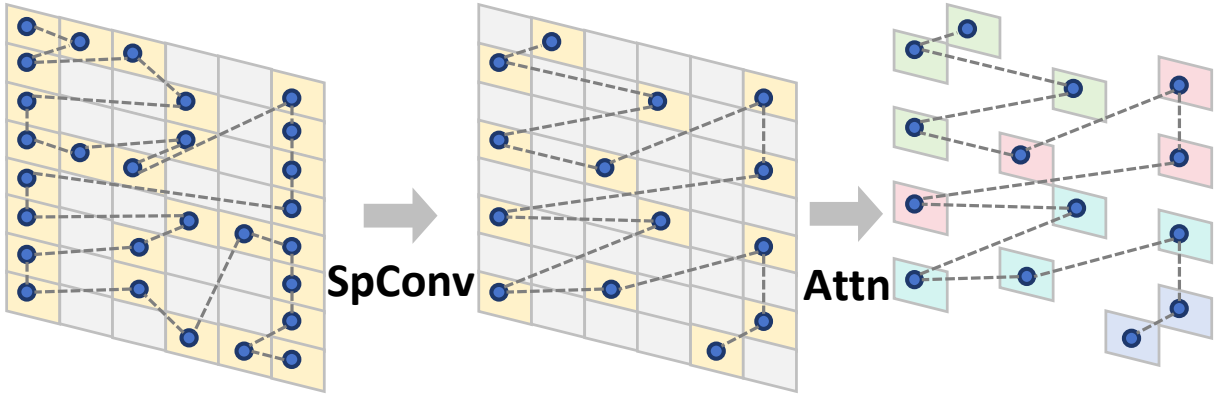
Point-Based Models



- **Farthest Point Sampling** (FPS) & **k-Nearest Neighbors** (kNN) for **down-sampling**
- **Attention** mechanisms in local windows for **feature computation**

Repeated point access leads to redundant computation & memory use, while window size restricts accuracy

Serialization-Based Models

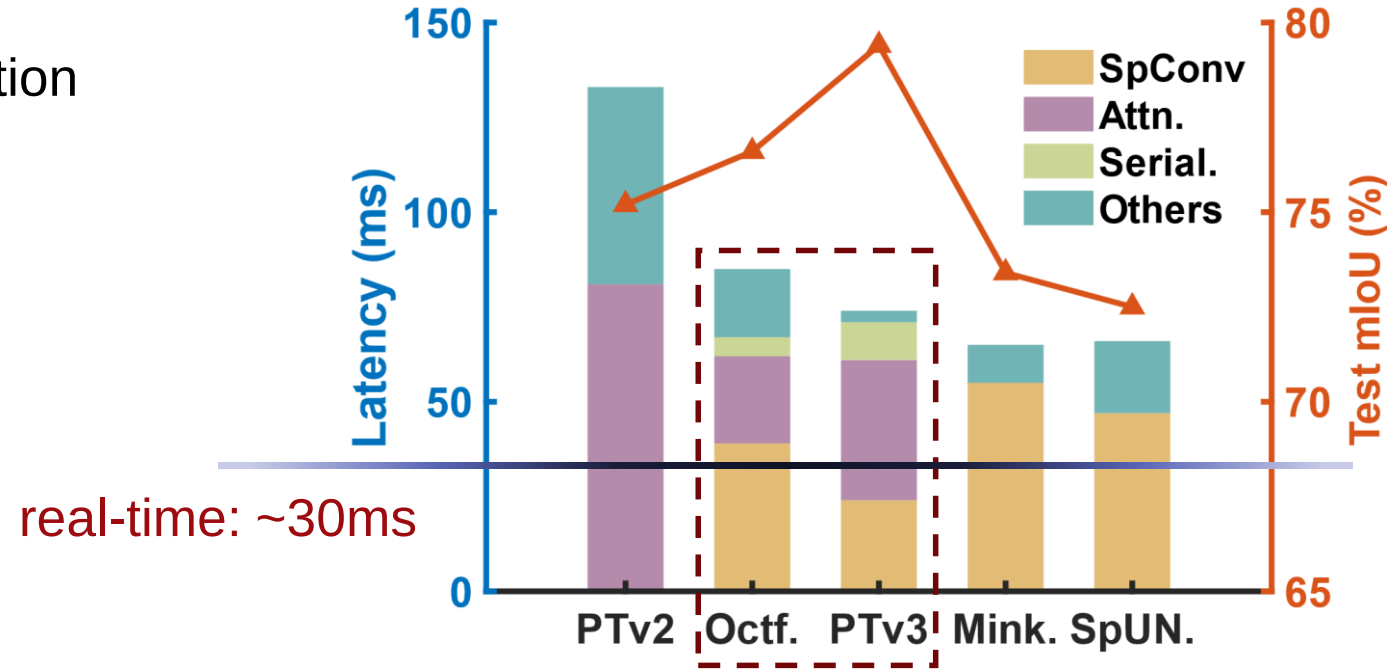


- Organizes points onto **a directed curve** (e.g. z-curve)
- **3D sparse convolution** for down-sampling
- **Larger attention windows** based on the curve

Regular memory access, less redundant computation, and better accuracy

Motivation: Performance Bottleneck Analysis

- Inference time breakdown:
  - Octformer: 46% SpConv & 27% attention
  - PTv3: 32% SpConv & 50% attention
  - Failed to meet real-time requirements



Serialization-based models

SpConv & attention are two key bottlenecks of serialization-based models

[1] Peng-Shuai Wang. 2023. Octformer: Octree-based transformers for 3d point clouds. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–11.

[2] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. 2024. Point Transformer V3: Simpler Faster Stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4840–4851.

Motivation

➢ Challenges

- Chlg.1: Inefficient Neighbor Search in SpConv
- Chlg.2: Useless Attention Computation
- Chlg.3: Data Dependencies in Softmax

➢ Optimizations

- Opt.1: Serialization-Based Parallel Neighbor Search
- Opt.2: Position-Aware Attention Pruning
- Opt.3: Fine-Grained Attention Dataflow

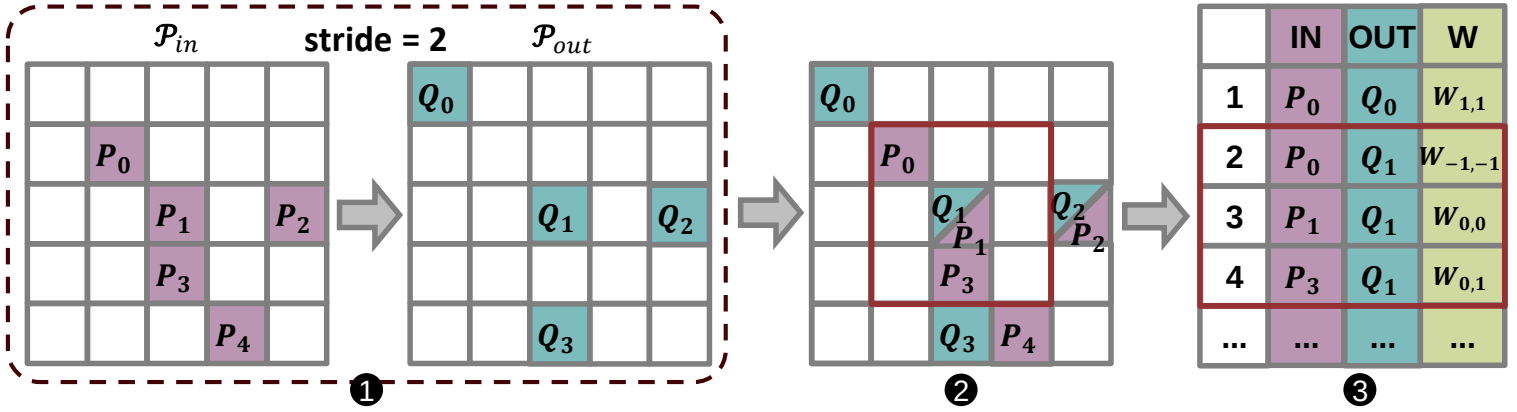
Chlg.1: Inefficient Neighbor Search in SpConv

➢ Kernel mapping in SpConv

- Stride > 1 for down-sampling
- Stride = 1 for positional encoding

❶ **Output Coordinate Generation**
out = floor (in / stride) * stride

❸ **Kernel Map** contains (in, out, weight) entries for aggregation



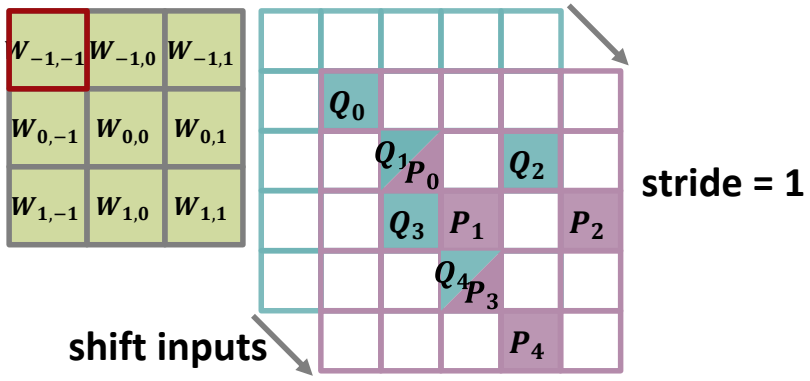
	IN	OUT	W
1	P_0	Q_0	$W_{1,1}$
2	P_0	Q_1	$W_{-1,-1}$
3	P_1	Q_1	$W_{0,0}$
4	P_3	Q_1	$W_{0,1}$
...	...	...	...

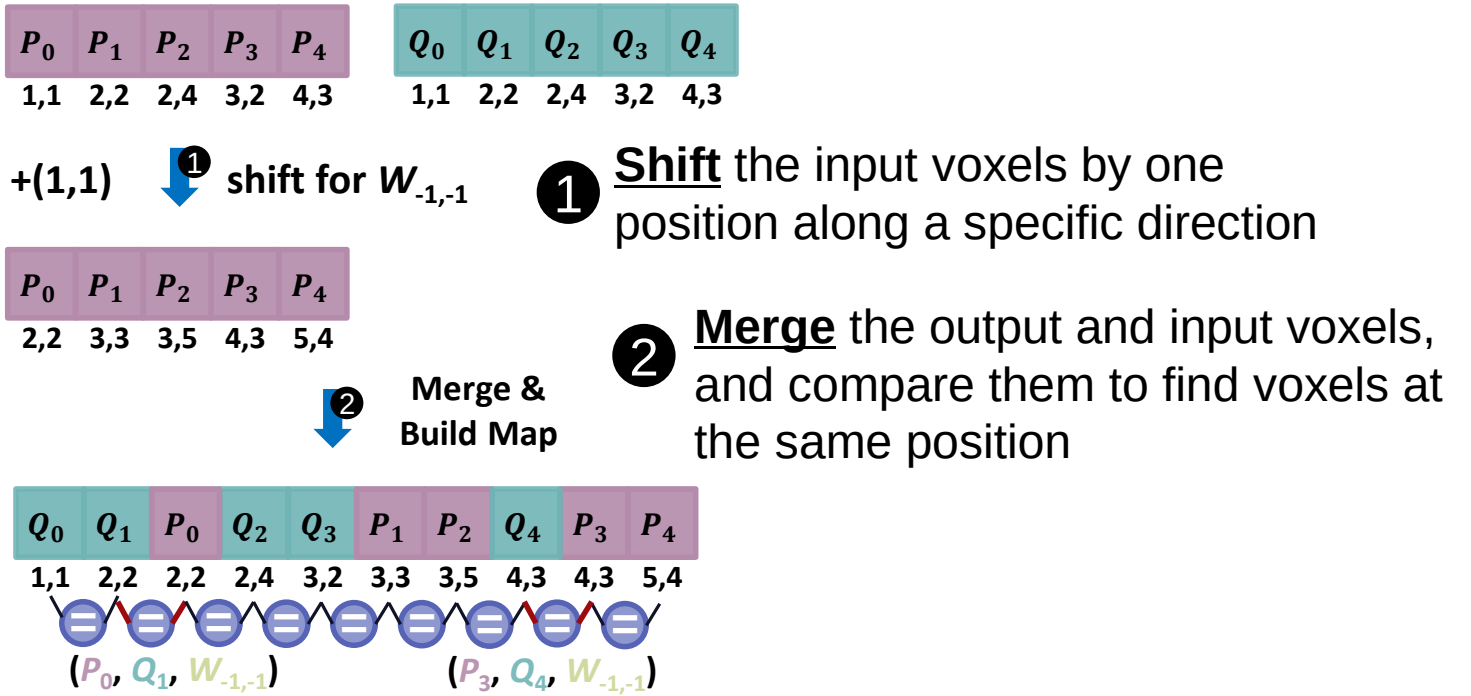
❷ **Neighbor Search** finds non-empty input voxels around the output voxel

Neighbor search is the most complex step

Chlg.1: Inefficient Neighbor Search in SpConv

- Prior works: voxel merging
  - Merge and compare all voxels (**>10^5**) serially





1. **Shift** the input voxels by one position along a specific direction
2. **Merge** the output and input voxels, and compare them to find voxels at the same position

Results in enormous operations and lacks parallelism

[1] Yujun Lin, Zhekai Zhang, Haotian Tang, Hanrui Wang, and Song Han. 2021. Pointacc: Efficient point cloud accelerator. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. 449–461.

Opt.1: Serialization-Based Parallel Neighbor Search

- Z-curve transform unstructured point clouds into ordered sequences

voxel coordinate **(x, y, z)**

$x = \{x_{n-1}\cdots x_1 x_0\}_2$

$y = \{y_{n-1}\cdots y_1 y_0\}_2$

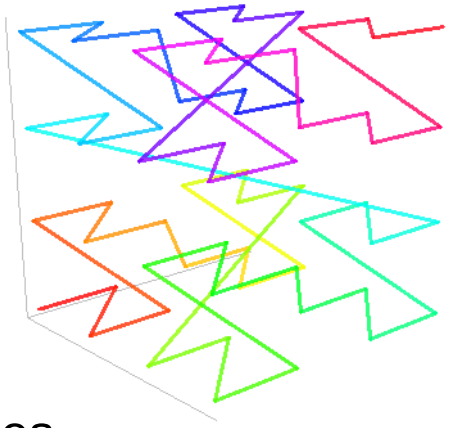
$z = \{z_{n-1}\cdots z_1 z_0\}_2$

combine x, y, z binaries bit-wise

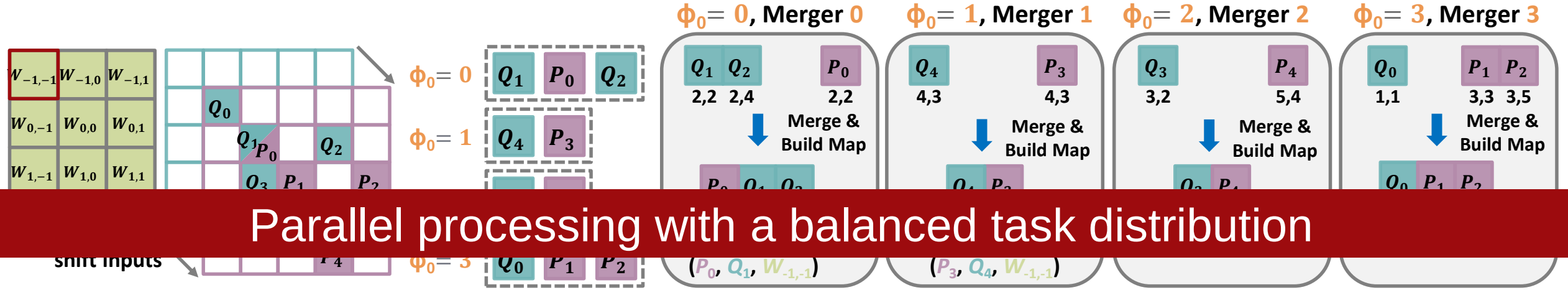
z-curve index Φ

$\Phi = \{\phi_{n-1}\cdots\phi_1\phi_0\}_8$

$= \{z_{n-1}y_{n-1}x_{n-1}\cdots z_1 y_1 x_1 z_0 y_0 x_0\}_2$



- Z-curve enables parallel search for voxels
  - Organize voxels into 8 sets based on **the last 3 bits** ϕ_0 of their z-curve indices
  - Each set performs merging and comparison **in parallel**

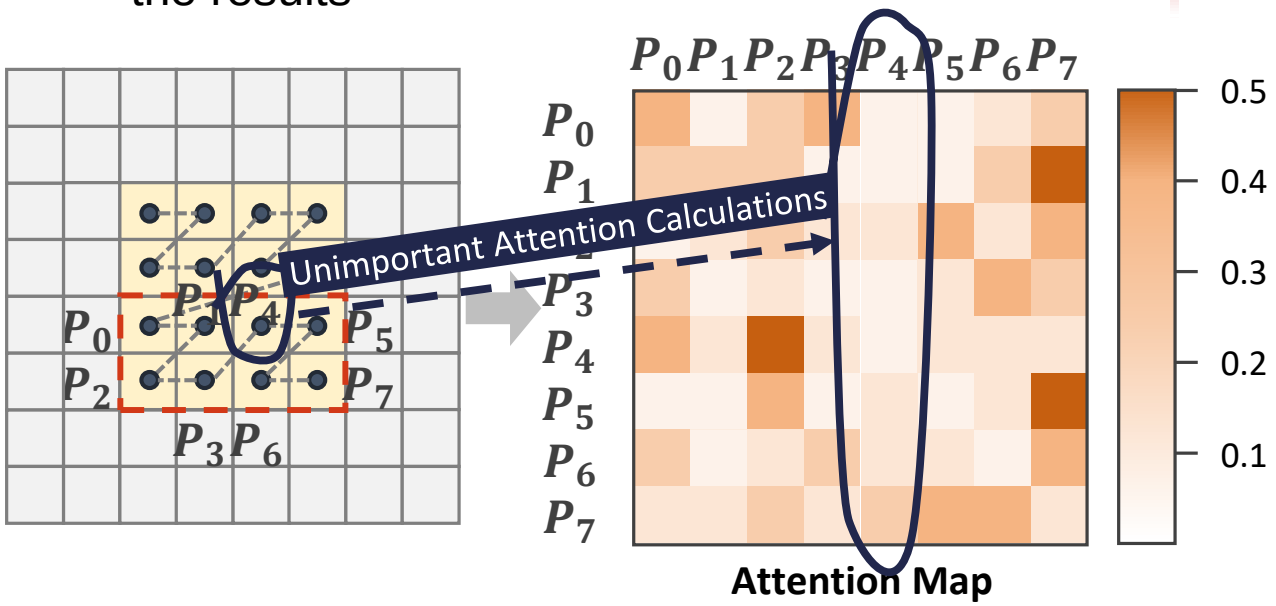


Parallel processing with a balanced task distribution

Chlg.2: Useless Attention Computation

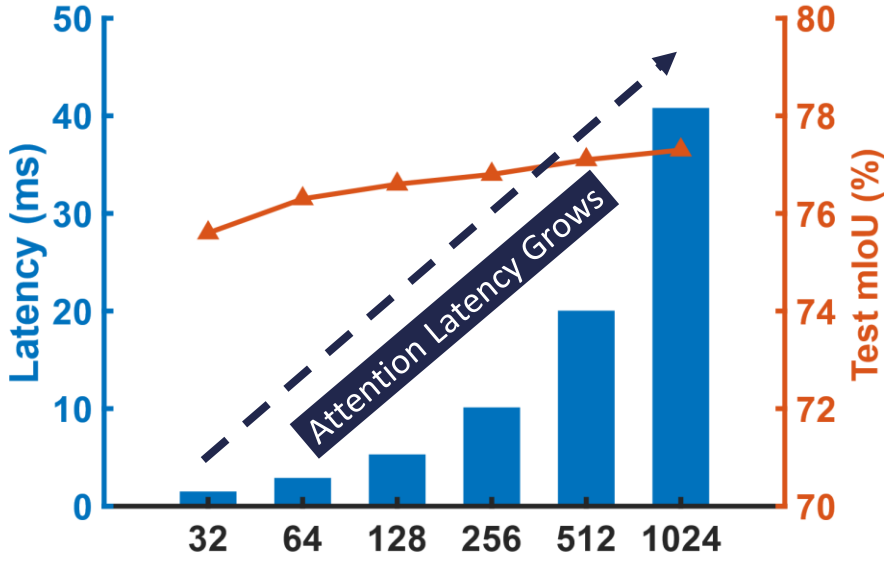
➢ Attention in local windows

- Each voxel engages in attention calculations with every other voxel
- **Unimportant voxels** have minimal impact on the results



➢ The increased window size

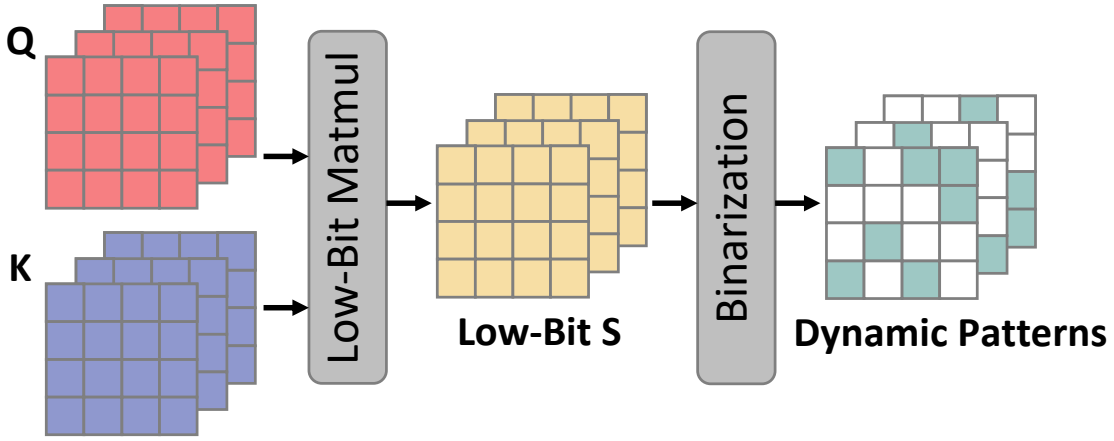
- A fixed point cloud size *N* and a window size *w*, the overall complexity is ***O(Nw)***
- Grows rapidly as *w* increases



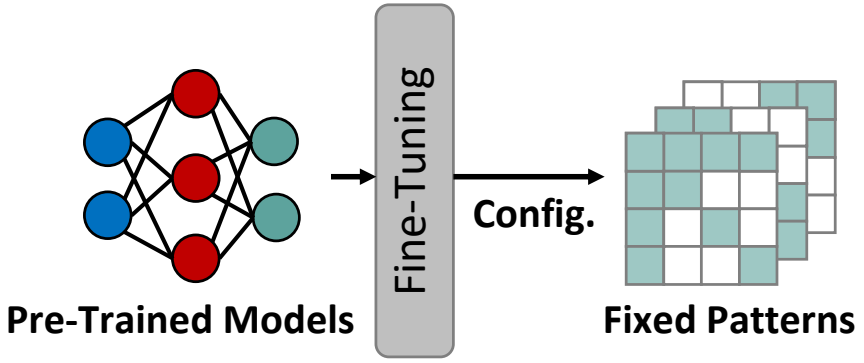
Attention suffers from significant useless computation

Chlg.2: Useless Attention Computation

- Dynamic pruning method
  - Pre-calculate a low-precision score matrix
  - Additional **computation & storage overhead**



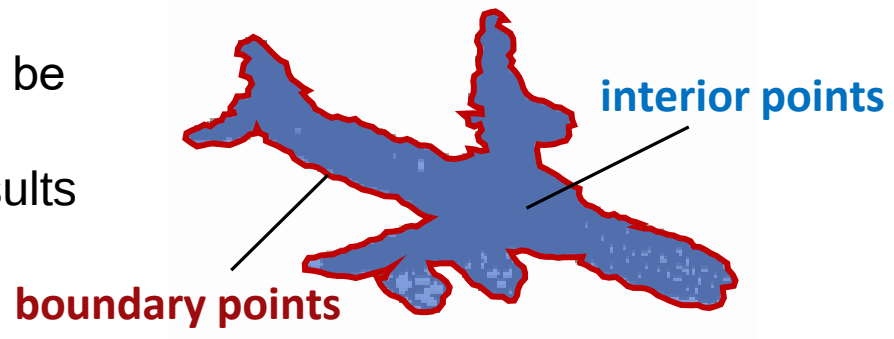
- Static pruning method
  - Adopt fixed pruning patterns
  - Significant **accuracy loss**



Both methods cannot suit serialization-based models

Opt.2: Position-Aware Attention Pruning

- Boundaries are more crucial than interiors
  - Once an object's boundary is identified, it becomes feasible to be segmented from the background
  - Pruning **tokens from interior points** minimally affects the results

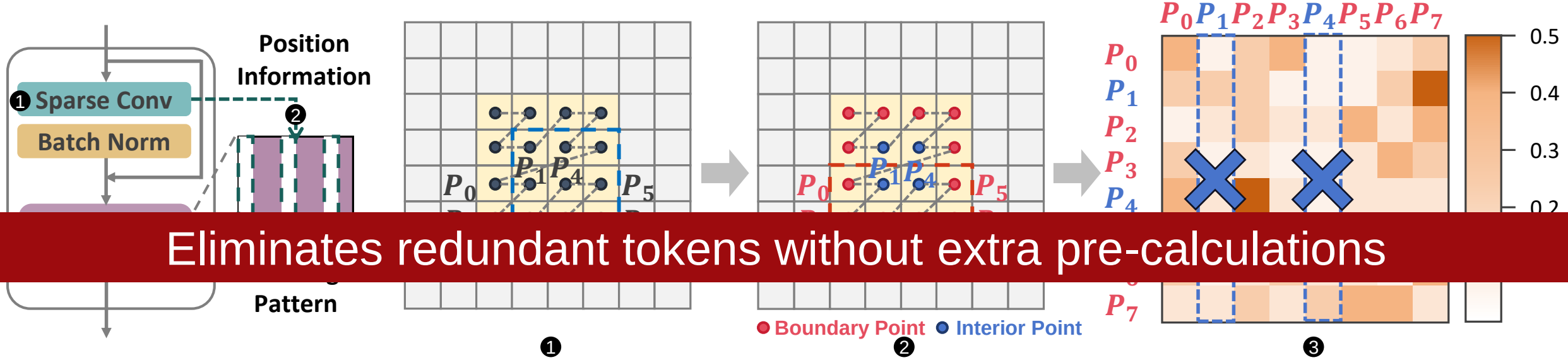


- Get neighbor counts from neighbor search in SpConv
  - **Boundary points**: voxels with fewer neighbors
  - **Interior points**: voxels with **more neighbors**

❶ Count **neighbor numbers** in SpConv

❷ Divide points into **boundary** & **interior** points

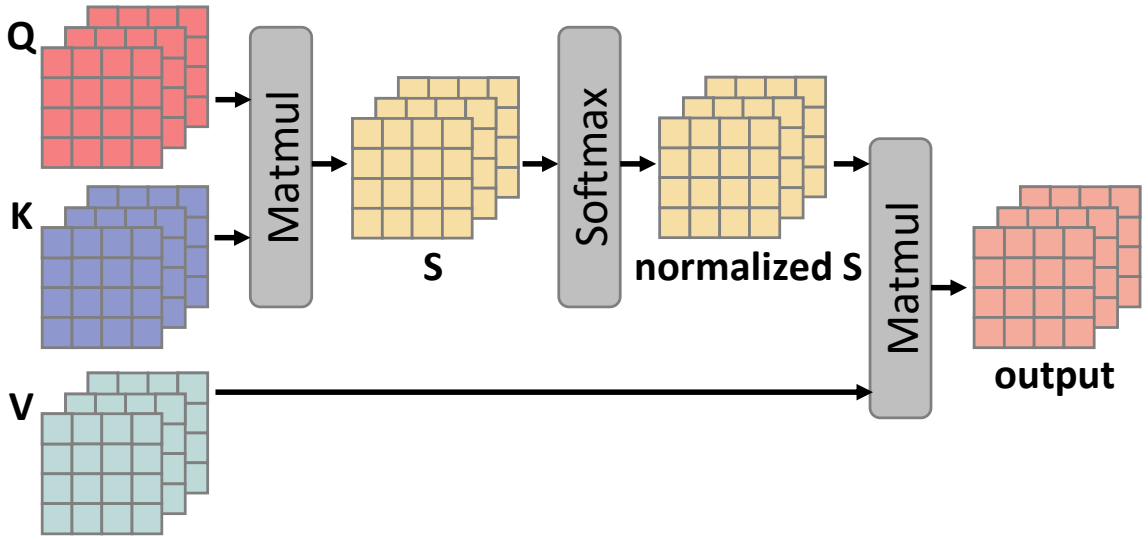
❸ Prune tokens generated from **interior points**



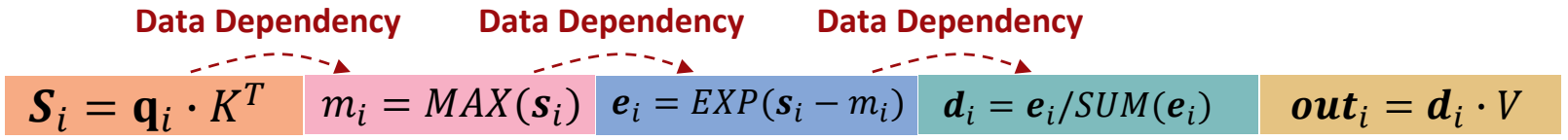
Eliminates redundant tokens without extra pre-calculations

Chlg.3: Data Dependencies in Softmax

- Avoid numerical overflow in exponentials
  - Subtract m_i from each element in **S**
  - m_i is the **maximum value** of each row in **S**
- Maximum value causes data dependencies
  - m_i calculated after all k_j multiplications
  - Exponential calculation waits for m_i
  - Exponential sum computed before Matmul with **V**



$$output_i = softmax(\boldsymbol{S}_i) \cdot \boldsymbol{V} = \frac{\sum_{j=1}^{n} exp(s_{ij} - m_i)\boldsymbol{v}_j}{\sum_{j=1}^{n} exp(s_{ij} - m_i)}$$

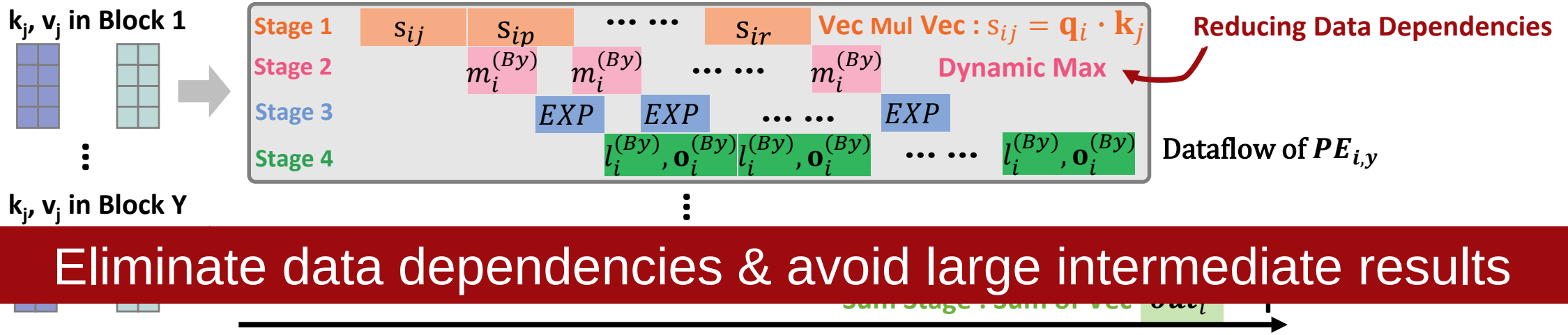


Inhibit efficient pipeline for computation of different kj, vj

Opt.3: Fine-Grained Attention Dataflow

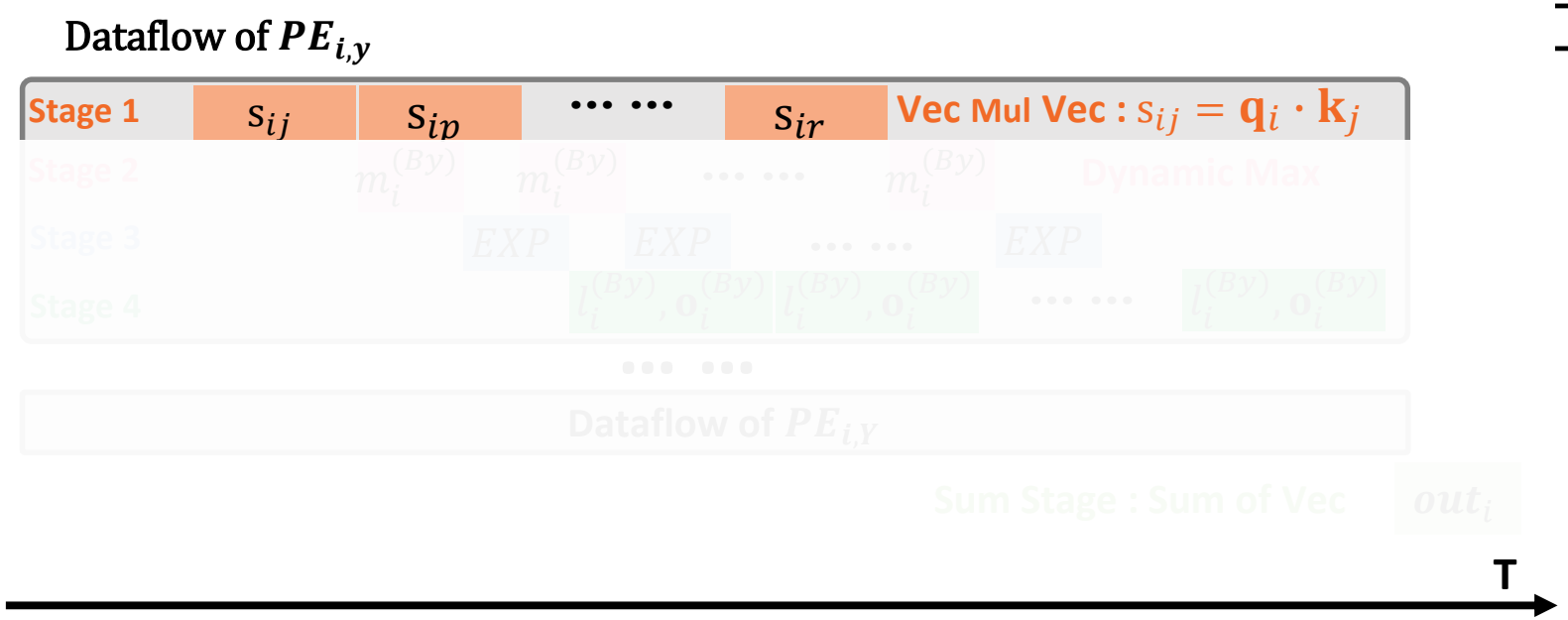
- Use dynamic maximum value m_i
  - Decouble the computation of each $\mathbf{k}_j$, $\mathbf{v}_j$ in **K**, **V**
  - Use the m_i of the scores s_{ij} already calculated

- Block-wise parallel attention calculation
  - Partition keys & values into blocks along **the sequence dimension**
  - Parallelize attention computation within different blocks



Eliminate data dependencies & avoid large intermediate results

Opt.3: Fine-Grained Attention Dataflow



Algorithm: Fine-Grained Attention Mechanism

INPUT: $\mathbf{q_1}, ..., \mathbf{q_n}, \mathbf{k_1}, ..., \mathbf{k_n}$ and $\mathbf{v_1}, ..., \mathbf{v_n}$ of size $1 \times d_k$ as rows of $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ of size $n \times d_k$

OUTPUT: $\mathbf{output_1}, ..., \mathbf{output_n}$ of size $1 \times d_k$ as rows of attention output matrix of size $n \times d_k$

Parallel for i **in** *range(n)* **do**

  Parallel for B_y **in** $\{B_1, B_2, ..., B_Y\}$ **do**

    Initialize $\mathbf{o}_i^{(B_y)} = (0)_{1 \times d_k}$, $l_i^{(B_y)} = 0$, $m'^{(B_y)}_i = -\infty$

    for j **in** B_y **do**

      $s_{ij} = \mathbf{q}_i \cdot \mathbf{k}_j^T$

      $m_i^{(B_y)} = max\left(m'^{(B_y)}_i, s_{ij}\right)$

      $\tilde{s}_{ij} = exp\left(s_{ij} - m_i^{(B_y)}\right)$

      $\tilde{m}_i^{(B_y)} = exp\left(m'^{(B_y)}_i - m_i^{(B_y)}\right)$

      $l_i^{(B_y)} = \tilde{m}_i^{(B_y)} l_i^{(B_y)} + \tilde{s}_{ij}$

      $\mathbf{o}_i^{(B_y)} = \tilde{m}_i^{(B_y)} \cdot \mathbf{o}_i^{(B_y)} + \tilde{s}_{ij} \cdot \mathbf{v}_j$

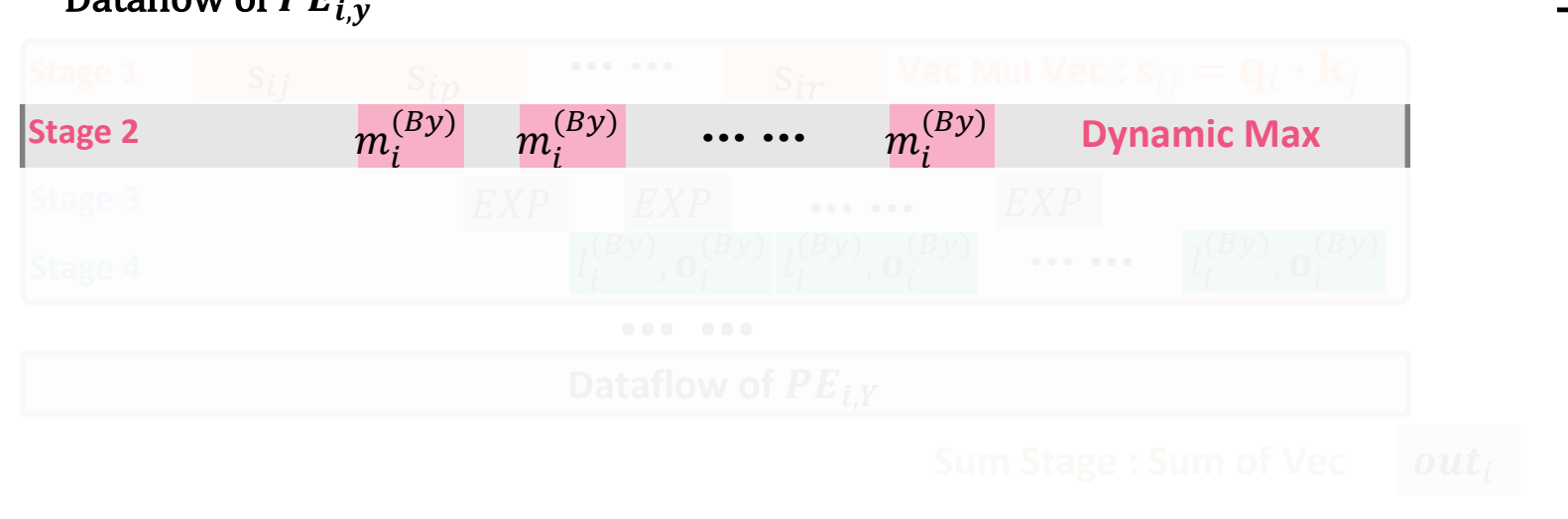
      $m'^{(B_y)}_i = m_i^{(B_y)}$

  $M_i = MAX\left(m_i^{(B_1)}, m_i^{(B_2)}, ..., m_i^{(B_Y)}\right)$

  $\mathbf{output}_i = \frac{\sum_y exp\left(m_i^{(B_y)} - M_i\right) \cdot \mathbf{o}_i^{(B_y)}}{\sum_y exp\left(m_i^{(B_y)} - M_i\right) l_i^{(B_y)}}$

- Stage 1
  - q_i and k_j are multiplied to obtain s_{ij}

Opt.3: Fine-Grained Attention Dataflow



Algorithm: Fine-Grained Attention Mechanism

INPUT: $\mathbf{q_1}, ..., \mathbf{q_n}, \mathbf{k_1}, ..., \mathbf{k_n}$ and $\mathbf{v_1}, ..., \mathbf{v_n}$ of size $1 \times d_k$ as rows of $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ of size $n \times d_k$

OUTPUT: $\mathbf{output_1}, ..., \mathbf{output_n}$ of size $1 \times d_k$ as rows of attention output matrix of size $n \times d_k$

Parallel for i **in** *range(n)* **do**

  Parallel for B_y **in** $\{B_1, B_2, ..., B_Y\}$ **do**

    Initialize $\mathbf{o}_i^{(B_y)} = (0)_{1\times d_k}$, $l_i^{(B_y)} = 0$, $m'^{(B_y)}_i = -\infty$

    for j **in** B_y **do**

      $s_{ij} = \mathbf{q}_i \cdot \mathbf{k}_j^T$

      $m_i^{(B_y)} = max\left(m'^{(B_y)}_i, s_{ij}\right)$

      $\tilde{s}_{ij} = exp\left(s_{ij} - m_i^{(B_y)}\right)$

      $\tilde{m}_i^{(B_y)} = exp\left(m'^{(B_y)}_i - m_i^{(B_y)}\right)$

      $l_i^{(B_y)} = \tilde{m}_i^{(B_y)} l_i^{(B_y)} + \tilde{s}_{ij}$

      $\mathbf{o}_i^{(B_y)} = \tilde{m}_i^{(B_y)} \cdot \mathbf{o}_i^{(B_y)} + \tilde{s}_{ij} \cdot \mathbf{v}_j$

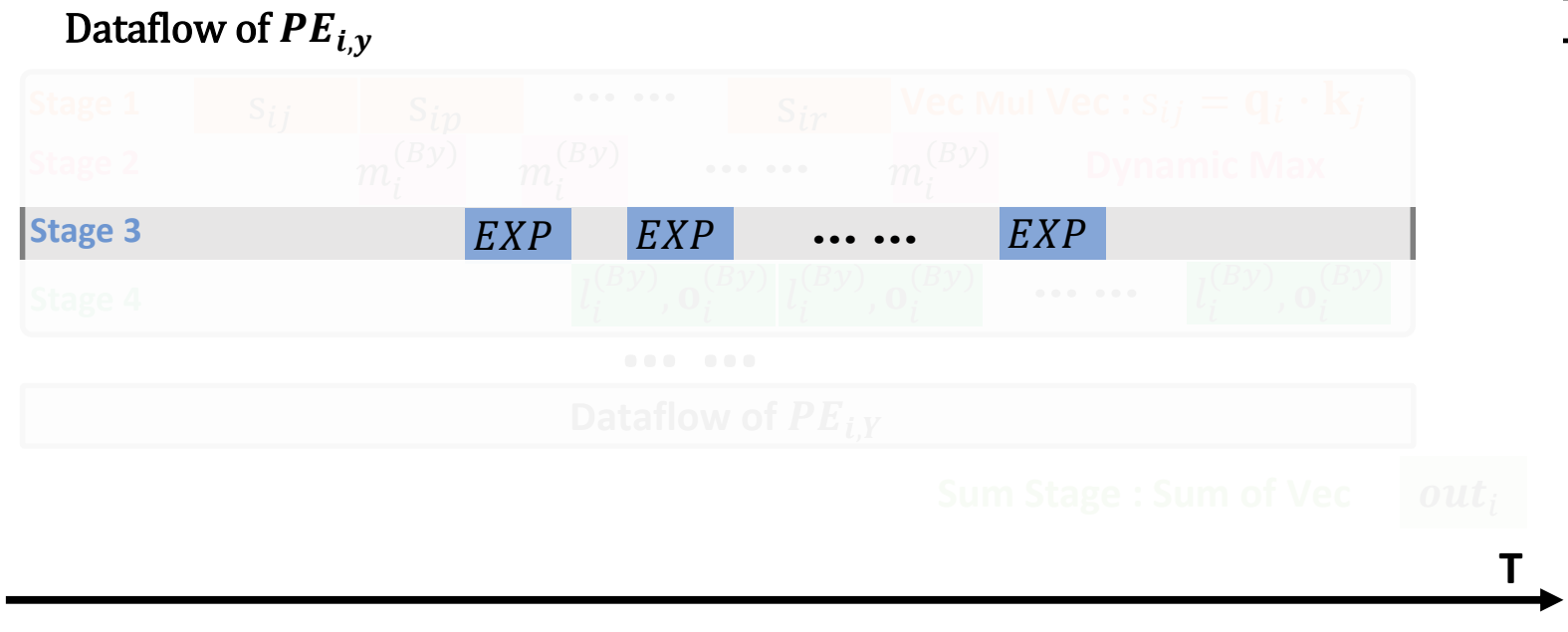
      $m'^{(B_y)}_i = m_i^{(B_y)}$

  $M_i = MAX\left(m_i^{(B_1)}, m_i^{(B_2)}, ..., m_i^{(B_Y)}\right)$

  $\mathbf{output}_i = \frac{\sum_y exp\left(m_i^{(B_y)} - M_i\right) \cdot \mathbf{o}_i^{(B_y)}}{\sum_y exp\left(m_i^{(B_y)} - M_i\right) l_i^{(B_y)}}$

- Stage 2
  - The stored m_i is compared with s_{ij} to get a new m_i

Opt.3: Fine-Grained Attention Dataflow



Algorithm: Fine-Grained Attention Mechanism

INPUT: $\mathbf{q_1}, ..., \mathbf{q_n}, \mathbf{k_1}, ..., \mathbf{k_n}$ and $\mathbf{v_1}, ..., \mathbf{v_n}$ of size $1 \times d_k$ as rows of $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ of size $n \times d_k$

OUTPUT: $\mathbf{output_1}, ..., \mathbf{output_n}$ of size $1 \times d_k$ as rows of attention output matrix of size $n \times d_k$

Parallel for i **in** *range(n)* **do**

  Parallel for B_y **in** $\{B_1, B_2, ..., B_Y\}$ **do**

    Initialize $\mathbf{o}_i^{(B_y)} = (0)_{1\times d_k}$, $l_i^{(B_y)} = 0$, $m'^{(B_y)}_i = -\infty$

    for j **in** B_y **do**

      $s_{ij} = \mathbf{q}_i \cdot \mathbf{k}_j^T$

      $m_i^{(B_y)} = max\left(m'^{(B_y)}_i, s_{ij}\right)$

      $\tilde{s}_{ij} = exp\left(s_{ij} - m_i^{(B_y)}\right)$

      $\tilde{m}_i^{(B_y)} = exp\left(m'^{(B_y)}_i - m_i^{(B_y)}\right)$

      $l_i^{(B_y)} = \tilde{m}_i^{(B_y)} l_i^{(B_y)} + \tilde{s}_{ij}$

      $\mathbf{o}_i^{(B_y)} = \tilde{m}_i^{(B_y)} \cdot \mathbf{o}_i^{(B_y)} + \tilde{s}_{ij} \cdot \mathbf{v}_j$

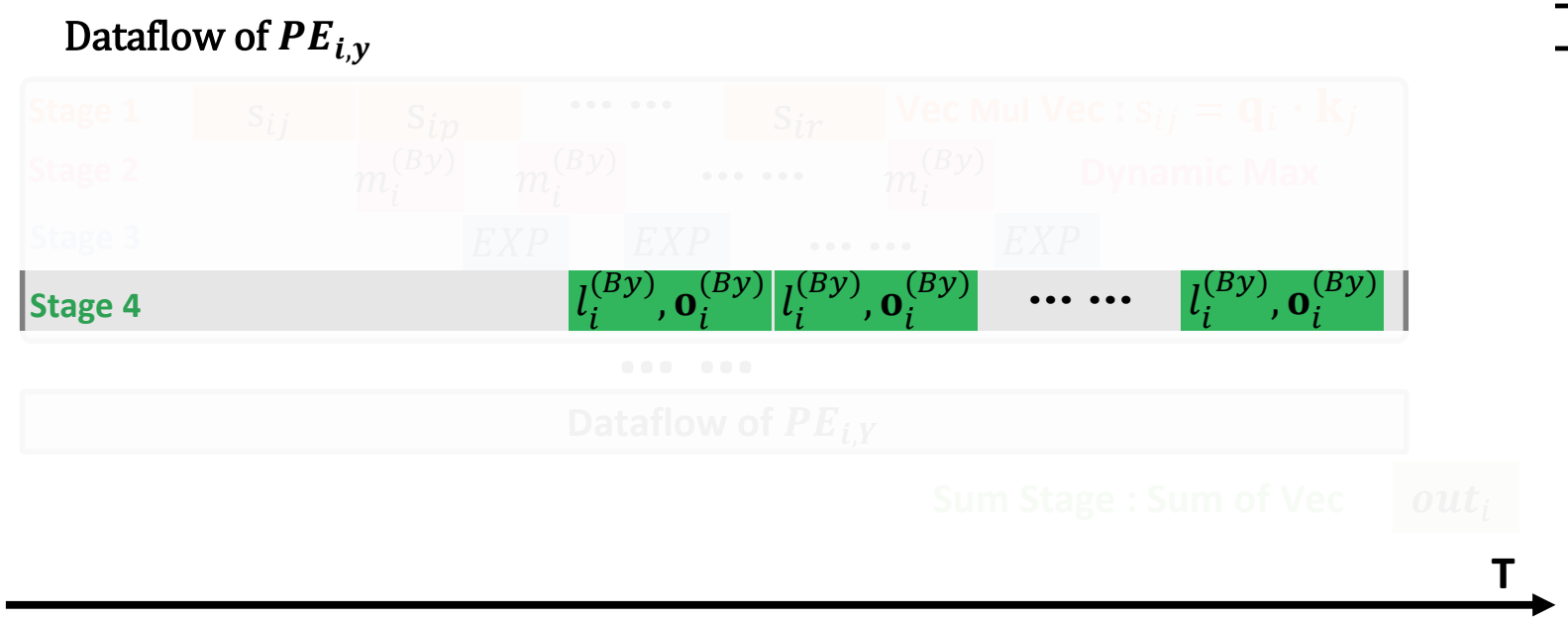
      $m'^{(B_y)}_i = m_i^{(B_y)}$

  $M_i = MAX\left(m_i^{(B_1)}, m_i^{(B_2)}, ..., m_i^{(B_Y)}\right)$

  $\mathbf{output}_i = \frac{\sum_y exp\left(m_i^{(B_y)} - M_i\right) \cdot \mathbf{o}_i^{(B_y)}}{\sum_y exp\left(m_i^{(B_y)} - M_i\right) l_i^{(B_y)}}$

- Stage 3
  - Exponentiation is calculated
  - Subtracting the new m_i

Opt.3: Fine-Grained Attention Dataflow



Algorithm: Fine-Grained Attention Mechanism

INPUT: $\mathbf{q}_1, ..., \mathbf{q}_n, \mathbf{k}_1, ..., \mathbf{k}_n$ and $\mathbf{v}_1, ..., \mathbf{v}_n$ of size $1 \times d_k$ as rows of $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ of size $n \times d_k$

OUTPUT: $\mathbf{output}_1, ..., \mathbf{output}_n$ of size $1 \times d_k$ as rows of attention output matrix of size $n \times d_k$

Parallel for i **in** *range(n)* **do**

  Parallel for B_y **in** $\{B_1, B_2, ..., B_Y\}$ **do**

    Initialize $\mathbf{o}_i^{(B_y)} = (0)_{1 \times d_k}$, $l_i^{(B_y)} = 0$, $m'^{(B_y)}_i = -\infty$

    for j **in** B_y **do**

      $s_{ij} = \mathbf{q}_i \cdot \mathbf{k}_j^T$

      $m_i^{(B_y)} = max\left(m'^{(B_y)}_i, s_{ij}\right)$

      $\tilde{s}_{ij} = exp\left(s_{ij} - m_i^{(B_y)}\right)$

      $\tilde{m}_i^{(B_y)} = exp\left(m'^{(B_y)}_i - m_i^{(B_y)}\right)$

      $l_i^{(B_y)} = \tilde{m}_i^{(B_y)} l_i^{(B_y)} + \tilde{s}_{ij}$

      $\mathbf{o}_i^{(B_y)} = \tilde{m}_i^{(B_y)} \cdot \mathbf{o}_i^{(B_y)} + \tilde{s}_{ij} \cdot \mathbf{v}_j$

      $m'^{(B_y)}_i = m_i^{(B_y)}$

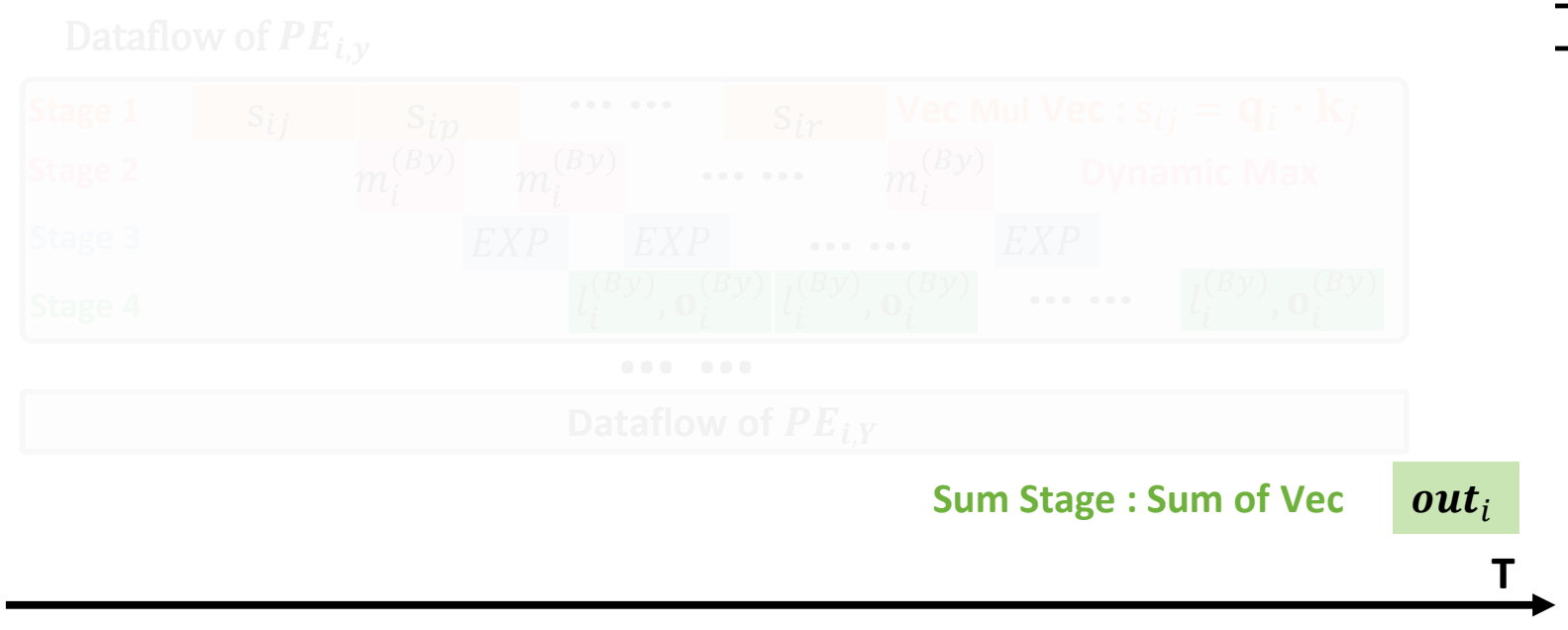
  $M_i = MAX\left(m_i^{(B_1)}, m_i^{(B_2)}, ..., m_i^{(B_Y)}\right)$

  $\mathbf{output}_i = \frac{\sum_y exp\left(m_i^{(B_y)} - M_i\right) \cdot \mathbf{o}_i^{(B_y)}}{\sum_y exp\left(m_i^{(B_y)} - M_i\right) l_i^{(B_y)}}$

➢ Stage 4

- Multiplication of s_{ij} and v_j is conducted
- The l_i, $\mathbf{o}_i$ results updated by the difference between the stored and new m_i
- The stored m_i is updated by the new one

Opt.3: Fine-Grained Attention Dataflow



Algorithm: Fine-Grained Attention Mechanism

INPUT: $\mathbf{q_1}, ..., \mathbf{q_n}, \mathbf{k_1}, ..., \mathbf{k_n}$ and $\mathbf{v_1}, ..., \mathbf{v_n}$ of size $1 \times d_k$ as rows of $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ of size $n \times d_k$

OUTPUT: $\mathbf{output_1}, ..., \mathbf{output_n}$ of size $1 \times d_k$ as rows of attention output matrix of size $n \times d_k$

Parallel for i **in** *range(n)* **do**

  Parallel for B_y **in** $\{B_1, B_2, ..., B_Y\}$ **do**

    Initialize $\mathbf{o}_i^{(B_y)} = (0)_{1\times d_k}$, $l_i^{(B_y)} = 0$, $m'^{(B_y)}_i = -\infty$

    for j **in** B_y **do**

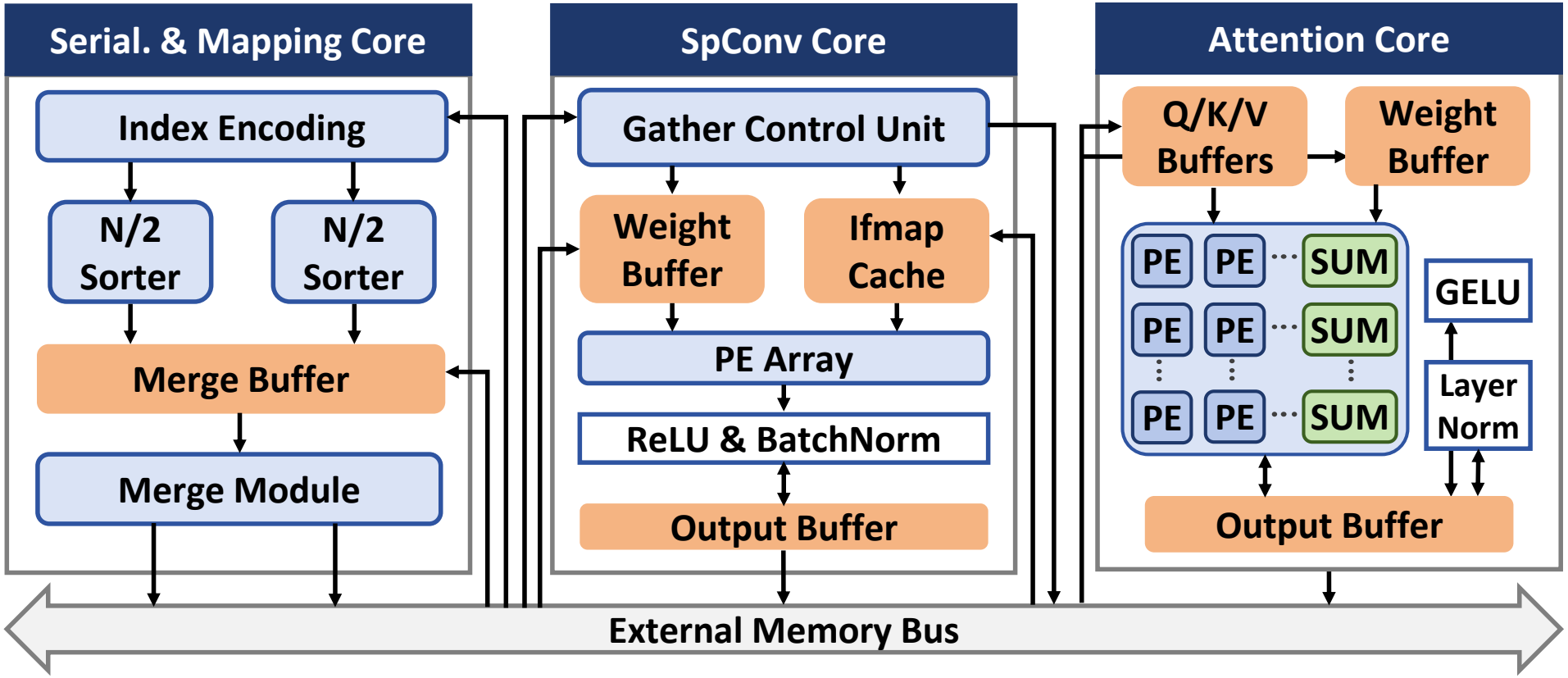
$$s_{ij} = \mathbf{q}_i \cdot \mathbf{k}_j^T$$
$$m_i^{(B_y)} = max\left(m'^{(B_y)}_i, s_{ij}\right)$$
$$\tilde{s}_{ij} = exp\left(s_{ij} - m_i^{(B_y)}\right)$$
$$\tilde{m}_i^{(B_y)} = exp\left(m'^{(B_y)}_i - m_i^{(B_y)}\right)$$
$$l_i^{(B_y)} = \tilde{m}_i^{(B_y)} l_i^{(B_y)} + \tilde{s}_{ij}$$
$$\mathbf{o}_i^{(B_y)} = \tilde{m}_i^{(B_y)} \cdot \mathbf{o}_i^{(B_y)} + \tilde{s}_{ij} \cdot \mathbf{v}_j$$
$$m'^{(B_y)}_i = m_i^{(B_y)}$$

$$M_i = MAX\left(m_i^{(B_1)}, m_i^{(B_2)}, ..., m_i^{(B_Y)}\right)$$

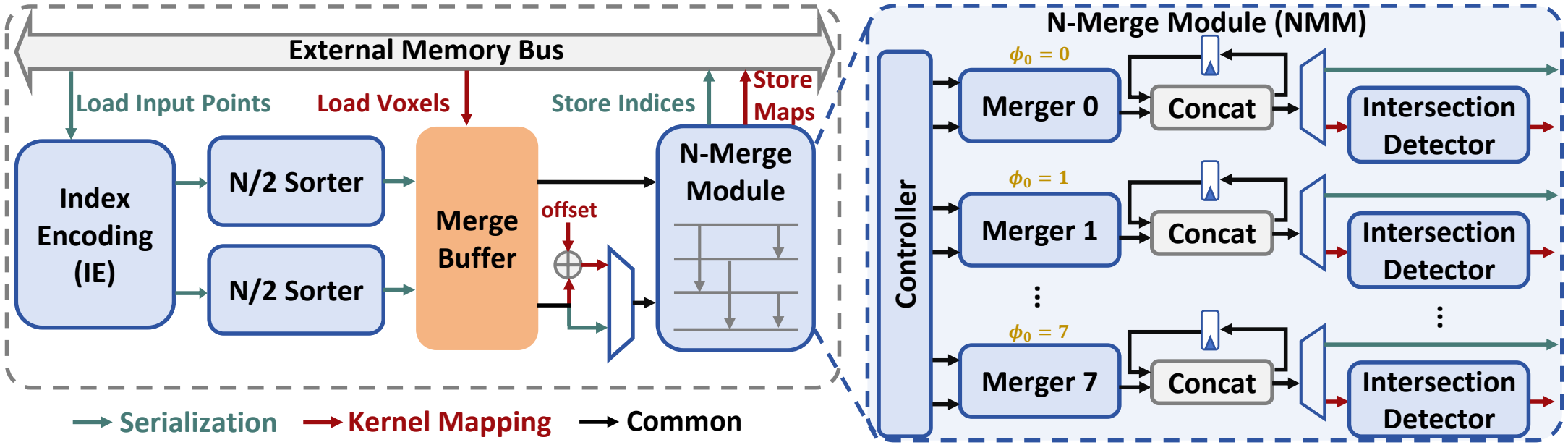
$$\mathbf{output}_i = \frac{\sum_y exp\left(m_i^{(B_y)} - M_i\right) \cdot \mathbf{o}_i^{(B_y)}}{\sum_y exp\left(m_i^{(B_y)} - M_i\right) l_i^{(B_y)}}$$

- Sum Stage
  - Obtain the global maximum value M_i of all local m_i
  - Updating l_i, $\mathbf{o}_i$ from each block with the difference between their m_i and M_i
  - The overall attention result $output_i$ for q_i is achieved

Overview of APTO Architecture



Serialization & Mapping Core



➢ Serialization

- Encodes and sorts points' **z-curve indices**

① IE converts coordinates into z-curve indices

② N/2 sorter sorts N/2 indices

③ NMM combines two sorted arrays into one

➢ Kernel map creation

- Supports **parallel neighbor searching**

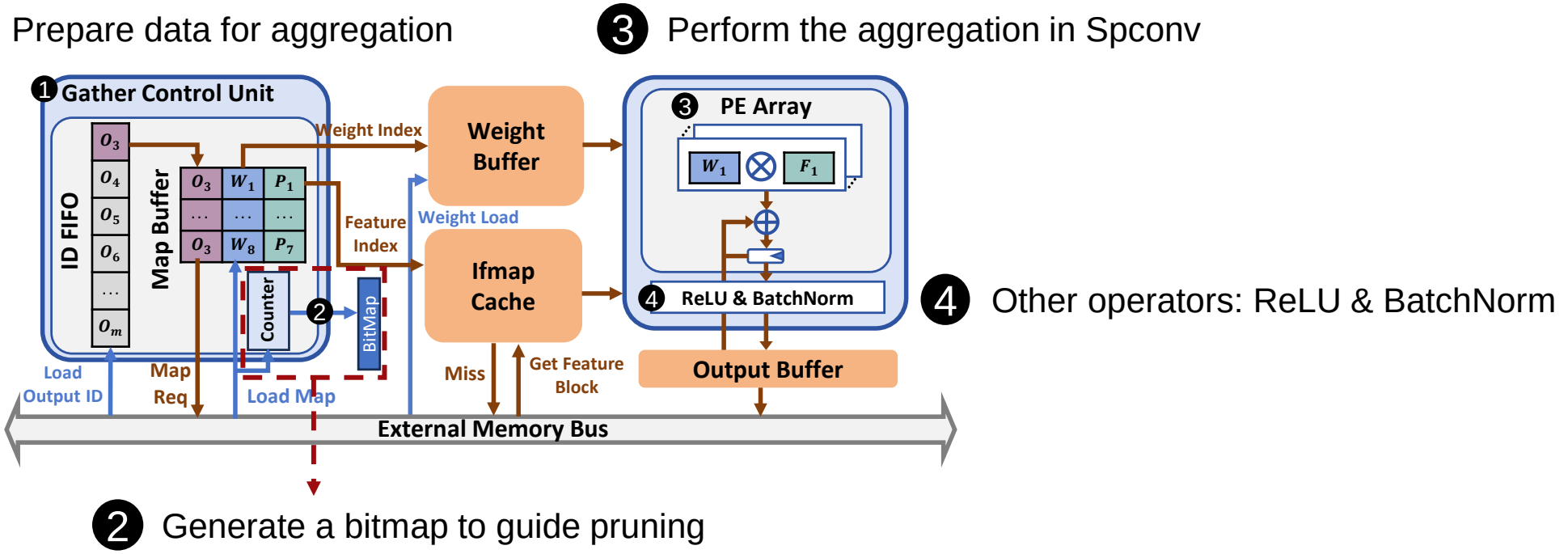
① Divides voxels into 8 sets based on ϕ_0

② NMM merges the shifted input and output voxels

③ Intersection detector finds voxels in the same position

Enables parallel processing based on the z-curve indices

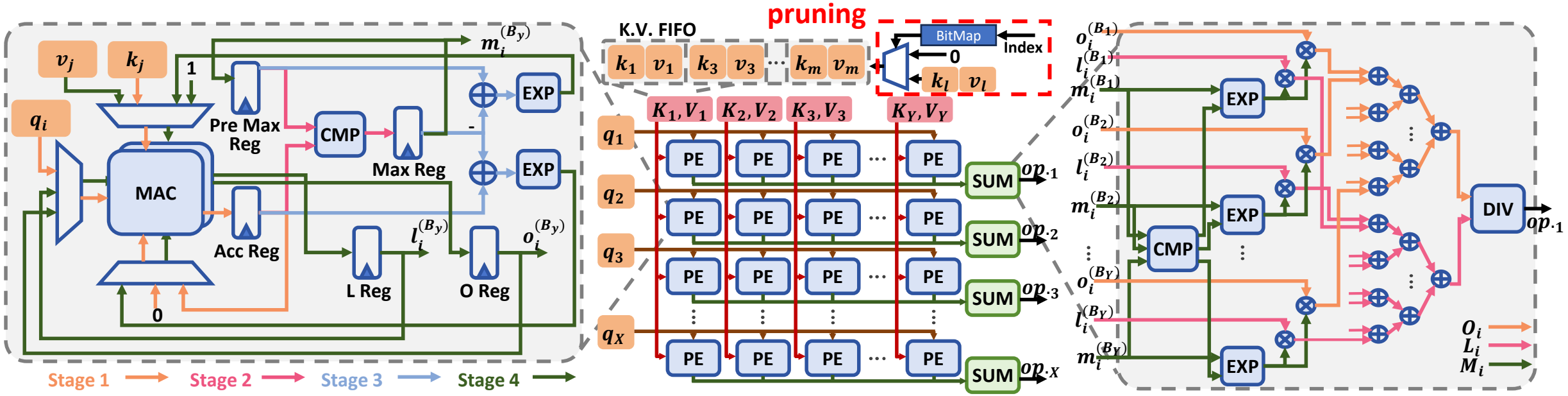
SpConv Core



- Gather control unit
  - ① Appends output coordinates to the **ID FIFO**
  - ② Retrives entries based on the FIFO's front element
- Counter
  - ① Resets to 0 when starts to load an output's entries
  - ② Counts the total number of entries
  - ③ Sets the bitmap element to 1 if exceeds the threshold

Performs SpConv & uses neighbor counts to guide pruning

Attention Core



- Each unit in the attention core performs 4 stages in the fine-grained attention dataflow

- Parallel processing along the sequence dimension
  - $\mathbf{q}_i$ is sent to each row of the array
  - **K**, **V** is sent to each column in blocks
- Support the position-aware pruning strategy
  - Pruned $\mathbf{k}_j$ & $\mathbf{v}_j$ are skipped according to the bitmap

High-throughput pipeline & efficient utilization of on-chip buffers

Experimental Setup

- Baseline
  - Octformer & PTv3
  - Implemented in PyTorch 1.12.1 & CUDA 11.8
- Dataset
  - ModelNet40
    - Small-scale, for object classification
  - ScanNet & S3DIS
    - Large-scale, for scene semantic segmentation
- Hardware platform
  - NVIDIA RTX 4090 GPU
  - State-of-the-art point cloud accelerators
    - PointAcc & SpOctA

Table 1: Models and datasets.

Application	Dataset	Model	#Points
Classification	ModelNet40	Octformer- MN	1k
Segmentation	ScanNet	Octformer- SN PTv3- SN	100k
	S3DIS	PTv3- S3D	1M

[1] Dongxu Lyu, Zhenyu Lil, Yuzhou Chen, Jinming Zhang, Ningyi Xu, and Guanghui He. 2023. SpOctA: A 3D Sparse Convolution Accelerator with Octree-Encoding Based Map Search and Inherent Sparsity-Aware Processing. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 1–9.

Effectiveness of the pruning strategy

- Compared to BigBird, ours achieves **higher accuracy** under the same sparsity

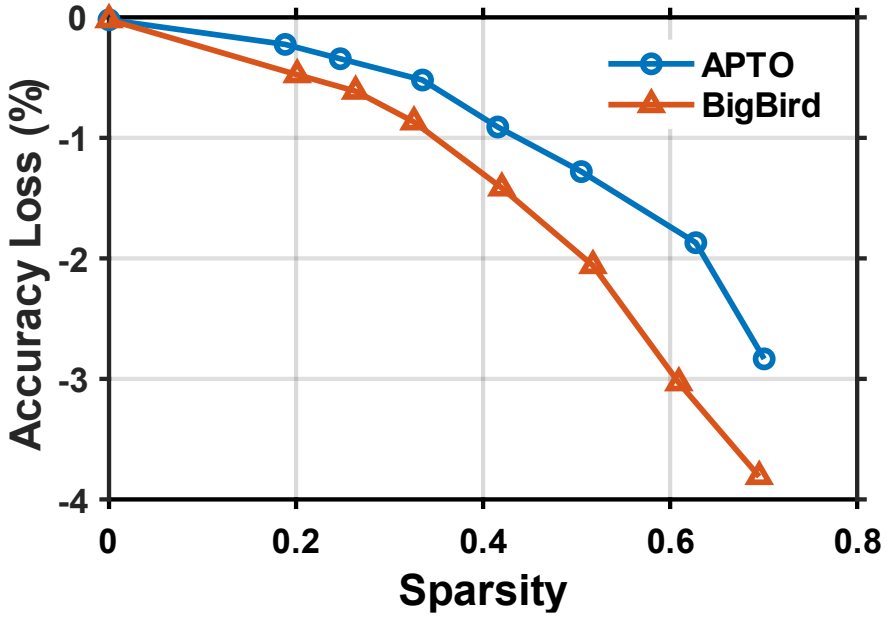


Table 2: Impacts of the position-aware pruning strategy.

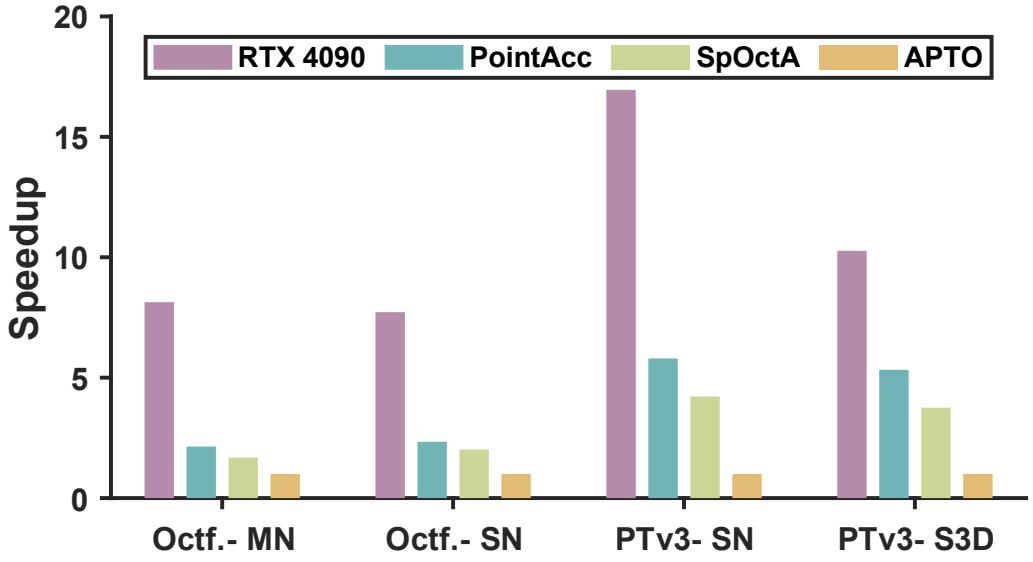
Model	Result	Loss	Sparsity	Window Size
Octformer- MN	91.75%	0.43%	32.20%	32
Octformer- SN	73.90%	0.52%	33.52%	32
PTv3- SN	76.63%	0.64%	34.02%	1024
PTv3- S3D	72.12%	0.30%	41.35%	128

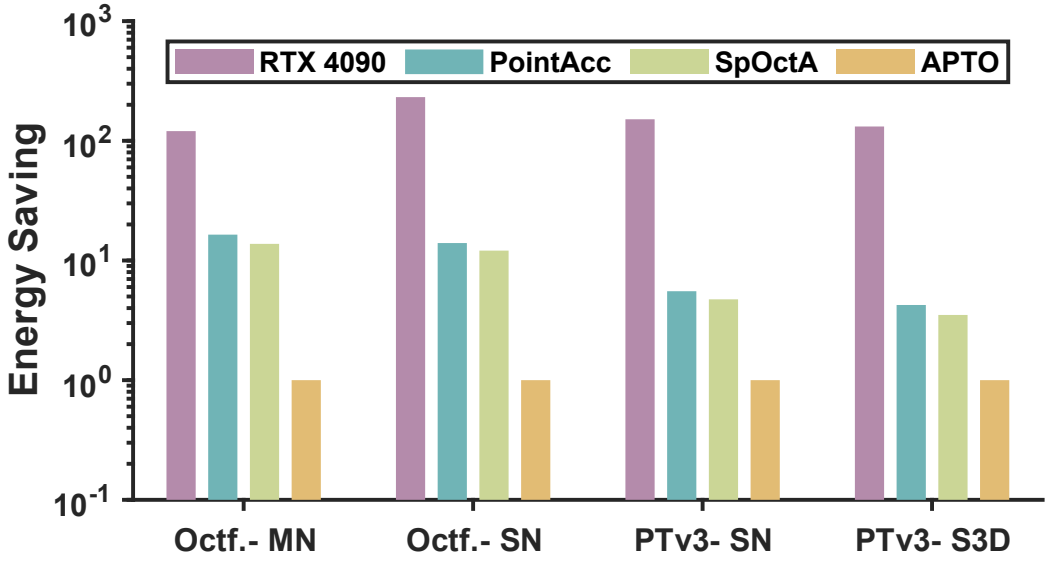
Achieves 35.27% sparsity with less than 1% accuracy loss

[1] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems* 33 (2020), 17283–17297.

End-to-End Performance

- Compared with RTX 4090 GPU, PointAcc & SpOctA
  - Speedups of **10.22x**, **3.53x**, **2.70x**
  - Energy savings of **153.59x**, **8.75x**, **7.25x**

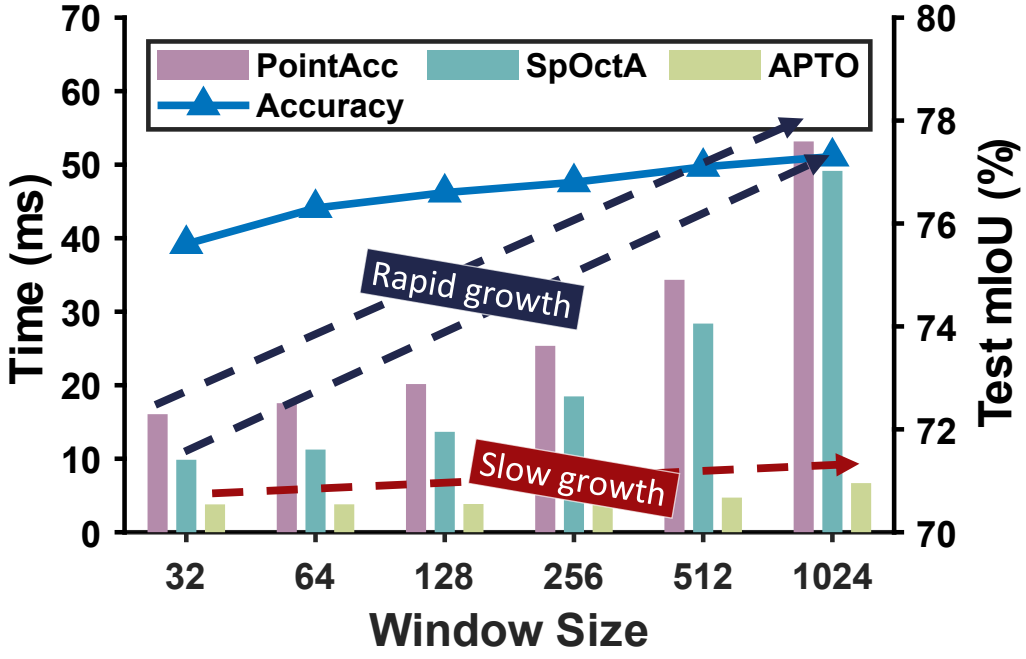




Meets real-time processing requirements

Attention Optimization

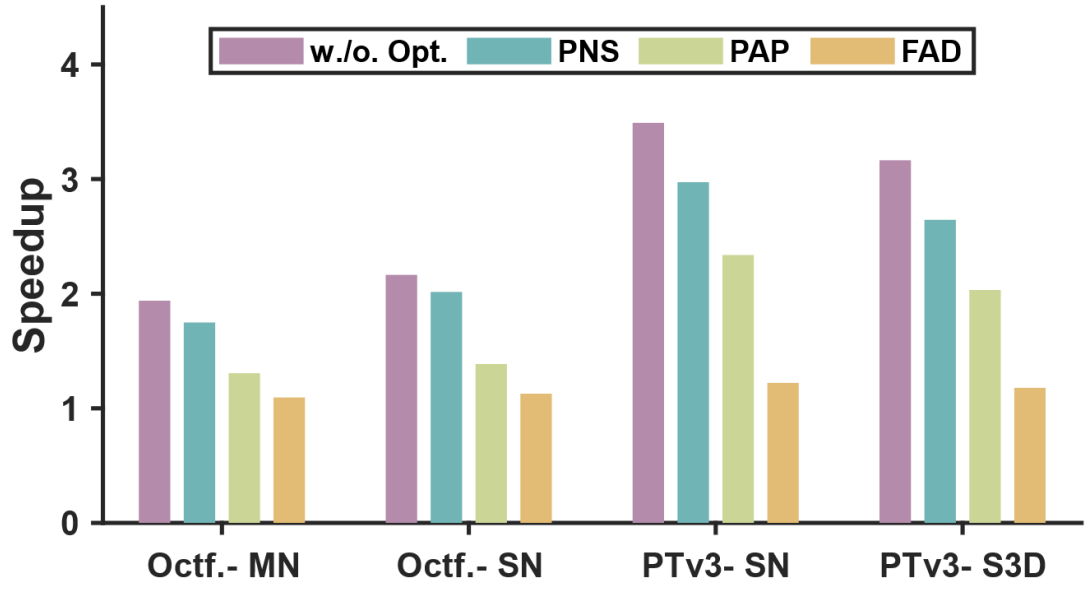
- Attention acceleration as window size increases
  - Window size is 32: **4.24x** over PointAcc & **2.60x** over SpOctA
  - Window size is 1024: **7.95x** over PointAcc & **7.35x** over SpOctA



Efficiently accelerate attention calculation in large-size windows

Ablation Study

- Optimizations for serialization-based models
  - PNS (Parallel Neighbor Search): **1.14x**
  - PAP (Position-Aware Pruning): **1.52x**
  - FAD (Fine-Grained Attention Dataflow): **2.26x**



Majority of optimization effects stems from PAP and FAD

Conclusion

- APTO optimizes SpConv's mapping stage by **searching neighbor voxels in parallel** based on the lowest 3 bits of their z-curve indices

- APTO takes a **position-aware attention pruning** strategy to reduce redundant attention computations, which prunes tokens generated from voxels

This is the first work to accelerate serialization-based point cloud transformers

- APTO adopts a **fine-grained attention dataflow** with dynamic maximum values to remove data dependencies and splits K, V into blocks for parallel processing

- APTO achieves average 10.22×, 3.53× and 2.70× speedups over RTX 4090 GPU, PointAcc, and SpOctA, with 153.59×, 8.57× and 7.25× energy savings

30th Asia and South Pacific Design Automation Conference (ASPDAC '25)
Tokyo, Japan

Thank you

Q & A