Exploiting Hardware Utilization and Adaptive Dataflow for Efficient Sparse Convolution in 3D Point Clouds

Ke Hong*[1], Zhongming Yu*[2], Guohao Dai[3], Xinhao Yang[1], Yaoxiu Lian[3], Zehao Liu[1], Ningyi Xu[3], Yuhan Dong[1], Yu Wang[1]

Tsinghua University[1] , UC San Diego[2] and Shanghai Jiao Tong University[3]

*Equal contribution

Correspondence to:

Yu Wang <yu-wang@tsinghua.edu.cn>

Quick View

- Methodologies:
  - Coded-CSR format **mapping storage**
  - Indicator-assisted FGMS (Fused Gather-MM-Scatter) **fusion**
  - Heuristic **adaptive dataflow selection**
- Compared to SOTA Designs:
  - **1.23x/1.64x end-to-end** speedup on average
  - **1.76x/1.81x sparse convolution** speedup on average

Outline

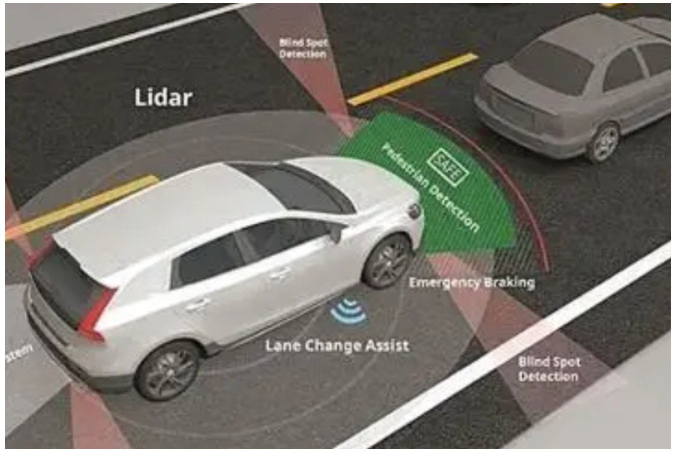
- Backgrounds and Motivations
- Design Overview
- Solutions
  - Coded-CSR format **mapping storage**
  - Indicator-assisted FGMS (Fused Gather-MM-Scatter) **fusion**
  - Heuristic **adaptive dataflow selection**
- Experiments
- dgSPARSE: Fast and Efficient Processing with Diverse Sparsity

Outline

- Backgrounds and Motivations
- Design Overview
- Solutions
  - Coded-CSR format **mapping storage**
  - Indicator-assisted FGMS (Fused Gather-MM-Scatter) **fusion**
  - Heuristic **adaptive dataflow selection**
- Experiments
- dgSPARSE: Fast and Efficient Processing with Diverse Sparsity

Point Cloud Neural Networks

Areas

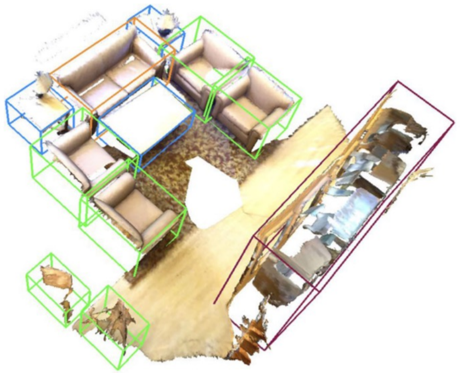


Autonomous Driving

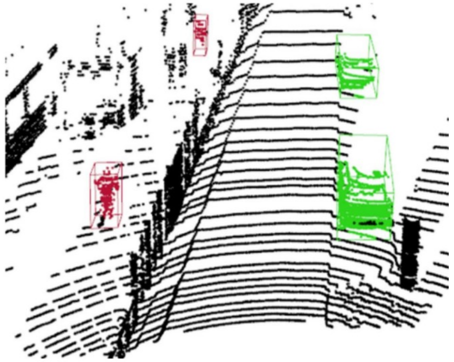
Robotics

Remote Sensing

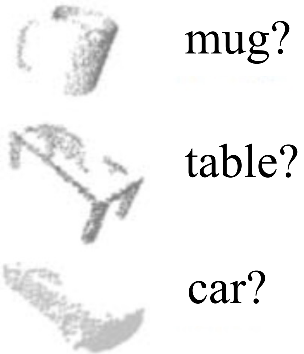
Tasks



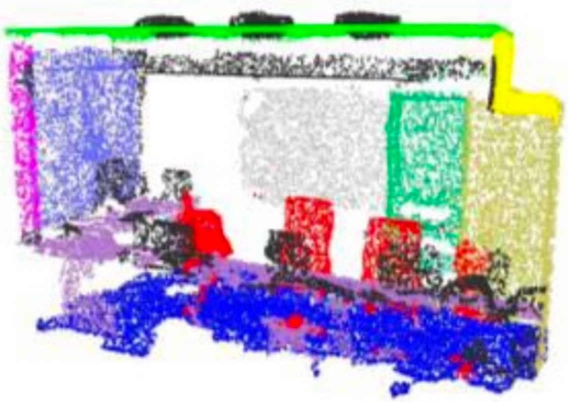
Detection



Tracking



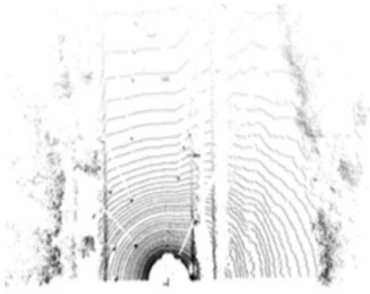
Classification



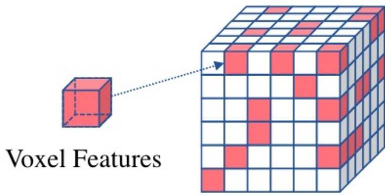
Segmentation

Point Cloud Neural Networks

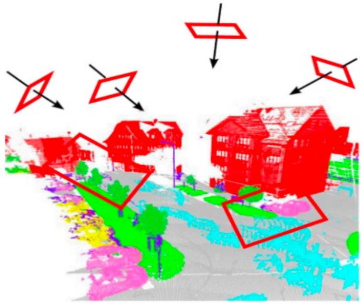
Data



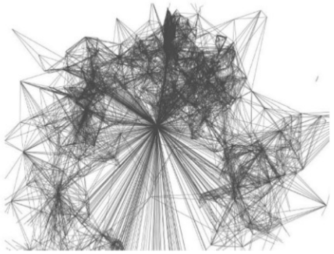
Point



Voxel



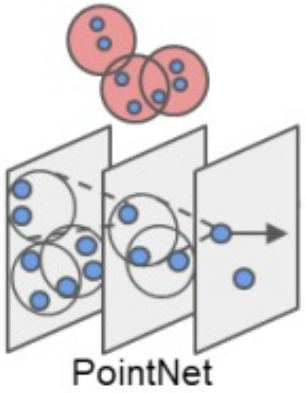
BEV (Bird's Eye View)



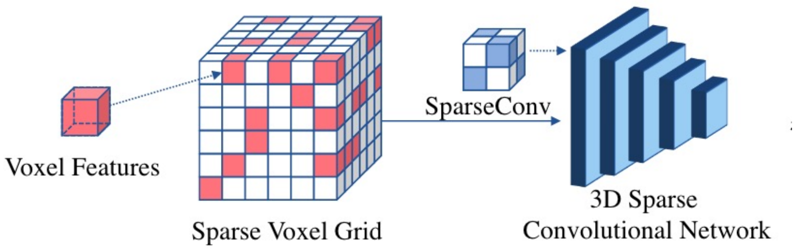
Graph

...

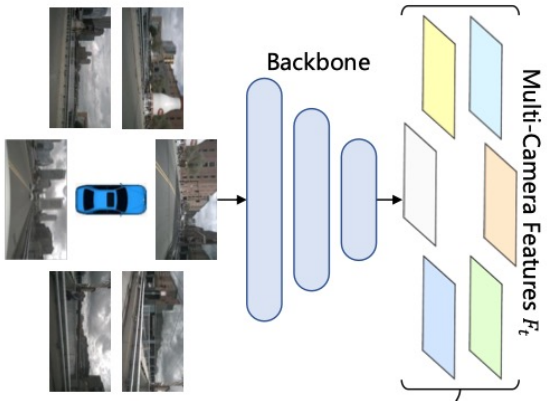
Methods



Point-based



Voxel-based

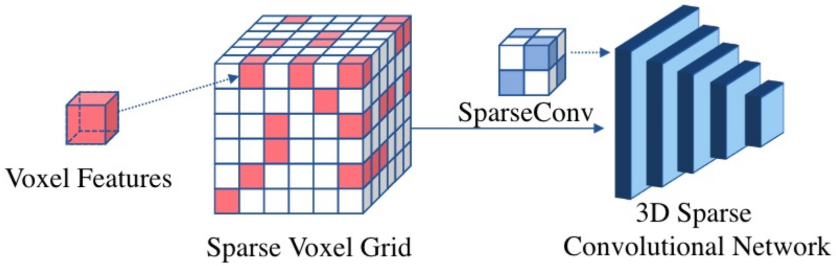


BEV-based

...

Sparse Convolution Neural Networks

Methods



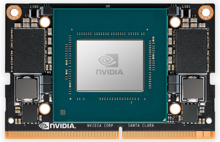
Voxel-based

SOTA Accuracy

nuScenes lidar segmentation task

Leaderboard

			Any	All	All				
>	2022-07-22	LidarMultiNet	Lidar	no	no	0.814	0.909	n/a	
>	2021-11-16	SVQNet	Lidar	no	no	0.813	0.911	n/a	
>	2022-07-18	MSeg3D	Camera, Lidar	no	no	0.811	0.914	n/a	
>	2021-05-27	SPVCNN++	Lidar	no	no	0.811	0.910	n/a	



<5FPS

Nvidia
Xavier GPU

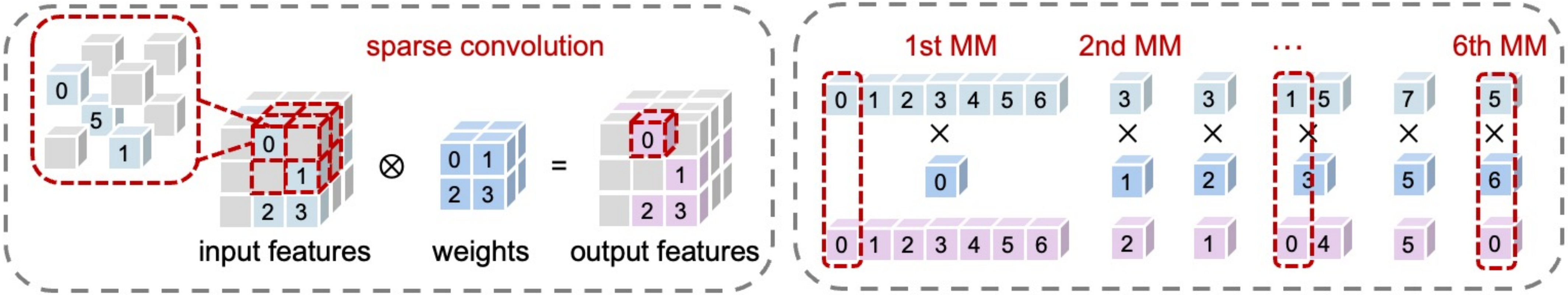
~10FPS

Nvidia
RTX 2080 GPU

Real-time Demand: 25~30FPS

Sparse Convolution Neural Networks

MM: Matrix Multiplication



Sparse Input
Only 0.01%-1% density

Irregular Access
Gather valid features

Dense Computing
MMs between features and weights

Irregular Access
Scatter partial sums

Implementation Dataflows

Device memory

Shared memory /register

GPU kernel

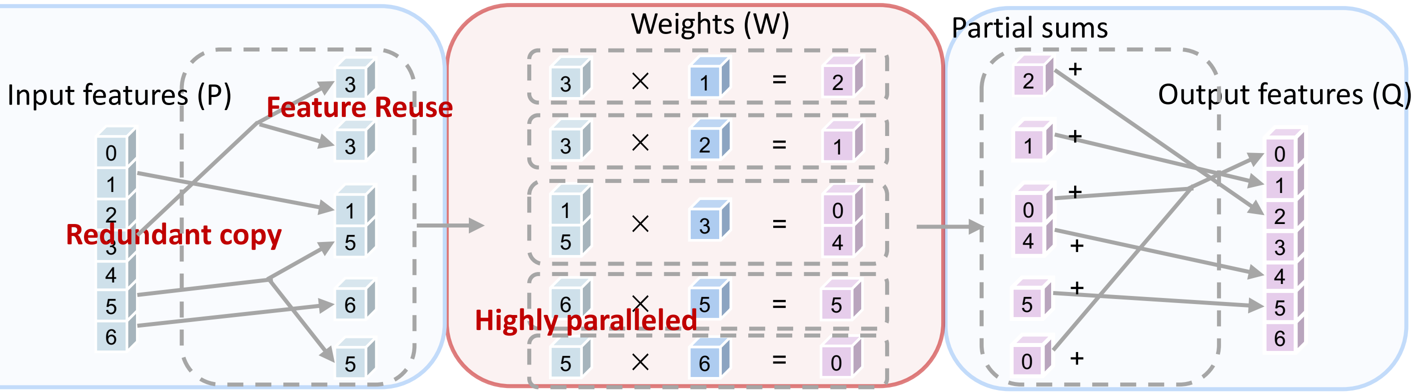
MM: Matrix Multiplication

Sparse Input	**Irregular Access**	**Dense Computing**	**Irregular Access**
Only 0.01%-1% density	**Gather** valid features	**MMs** between features and weights	**Scatter** partial sums

Dataflow 1
Gather-MM-Scatter

(input, weight)	(output, weight)
P_1, W_3	Q_0, W_3
P_3, W_1	Q_0, W_6
…	
P_5, W_6	Q_4, W_3
P_6, W_5	Q_5, W_5

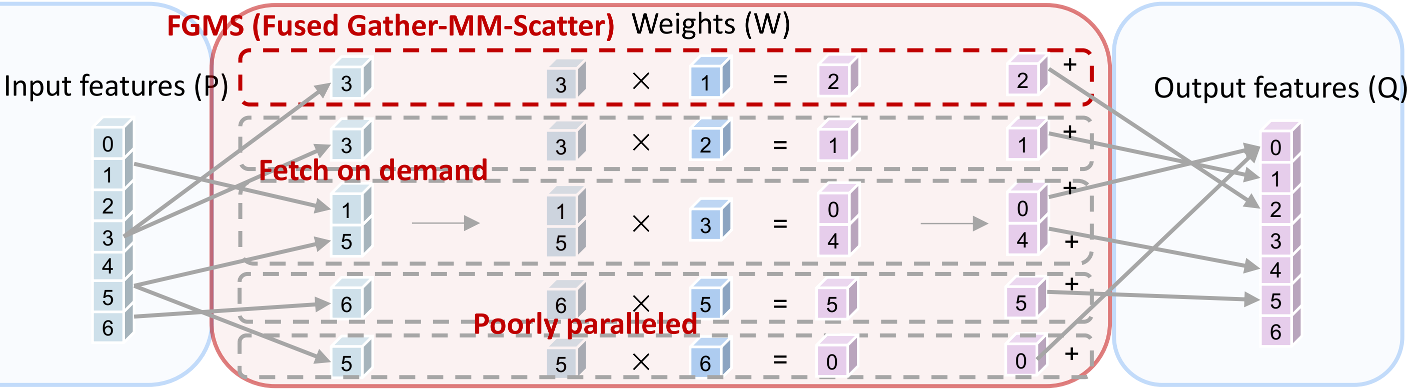
Input/output-major mapping



Dataflow 2
Fetch-on-Demand

(weight, input, output)
W_1, P_3, Q_2
W_2, P_3, Q_1
…
W_5, P_6, Q_5
W_6, P_5, Q_0

Weight-major mapping



Outline

- Backgrounds and Motivations
- Design Overview
- Solutions
  - Coded-CSR format **mapping storage**
  - Indicator-assisted FGMS (Fused Gather-MM-Scatter) **fusion**
  - Heuristics **adaptive dataflow selection**
- Experiments
- dgSPARSE: Fast and Efficient Processing with Various Sparsity

Design Overview

3. **Heuristic adaptive dataflow selection**

Performance loss against input dynamics:
Dataflow 2 benefits light workload

1. **Coded-CSR format mapping storage**

Redundant mapping loadings:
Up to 79.97% mappings are empty

Dataflow 1
Gather-MM-Scatter

Dataflow 2
Fetch-on-Demand

Irregular Access
Gather valid features

Dense Computing
MMs between features and weights

Irregular Access
Scatter partial sums

Sparse Computing
Sequential FGMS (Fused Gather-MM-Scatter) kernels

Computation under-utilization:
Only 22.84% GPU utilization

2. **Indicator-assisted FGMS fusion**

Outline

- Backgrounds and Motivations
- Design Overview
- Solutions
  - Coded-CSR format **mapping storage**
  - Indicator-assisted FGMS (Fused Gather-MM-Scatter) **fusion**
  - Heuristics **adaptive dataflow selection**
- Experiments
- dgSPARSE: Fast and Efficient Processing with Diverse Sparsity

Coded-CSR Format Mapping Storage

➢Observation:

- **Redundant loadings** of invalid mappings

	W_0	W_1	W_2	W_3	W_4	W_5	W_6
P_0	0	-1	-1	0	0	-1	0
P_1	-1	0	-1	-1	-1	-1	-1
P_2	-1	-1	0	-1	-1	-1	1
P_3	-1	-1	1	1	-1	-1	2
P_4	-1	1	-1	-1	-1	0	-1

35 integers

(a) Matrix

1) Load input feature (P)

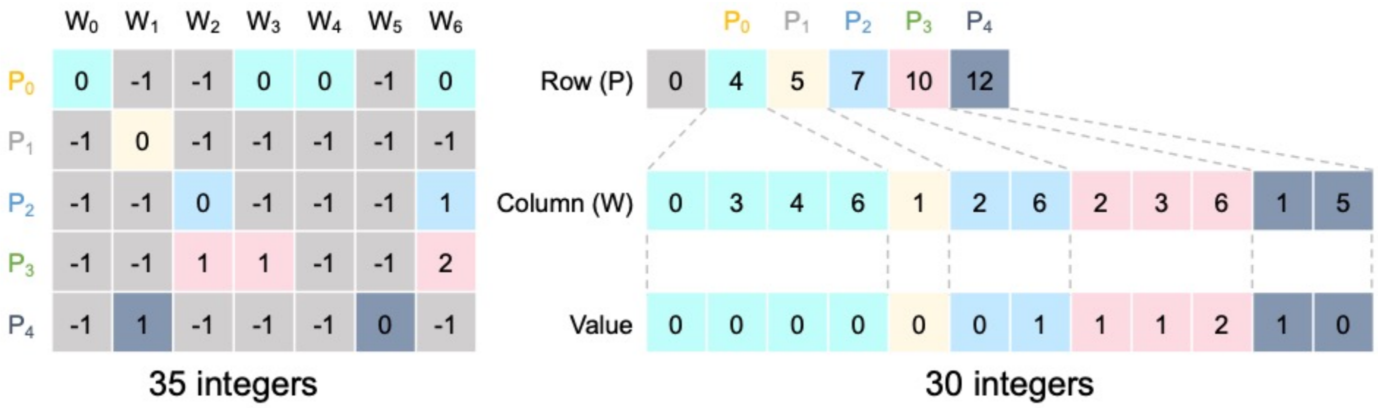
2) Traverse mappings of all weights (W)

3) If mapping is valid, write feature to the buffer

Redundant mapping loadings:
Up to 79.97% mappings are empty

Coded-CSR Format Mapping Storage

- **Observation:**
  - **Redundant loadings** of invalid mappings
  - Column array and value array can be **further compressed**
    - Column element < K (kernel size), value element < N (number of input non-zeros)

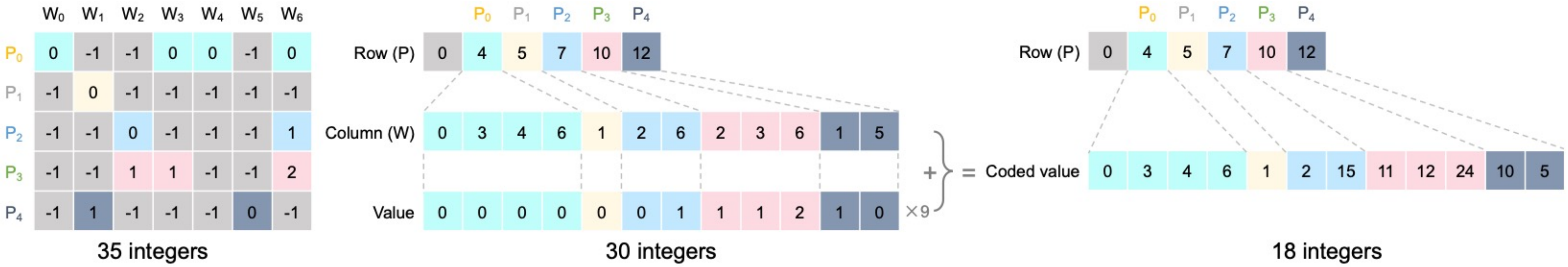


(a) Matrix (b) CSR (Compress Sparse Row)

Only valid mappings are stored

Coded-CSR Format Mapping Storage

- Solution:
  - **Sparse format** for mapping storage
  - Column array and value array are further **coded into one array**
    - Encoder function: c = a + *M*b, *M* ≥ *K.*



(a) Matrix

(b) CSR (Compress Sparse Row)

(c) Coded-CSR

about 1 (int32) loading for a valid mapping

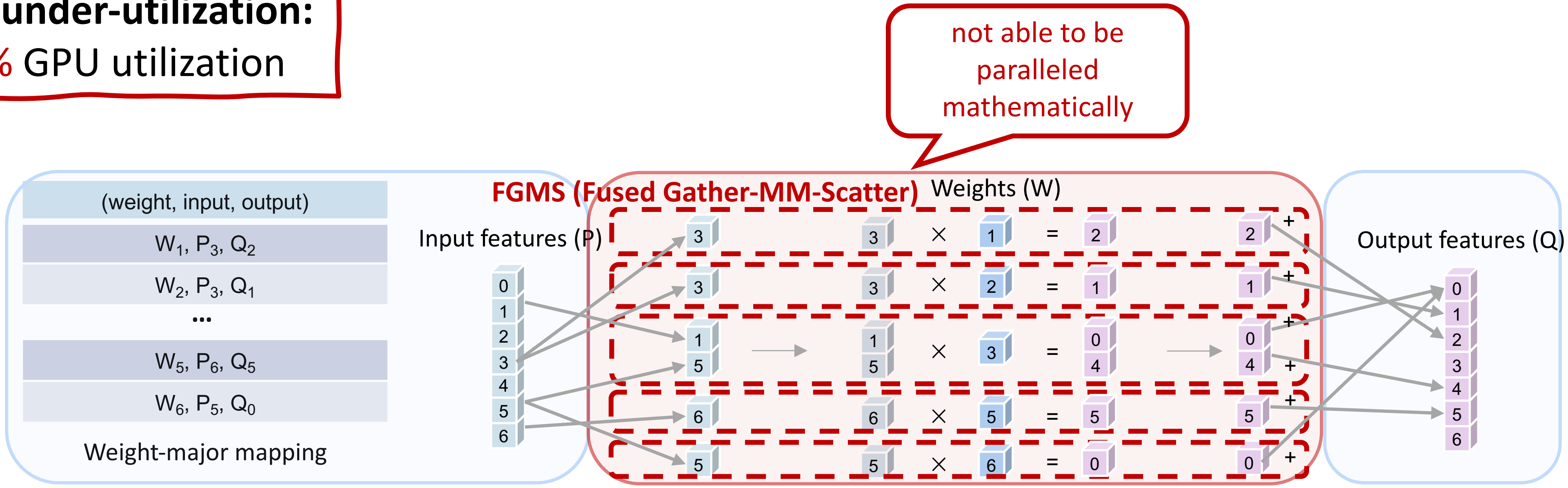
Indicator-assisted FGMS Fusion Scheme

➢ Observation:

- Sequentially computing each FGMS leads to **under-utilization**

Computation under-utilization:
Only 22.84% GPU utilization

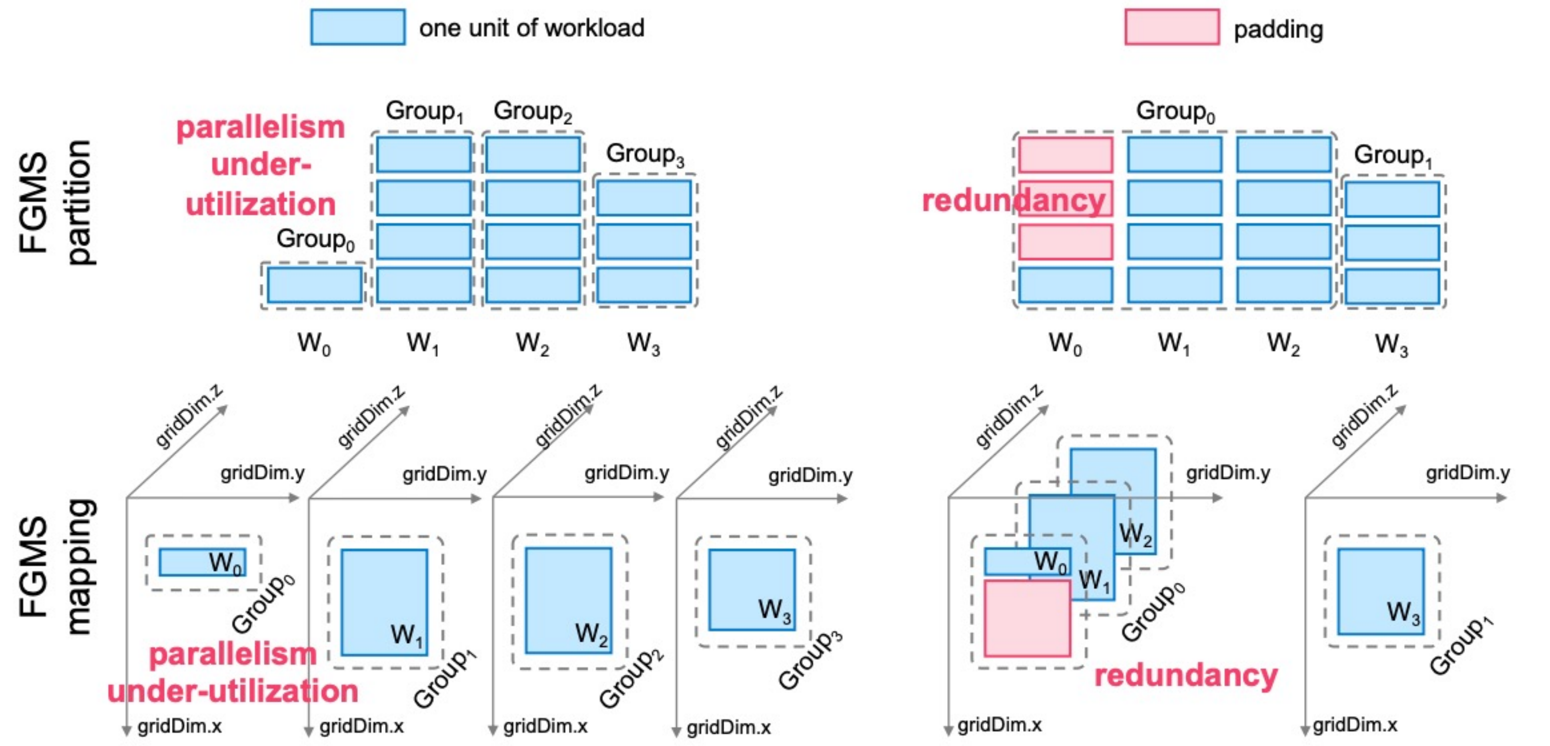
Dataflow 2
Fetch-on-Demand



Indicator-assisted FGMS Fusion Scheme

➢Observation:

- Sequentially computing each FGMS leads to **under-utilization**
- Segmented FGMSs can be paralleled through **hardware mapping strategy**
  - Batched FGMS scheme still leads to **computing redundancy** and **under-utilization**



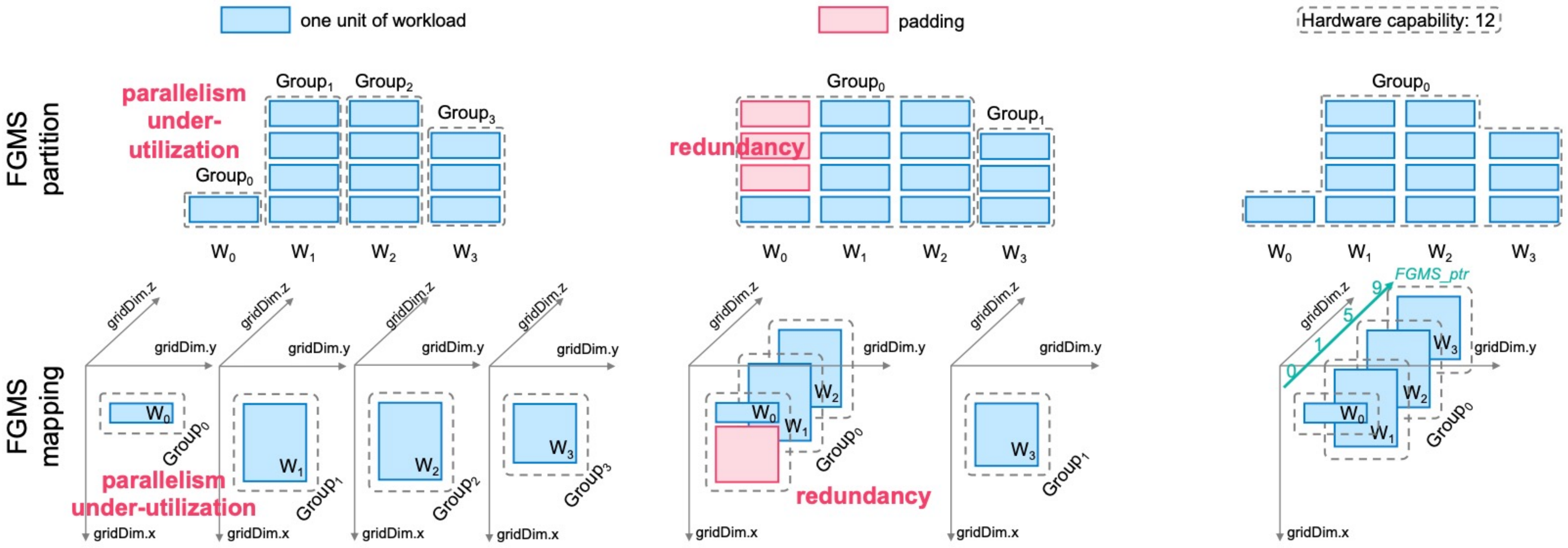
(a) Separate FGMS with parallelism under-utilization (b) Batched FGMS with redundant padding

Computation under-utilization:
Only 32.67% GPU utilization

Indicator-assisted FGMS Fusion Scheme

➢ Solution:

- Segmented FGMSs are fused with a **sparse indicator**, which indicates the mapping table address from each FGMS

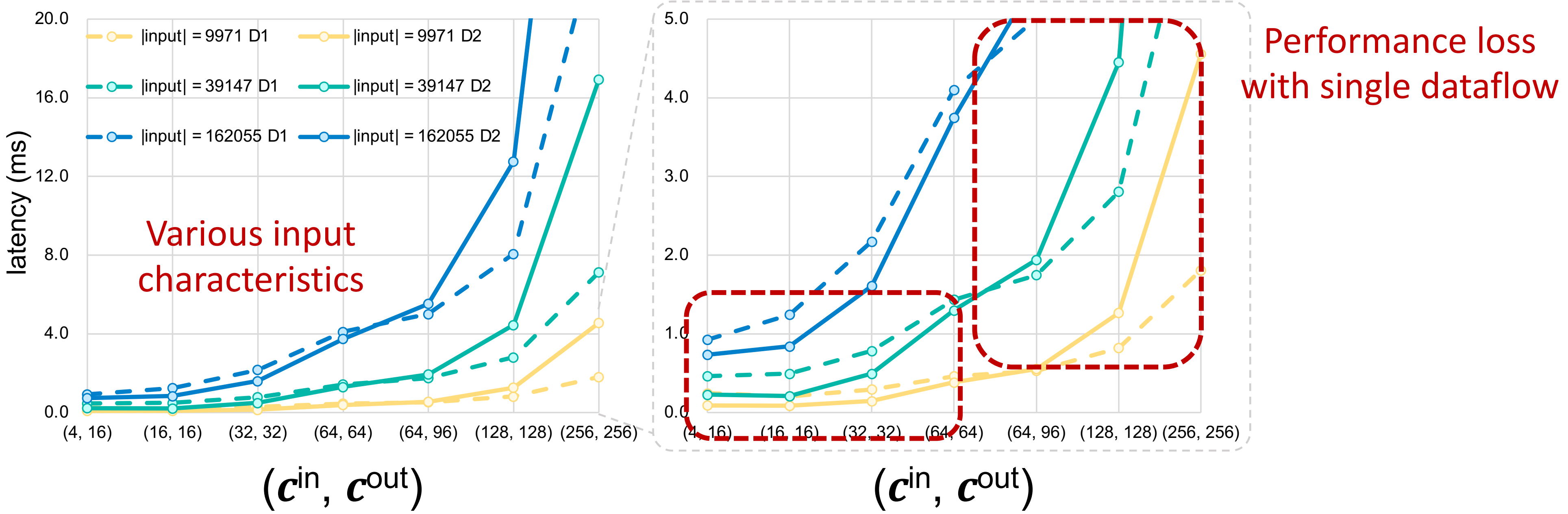


(a) Separate FGMS with parallelism under-utilization (b) Batched FGMS with redundant padding (c) Indicator-assisted segmented FGMS fusion

Heuristic Adaptive Dataflow Selection

➢Observation:

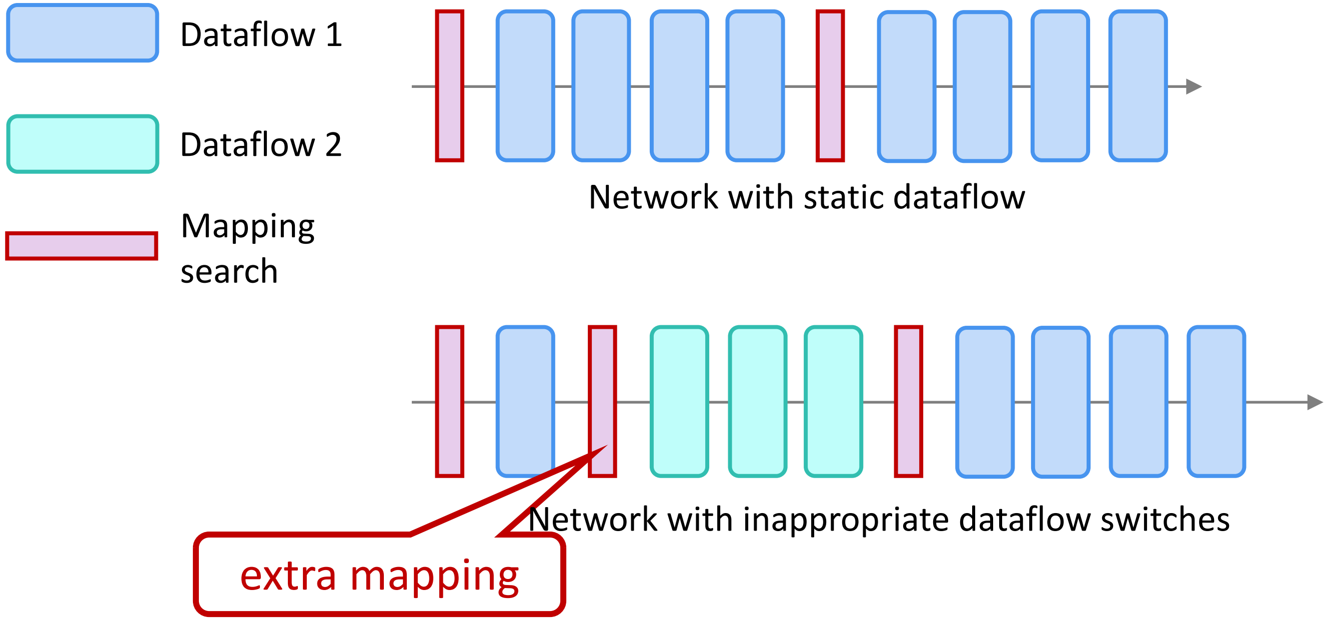
- Static dataflow brings **performance loss against input dynamics**

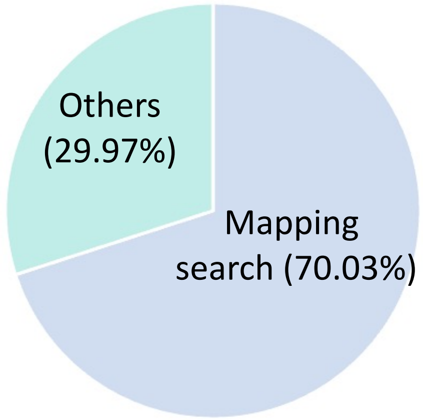


Heuristic Adaptive Dataflow Selection

➢Observation:

- Static dataflow brings **performance loss against input dynamics**
- Different mapping patterns are used in different dataflows, which may lead **extra mapping searches upon dataflow switches**
  - Mapping reuse can be exploited based on network structure



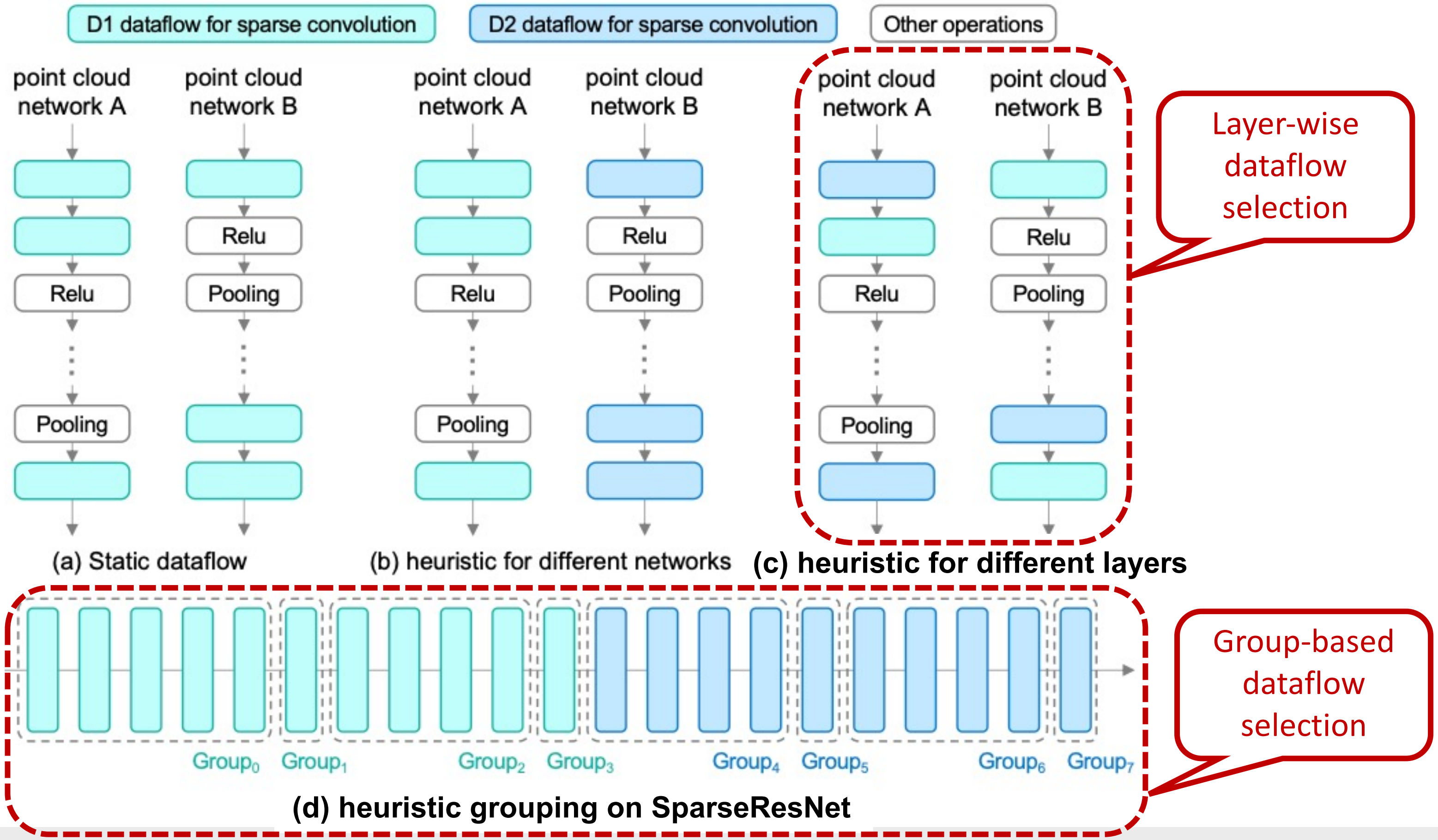


Runtime breakdown of a strided sparse convolutional layer

Heuristic Adaptive Dataflow Selection

➢Solution:

- **Layer-wise dataflow selection** to exploit network performance improvement
- **Group-based dataflow selection** to avoid extra mapping searches



Outline

- Backgrounds and Motivations
- Design Overview
- Solutions
  - Coded-CSR format **mapping storage**
  - Indicator-assisted FGMS (Fused Gather-MM-Scatter) **fusion**
  - Heuristic **adaptive dataflow selection**
- Experiments
- dgSPARSE: Fast and Efficient Processing with Diverse Sparsity

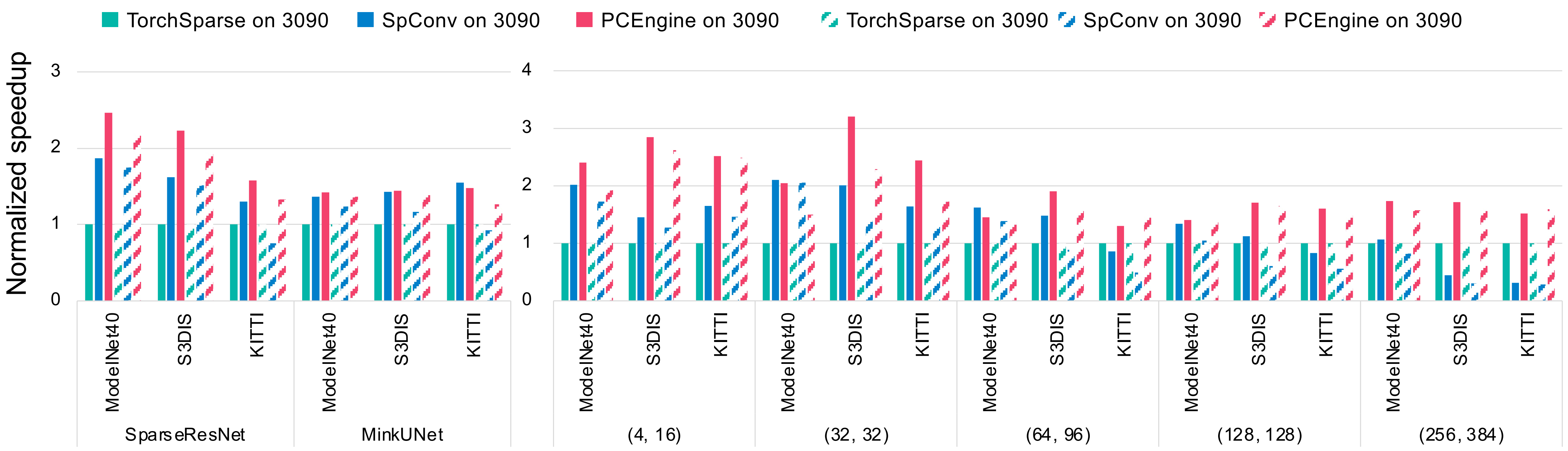
Experiment Settings

- Baselines:
  - TorchSparse v2.0.0 [https://github.com/mit-han-lab/torchsparse]
  - SpConv v2.2.3 [https://github.com/traveller59/spconv]
- Models:
  - SparseResNet [21 sparse convolutional layers]
  - MinkUNet [42 sparse convolutional layers]
- Datasets:
  - ModelNet40 [classification]
  - KITTI [detection]
  - S3DIS [segmentation]

Dataset	Size	Density
ModelNet40	~25k	1.59%
KITTI	~40k	0.04%
S3DIS	~10k	6.92%

- Platforms
  - A 10-core 20-thread Intel Xeon Silver 4210 CPU running @ 2.2GHz
  - An NVIDIA **RTX 3090** GPU and an NVIDIA **RTX 2080** GPU with CUDA 11.1.

Experiment: End-to-End



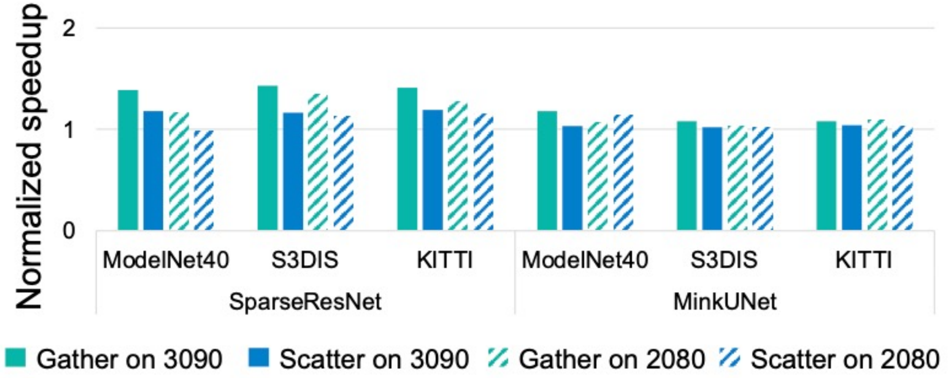
(a) end-to-end performance

(b) kernel performance

End-to-end: 1.64x, 1.23x speedup over TorchSparse, SpConv on average
Sparse convolution: 1.81x, 1.76x speedup over TorchSparse, SpConv on average

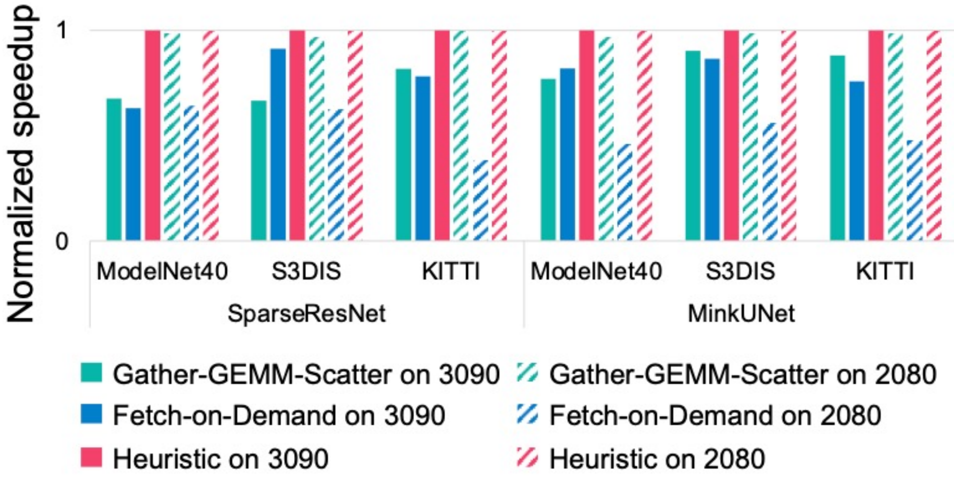
Experiment: Ablation & Additional Studies

Coded-CSR Format Mapping Storage



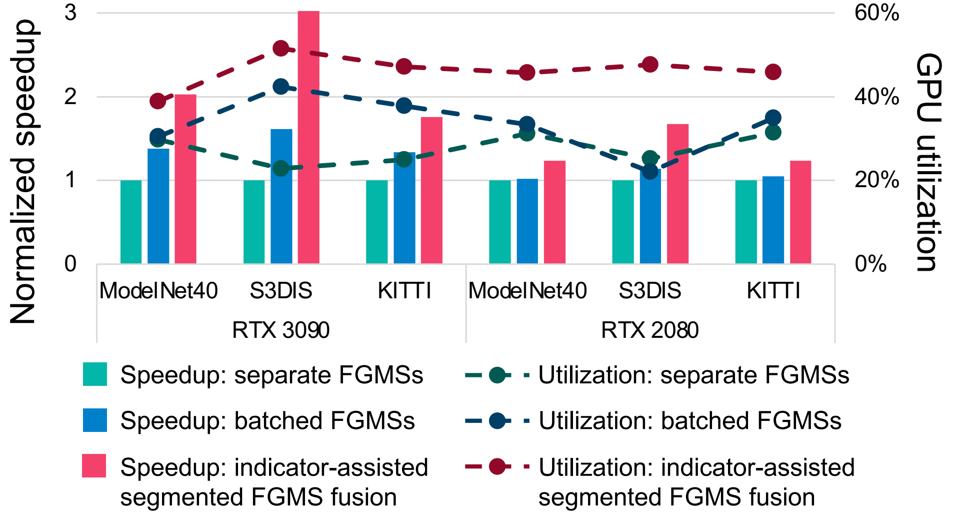
On average 1.21x/1.10x speedup, 21.18% less memory access

Heuristics Adaptive Dataflow Selection



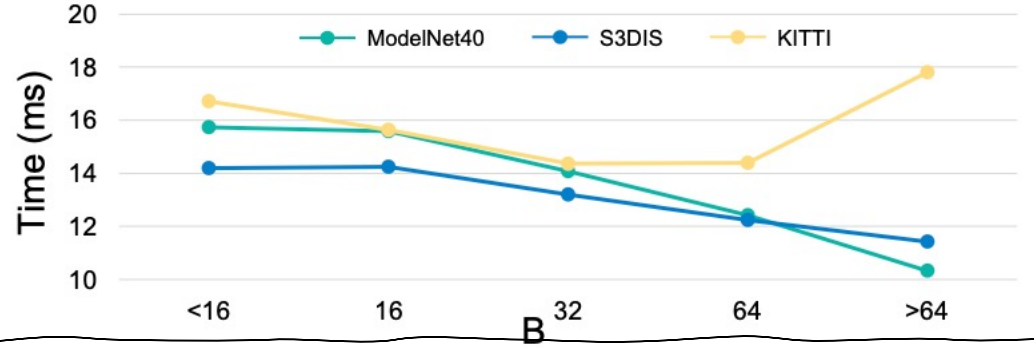
On average 1.15x/1.57x speedup over static dataflows

Indicator-assisted FGMS fusion scheme



On average 1.75x/1.41x speedup, 1.68x/1.40x higher utilization

Threshold choice for dataflow heuristic algorithm



Tuning *B* on different datasets brings further improvement

Key Challenges & Contributions

*FGMS: Fused Gather-MM-Scatter

✗ Redundant memory access caused by mapping storage

✓ Coded-CSR mapping storage

✗ Computation under-utilization caused by sequential FGMSs

✓ Indicator-assisted FGMS fusion scheme

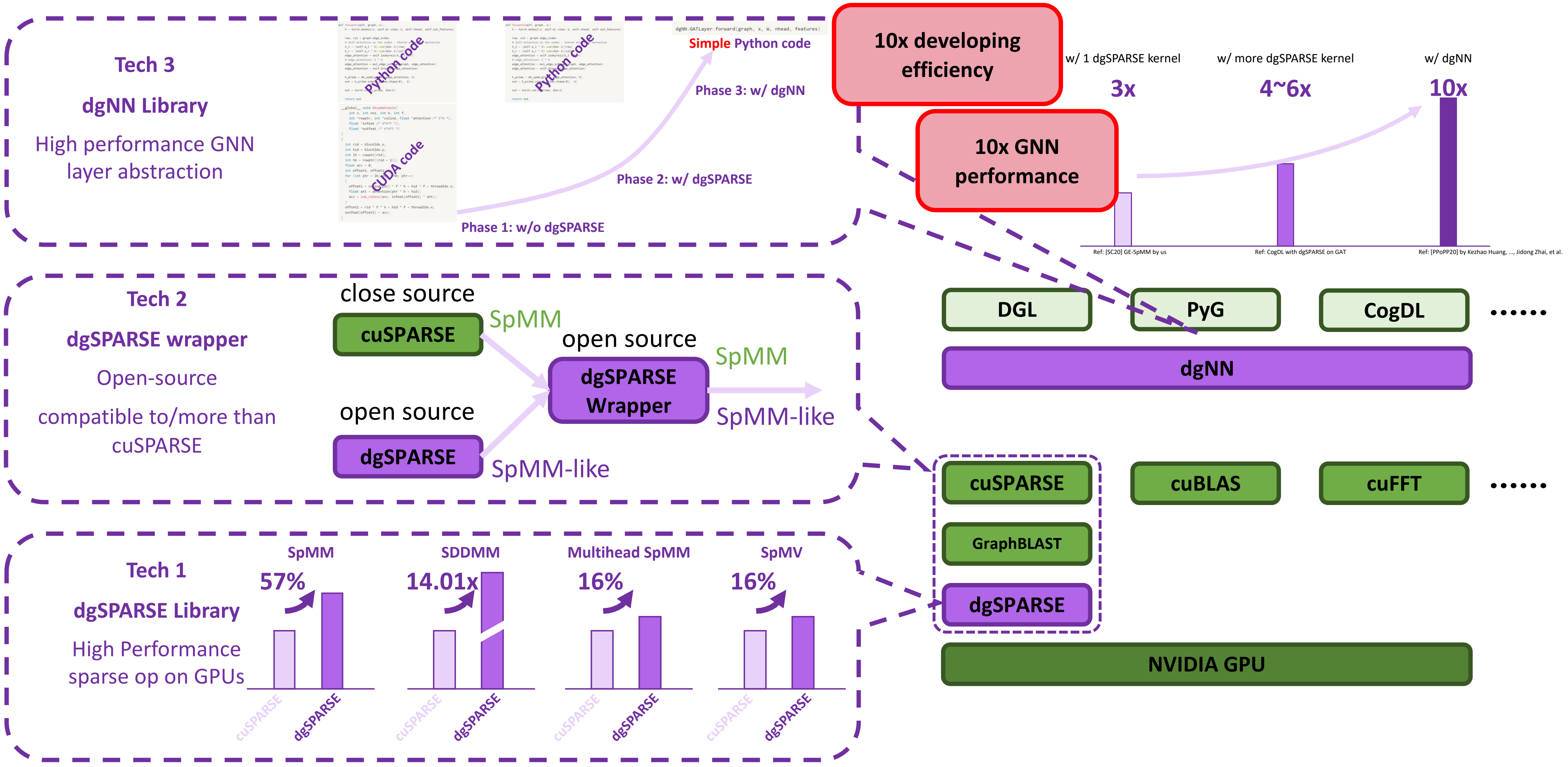
✗ Input dynamics performance loss caused by static dataflow

✓ Heuristic adaptive dataflow selection

Outline

- Backgrounds and Motivations
- Design Overview
- Solutions
  - Coded-CSR format **mapping storage**
  - Indicator-assisted FGMS (Fused Gather-MM-Scatter) **fusion**
  - Heuristic **adaptive dataflow selection**
- Experiments
- dgSPARSE: Fast and Efficient Processing with Diverse Sparsity

dgSPARSE

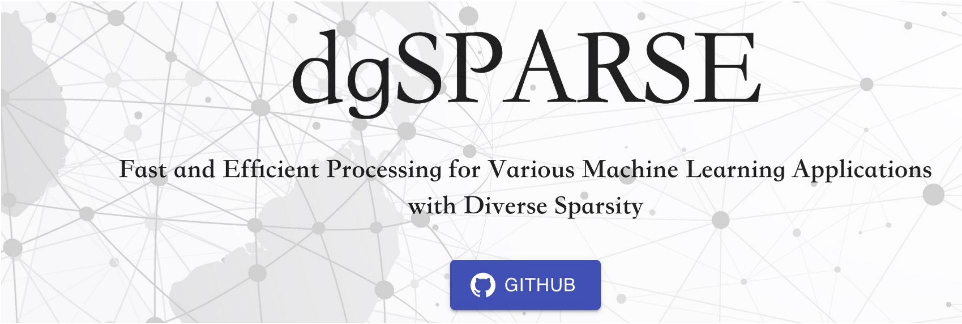


Tech 3
dgNN Library
High performance GNN layer abstraction
Python code
CUDA code
Python code
Simple Python code
Phase 1: w/o dgSPARSE
Phase 2: w/ dgSPARSE
Phase 3: w/ dgNN
10x developing efficiency
10x GNN performance
w/ 1 dgSPARSE kernel
3x
w/ more dgSPARSE kernel
4~6x
w/ dgNN
10x
Ref: [SC20] GE-SpMM by us
Ref: CogDL with dgSPARSE on GAT
Ref: [PPoPP20] by Kezhao Huang, ..., Jidong Zhai, et al.
Tech 2
dgSPARSE wrapper
Open-source
compatible to/more than cuSPARSE
close source
cuSPARSE
SpMM
open source
dgSPARSE Wrapper
SpMM
SpMM-like
open source
dgSPARSE
SpMM-like
DGL
PyG
CogDL
......
dgNN
cuSPARSE
cuBLAS
cuFFT
......
GraphBLAST
dgSPARSE
NVIDIA GPU
Tech 1
dgSPARSE Library
High Performance sparse op on GPUs
SpMM
57%
SDDMM
14.01x
Multihead SpMM
16%
SpMV
16%
cuSPARSE
dgSPARSE
cuSPARSE
dgSPARSE
cuSPARSE
dgSPARSE
cuSPARSE
dgSPARSE

dgSPARSE

➢dgSPARSE papers

- [MLSys 23] Yu, Z., Dai, G., Yang, S., Zhang, G., Zhang, H., Zhu, F., Yang, J., Zhao, J. & Wang, Y. HyperGef: A Framework Enabling Efficient Fusion for Hypergraph Neural Network on GPUs. To appear in Sixth Conference on Machine Learning and Systems (MLSys), 2023.
- [MLSys 23] Hong, K., Yu, Z., Dai, G., Yang, X., Lian, Y., Liu, Z., Xu, N., Dong, Y., & Wang, Y. Exploiting Hardware Utilization and Adaptive Dataflow for Efficient Sparse Convolution in 3D Point Clouds. To appear in Sixth Conference on Machine Learning and Systems (MLSys), 2023.
- [DAC 22] Dai, G., Huang, G., Yang, S., Yu, Z., Zhang, H., Ding, Y., Xie, Y., Yang, H., Wang, Y. Heuristic Adaptability to Input Dynamics for Sparse Matrix-Matrix Multiplication on GPUs. (Best Paper nomination)
- [MLSys 22] Zhang, H., Yu, Z., Dai, G., Huang, G., Ding, Y., Xie, Y., & Wang, Y. Understanding GNN Computational Graph: A Coordinated Computation, IO, and Memory Perspective. arXiv preprint arXiv:2110.09524.
- [ACM-SRC 21] Huang, G., Dai, G., Ding, Y., Wang, Y., Xie, Y. Efficient Sparse Matrix Kernels based on Adaptive Workload-Balancing and Parallel-Reduction. In ACM Student Research Competition (ACM-SRC), 2021.
- [ICCD 21] Yu, Z., Dai, G., Huang, G., Wang, Y., & Yang, H. Exploiting Online Locality and Reduction Parallelism for Sampled Dense Matrix Multiplication on GPUs. To appear in International Conference on Computer Design (ICCD), 2021.
- [MICRO-SRC 20] Huang, G., Dai, G., Wang, Y., & Yang, H. Towards Fast Graph Neural Network Training with Efficient and Framework-Compatible Sparse-Dense Matrix Multiplication. In MICRO-53 Student Research Competition (MICRO-SRC), 2020.
- [SC 20] Huang, G., Dai, G., Wang, Y., & Yang, H. GE-SpMM: General-purpose Sparse Matrix-Matrix Multiplication on GPUs for Graph Neural Networks. In International Conference for High Performance Computing, Networking, Storage and Analysis (SC), pp. 1-12, 2020.



https://dgsparse.github.io/

Thank you
Q&A

Ke Hong

hongk21@mails.tsinghua.edu.cn