HyperGef: A Framework Enabling Efficient Fusion For Hypergraph Neural Network on GPU

Zhongming Yu (https://fishmingyu.github.io),

Guohao Dai, Shang Yang, Genghan Zhang, Hengrui Zhang, Feiwen Zhu, Jun Yang,

Jishen Zhao, Yu Wang

*Work was done during the internship at Nvidia

Executive Summary

- Design Principles:
  - Workload-Balanced Efficient Partition
  - Fusion Workload Aware Format
  - Shared Memory Aware Grouping
- Compared to SOTA Designs:
  - 2.25x-3.39x end-to-end speedup over DGL/PyG on average
  - 3.31x kernel-wise speedup over cuSPARSE on average

Outline

- Background and Motivation
- Challenges and Solutions
  - Partition: Workload-Balanced Efficient Partition
  - Format: Fusion Workload Aware Format
  - Writing conflicts: Shared Memory Aware Grouping
- Experiments

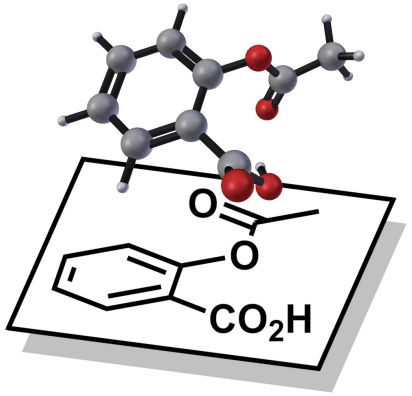
Outline

- Background and Motivation
- Challenges and Solutions
  - Partition: Workload-Balanced Efficient Partition
  - Format: Fusion Workload Aware Format
  - Writing conflicts: Shared Memory Aware Grouping
- Experiments

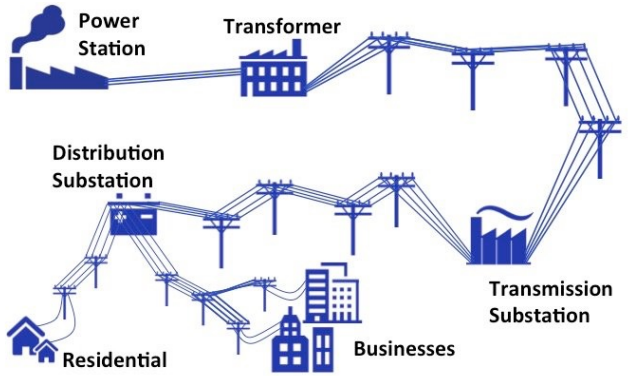
Background and Motivation

- Various GNN applications

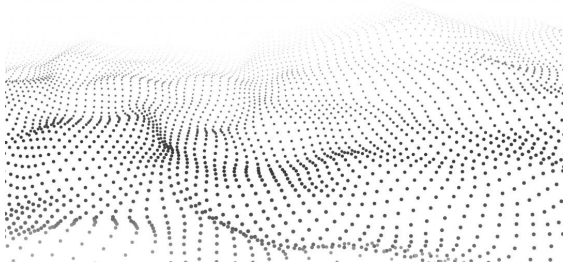
Social Network



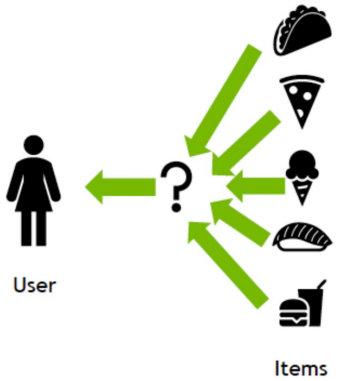
Drug Discovery



Power Grid



Point Cloud

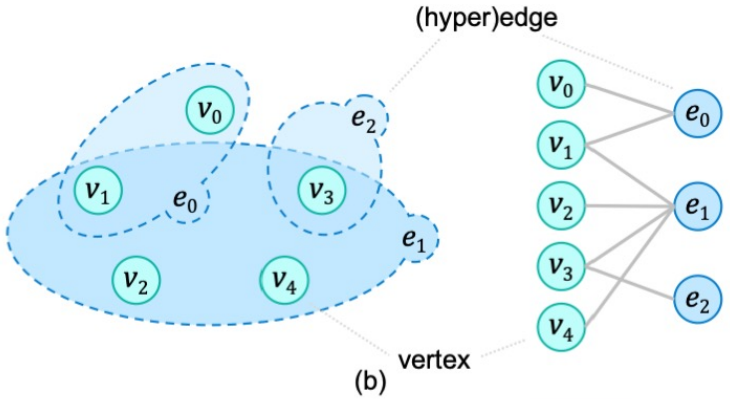


Recommendation System

Financial Service

Background and Motivation

- A prospective branch of GNN
  - Capability of representing high-order structure
  - Higher performance



	v_0	v_1	v_2	v_3	v_4
e_0	1	1	0	0	0
e_1	0	1	1	1	1
e_2	0	0	0	1	0

H^T: edge to vertex

	e_0	e_1	e_2
v_0	1	0	0
v_1	1	1	0
v_2	0	1	0
v_3	0	1	1
v_4	0	1	0

H: vertex to edge

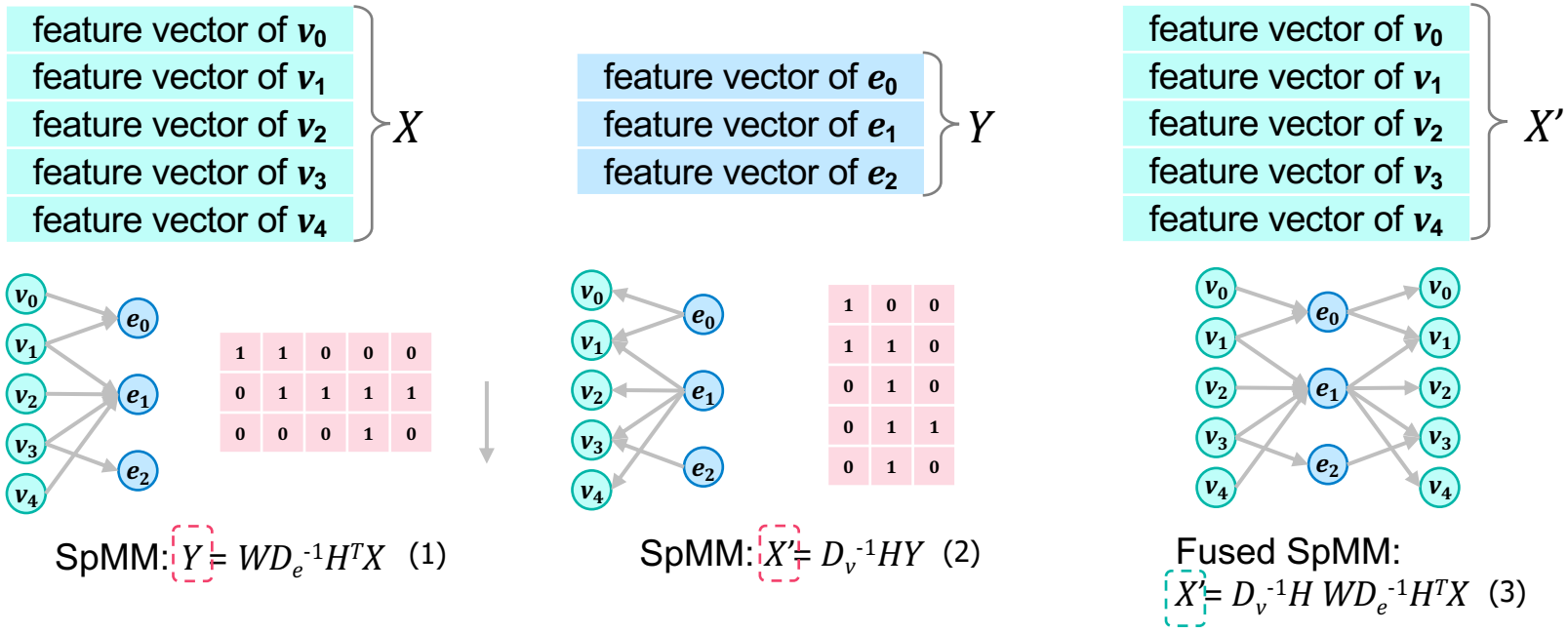
(c)

Dataset	Metric	Category						
		MF		GNN			HGNN	
		SoReg	SocialMF	DiffNet	NGCF	LightGCN	DHCF	MHCN
Douban	Precision@10	0.026%	0.026%	0.910%	0.699%	1.121%	0.958%	1.060%
	Recall@10	0.194%	0.168%	3.060%	0.446%	4.007%	3.311%	3.848%
	NDCG@10	0.094%	0.093%	2.035%	1.642%	2.770%	2.204%	2.605%
	HR@10	0.168%	0.166%	3.449%	2.649%	4.250%	3.634%	4.111%
Yelp	Precision@10	0.033%	0.055%	3.543%	4.531%	6.039%	4.637%	5.836%
	Recall@10	0.127%	0.204%	2.588%	3.467%	4.598%	3.563%	4.473%
	NDCG@10	0.080%	0.133%	3.848%	5.072%	6.742%	5.161%	6.572%
	HR@10	0.123%	0.205%	2.422%	3.097%	4.128%	3.169%	3.989%

Method	Classification Accuracy
PointNet (Qi et al. 2017a)	89.2%
PointNet++ (Qi et al. 2017b)	90.7%
PointCNN (Li et al. 2018)	91.8%
SO-Net (Li, Chen, and Lee 2018)	93.4%
HGNN	**96.7%**

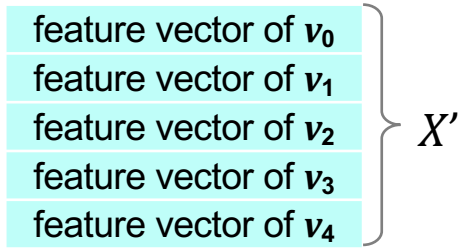
Background and Motivation

- Fusion benefits a lot in
  - Saving memory accesses
  - Eliminating the temporary tensor

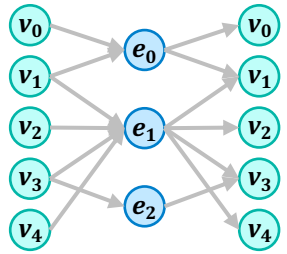


Background and Motivation

- Efficient fusion is hard for hypergraph neural networks
  - Variable shape
  - Sparse data attribute
    - Sparse tensor
    - Imbalance

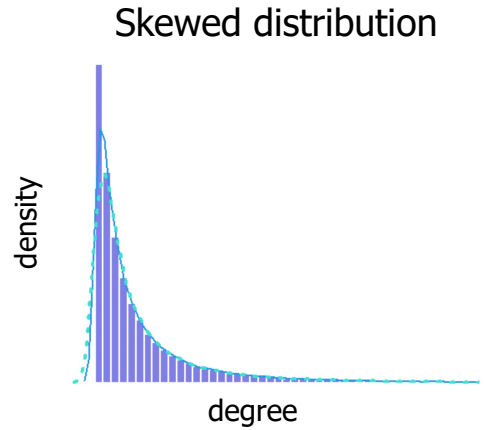


Different hypergraphs have varied vertexes and hyperedges number



Fused SpMM:

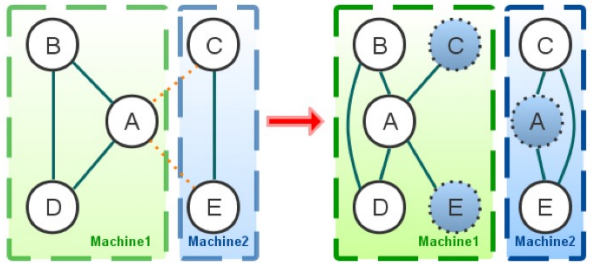
$$X' = D_v^{-1} H W D_e^{-1} H^T X \quad (3)$$



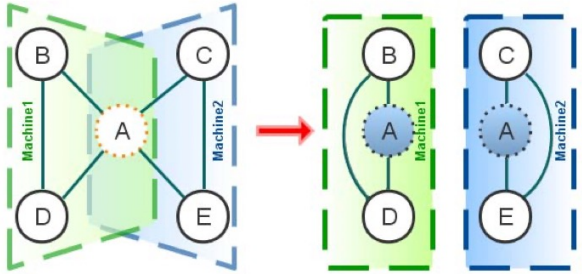
Outline

- Background and Motivation
- Challenges and Solutions
  - Partition: Workload-Balanced Efficient Partition
  - Format: Fusion Workload Aware Format
  - Writing conflicts: Shared Memory Aware Grouping
- Experiments

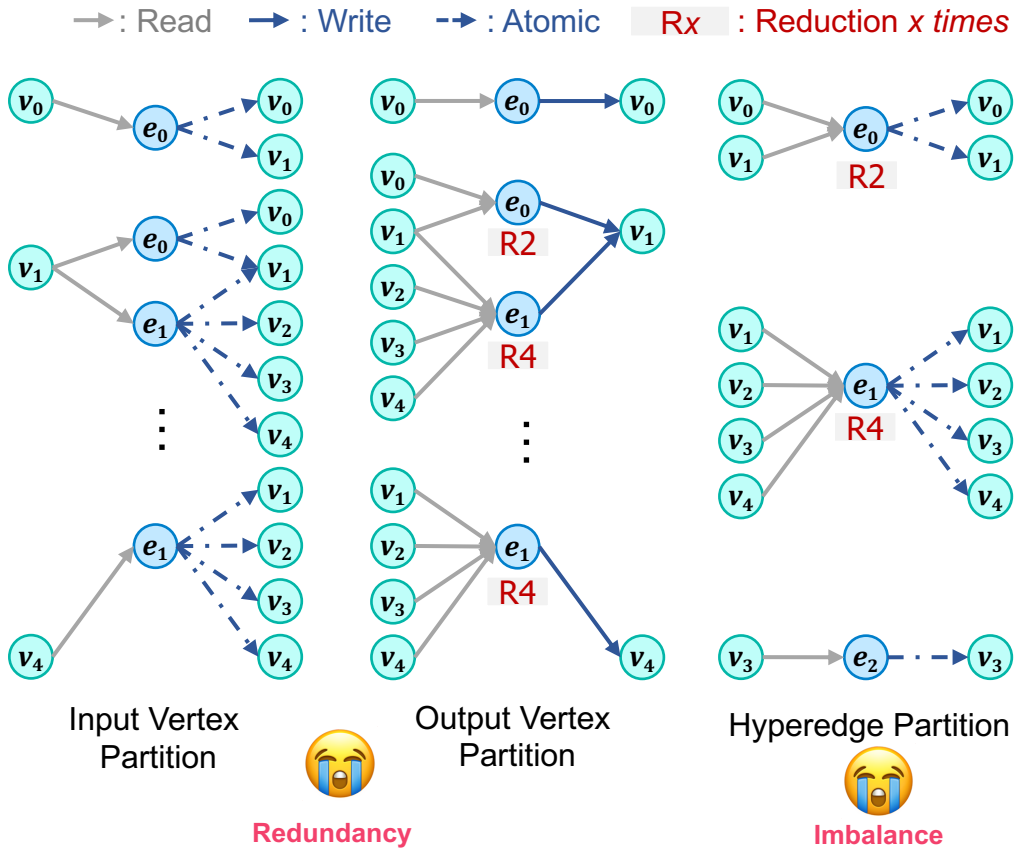
Challenge 1: Inefficient Partition



(a) Edge-Cut

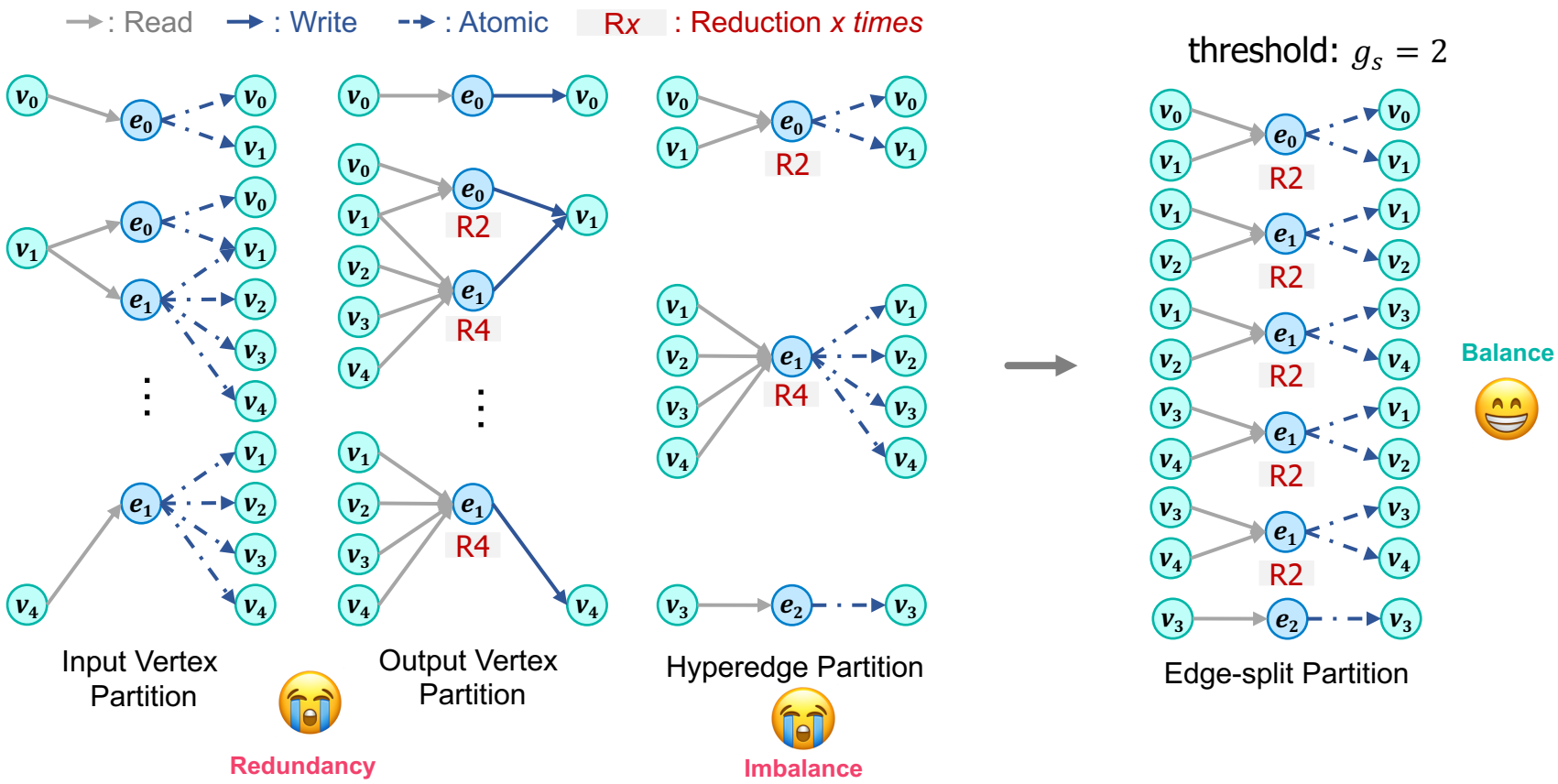


(b) Vertex-Cut



Solution 1

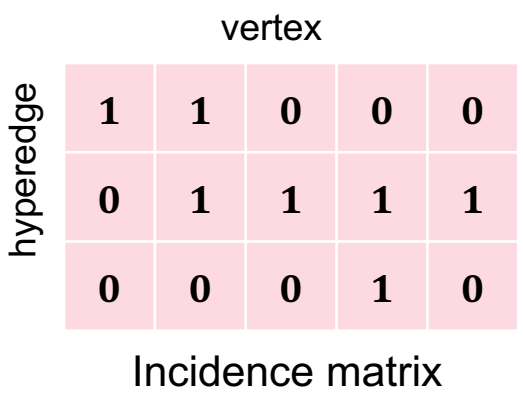
- Edge-split workload balance partition

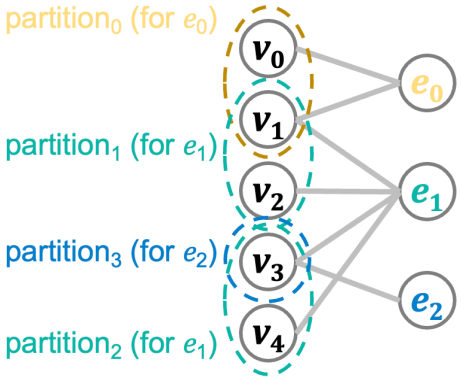


Outline

- Background and Motivation
- Challenges and Solutions
  - Partition: Workload-Balanced Efficient Partition
  - Format: Fusion Workload Aware Format
  - Writing conflicts: Shared Memory Aware Grouping
- Experiments

Challenge 2: Workload-Agnostic format





no workload identifier for efficient partition

hyperedge	0	2	6	7			
vertex	0	1	1	2	3	4	3

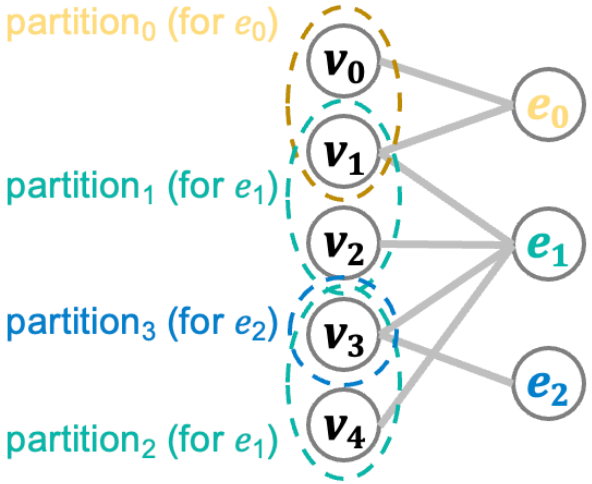
Compressed Sparse Row (CSR)

hyperedge	0	2	6	7			
vertex	0	1	1	2	3	4	3

Dynamic identify partition is harmful to runtime kernel's performance

Solution 2

- Workload-aware data format



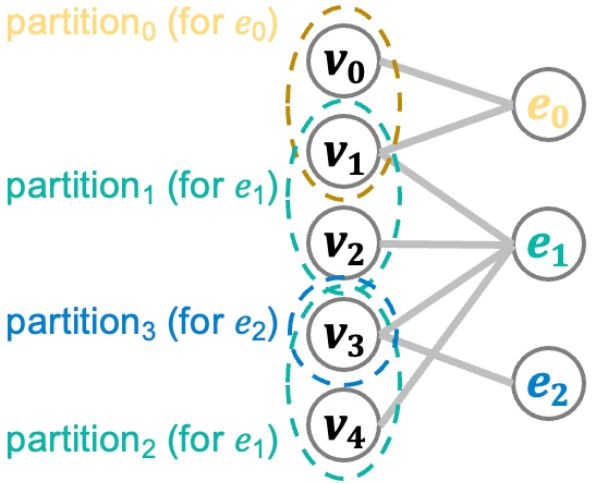
(1)

hyperedge	0	2	6	7			
vertex	0	1	1	2	3	4	3

CSR representation

Solution 2

- Workload-aware data format



(1)

hyperedge	0	2	6	7			
vertex	0	1	1	2	3	4	3

CSR representation

edge-split partition

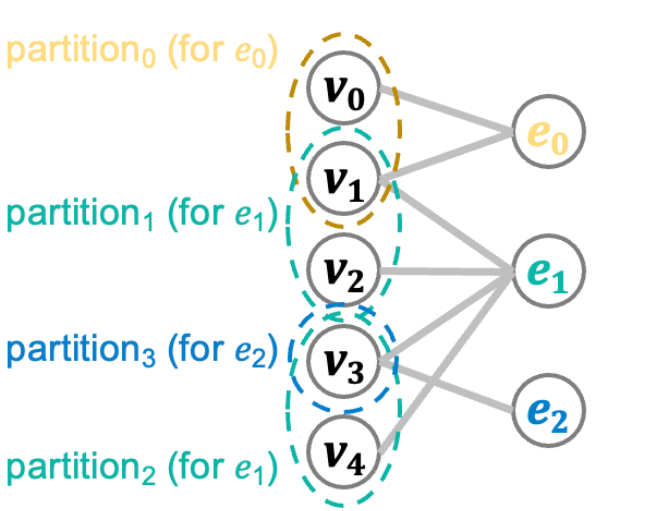
(2)

hyperedge	0	2	6	7			
partition	0	2	4	6	7		
vertex	0	1	1	2	3	4	3

partition$_1$ (for e_1) partition$_2$ (for e_1)

Solution 2

- Workload-aware data format



(1)

hyperedge	0	2	6	7			
vertex	0	1	1	2	3	4	3

CSR representation

edge-split partition

hyperedge	0	2	6	7			
partition	0	2	4	6	7		
vertex	0	1	1	2	3	4	3

partition$_1$ (for e_1) partition$_2$ (for e_1)

(2)

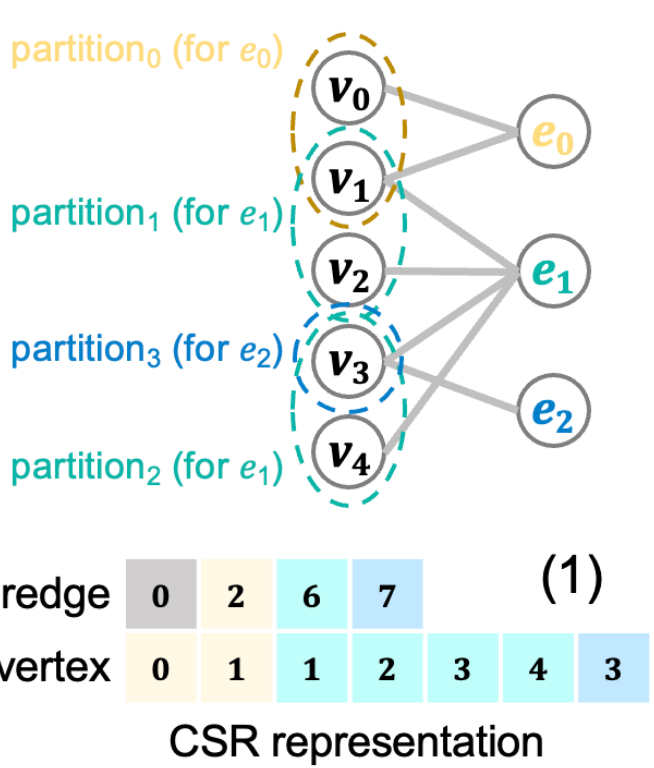
workload calculation

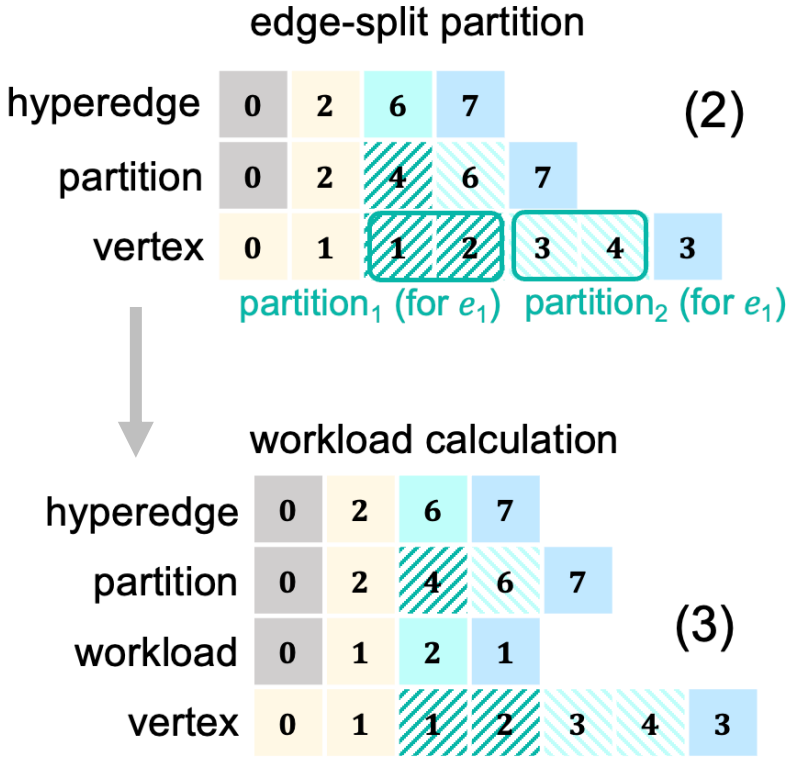
hyperedge	0	2	6	7			
partition	0	2	4	6	7		
workload	0	1	2	1			
vertex	0	1	1	2	3	4	3

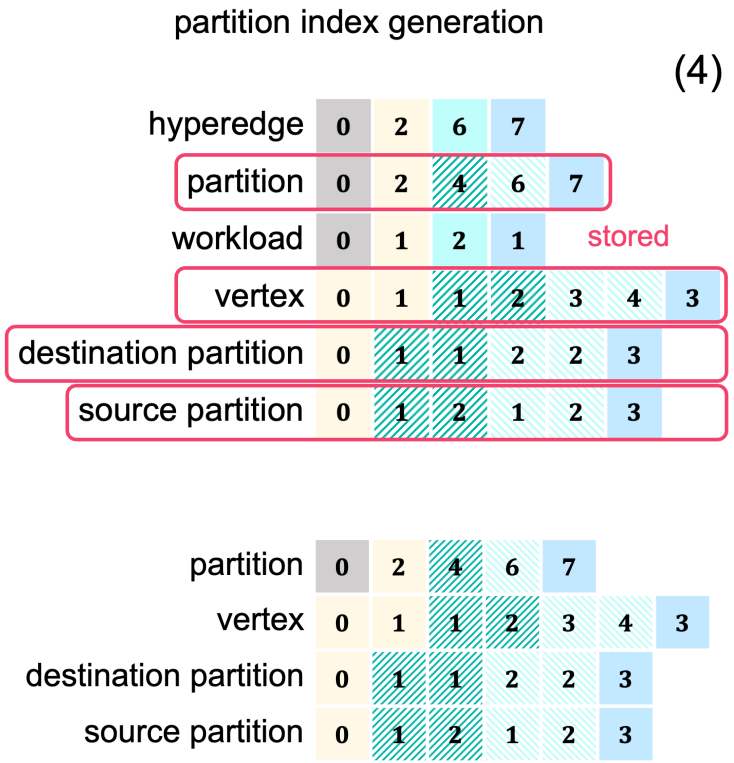
(3)

Solution 2

- Workload-aware data format

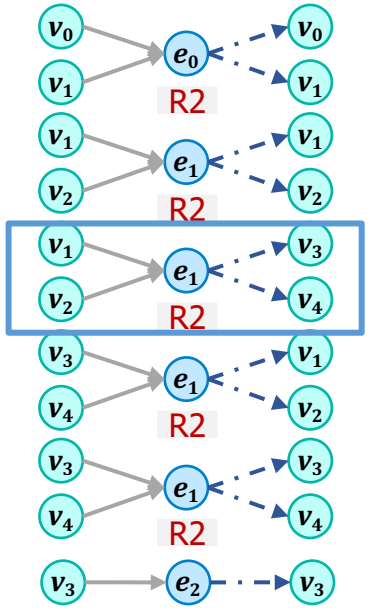






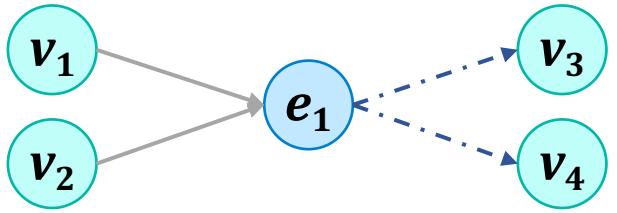
Solution 2

- Workload-aware data format-Example



Overall Partition mapping 2

destination partition	0	1	1	2	2	3
source partition	0	1	2	1	2	3



Partition 1

partition	0	2	4	6	7		
vertex	0	1	1	2	3	4	3

Vertexes 1&2

Partition 2

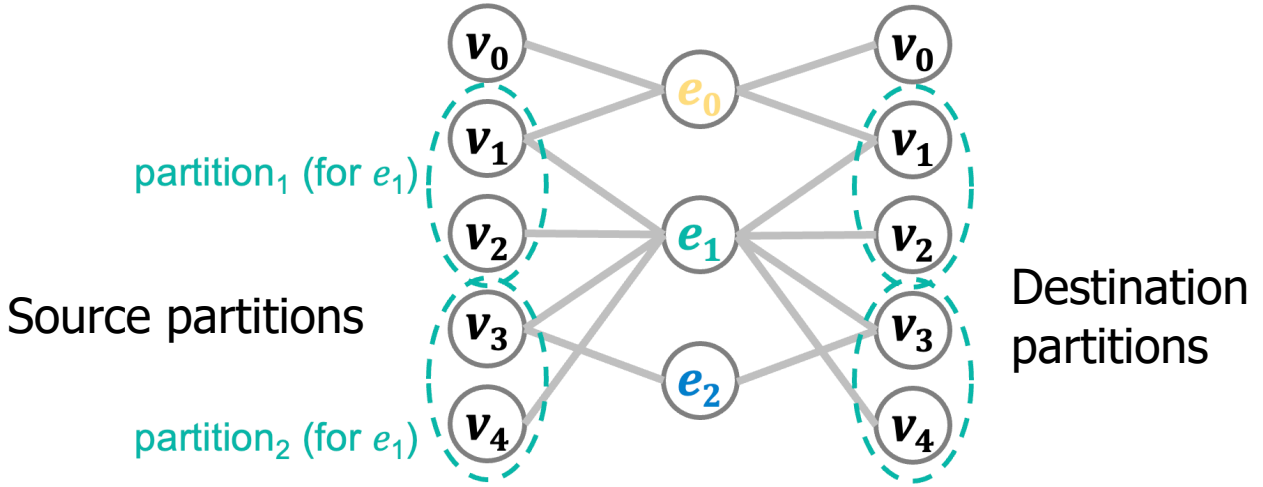
partition	0	2	4	6	7		
vertex	0	1	1	2	3	4	3

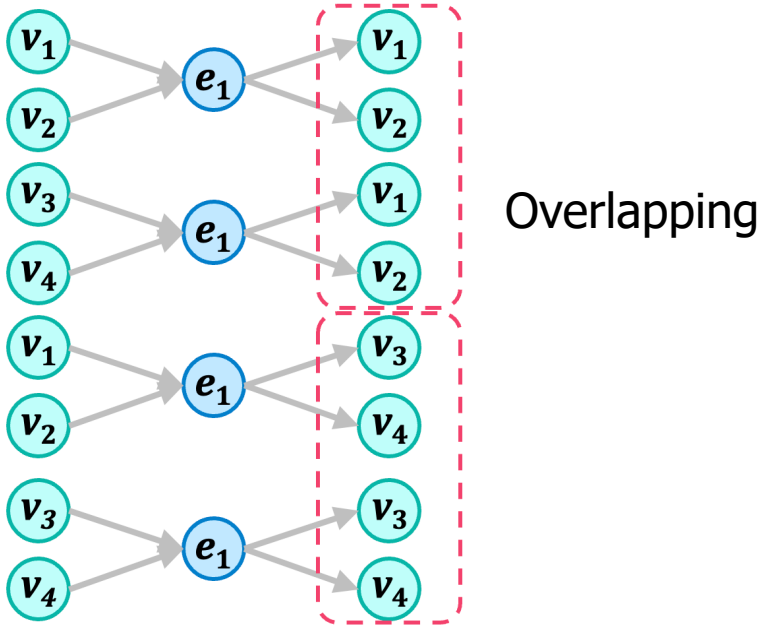
Vertexes 3&4

Outline

- Background and Motivation
- Challenges and Solutions
  - Partition: Workload-Balanced Efficient Partition
  - Format: Fusion Workload Aware Format
  - Writing conflicts: Shared Memory Aware Grouping
- Experiments

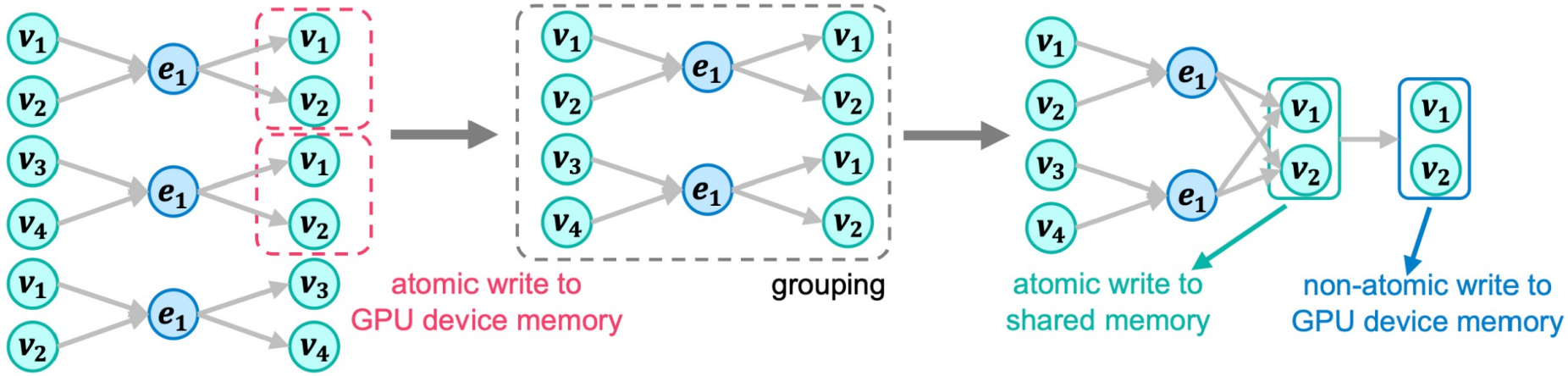
Challenge 3: Heavy atomics operations





Solution 3

- Shared-memory-aware grouping



Outline

- Background and Motivation
- Challenges and Solutions
  - Partition: Workload-Balanced Efficient Partition
  - Format: Fusion Workload Aware Format
  - Writing conflicts: Shared Memory Aware Grouping
- Experiments

Experimental Setup

- Baselines
  - PyG 2.0, DGL 0.9
  - cuSPARSE (CUDA 11.3)
- Models
  - HGNN(AAAI'19)
  - UniGNN(IJCAI'21)
    - UniGIN
    - UniGCNII
- Platforms
  - NVIDIA 3090 GPU (CUDA 11.3)
  - Intel Xeon Silver 4210, Ubuntu 20.04
- Datasets
  - Datasets from [Allset (ICLR'22)]

$$\mathbf{h}_e^{(l)} = \frac{1}{\sqrt{d_{v_j}}} \sum_{v_j \in e} \left\{ \mathbf{x}_j^{(l-1)} \right\}$$

$$\mathbf{x}_i^{(l)} = \frac{1}{\sqrt{d_{v_i}}} \sum_{e \in \widetilde{\mathcal{N}}(v_i)} \left\{ \frac{w_e}{d_e} \mathbf{h}_e^{(l)} \mathbf{\Theta} \right\}$$

HGNN

$$\mathbf{h}_e^{(l)} = \sum_{v_j \in e} \mathbf{x}_j^{(l-1)}$$

$$\mathbf{x}_i^{(l)} = \left((1+\epsilon)\mathbf{x}_i^{l-1} + \sum_{e \in \widetilde{\mathcal{N}}(v_i)} \mathbf{h}_e^{(l)} \right) \mathbf{\Theta}$$

UniGIN

$$\mathbf{h}_e^{(l)} = \frac{1}{\sqrt{d_{v_j}}} \sum_{v_j \in e} \left\{ \mathbf{x}_j^{(l-1)} \right\}$$

$$\widetilde{\mathbf{x}}_i = \frac{1}{\sqrt{d_e}} \sum_{e \in \widetilde{\mathcal{N}}(v_i)} \left\{ \mathbf{h}_e^{(l)} \right\}$$

$$\mathbf{x}_i^{(l)} = (1-\alpha)\widetilde{\mathbf{x}}_i + \alpha \mathbf{x}_i^{(l-1)} \left\{ (1-\beta)\mathbf{I} + \beta \mathbf{\Theta} \right\}$$

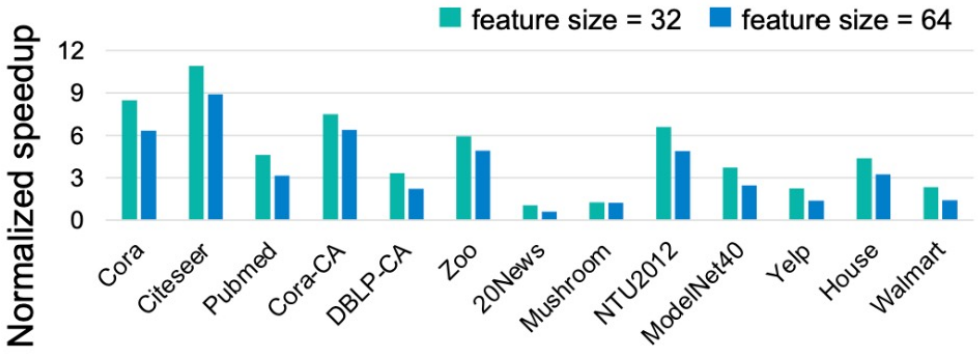
UniGCNII

Table 3. Hypergraph datasets used in our experiments.

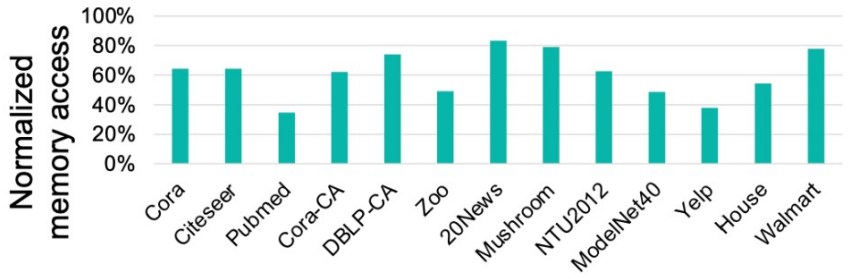
	Cora	Citeseer	Pubmed	Cora-CA	DBLP-CA	Zoo	20News	Mushroom	NTU2012	ModelNet40	Yelp	House	Walmart
#Vertex	2708	3312	19717	2708	41302	101	16242	8124	2012	12311	50758	1290	88860
#Edge	1579	1079	7963	1072	22363	43	100	298	2012	12311	679302	341	69906
#Feature	1433	3703	500	1433	1425	16	100	22	100	100	1862	100	100
#Class	7	6	3	7	6	7	4	2	67	40	9	2	11
max $\|e\|$	5	26	171	43	202	93	2241	1808	5	5	2838	81	25

Overall Performance

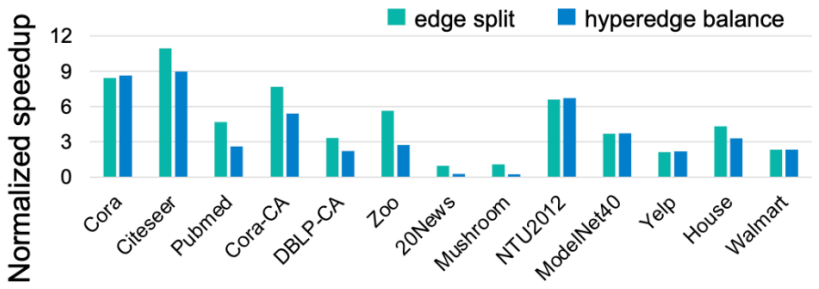
- **2.25x-3.39x** end-to-end speedup over DGL/PyG on average
- **3.31x** kernel-wise speedup over cuSPARSE on average



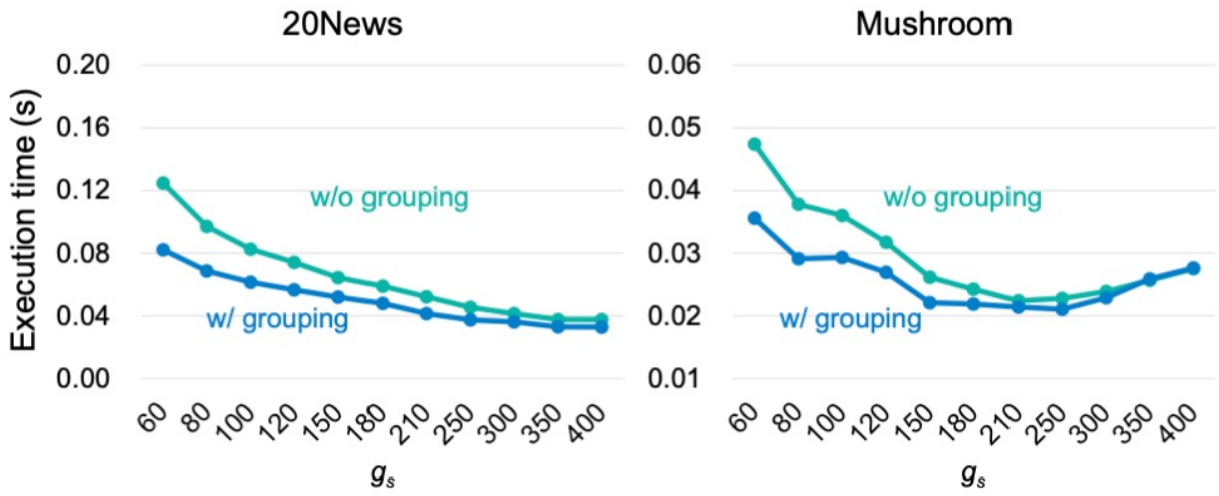
Ablation Study



- **39%** global memory access average saving compared to cuSPRARSE-based SpMM



- **1.53**× average speedup brought by edge-split partition



- Kernels using shared memory exhibit better performance compared to those without shared memory grouping. In the overall kernel, we tune the threshold parameter g_s to get the best configuration.

Summary

- The first framework targets HypergraphNN acceleration on GPU
- Make fusion kernel efficient on HypergraphNN
  - Workload-Balanced Efficient Partition
  - Fusion Workload Aware Format
  - Shared Memory Aware Grouping
- Significant Performance Gains
  - 2.25x-3.39x end-to-end speedup over DGL/PyG on average
  - 3.31x kernel-wise speedup over cuSPARSE on average

Github Code

HyperGef: A Framework Enabling Efficient Fusion For Hypergraph Neural Network on GPU

Zhongming Yu (https://fishmingyu.github.io),

Guohao Dai, Shang Yang, Genghan Zhang, Hengrui Zhang, Feiwen Zhu, Jun Yang,

Jishen Zhao, Yu Wang

*Work was done during the internship at Nvidia