Comprehensive Evaluation of GNN Training Systems: A Data Management Perspective

Hao Yuan, Yajiong Liu, Yanfeng Zhang, Xin Ai,
Qiange Wang, Chaoyi Chen, Yu Gu, Ge Yu

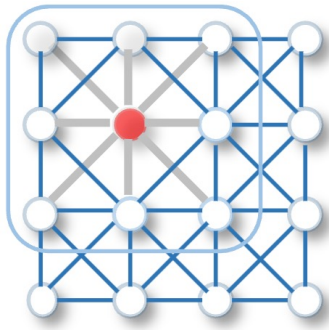
Northeastern University, China

VLDB 2024

Graph Neural Network (GNN)

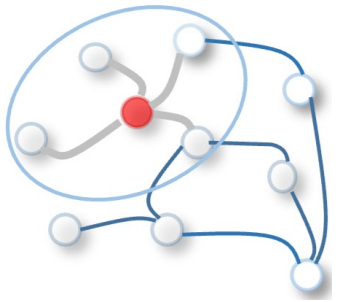
why is it emerging?

Conventional CNN Model



Regular data in Euclidean space
(Learning information: **Euclidean distance**)

Emerging GNN Model



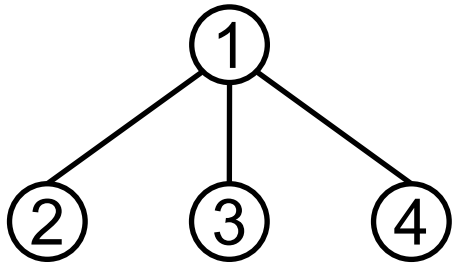
Irregular data in non-Euclidean space
(Learning information: **Relationship)**

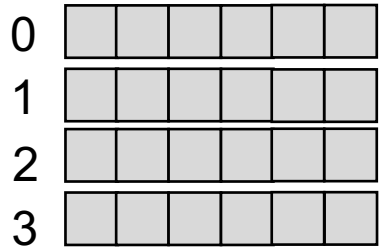
Image source: A Comprehensive Survey on Graph Neural Networks (TNNLS'20)

Graph Neural Network (GNN)

GNN algorithm

Input data

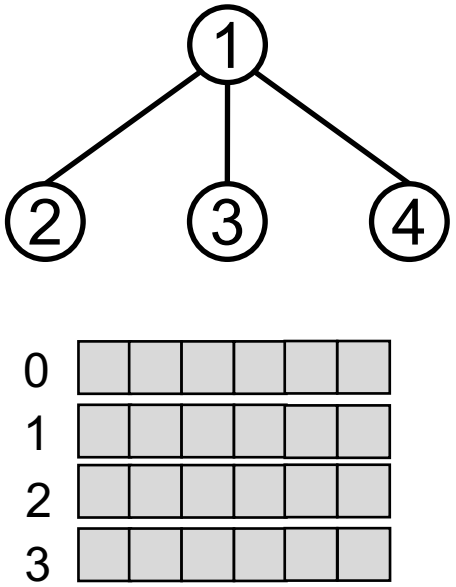




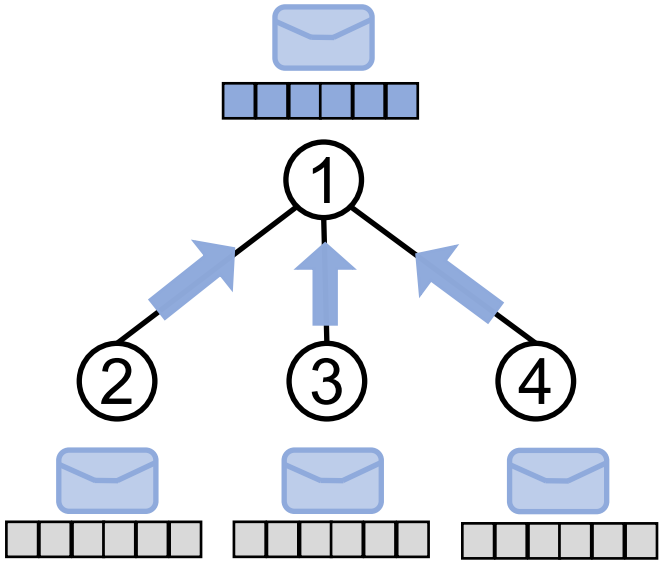
Graph Neural Network (GNN)

GNN algorithm

Input data

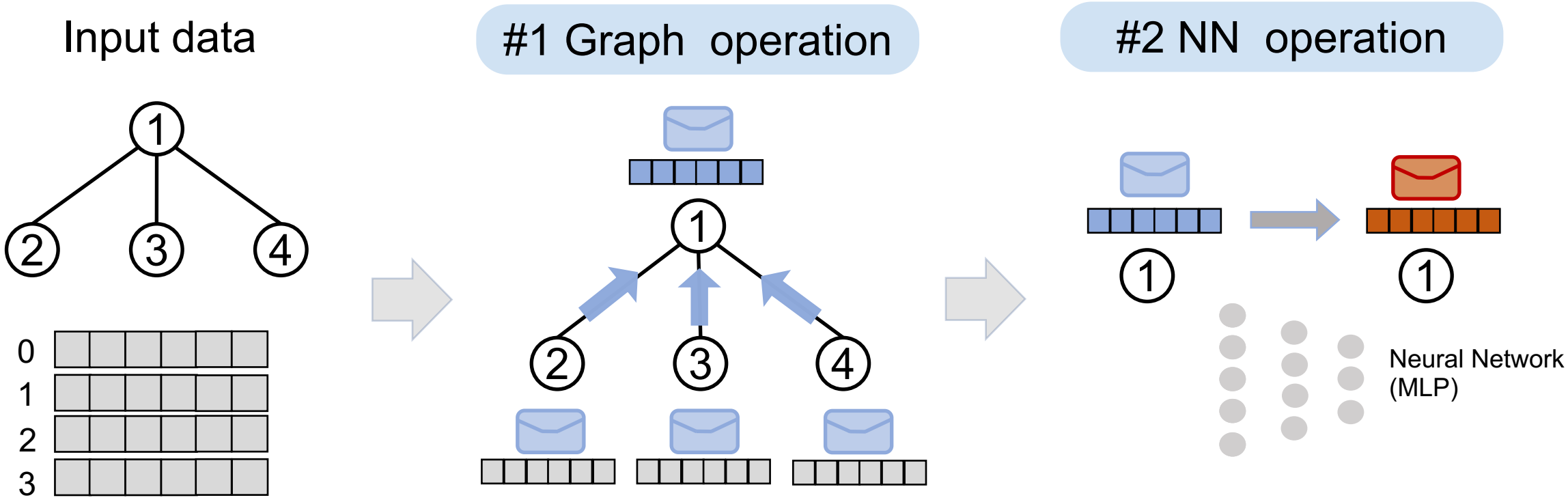


#1 Graph operation



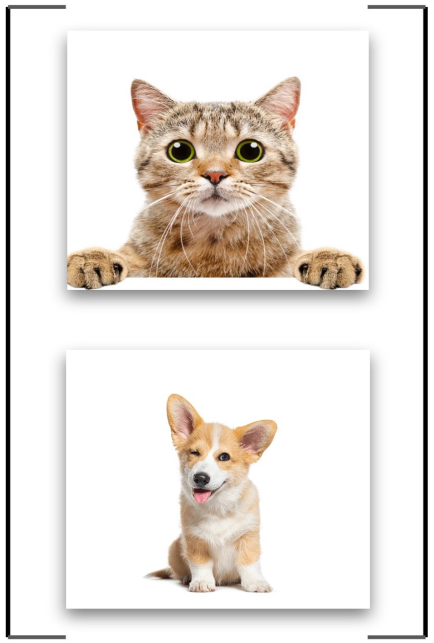
Graph Neural Network (GNN)

GNN algorithm

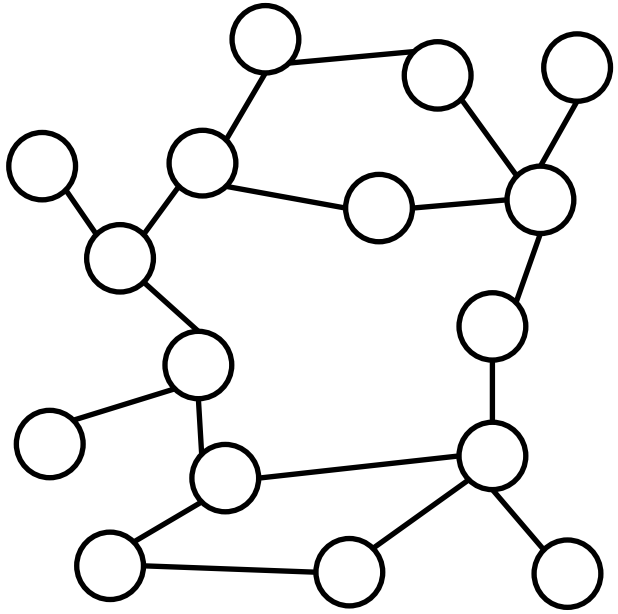


Graph Neural Network (GNN)

Dependency of GNN data samples



DNN inputs



GNN inputs

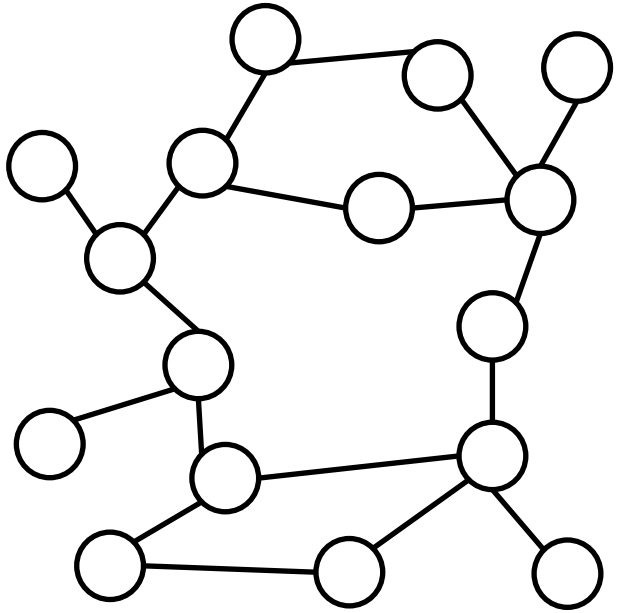
Independence between data samples

Complex dependencies between data samples

Graph Neural Network (GNN)

Dependency of GNN data samples

GNNs require additional **data management** steps

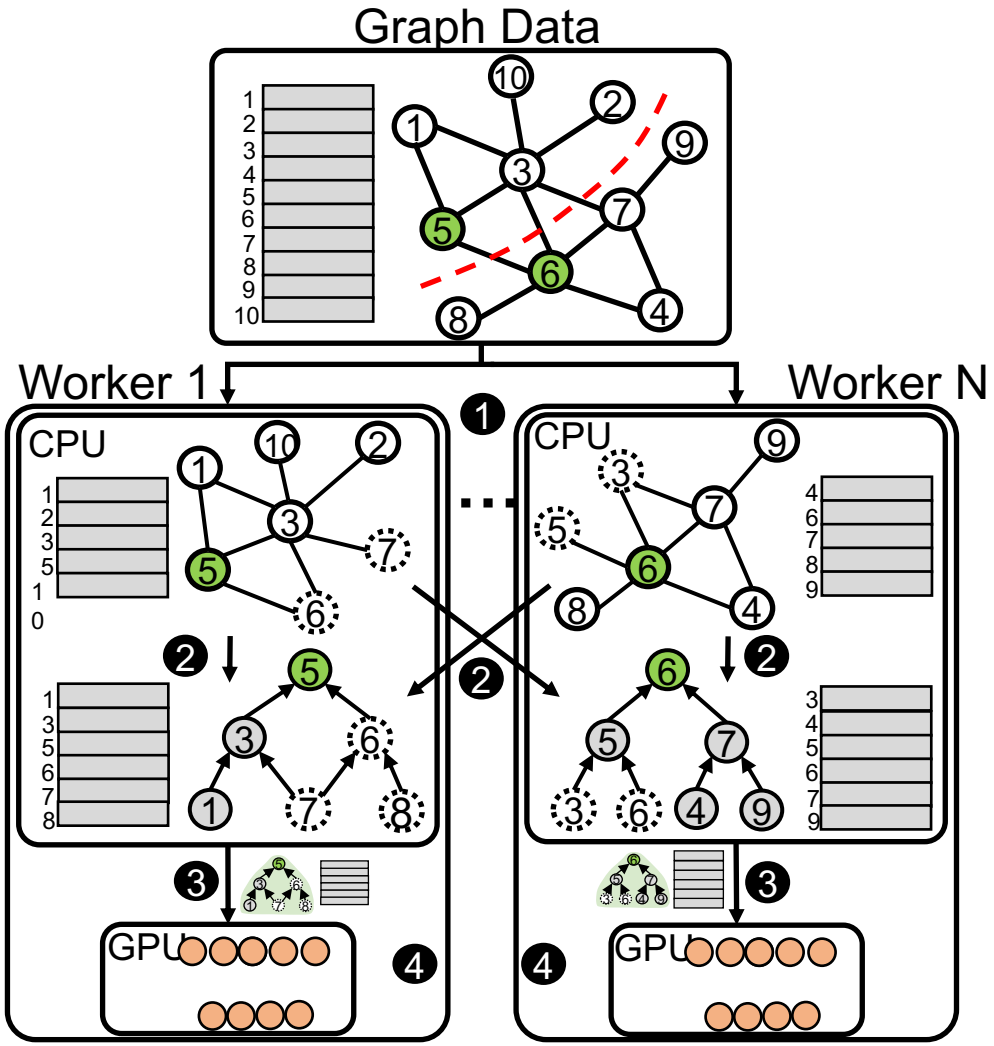


GNN inputs

Complex dependencies between data samples

End-to-End GNN training

Distributed GNN training processing



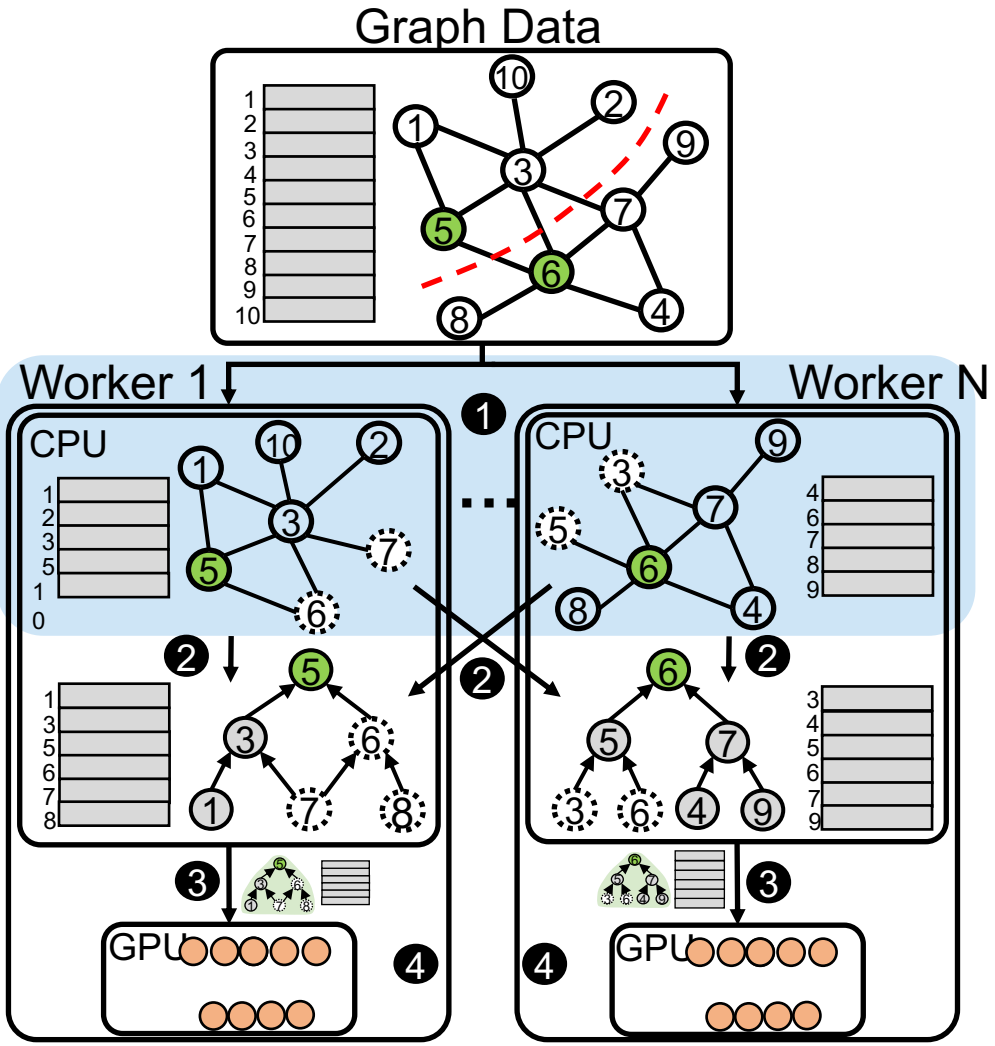
1. Data partitioning
2. Batch preparation
3. Data transferring
4. NN computation

Data management

Actual computation

End-to-End GNN training

Distributed GNN training processing



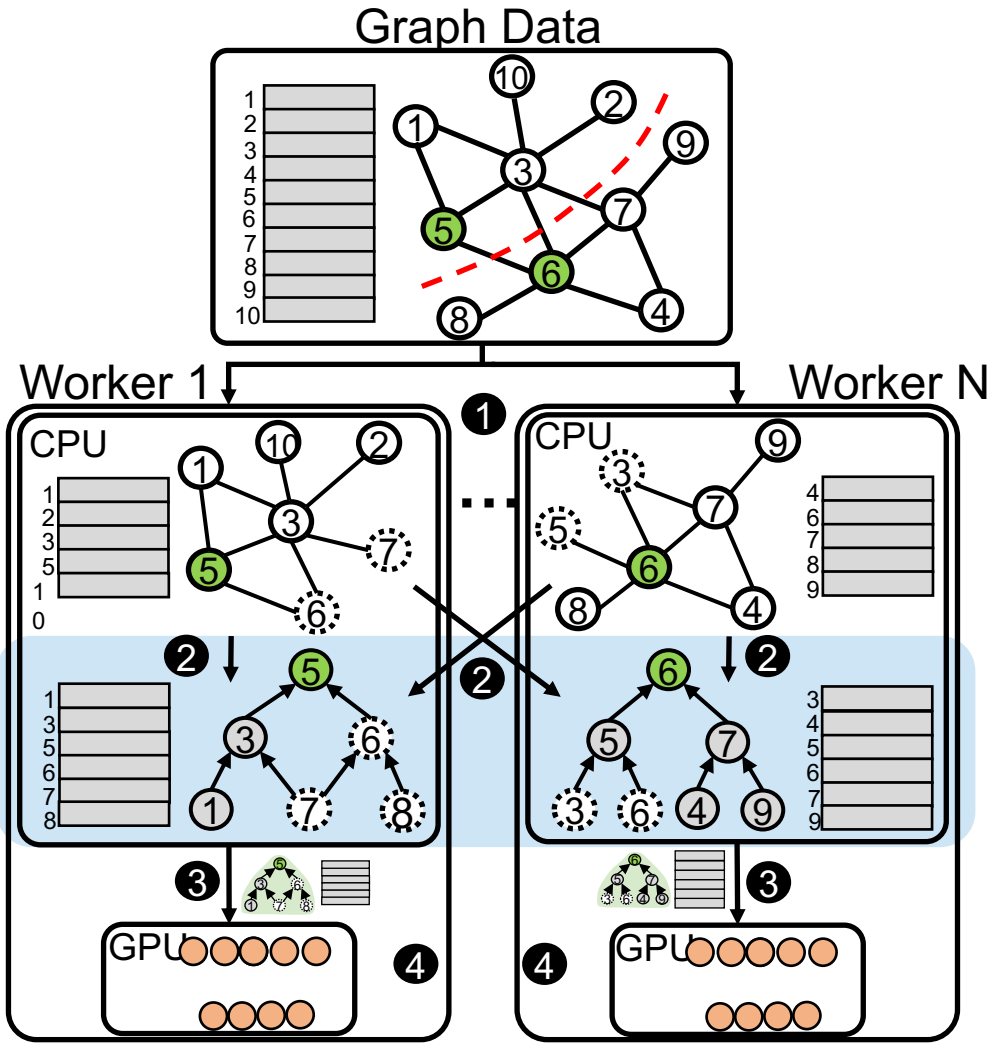
1. Data partitioning
2. Batch preparation
3. Data transferring
4. NN computation

Data management

Actual computation

End-to-End GNN training

Distributed GNN training processing



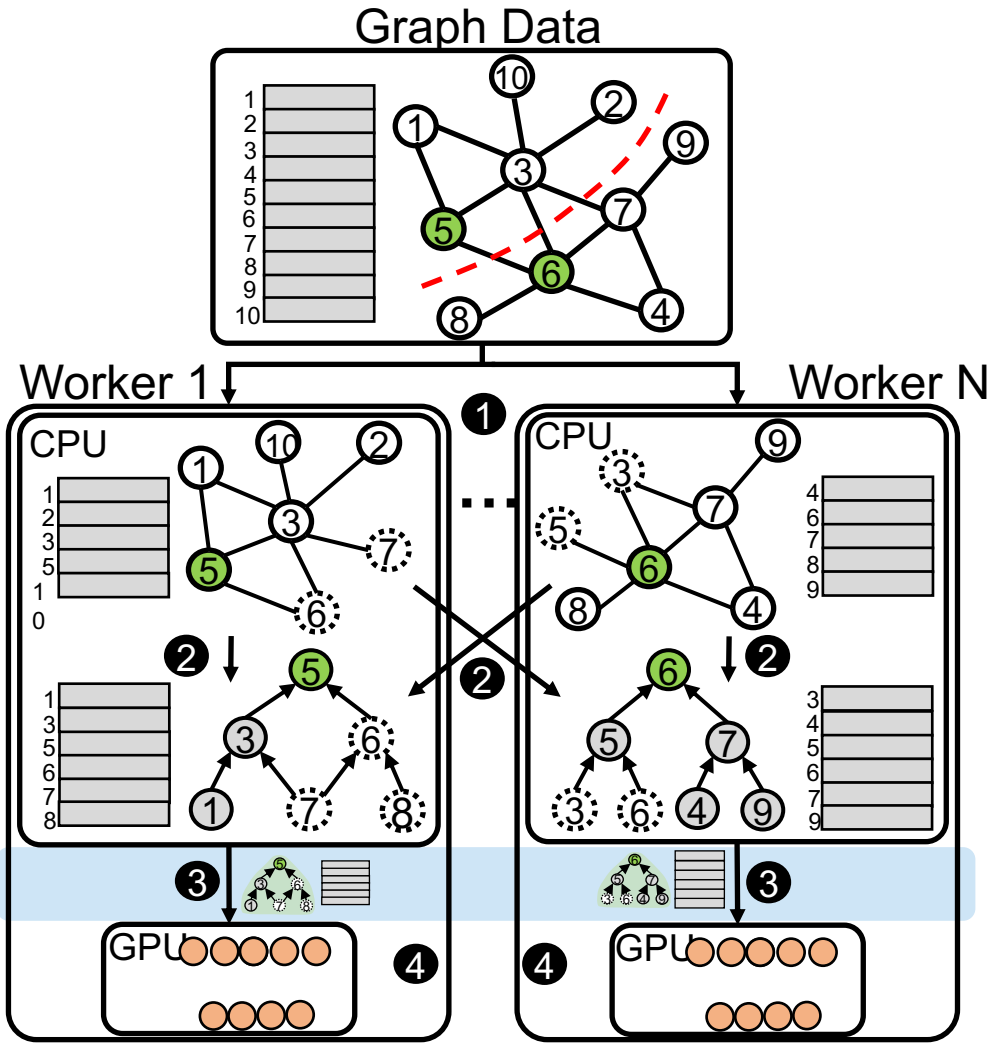
1. Data partitioning
2. Batch preparation
3. Data transferring
4. NN computation

Data management

Actual computation

End-to-End GNN training

Distributed GNN training processing



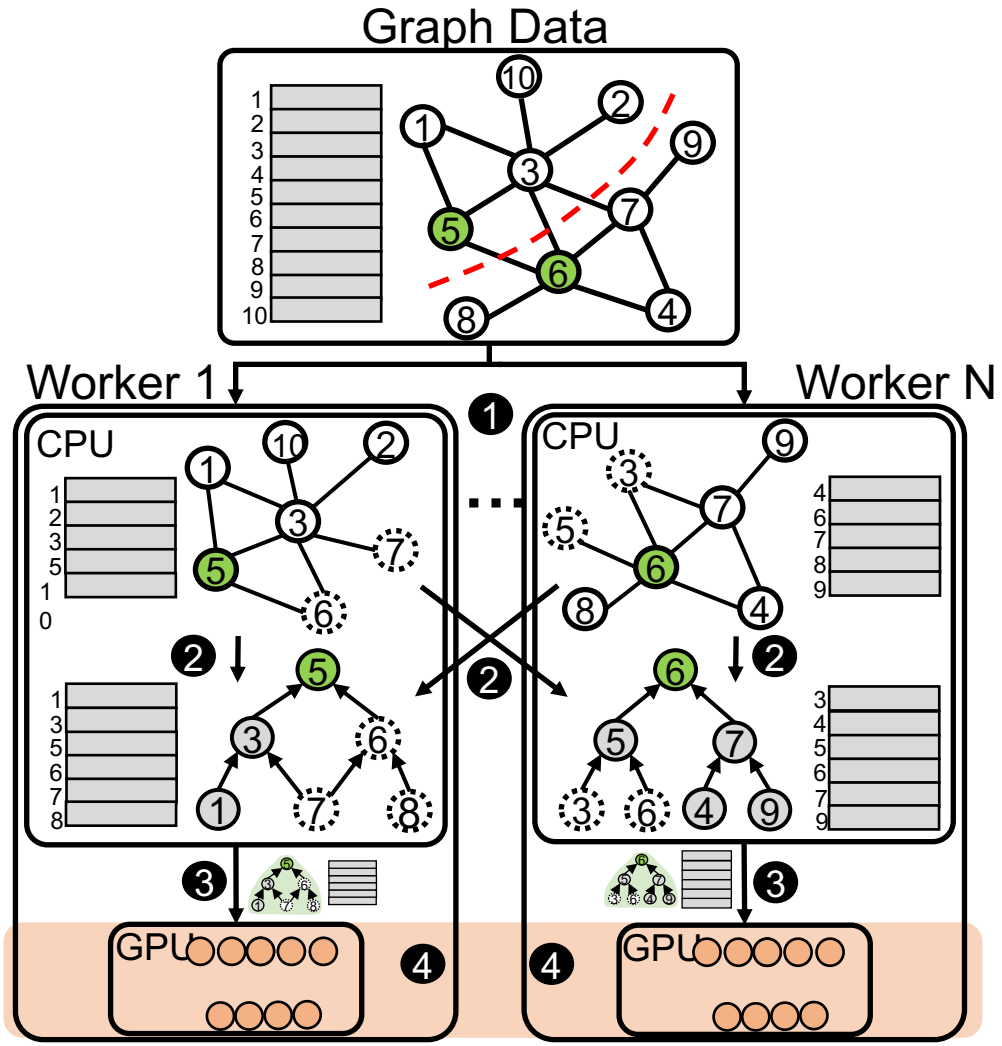
1. Data partitioning
2. Batch preparation
3. Data transferring
4. NN computation

Data management

Actual computation

End-to-End GNN training

Distributed GNN training processing



1. Data partitioning
2. Batch preparation
3. Data transferring

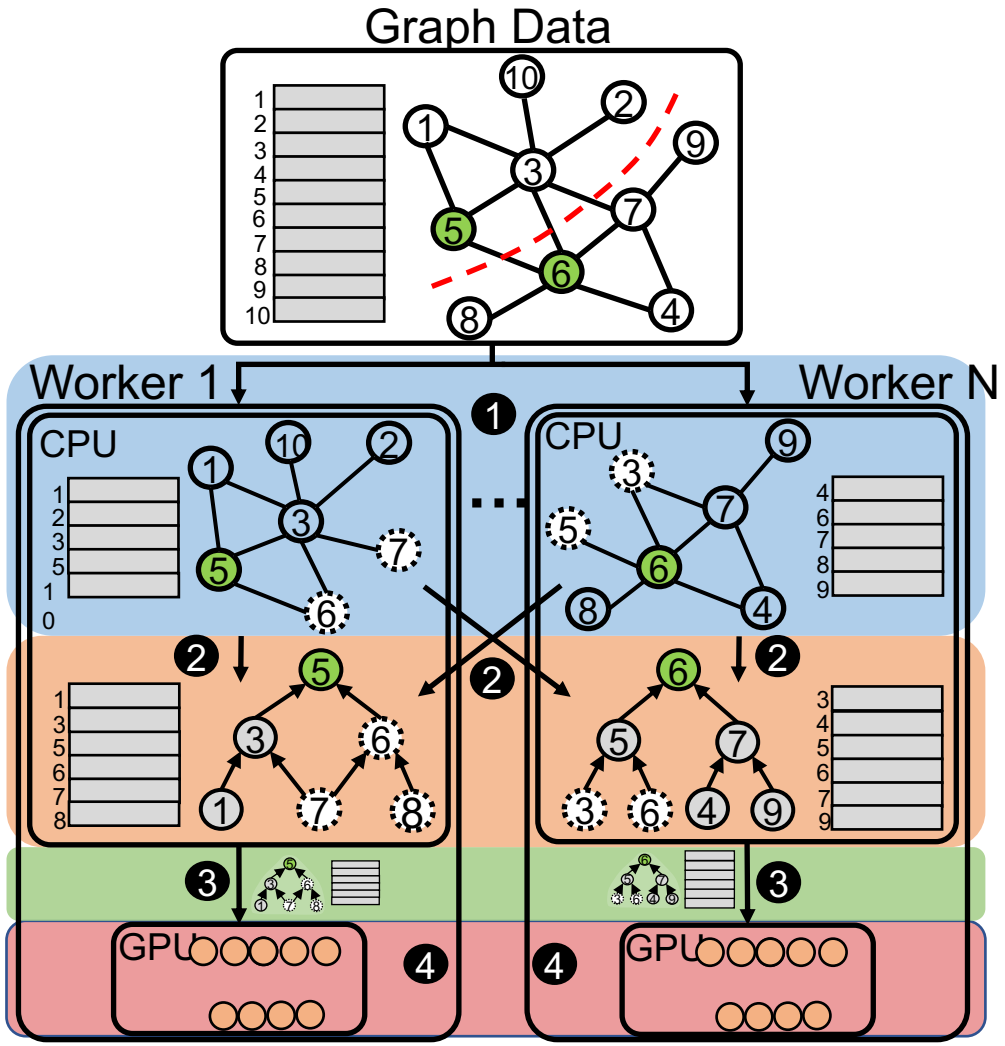
Data management

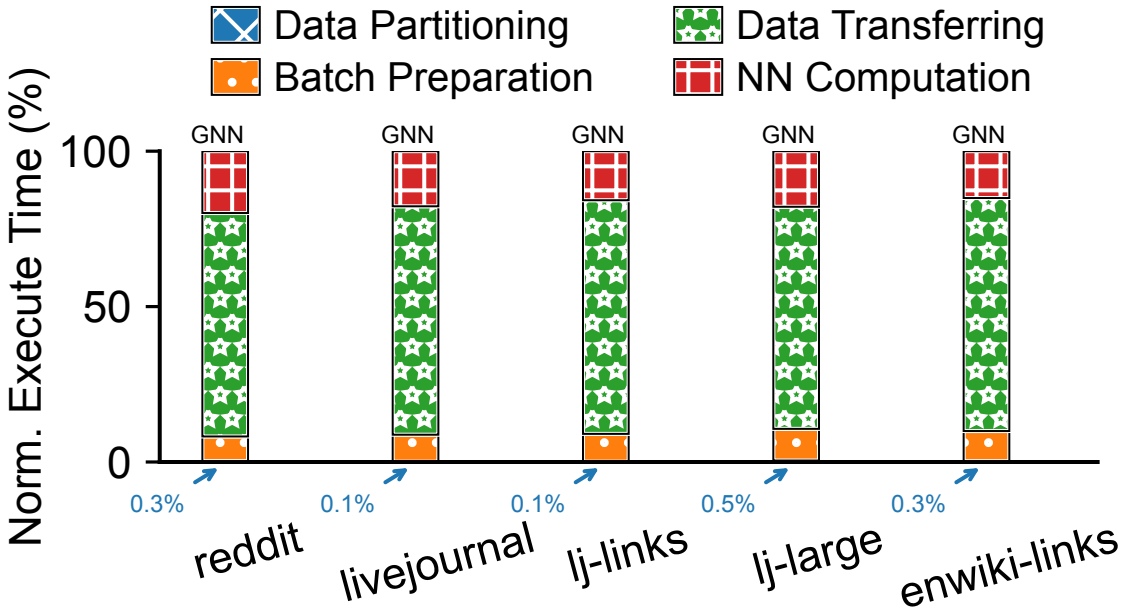
4. NN computation

Actual computation

End-to-End GNN training

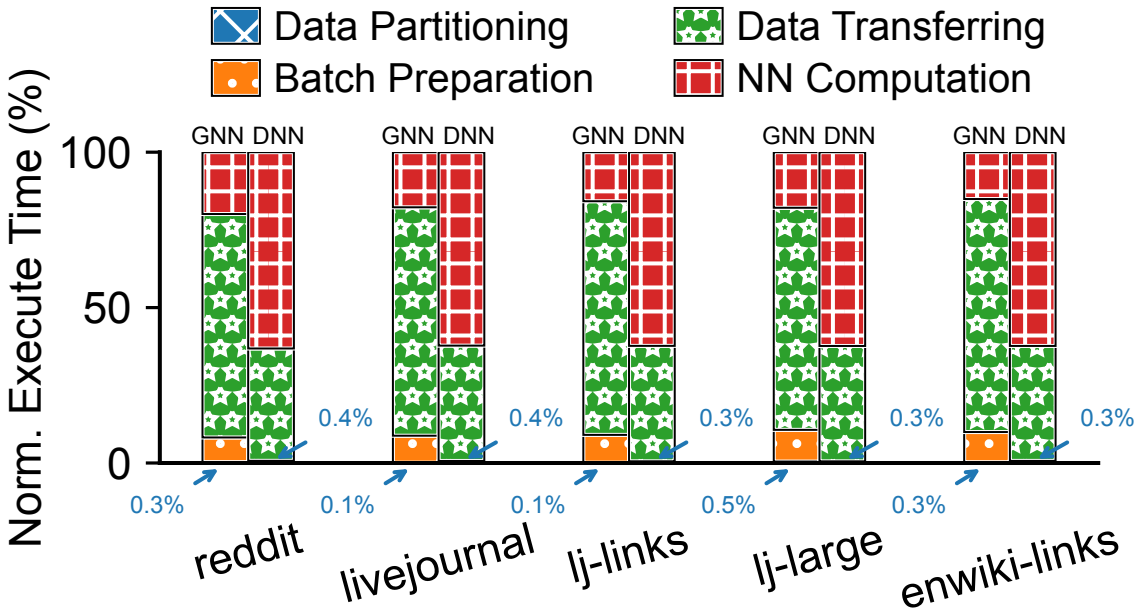
Step-level time breakdown





End-to-End GNN training

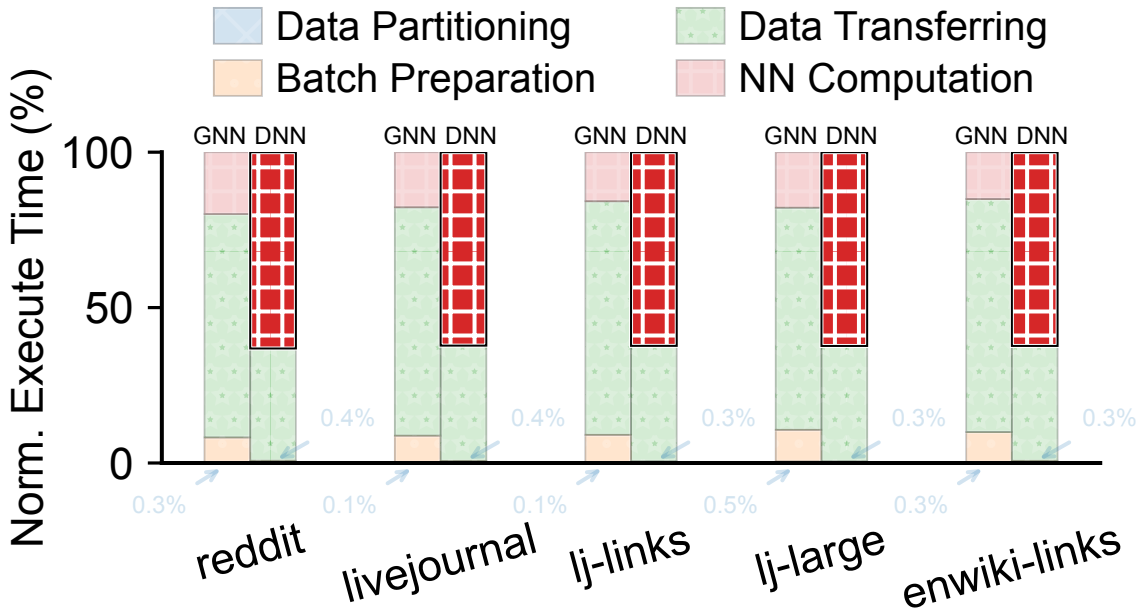
Step-level time breakdown



End-to-End GNN training

Step-level time breakdown

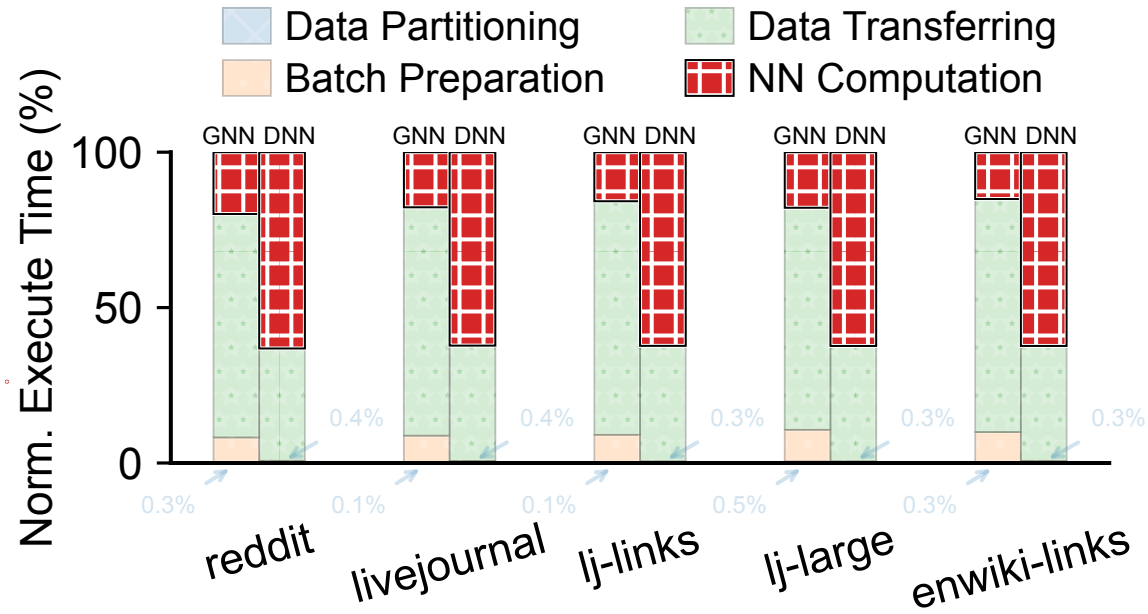
NN computation dominates DNN training



End-to-End GNN training

Step-level time breakdown

NN computation dominates DNN training

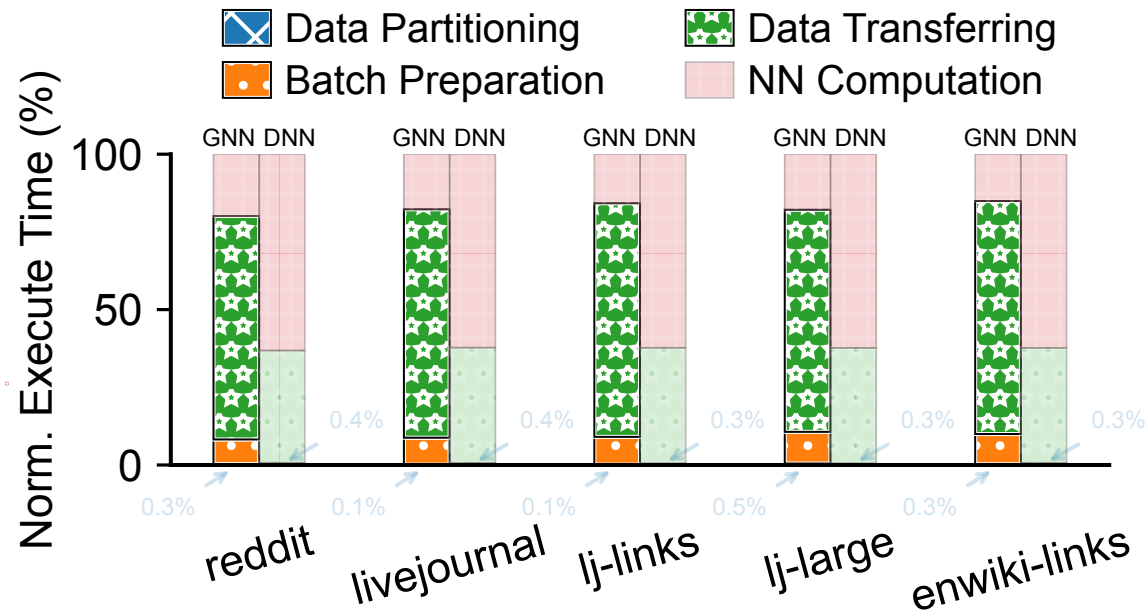


End-to-End GNN training

Step-level time breakdown

NN computation dominates DNN training

Data management dominates GNN training

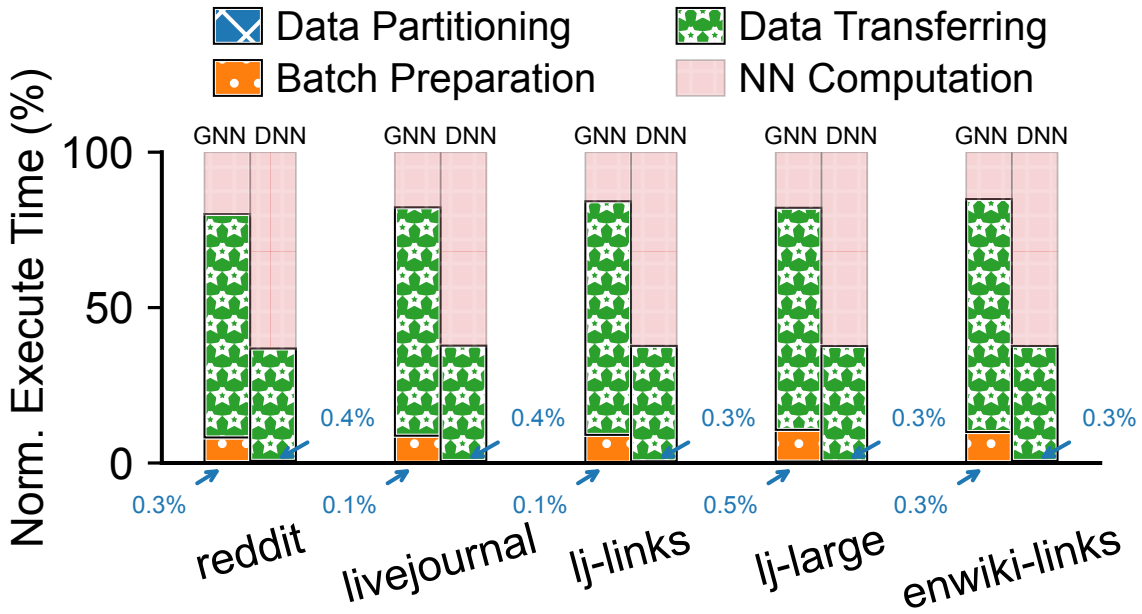


End-to-End GNN training

Step-level time breakdown

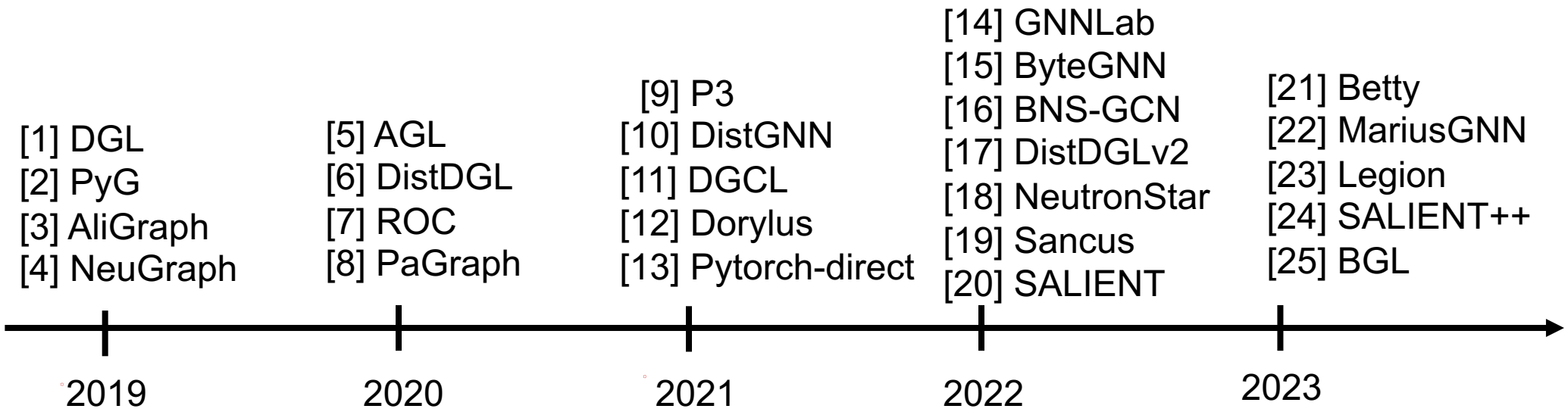
NN computation dominates DNN training

Data management dominates GNN training



GNN Training Systems

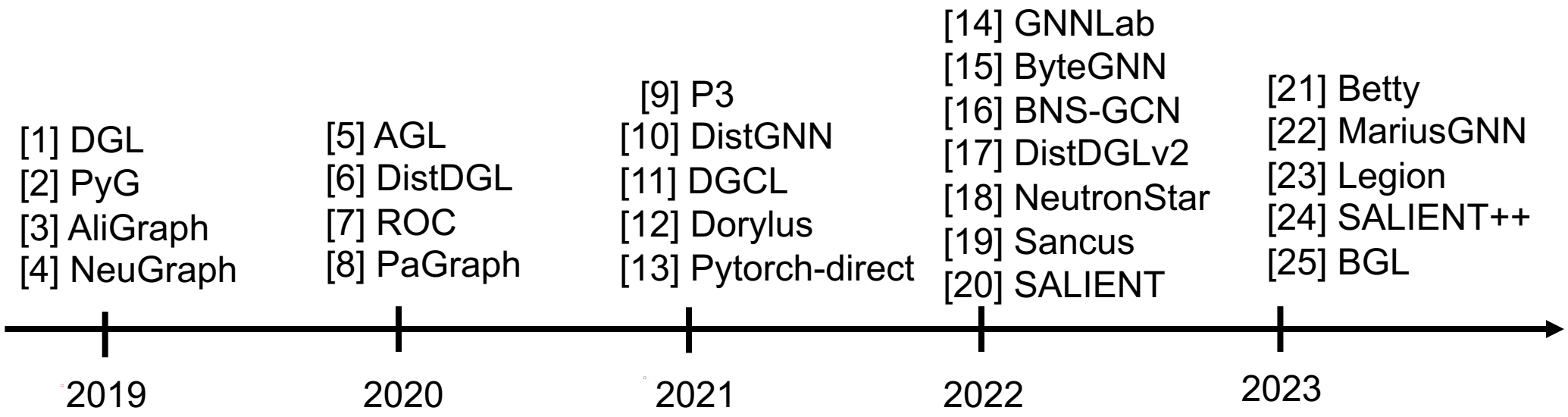
GNN training system: *development history*



[1]M Wang et al., Deep graph library: A graph-centric, highly-performant package for graph neural networks. arXiv 1009.01315
[2] M Fey et al., PyG: Fast graph representation learning with PyTorch Geometric. arXiv 1903.02428
[3] R Zhu et al., Aligraph: A comprehensive graph neural network platform VLDB'19
[4] L Ma et al., NeuGraph: Parallel deep neural network computation on large graphs. ATC'19
[5] D Zhang et al., AGL: a scalable system for industrial-purpose graph machine learning. VLDB'20.
[6] D Zheng et al., DistDGL: Distributed graph neural network training for billion-scale graphs. Arxiv 2010.05337.
[7] Z Jia et al., Improving the accuracy, scalability, and performance of graph neural networks with roc. MLSys'20
[8] Z Lin et al., Pagraph: Scaling gnn training on large graphs via computation-aware caching. SC'20
[9] S Gandhi et al., P3: Distributed deep graph learning at scale. OSDI'21
[10] V Md et al., Distgnn: Scalable distributed training for large-scale graph neural networks. SC'21
[11] Z Cai et al., DGCL: An efficient communication library for distributed GNN training. EuroSys'21
[12] J Thorpe et al., Dorylus: Affordable, scalable, and accurate GNN training with distributed CPU servers and serverless threads. OSDI'21
[13] SW Min et al., Large graph convolutional network training with GPU-oriented data communication architecture. VLDB'21
[14] J Yang et al., GNNLab: a factored system for sample-based GNN training over GPUs. EuroSys '22
[15] C Zheng et al., ByteGNN: efficient graph neural network training at large scale. VLDB'22
[16] C Wan et al., Bns-gcn: Efficient full-graph training of graph convolutional networks with partition-parallelism and random boundary node sampling. MLSYS'22
[17] D Zheng et al., Distributed hybrid cpu and gpu training for graph neural networks on billion-scale heterogeneous graphs. KDD'22
[18] Q Wang et al., Neutronstar: distributed GNN training with hybrid dependency management. SIGMOD'22
[19] J Peng et al., Sancus: sta le n ess-aware c omm u nication-avoiding full-graph decentralized training in large-scale graph neural networks. VLDB'22
[20] T Kaler et al., Accelerating training and inference of graph neural networks with fast sampling and pipelining. MLSYS'22
[21] S Yang et al., Betty: Enabling large-scale gnn training with batch-level graph partitioning. ASPLOS'23
[22] R Waleffe et al., Mariusgnn: Resource-efficient out-of-core training of graph neural networks. EuroSys'23
[23] J Sun et al., Legion: Automatically Pushing the Envelope of Multi-GPU System for Billion-Scale GNN Training. ATC'23
[24] Tim Kaler et al., Communication-Efficient Graph Neural Networks with Probabilistic Neighborhood Expansion Analysis and Caching. MLSYS'23
[25] T Liu et al., BGL: GPU-Efficient GNN training by optimizing graph data I/O and preprocessing. NSDI'23

GNN Training Systems

GNN training system: *development history*



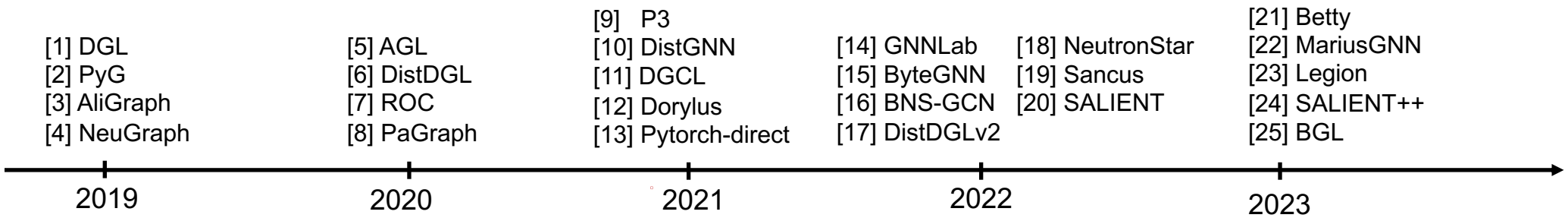
Mini-batch | Fanout | Pipelining | Zero-copy

Partitioning | Lots of optimization techniques | Sampling

Full-batch | Dependency Management | Cache | Unified memory

GNN Training Systems

High-level summary of our work



System Evaluation: Data Management Perspective

Data Partitioning

[3] AliGraph
[6] DistDGL
[9] P3
[15] ByteGNN
[24] SALIENT++
[25] BGL

Batch Preparation

[9] P3
[6] DistDGL
[8] PaGraph
[14] GNNLab
[15] ByteGNN
[16] BNS-GCN
[24] SALIENT++

Data Transferring

[8] PaGraph
[13] Pytorch-direct
[14] GNNLab
[15] ByteGNN
[17] DistDGLv2
[21] Betty
[22] MariusGNN
[23] Legion
[24] SALIENT++
[25] BGL

Experimental Setup

Platforms, algorithms, graph dataset, and environment

Platforms:

A 4-node Aliyun ECS cluster[1] (Each: 40 vCPUs, 155GB RAM, 1 NVIDIA-T4 GPU)

Algorithms and graphs:

- 2 Graph Neural Networks GCN, GraphSAGE
- 9 real-world graphs.

Environment

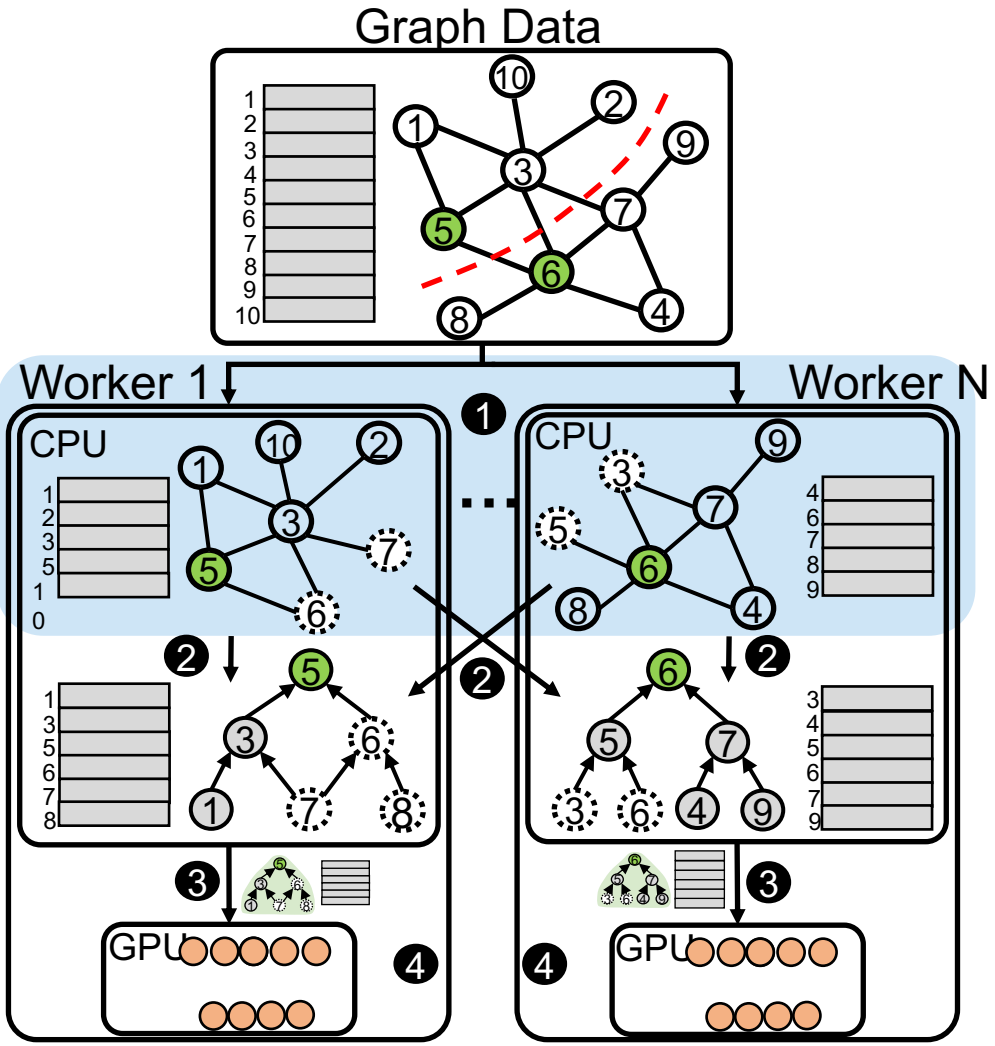
- Ubuntu 20.04 LTS
- CUDA 11.3

Table 2: Dataset description.

Dataset	|V|	|E|	#F	#L	#hidden
Reddit [13]	232.96K	114.85M	602	41	128
OGB-Arxiv [35]	169.34K	2.48M	128	40	128
OGB-Products [37]	2.45M	126.17M	100	47	128
OGB-Papers [36]	111.06M	1.6B	128	172	128
Amazon [66]	1.57M	264.34M	200	107	128
LiveJournal [61]	4.85M	90.55M	600	60	128
Lj-large [32]	7.49M	232.1M	600	60	128
Lj-links [22]	5.2M	205.25M	600	60	128
Enwiki-links [23]	13.59M	1.37B	600	60	128

[1] Clusters are connected via 10GigE

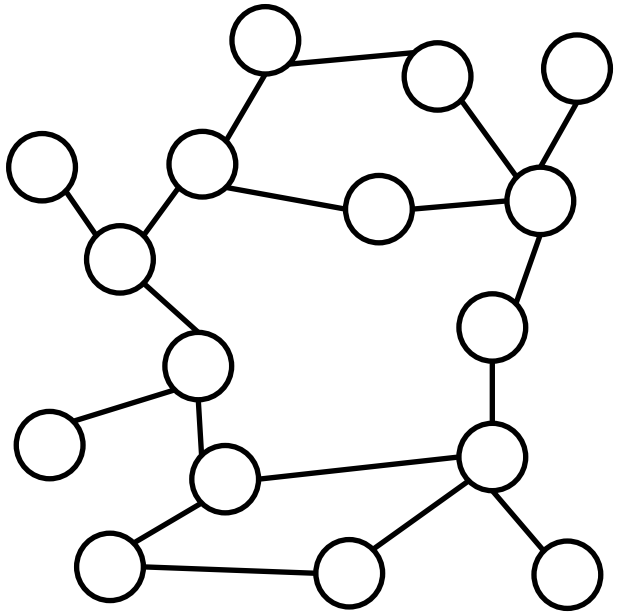
Data Partitioning



Part1: Data Partitioning

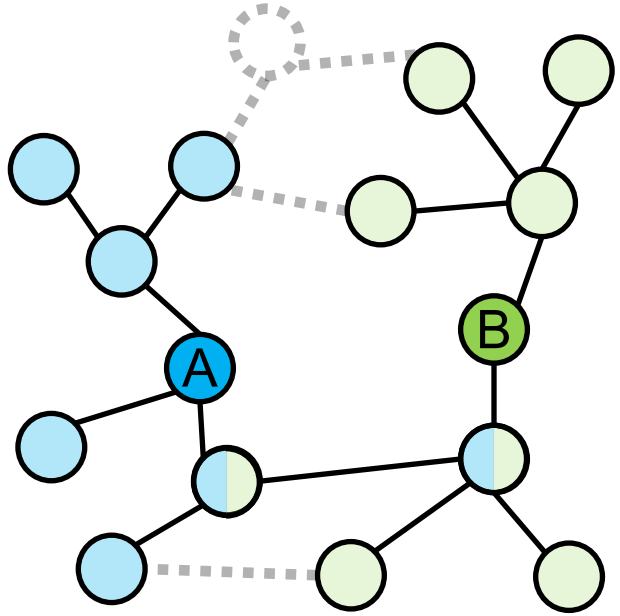
Data Partitioning

Compute Patterns in Graph Computations and GNNs



(a) Computation pattern of graph computation

All vertices + 1-hop neighbors



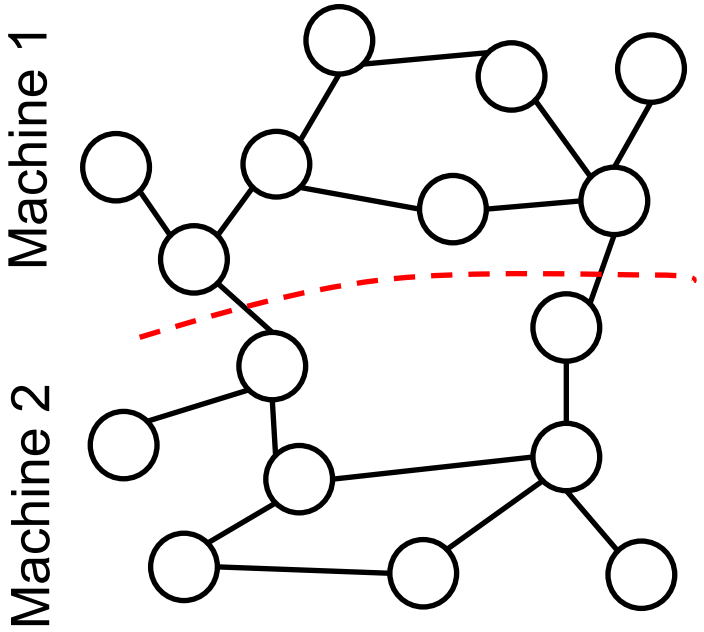
(b) Computation pattern of GNNs

Labeled vertices + L-hop subgraphs

GNN and graph computation have different compute patterns

Data Partitioning

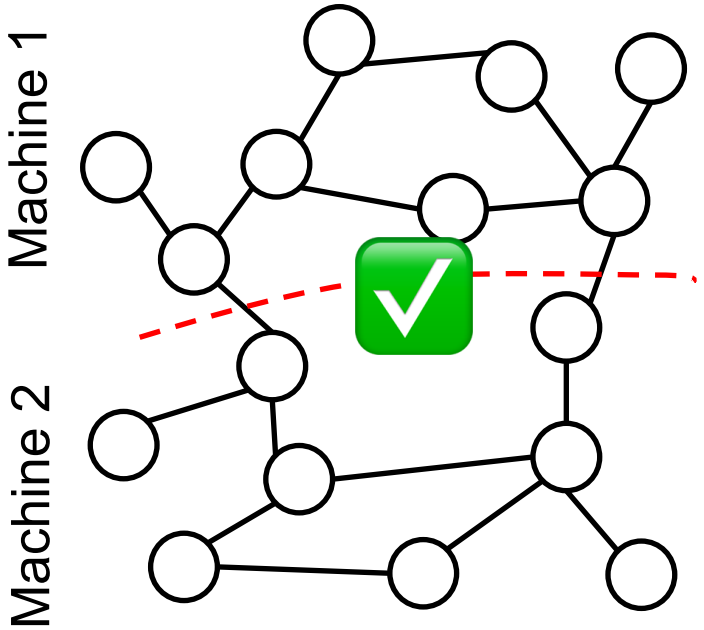
Traditional graph partitioning is not suitable for GNN training



(a) Minimum edge-cut partitioning for graph computation

Data Partitioning

Traditional graph partitioning is not suitable for GNN training

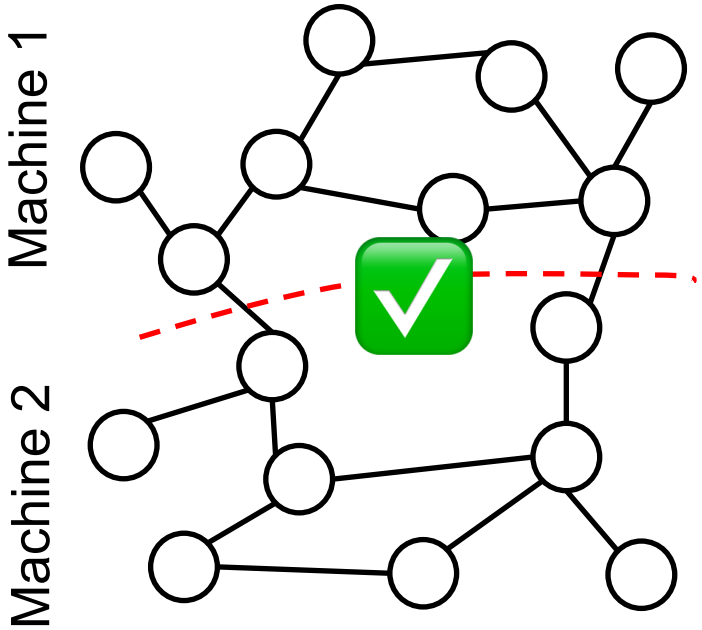


(a) Minimum edge-cut partitioning for graph computation

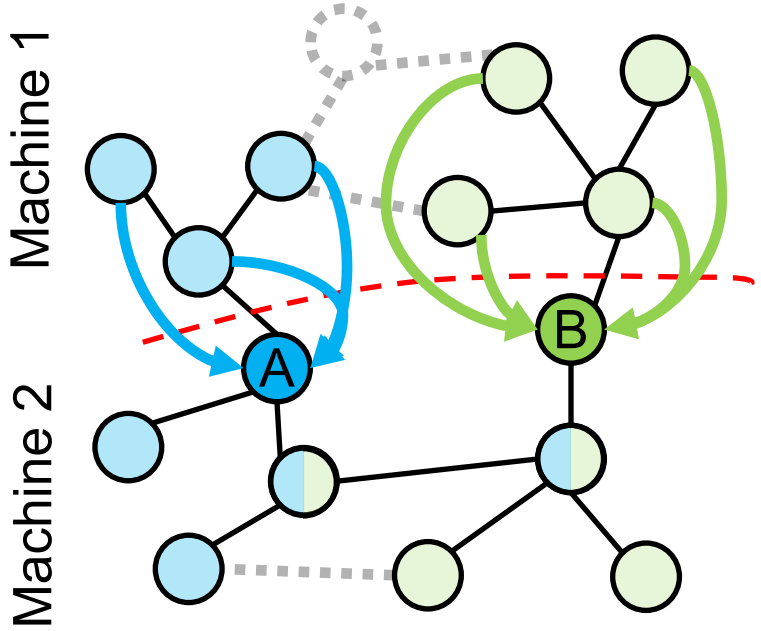
- Balanced computational load
- Minimized communication volume

Data Partitioning

Traditional graph partitioning is not suitable for GNN training



(a) Minimum edge-cut partitioning for graph computation

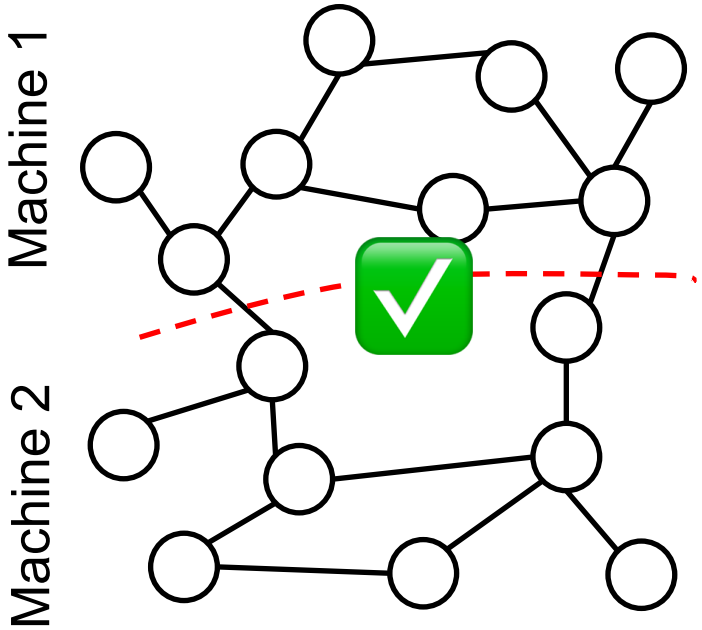


(b) Minimum edge-cut partitioning for GNNs

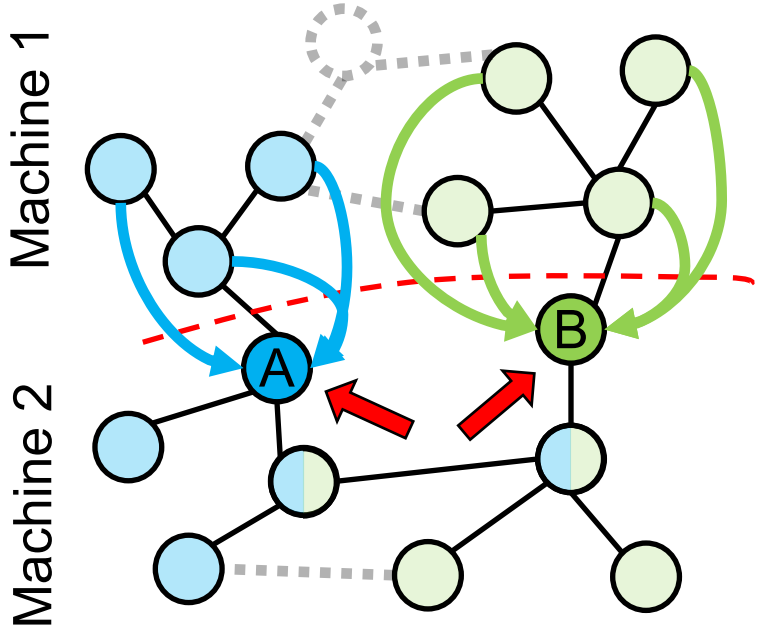
- Balanced computational load
- Minimized communication volume

Data Partitioning

Traditional graph partitioning is not suitable for GNN training



(a) Minimum edge-cut partitioning for graph computation



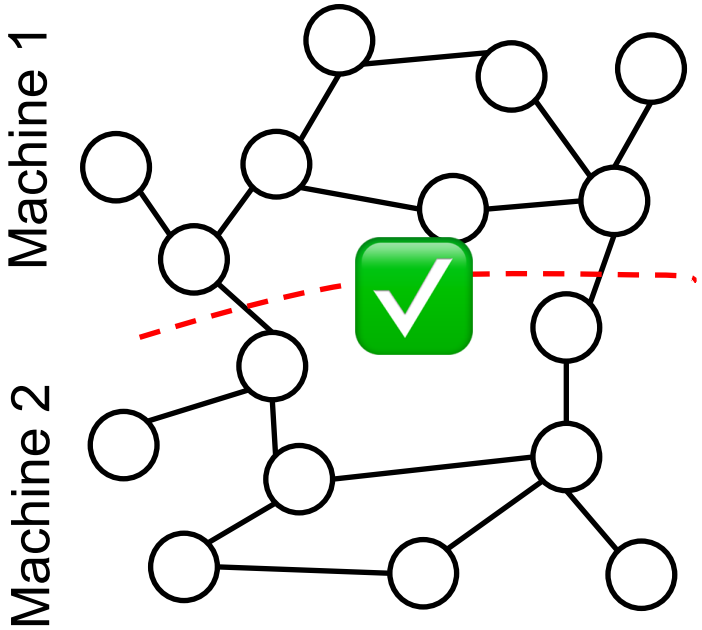
(b) Minimum edge-cut partitioning for GNNs

- Balanced computational load
- Minimized communication volume

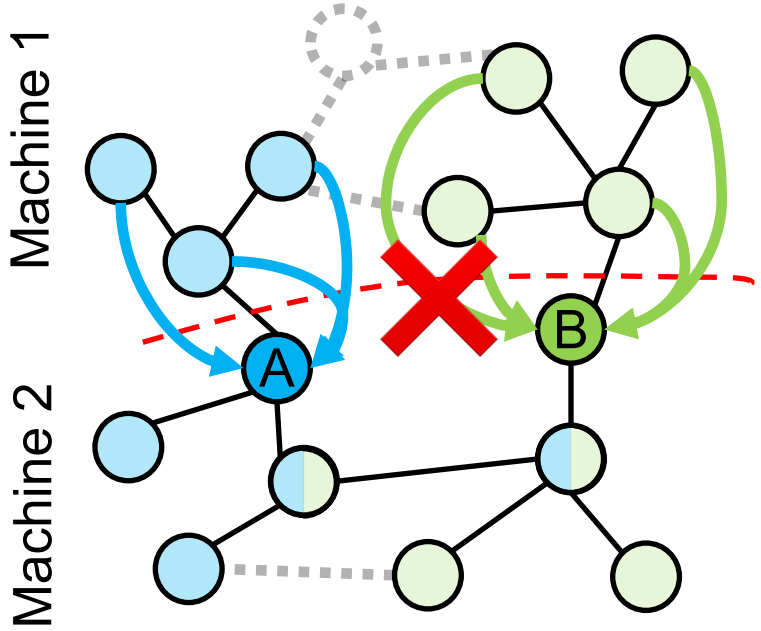
- Imbalanced computational load

Data Partitioning

Traditional graph partitioning is not suitable for GNN training



(a) Minimum edge-cut partitioning for graph computation



(b) Minimum edge-cut partitioning for GNNs

- Balanced computational load
- Minimized communication volume

- Imbalanced computational load
- High communication volume

Data Partitioning

Existing methods

(1) Hash: randomly assign vertices or edges.

(2) Metis-extend: extends Metis by adding additional constraints to balance vertices and edges.

(3) Streaming: Dynamically partitions the graph by setting different scoring functions.

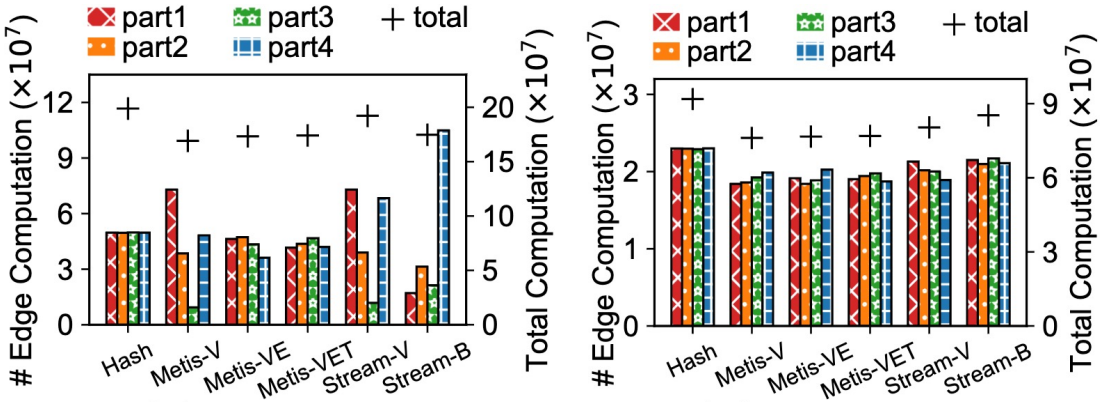
Summary of evaluated partitioning methods

Method	Strategy	Representative System
Hash	Randomly assign vertices or edges	P3
Metis-V	Extend Metis by adding constraints on training vertex masks.	N/A
Metis-VE	Extend Metis by adding constraints on training vertex masks and vertex degrees.	DistDGL
Metis-VET	Extend Metis by adding constraints on training/validation/test vertex masks and vertex degrees.	SALIENT++
Stream-V	Assign vertex v to a partition P that has the most edges connected to v. while balancing the number of train vertices and caching L-hop neighbors.	PaGraph
Stream-B	Assign a block B of vertices to a partition P that has the most edges connected to B. while balancing the number of train/val/test vertices.	ByteGNN

S Gandhi et al., P3: Distributed deep graph learning at scale. OSDI'21
D Zheng et al., DistDGL: Distributed graph neural network training for billion-scale graphs. Arxiv 2010.05337.
Tim Kaler et al., Communication-Efficient Graph Neural Networks with Probabilistic Neighborhood Expansion Analysis and Caching. MLSYS'23
Z Lin et al., Pagraph: Scaling gnn training on large graphs via computation-aware caching. SC'20
C Zheng et al., ByteGNN: efficient graph neural network training at large scale. VLDB'22

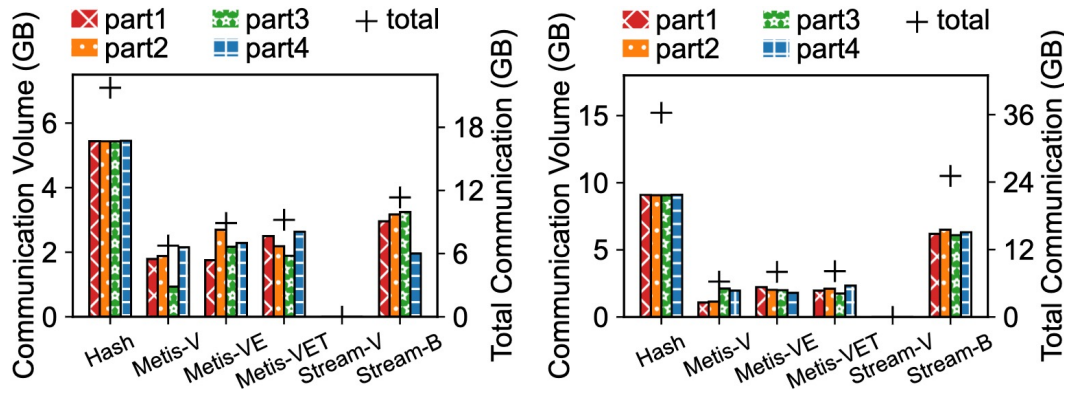
Data Partitioning

Computational and communication load analysis



(a) Amazon (b) Products

Computational load of different partitioning methods.



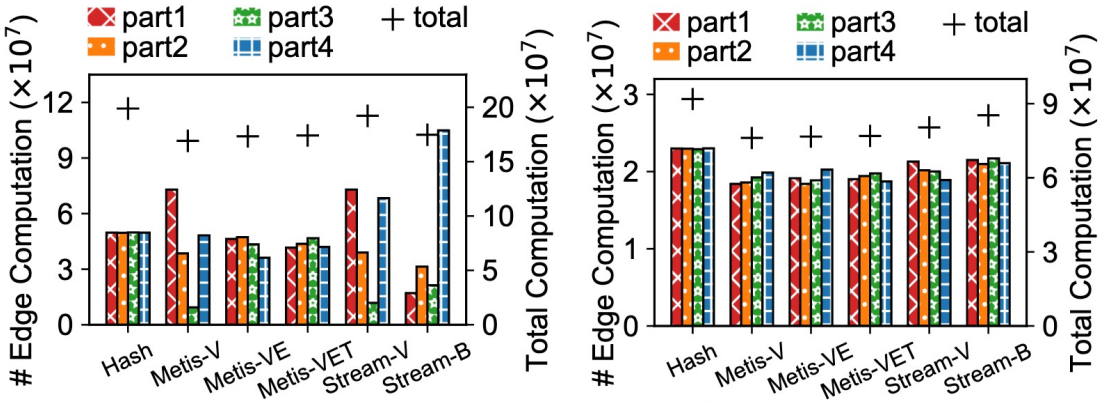
(a) Amazon (b) Products

Communication load of different partitioning methods.

- Hash partitioning achieves **optimal load balancing**, but it also has the **highest workload**.
- Metis-extend achieves **minimal computational and communication load**.
- Streaming partitioning suffers from **workload imbalance on power-law graphs**.

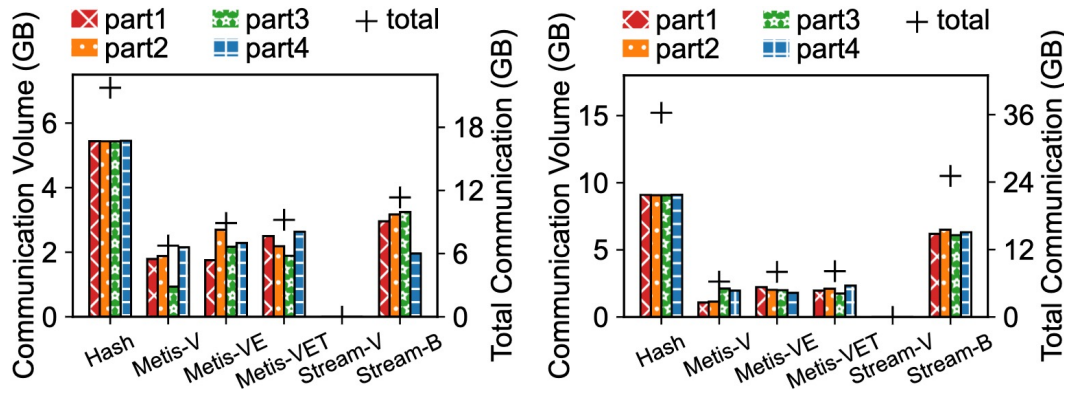
Data Partitioning

Computational and communication load analysis



(a) Amazon (b) Products

Computational load of different partitioning methods.



(a) Amazon (b) Products

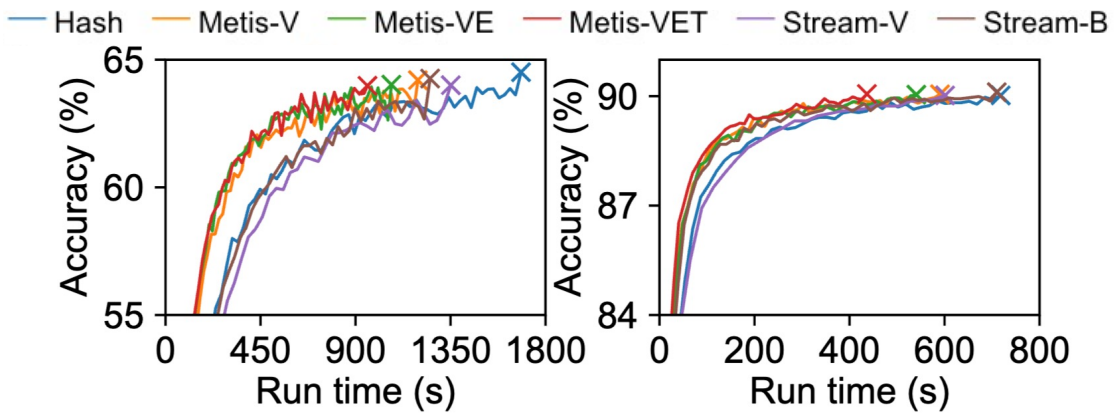
Communication load of different partitioning methods.

- Hash partitioning achieves **optimal load balancing**, but it also has the **highest workload**.
- Metis-extend achieves **minimal computational and communication load**.
- Streaming partitioning suffers from **workload imbalance on power-law graphs**.

- **Partitioning densely connected** vertices into the same partition can significantly **reduce computational and communication load**.
- Existing graph partitioning methods all suffer from an **imbalanced communication load**.

Data Partitioning

Effect to accuracy



Time-to accuracy

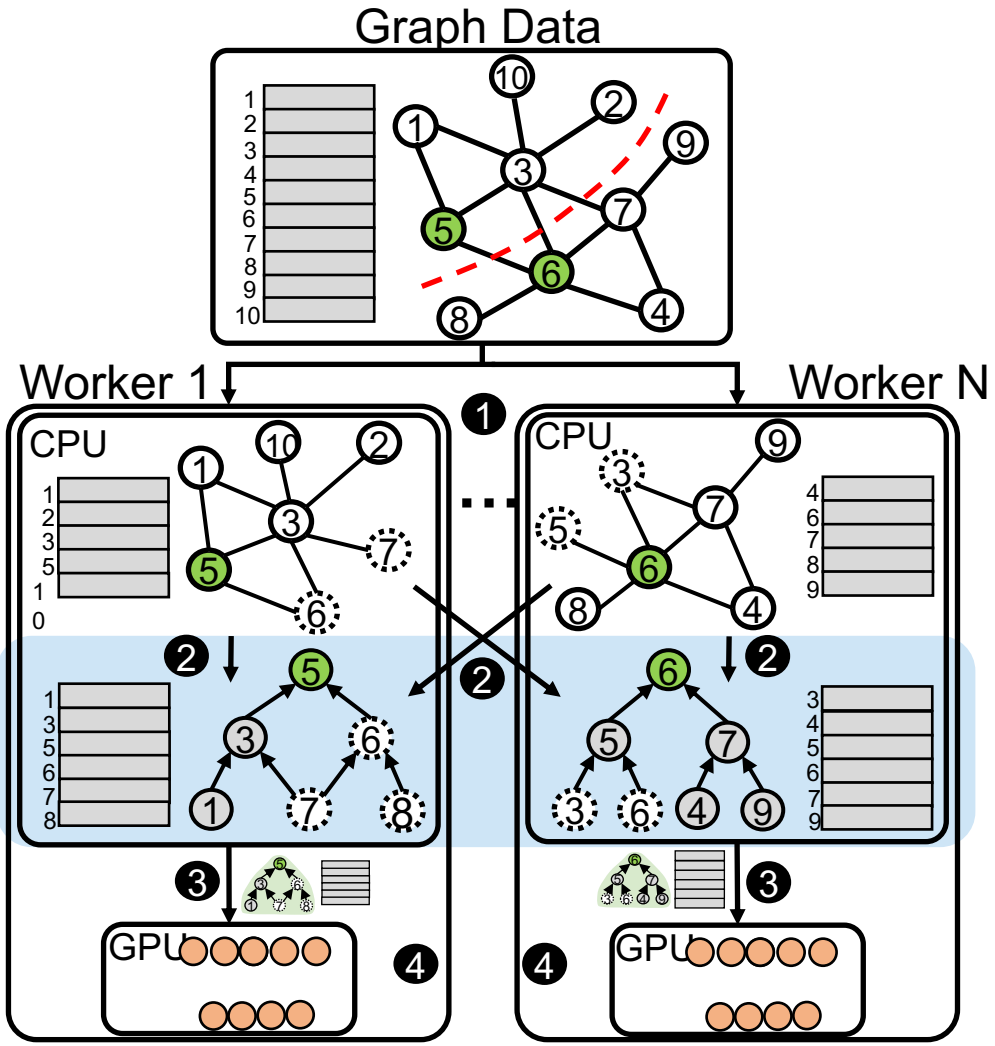
Model accuracy under different partition methods.

	Hash	Metis-V	Metis-VE	Metis-VET	Stream-V	Stream-B	Diff.
Reddit	96.2%	96.2%	96.2%	96.1%	96.4%	96.2%	±0.3%
Products	90.1%	90.3%	90.2%	90.2%	90.5%	90.2%	±0.4%
Amazon	64.5%	64.2%	64.4%	64.7%	65.1%	64.3%	±0.9%

The accuracy difference across different partitioning methods is **less than 1%.**

Graph partitioning does not affect the accuracy.

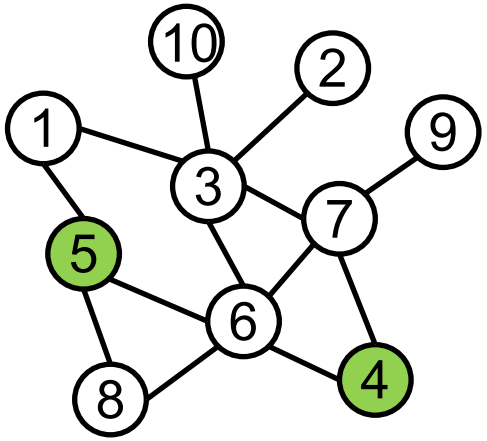
Batch Preparation



Part2: Batch Preparation

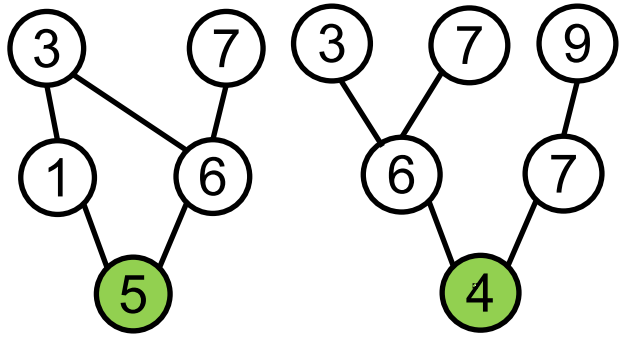
Batch Preparation

Batch size



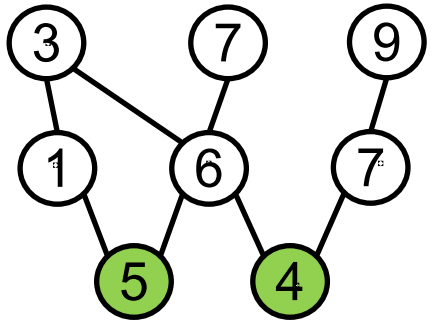
Input Graph

batch size = 1



2 updates per epoch

batch size = 2

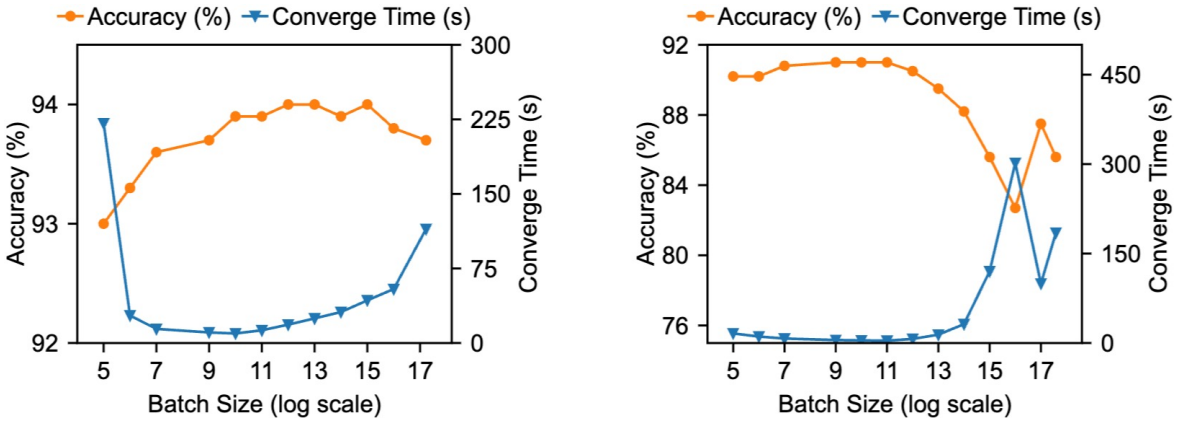


1 update per epoch

- Small batch size has higher model update frequency
- Large batch size contain more information.

Batch Preparation

Batch size



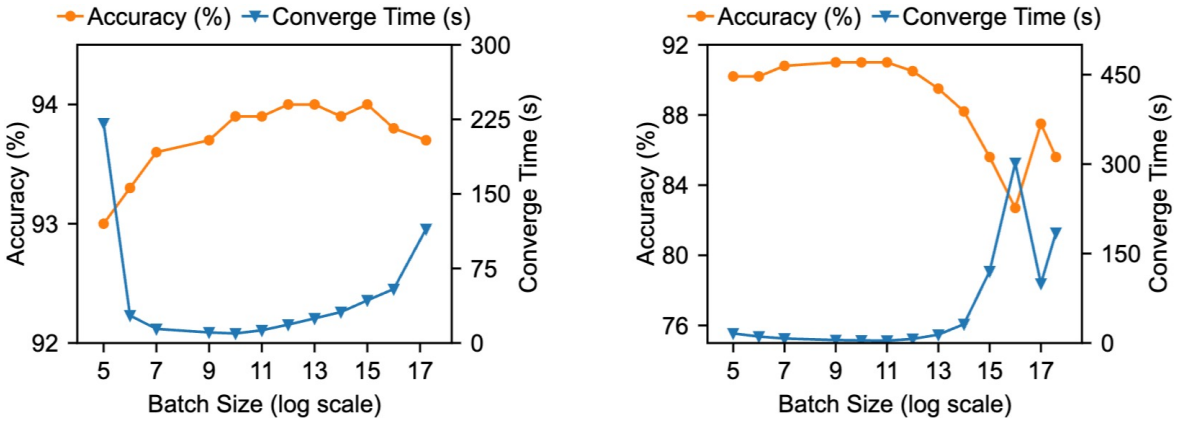
(a) Reddit (b) Products

Accuracy and convergence speed with varying batch size

- Small batch sizes converges quickly but have low accuracy.
- Large batch sizes converges slowly but have high accuracy.

Batch Preparation

Batch size



(a) Reddit (b) Products

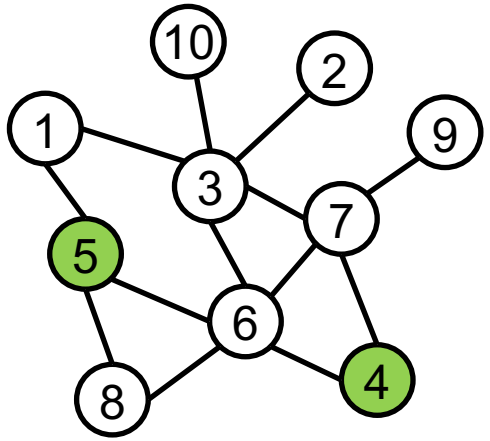
Accuracy and convergence speed with varying batch size

- Small batch sizes converges quickly but have low accuracy.
- Large batch sizes converges slowly but have high accuracy.

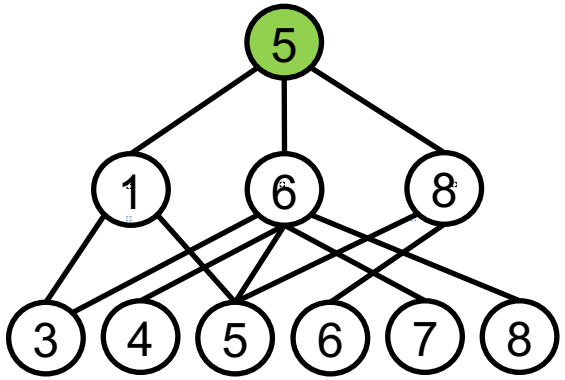
- Small batch size: larger gradients, faster descent direction finding.
- Large batch size: smaller gradients, better convergence to optimal point.

Batch Preparation

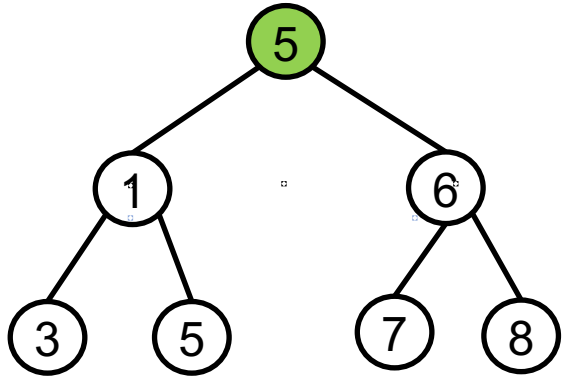
Sampling



Input Graph



Computational graph for vertex 5

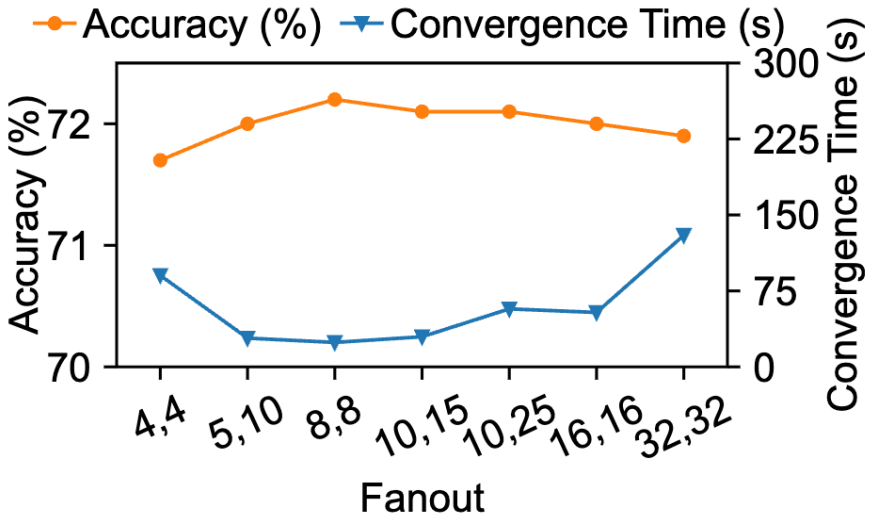


Sampled subgraph for vertex 5 (**fanout = 2**)

Sampling significantly reduces the training graph size

Batch Preparation

Fanout

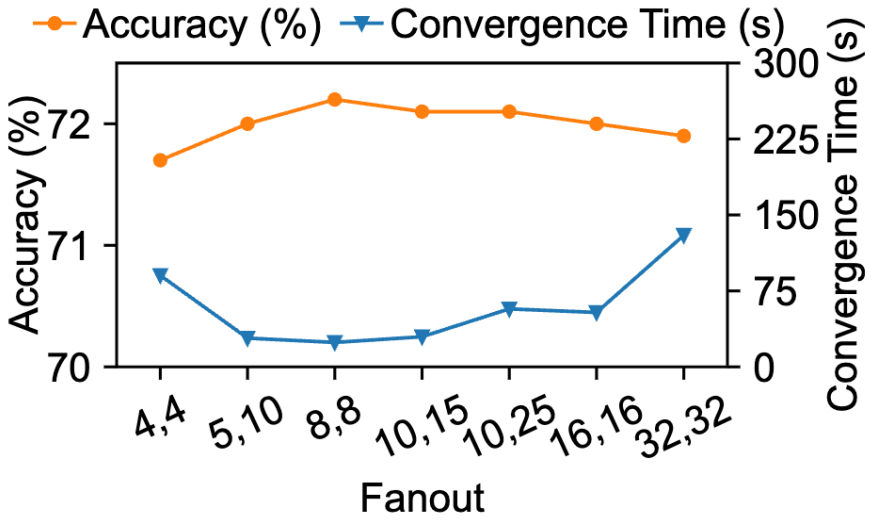


Accuracy and convergence speed with varying fanout

- Small fanout converges quickly but has low accuracy.
- Large fanout converges slowly but has high accuracy.

Batch Preparation

Fanout



Accuracy and convergence speed with varying fanout

- Small fanout converges quickly but has low accuracy.
- Large fanout converges slowly but has high accuracy.

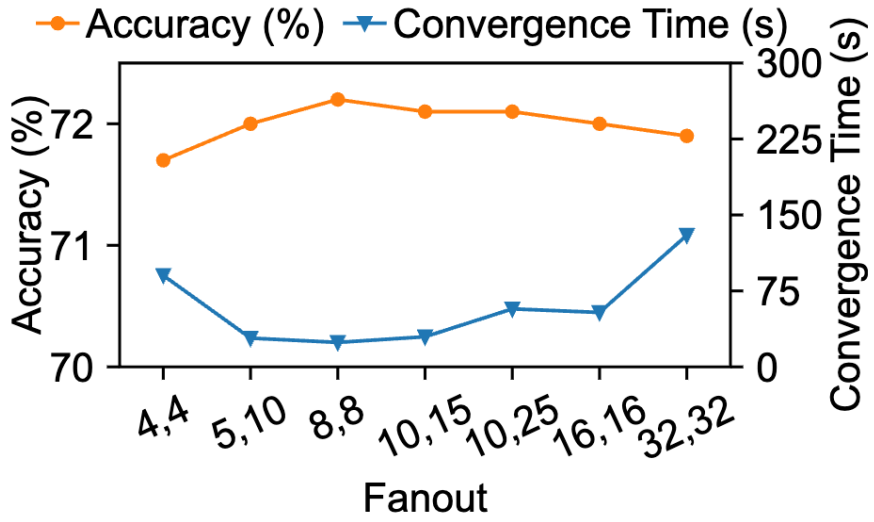
Accuracy of high and low degree vertices

Vertex type	Fanout			
	4, 4	8, 8	16, 16	32, 32
Low-degree vertices	0.718	**0.722**	0.720	0.718
High-degree vertices	0.786	0.794	0.810	**0.817**

- Low-degree vertices achieve higher accuracy with a smaller fanout.
- High-degree vertices achieve higher accuracy with a larger fanout.

Batch Preparation

Fanout



Accuracy and convergence speed with varying fanout

- Small fanout converges quickly but has low accuracy.
- Large fanout converges slowly but has high accuracy.

Accuracy of high and low degree vertices

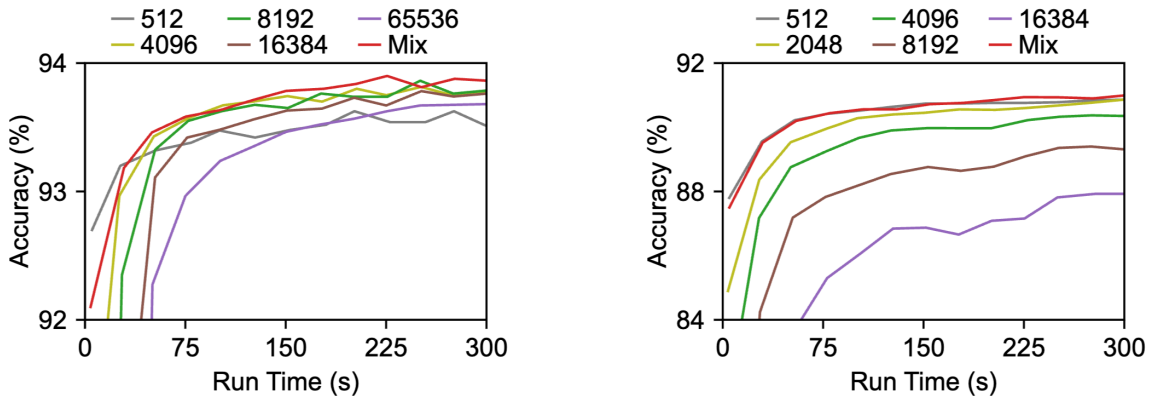
Vertex type	Fanout			
	4, 4	8, 8	16, 16	32, 32
Low-degree vertices	0.718	**0.722**	0.720	0.718
High-degree vertices	0.786	0.794	0.810	**0.817**

- Low-degree vertices achieve higher accuracy with a smaller fanout.
- High-degree vertices achieve higher accuracy with a larger fanout.

Fixed fanout is disadvantageous for power-law graphs.

Batch Preparation

Adaptive batch size and fanout-rate hybrid sampling



(a) Reddit **(b) Products**

Performance with adaptive batch size

Convergence speed Increased by **1.52-1.64**x

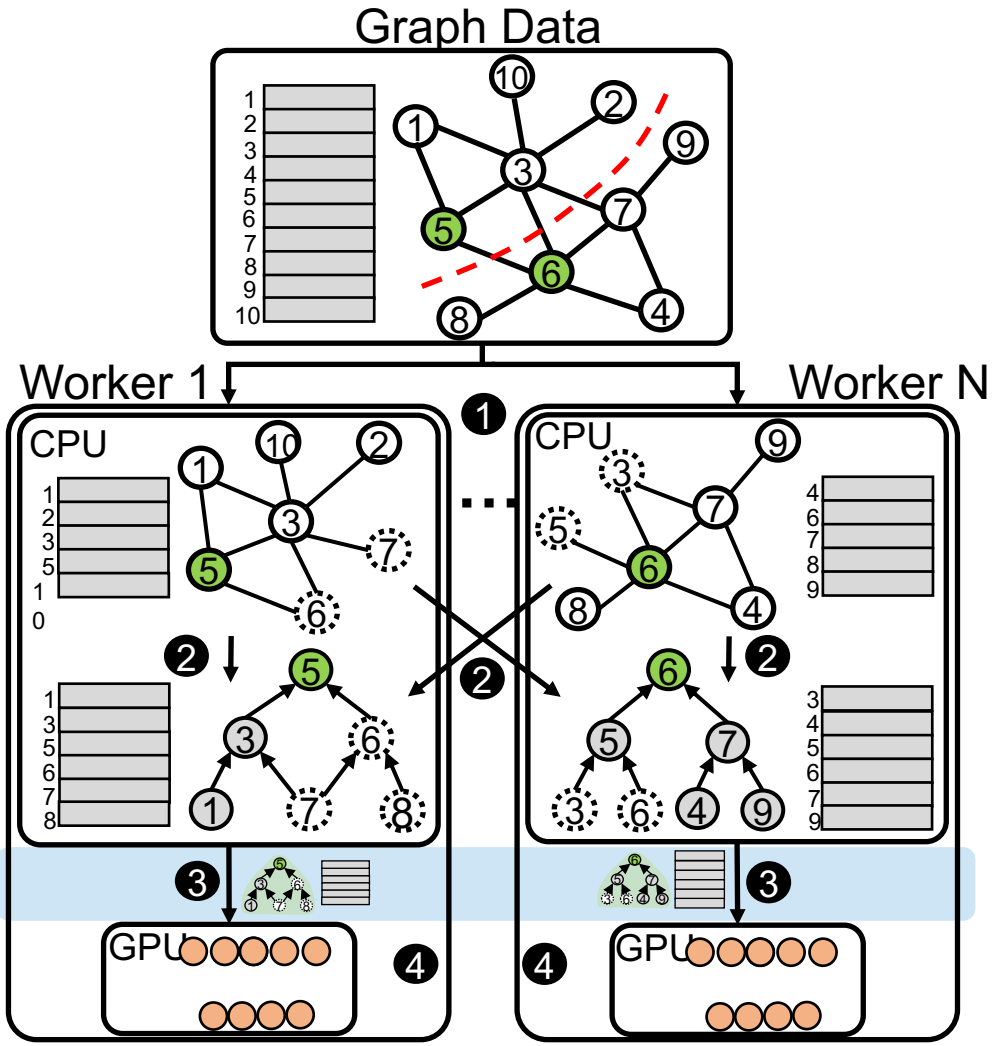
Performance with fanout-rate hybrid sampling

	4, 4	8, 8	10,15	10,25	32, 32	hybrid
Accuracy (%)	71.5	72.1	72.1	72.1	71.5	72.1
Time (s)	300	172	165	237	300	**99**

Convergence speed Increased by **1.74**x

Dynamic parameter tuning further improves model training efficiency.

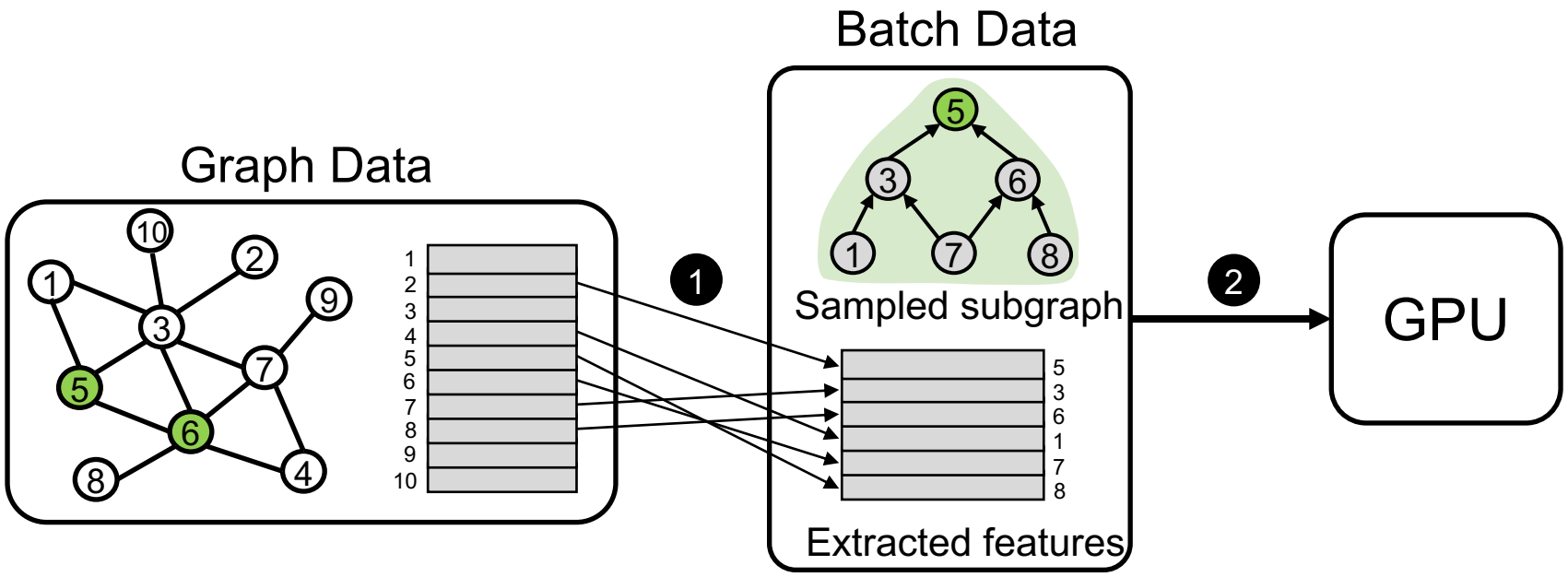
Data Transferring



Part3: Data Transferring

Data Transferring

Data transferring process

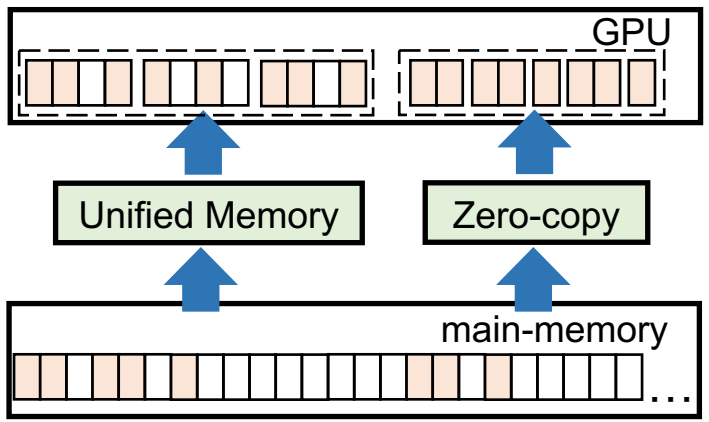


❶ Feature Extraction: **Irregular memory access**

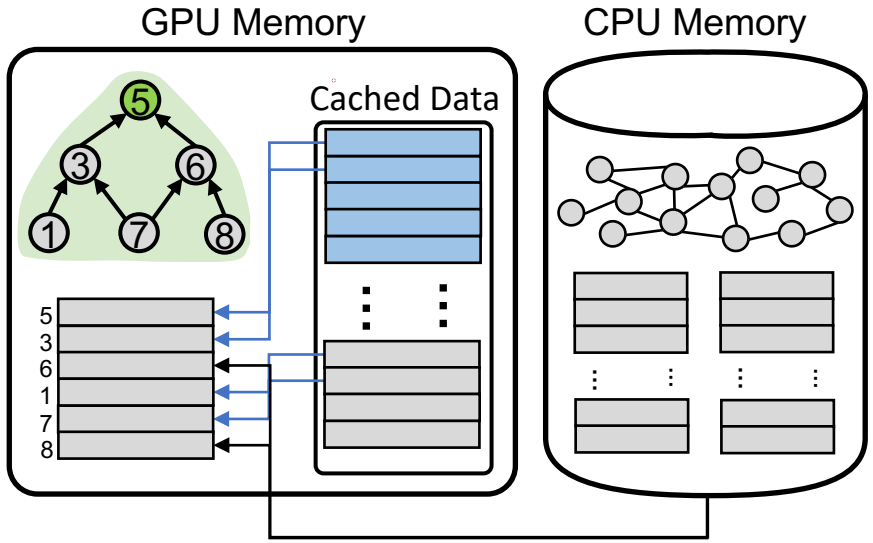
❷ Data Transfer: **Redundant communication overhead**

Data Transferring

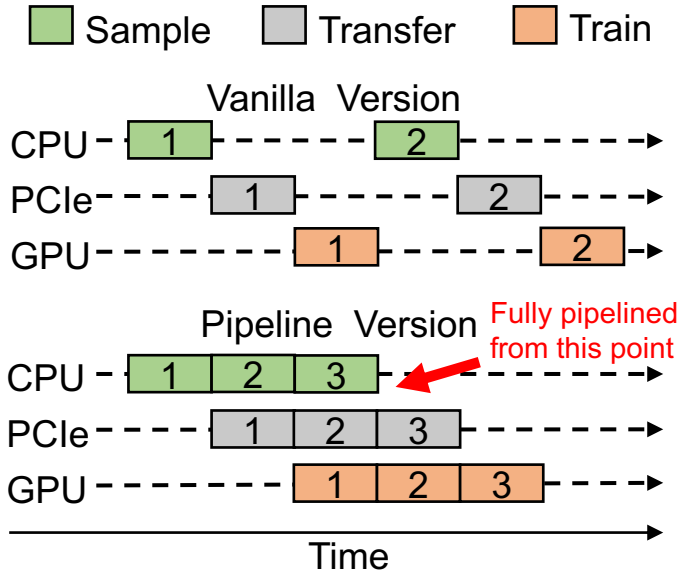
Existing methods



Implicit Transfer



Cache-based Data Reusing

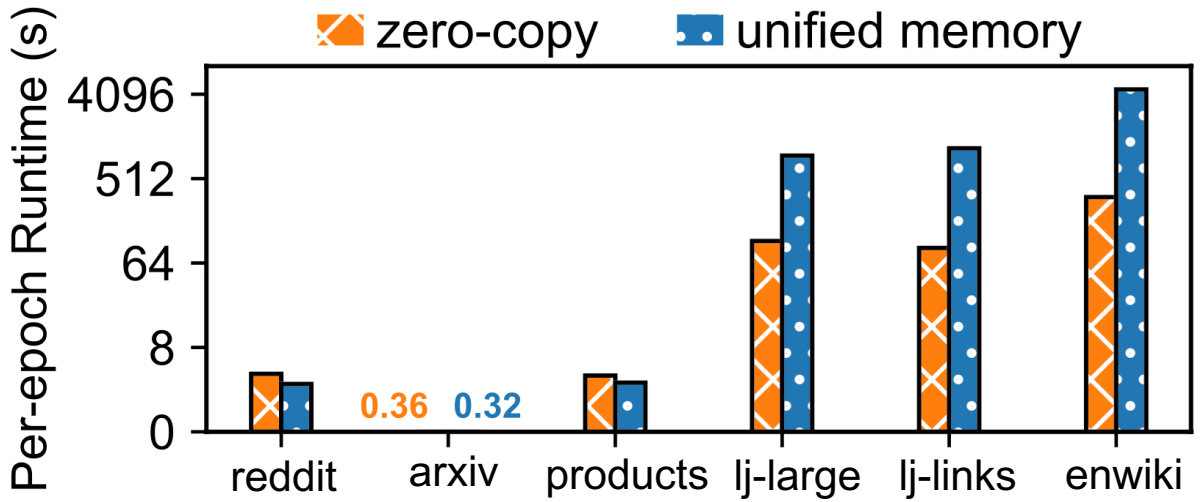


Task Pipeling

SW Min et al., Large graph convolutional network training with GPU-oriented data communication architecture. VLDB'21
Q Wang, et al., HyTGraph: GPU-Accelerated Graph Processing with Hybrid Transfer Management. ICDE'23
J Yang et al., GNNLab: a factored system for sample-based GNN training over GPUs. EuroSys '22
Z Lin et al., Pagraph: Scaling gnn training on large graphs via computation-aware caching. SC'20
T Kaler et al., Accelerating training and inference of graph neural networks with fast sampling and pipelining. MLSYS'22
R Waleffe et al., Mariusgnn: Resource-efficient out-of-core training of graph neural networks. EuroSys'23

Data Transferring

Implicit transfer method



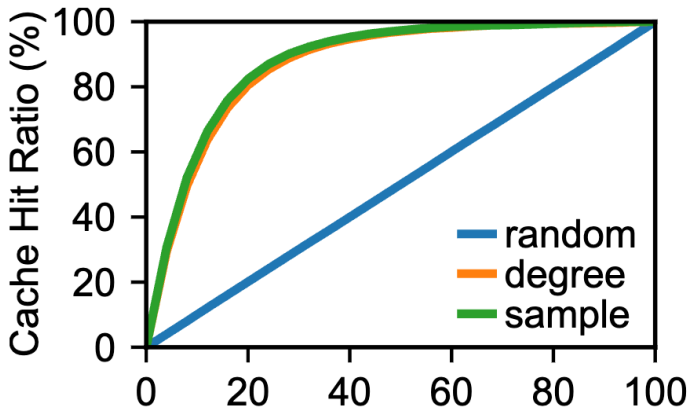
Per-epoch runtime comparison: zero-copy vs. unified memory.

Unified memory is **1.21x faster on small-scale** and **10.59x slower on large-scale** datasets for per-epoch runtime compared to zero-copy.

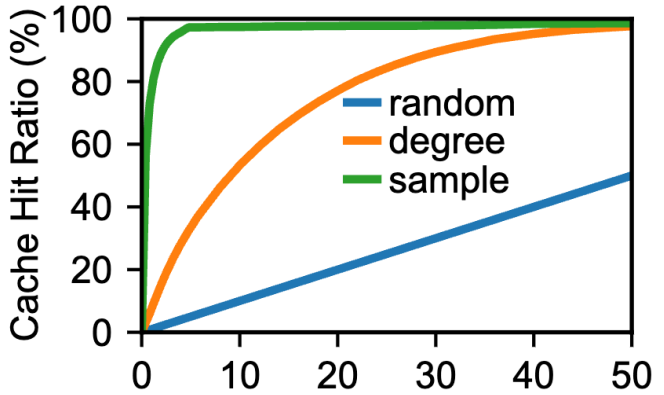
Unified memory unsuitable for large-scale datasets due to high page migration overhead.

Data Transferring

Cache-based data reusing



(a) Amazon (power-law graph)



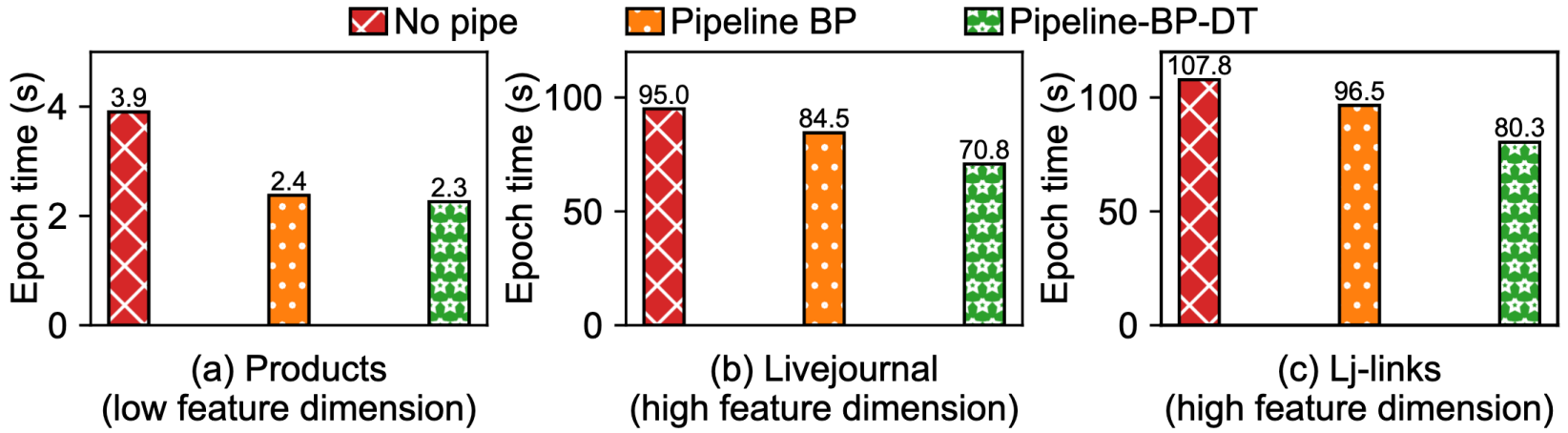
(b) OGB-Paper (non-power-law graph)

Performance comparison of caching policies.

The pre-sampling-based caching method offers better performance and robustness.

Data Transferring

Task pipelining



Pipeline training ablation study.

The effect of pipeline optimization is not outstanding (less than 50% improvement)

Opportunity: orchestrate CPU and GPU workloads for improved pipeline performance.

Summary

Comprehensive Evaluation of GNN Training Systems: A Data Management Perspective.

- **Evaluation of optimization techniques in GNN systems**
  we conduct a comprehensive evaluation of the optimization techniques adopted by existing GNN systems from a data management perspective.

Summary

Comprehensive Evaluation of GNN Training Systems: A Data Management Perspective.

- **Evaluation of optimization techniques in GNN systems**
  we conduct a comprehensive evaluation of the optimization techniques adopted by existing GNN systems from a data management perspective.
- **Lessons learned for designing future GNN training systems**
  We provide some practical tips learned from the experiment results, which are helpful for designing GNN training systems in the future.

Summary

Comprehensive Evaluation of GNN Training Systems: A Data Management Perspective.

- **Evaluation of optimization techniques in GNN systems**
  we conduct a comprehensive evaluation of the optimization techniques adopted by existing GNN systems from a data management perspective.
- **Lessons learned for designing future GNN training systems**
  We provide some practical tips learned from the experiment results, which are helpful for designing GNN training systems in the future.
  - Partitioning densely connected vertices together can effectively reduce computation but may disrupt load balancing.
  - There are significant differences in the accuracy and convergence speed of GNN under different parameter settings.
  - Adaptive batch size and hybrid sampling training methods can accelerate convergence without sacrificing accuracy.
  - There is an opportunity to improve pipeline performance by orchestrate CPU and GPU workloads.

Summary

Comprehensive Evaluation of GNN Training Systems: A Data Management Perspective.

- **Evaluation of optimization techniques in GNN systems**
  we conduct a comprehensive evaluation of the optimization techniques adopted by existing GNN systems from a data management perspective.
- **Lessons learned for designing future GNN training systems**
  We provide some practical tips learned from the experiment results, which are helpful for designing GNN training systems in the future.
  - Partitioning densely connected vertices together can effectively reduce computation but may disrupt load balancing.
  - There are significant differences in the accuracy and convergence speed of GNN under different parameter settings.
  - Adaptive batch size and hybrid sampling training methods can accelerate convergence without sacrificing accuracy.
  - There is an opportunity to improve pipeline performance by orchestrate CPU and GPU workloads.
- **The codes are publicly available on github.**
  https://github.com/iDC-NEU/NeutronBench

Thanks for your listening

Questions