NeutronTP: **Load-Balanced** Distributed Full-Graph **GNN Training** with **Tensor Parallelism**

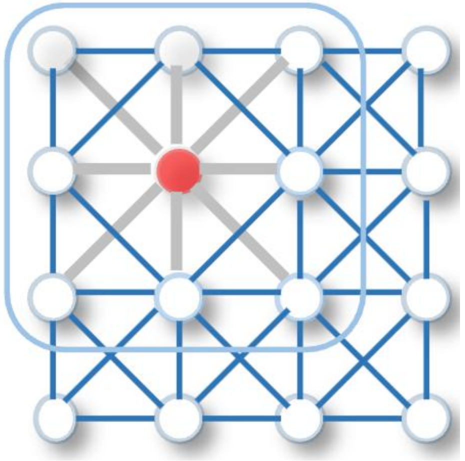
Xin Ai, Hao Yuan, Zeyu Ling, Qiange Wang, Yanfeng Zhang,
Zhenbo Fu, Chaoyi Chen, Yu Gu, Ge Yu
School of Computer Science and Engineering
Northeastern University, Shenyang, China

VLDB 2025

Graph Neural Network (GNN)

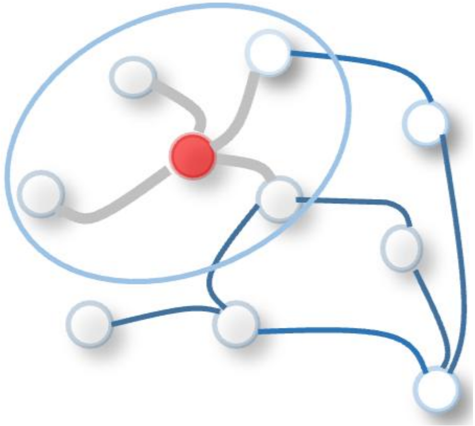
Why is it emerging?

DNN



Regular data
in Euclidean space

GNN



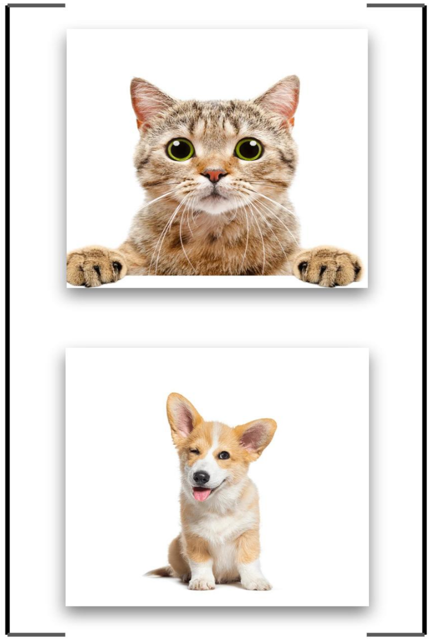
Irregular data
in non-Euclidean space

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

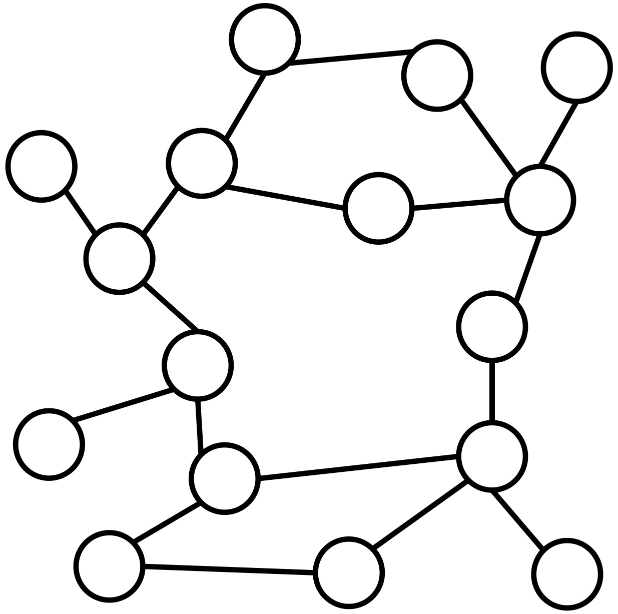
Graph Neural Network (GNN)

Dependency of GNN data samples

DNN inputs



GNN inputs



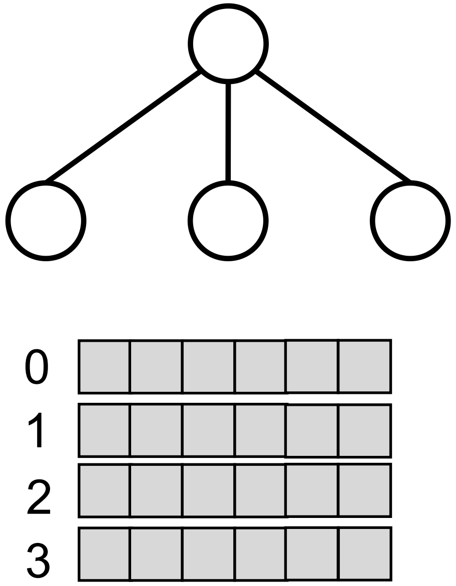
Independence

Dependencies

Graph Neural Network (GNN)

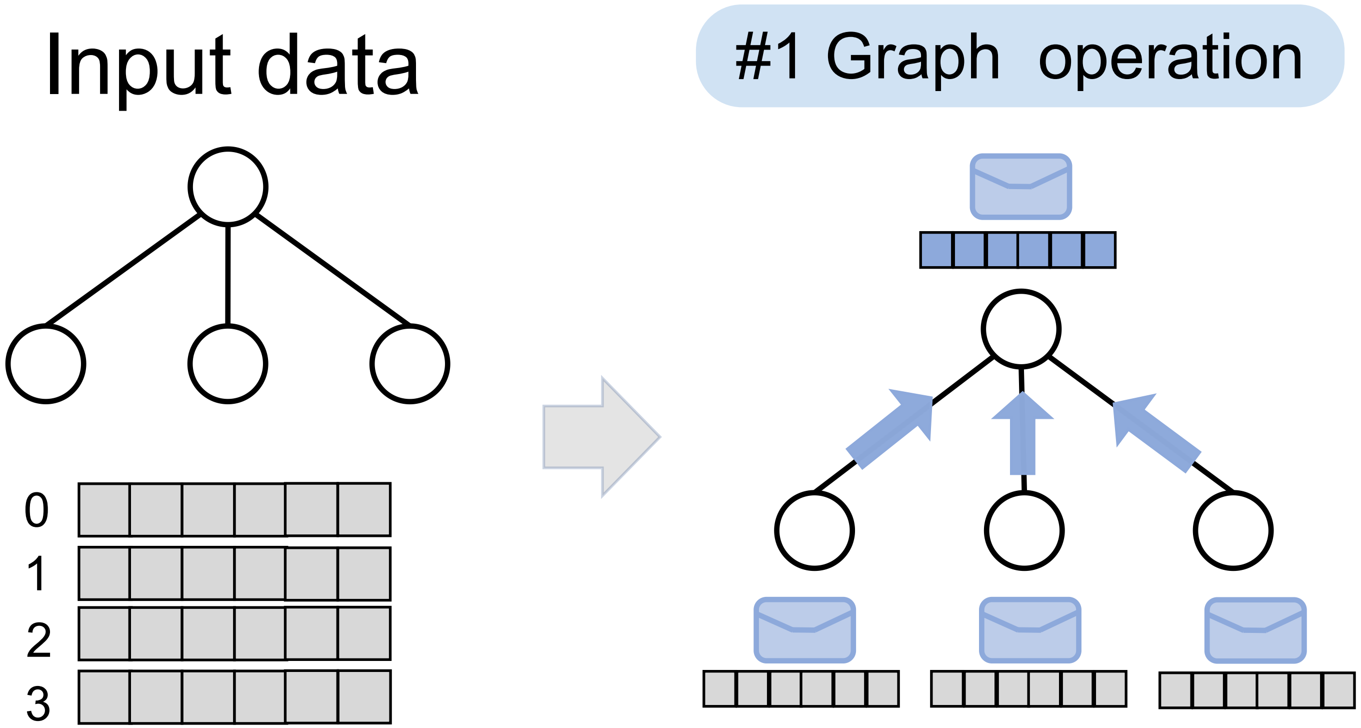
Two key stages of a GNN model

Input data



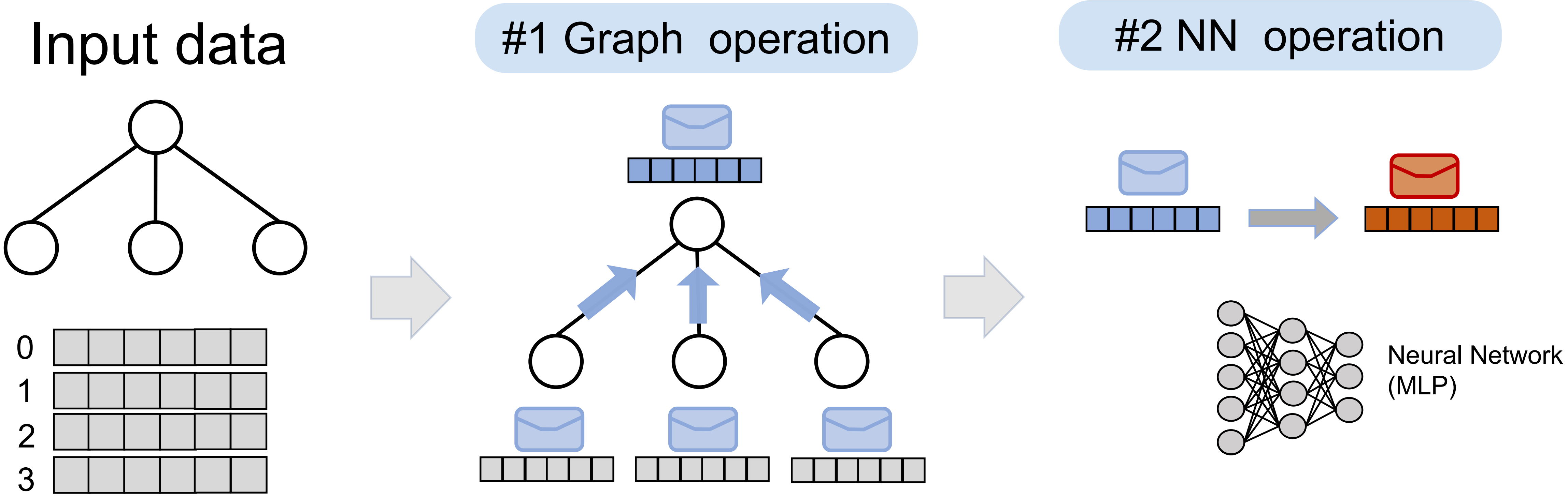
Graph Neural Network (GNN)

Two key stages of a GNN model



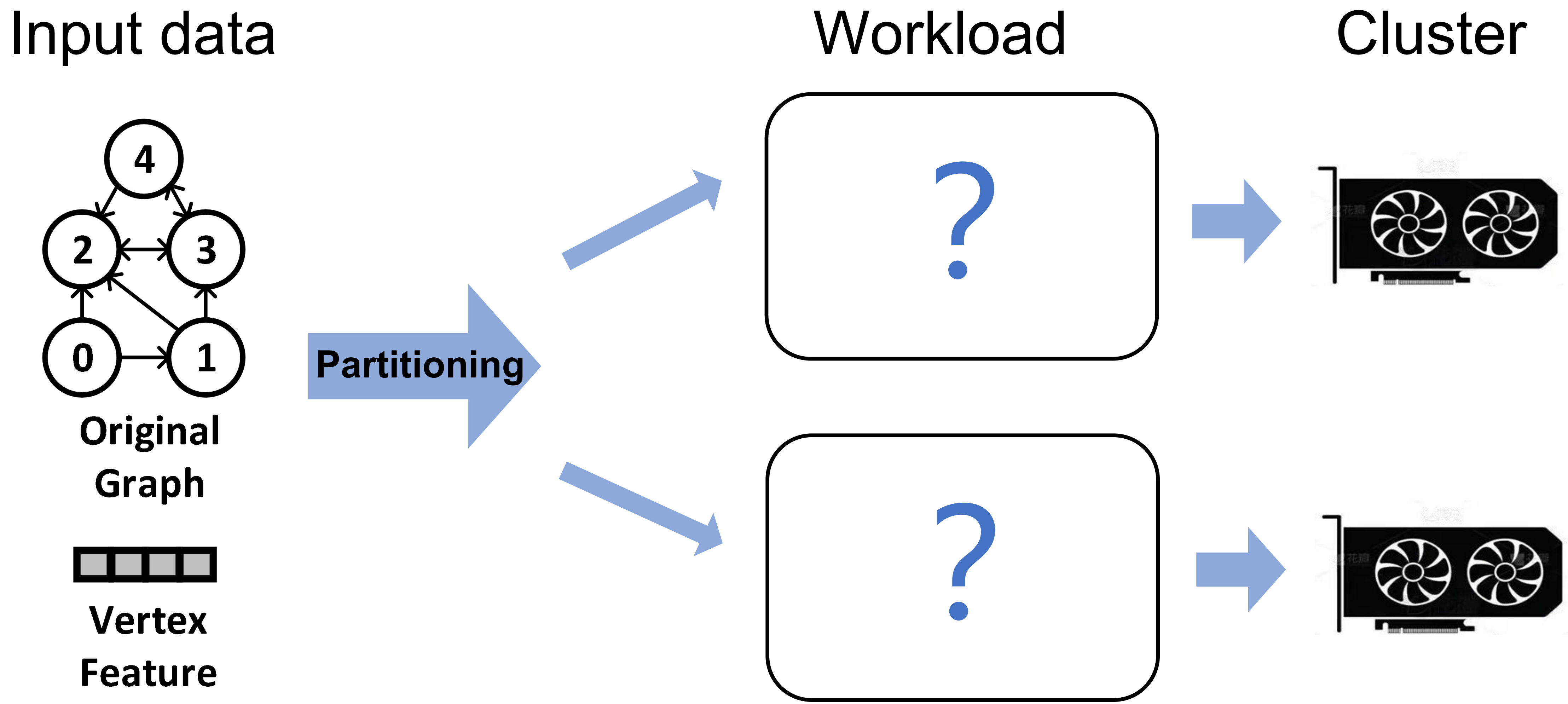
Graph Neural Network (GNN)

Two key stages of a GNN model



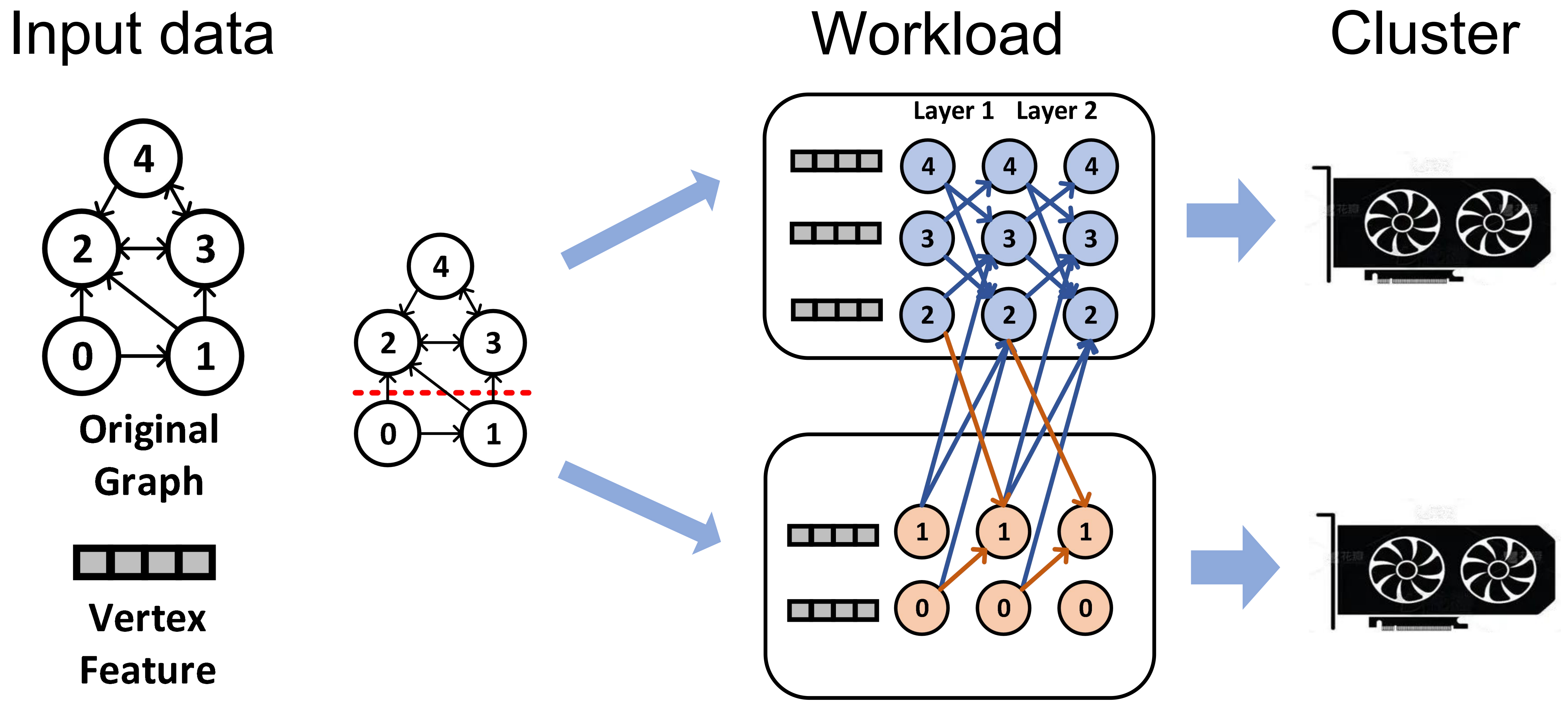
Distributed GNN Training

Workload partitioning is the key problem



Distributed GNN Training with Data Parallelism

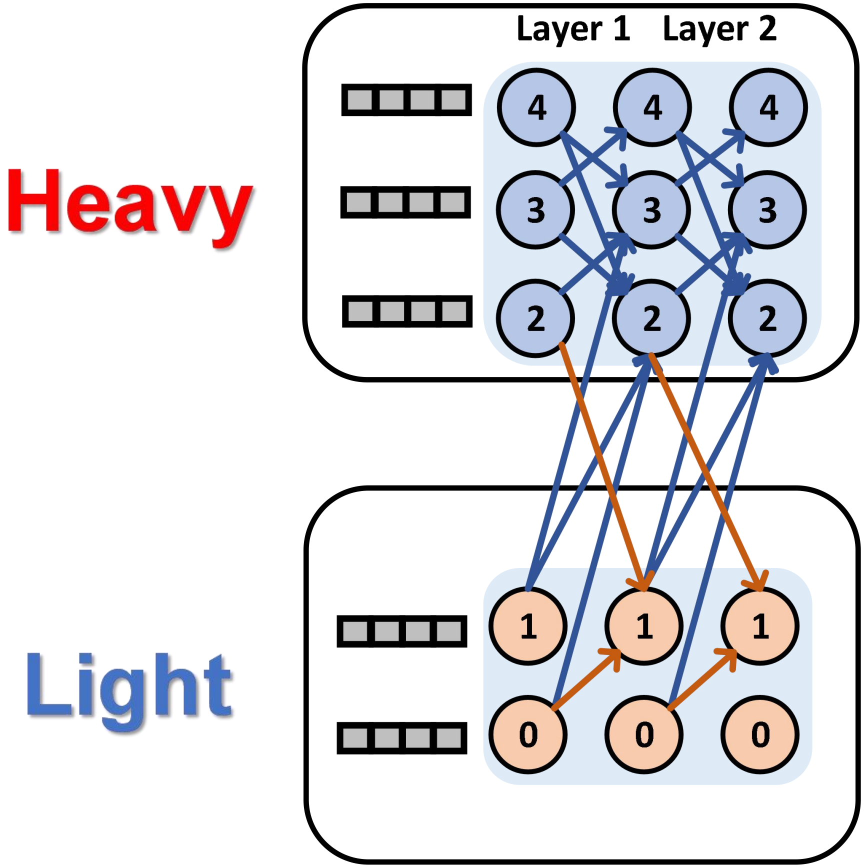
Graph partitioning is a common choice



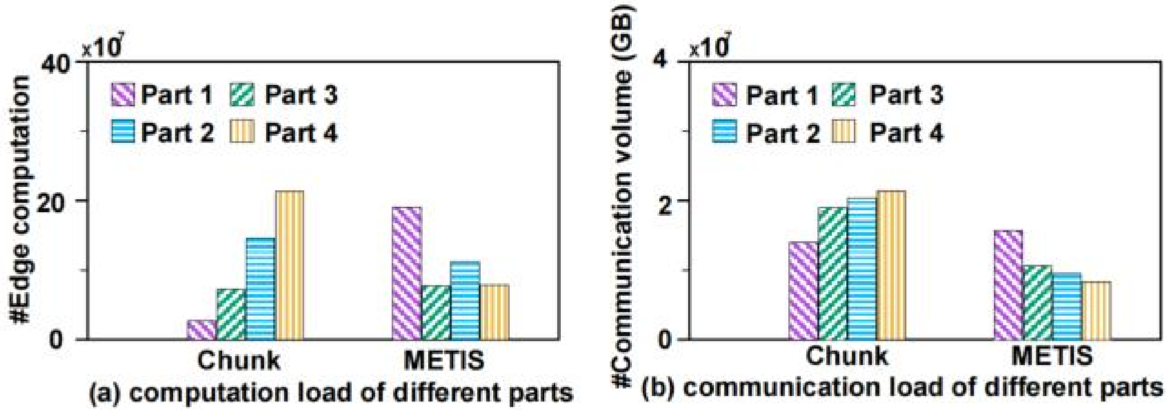
Distributed GNN Training with Data Parallelism

Graphs are difficult to partition uniformly

Workload



☹ **Workload imbalance**

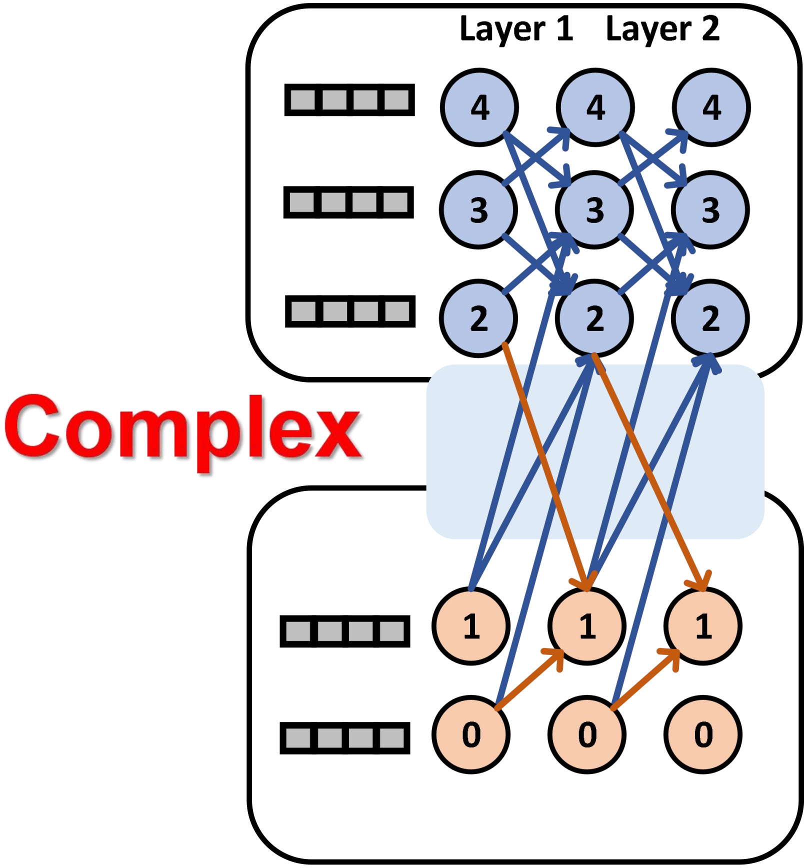


- The workload difference is up to **7.1X**.

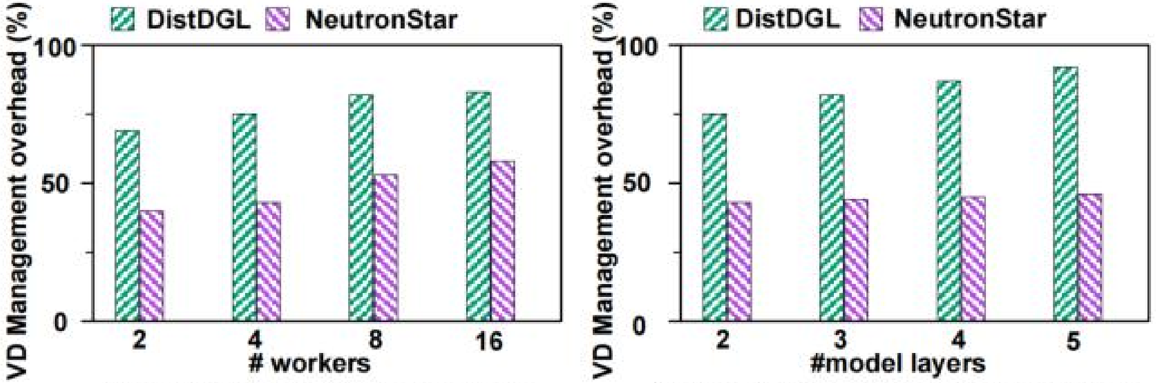
Distributed GNN Training with Data Parallelism

Each node may have many remote neighbors

Workload



☹ **Complex vertex dependencis**

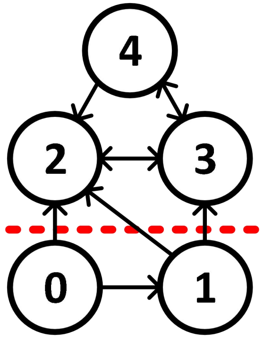


(a) varying the number of workers (b) varying the number of model layers

- VD management overhead dominates the training process, accounting for **40%–90%**.

Distributed GNN Training with Data Parallelism

Graph partitioning is the root of both problems



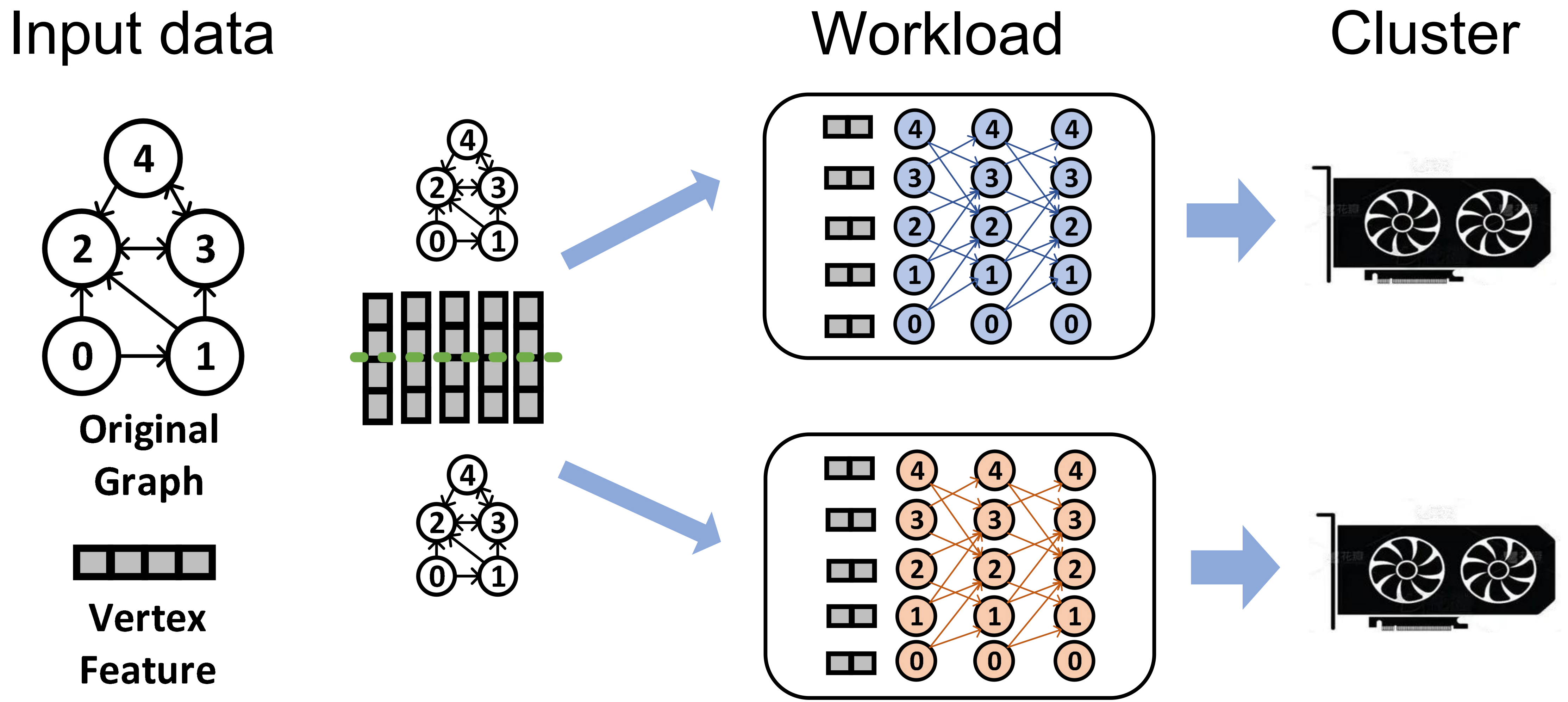
☹ **Workload imbalace**

☹ **Complex vertex dependencis**

Do we really need to **partition the graph?**

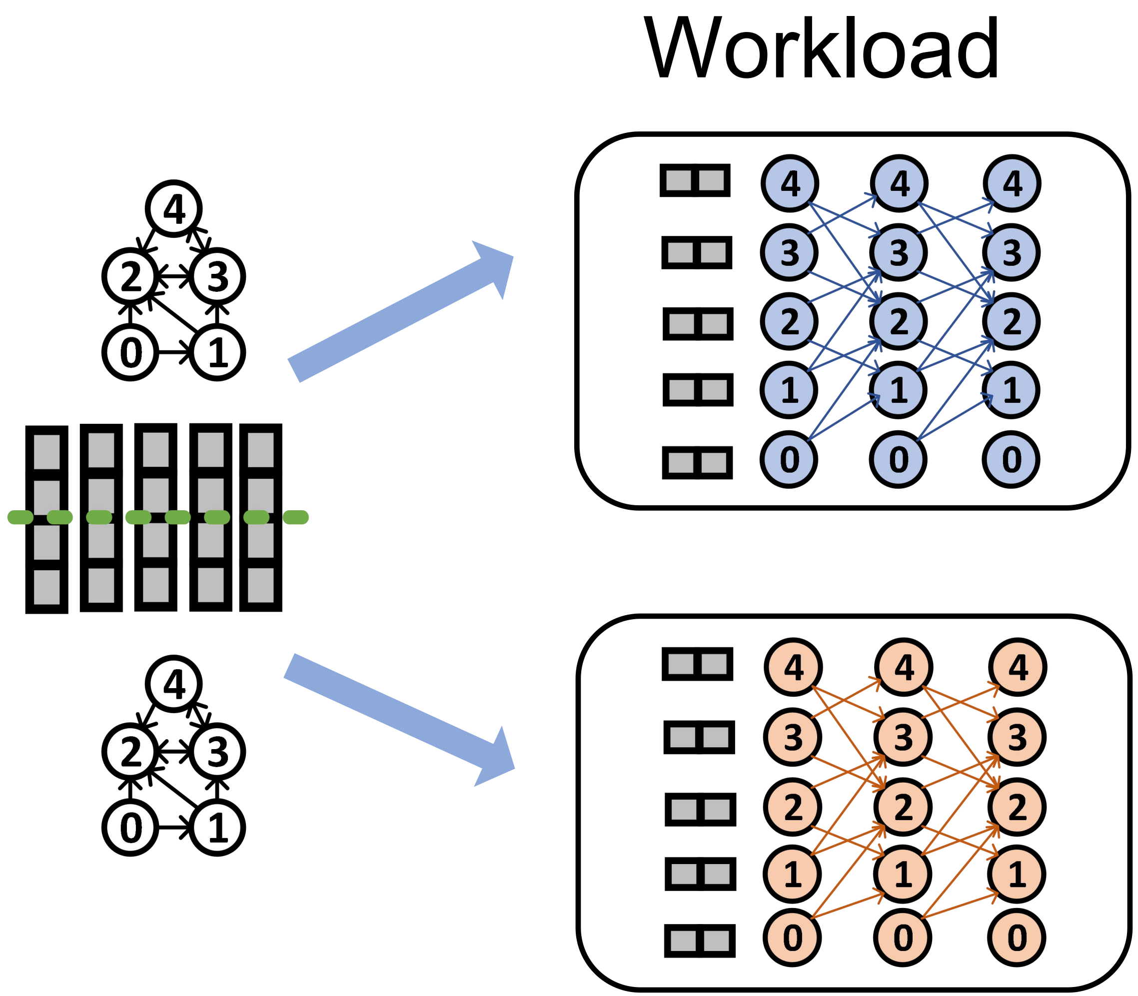
Our Solution: Tensor Parallelism

Partitioning features instead of graph structures



Our Solution: Tensor Parallelism

Partitioning features achieves balanced workload

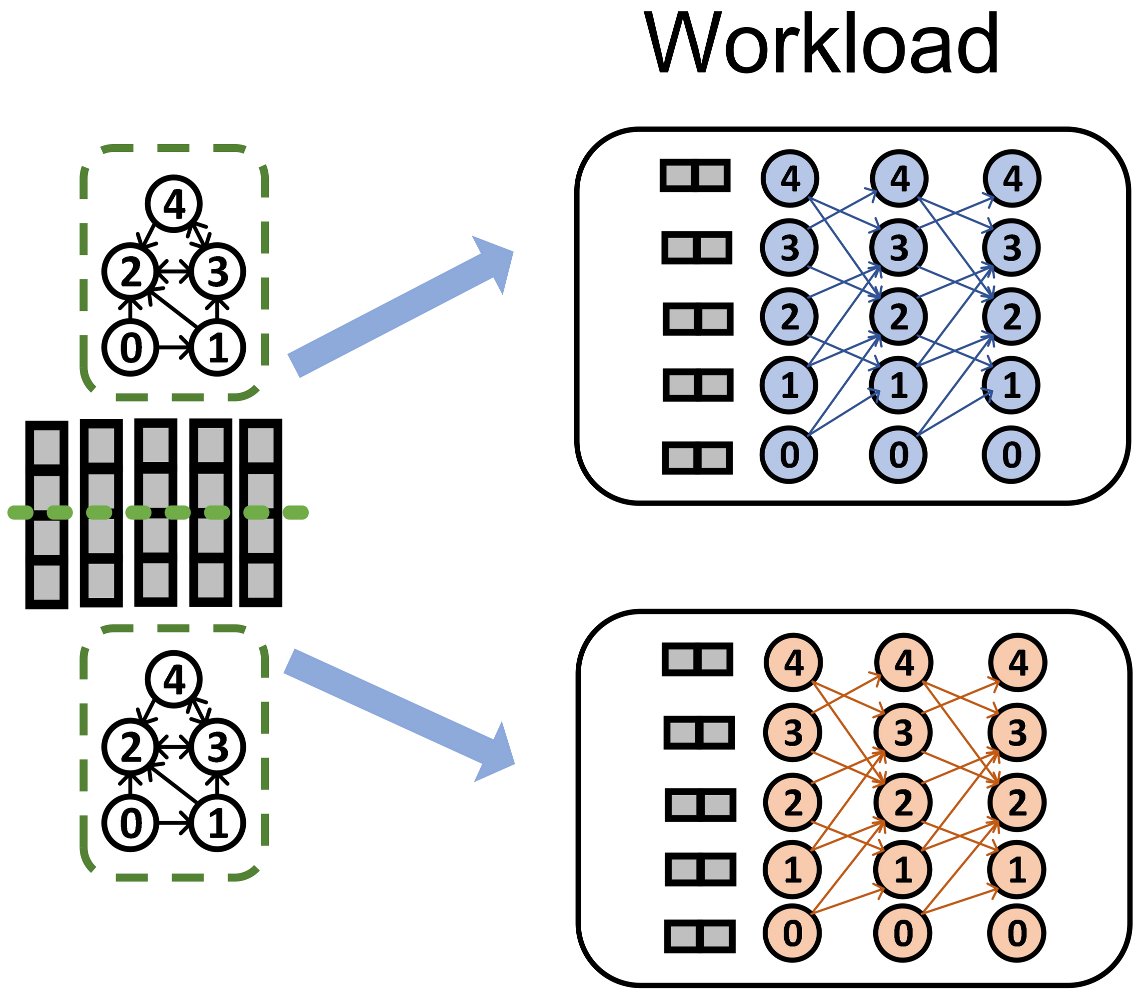


Workload balance

- Evenly partitioning feature
- The graph is replicated
- Completely balanced workload

Our Solution: Tensor Parallelism

Each worker holds a full replica of the graph



☺ **Workload balance**

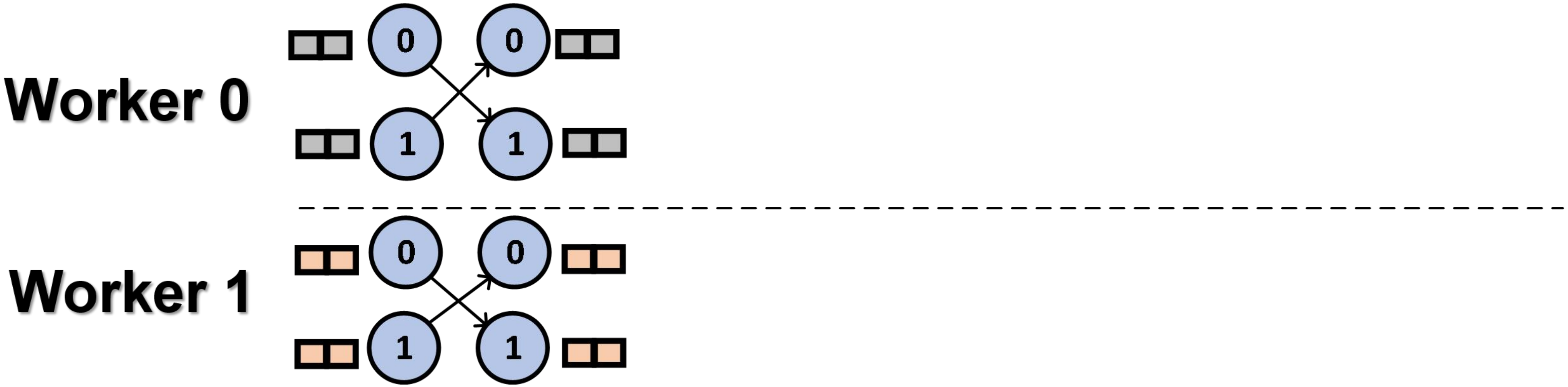
- **Evenly partitioning feature**
- **The graph is replicated**
- **Completely balanced workload**

☺ **No vertex dependencies**

Workflow of Tensor Parallelism

Workflow for a single layer

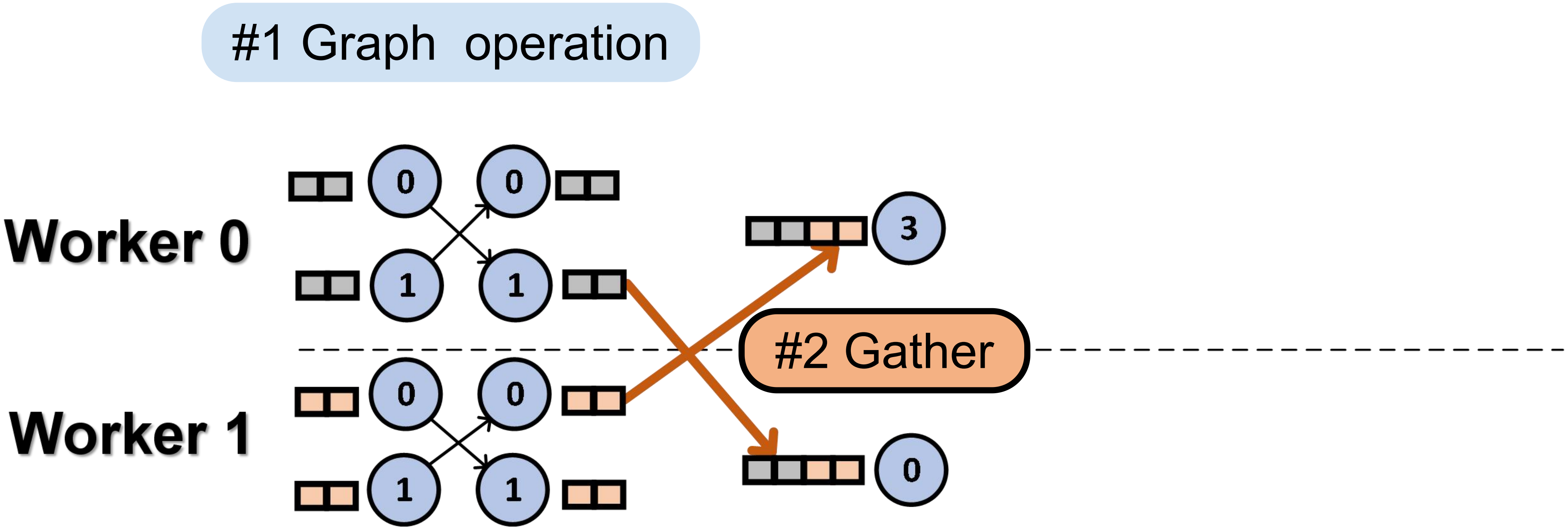
#1 Graph operation



Graph aggregation with feature slicing

Workflow of Tensor Parallelism

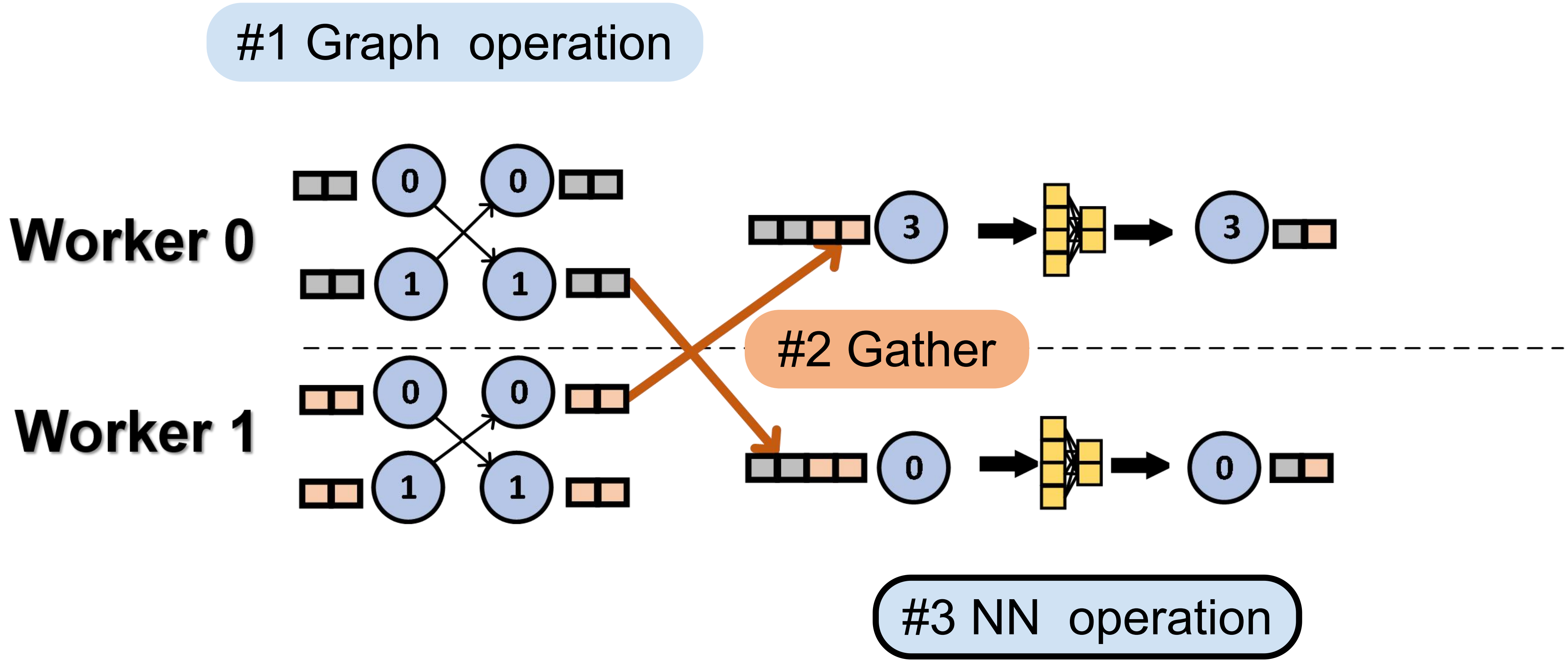
Workflow for a single layer



Gathering full features

Workflow of Tensor Parallelism

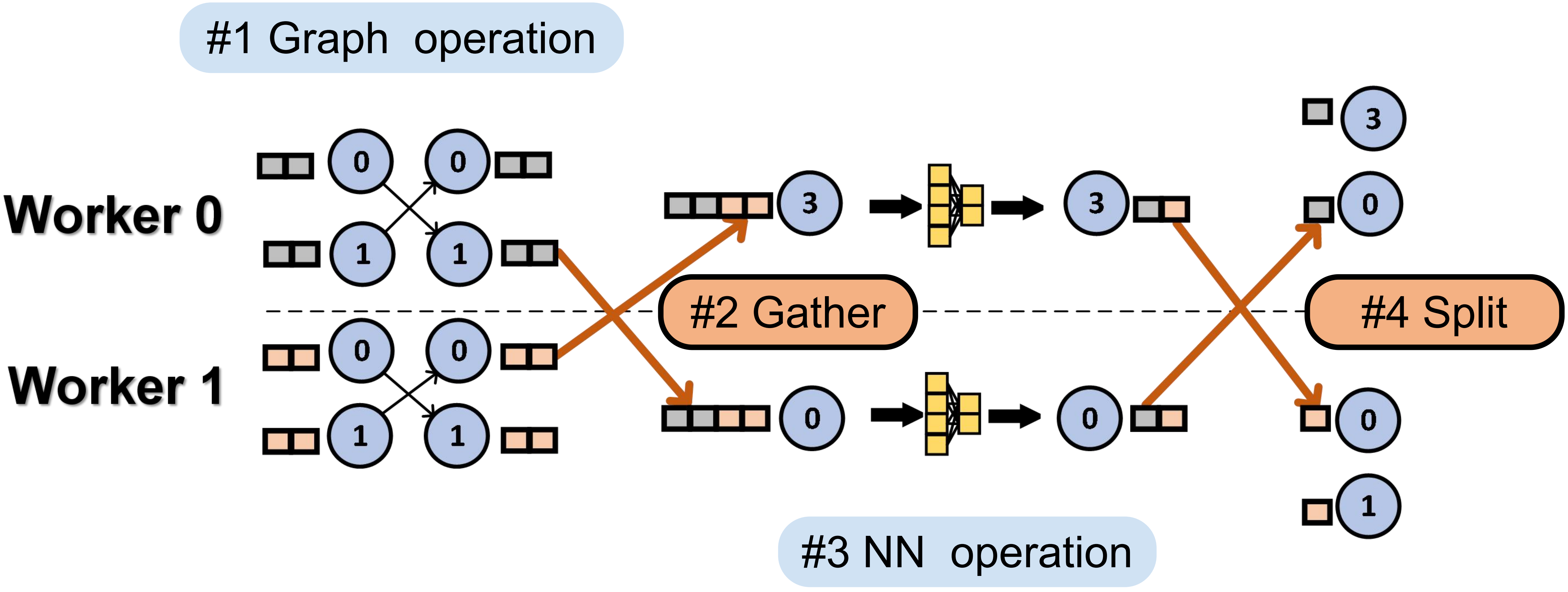
Workflow for a single layer



NN operation with full feature

Workflow of Tensor Parallelism

Workflow for a single layer



Splitting feature slicing

Challenges in Tensor Parallelism

Tensor parallelism has two major challenges

Challeges #1: Frequent collective communication

- Gather and Split in every layer
- Substantial layer-wise sync

Challeges #2: Processing the entire graph on a single worker

- Entire graph and corresponding embedding slices
- Limited GPU memory in each worker

NeutronTP

A distributed GNN system based on tensor parallelism

Generalized decoupled training method

- Decouple two opertions in GNN training
- Keep comparable model accuracy

Challeges #1

Memory-efficient task scheduling method

- Chunk-based task scheduling
- Inter-chunk pipelining

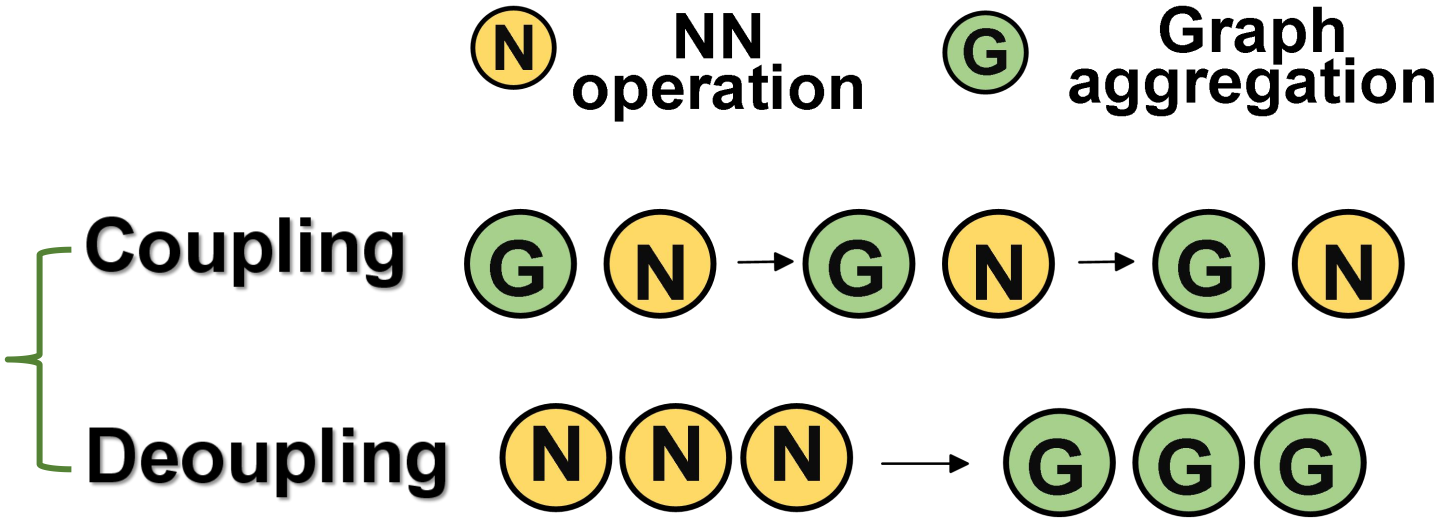
Challeges #2

Generalized Decoupled Training Method

Decoupling two opertions in GNN training

Observation: The power of GNNs stems from NN and graph operations, not their coupling execution

Comparable expressive power
Accuracy gap ≤ 1%



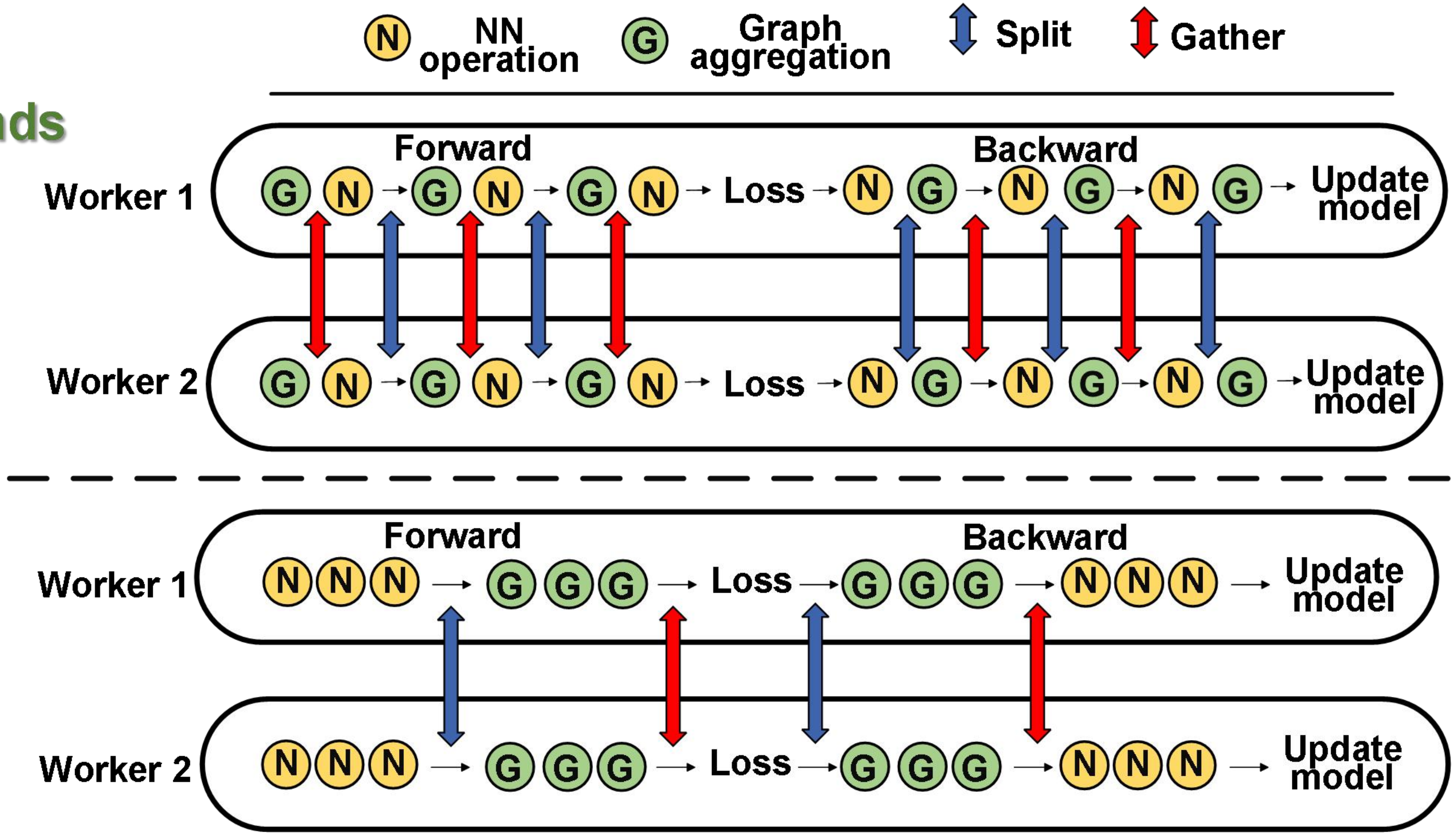
Generalized Decoupled Training Method

Reduce collective communication frequency

Reduce communication rounds

4× model depth - 2

Just 4 times



Generalized Decoupled Training Method

Provide Convergence analysis

Convergence Analysis

- **Decoupled training converges under infinite iterations**
- **Experiments demonstrate comparable accuracy**

Support decoupled training for edge-based NN operation (e.g., GAT)

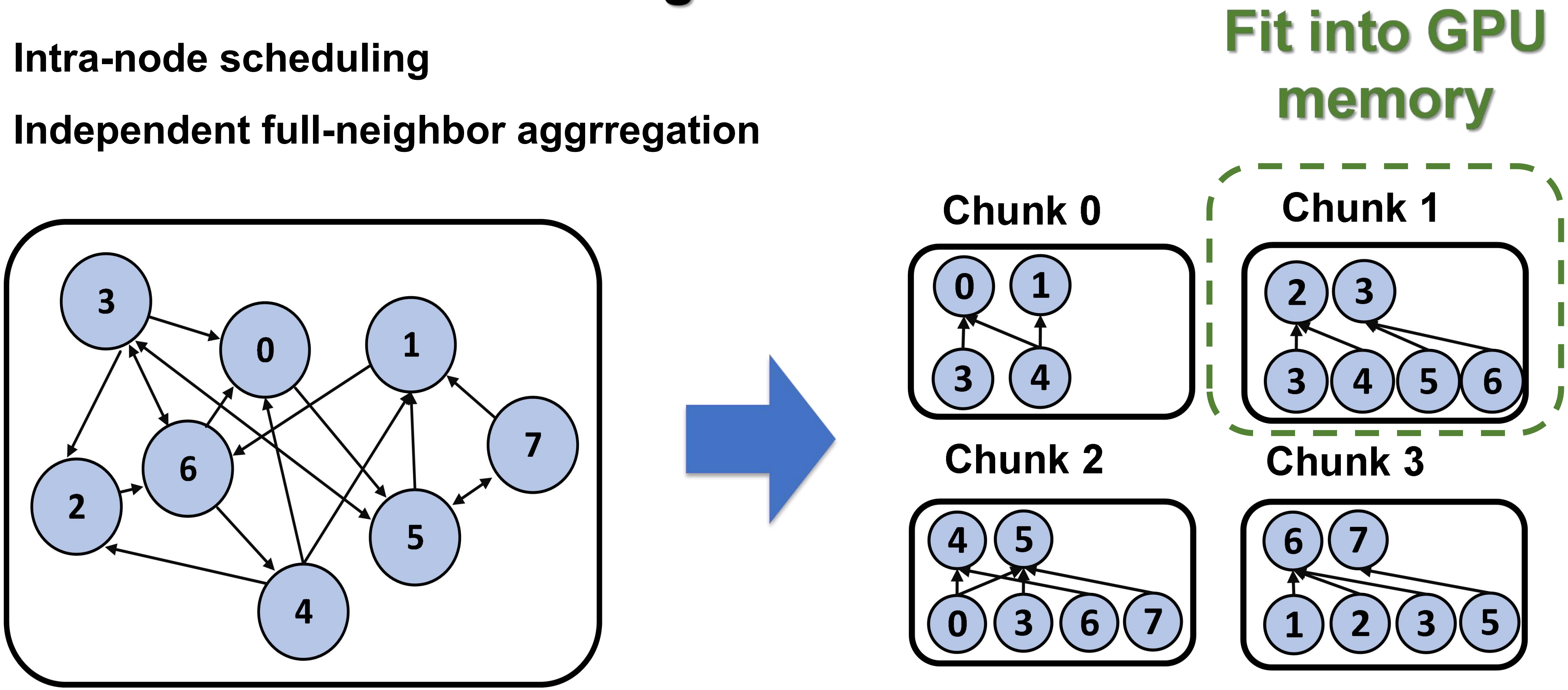
- **Edge weight precomputing**
- **Tensor-data hybrid parallelism**

Memory-efficient task scheduling method

Partition graph into chunks that can fit into GPU memory

Chunk-based task Scheduling

- Intra-node scheduling
- Independent full-neighbor aggrregation

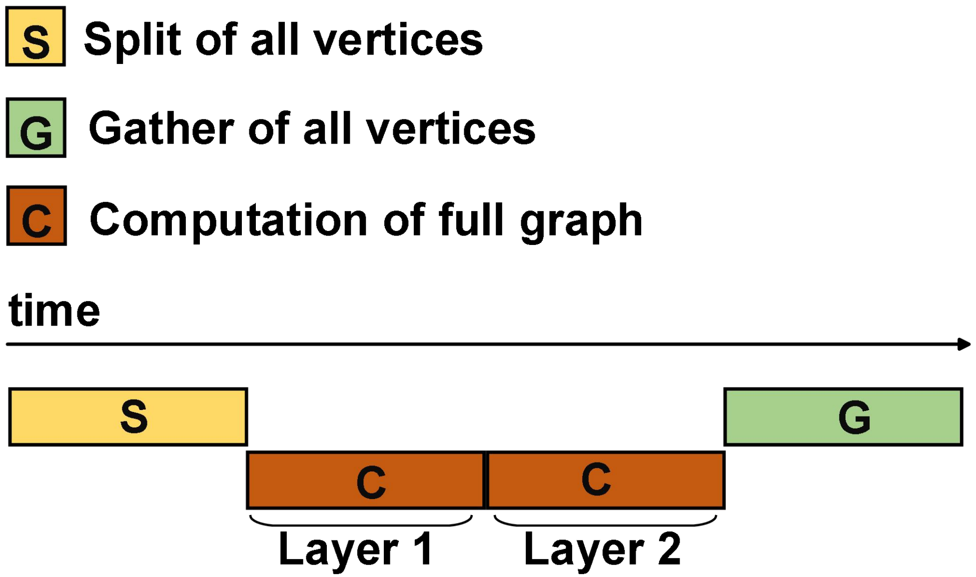


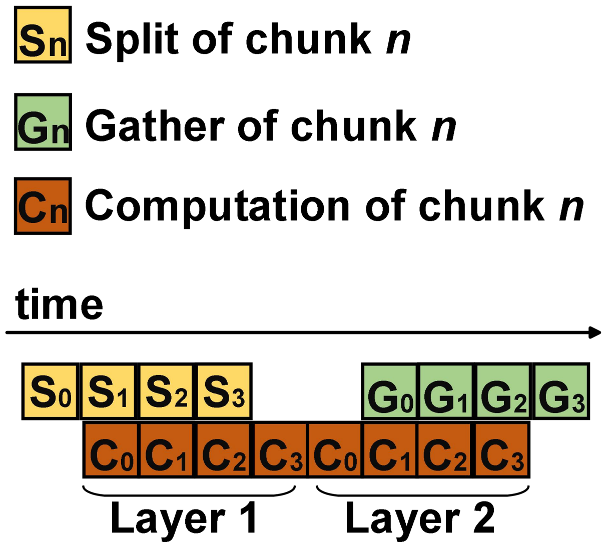
Memory-efficient task scheduling method

Hidden communication latency

Inter-chunk pipelining

- Chunk-level gather and split
- Computation-communication overlap





Experimental Setting

Competitors: **DistDGL** [Arxiv'20], **Sancus** [VLDB'22], **NeutronStar** [Sigmod'22].

Test Platforms:

A 16-node Aliyun ECS cluster[1] (Each: 16 vCPUs, 62GB RAM, 1 NVIDIA-T4 GPU)

Algorithms and Datasets:

- 2 Graph Neural Networks
  GCN, GAT
- 6 real world graphs

Softeware Environment:

- Ubuntu 18.04 LTS
- CUDA 10.1 (418.67 driver)

Table 1: Dataset description

Dataset	|V|	|E|	ftr. dim	#$\mathbb{L}$	hid. dim
Reddit (RDT)	0.23M	114M	602	41	256
Ogbn-products (OPT)	2.45M	61.68M	100	47	64
Ogbn-paper (OPR)	111.1M	1.616B	128	172	128
Friendster (FS)	65.6M	2.5B	256	64	128
Ogbn-mag (MAG)	1.9M	21M	128	349	64
Mag-lsc (LSC)	244.2M	1.7B	768	153	256

[1] Clusters are connected via 6GigE

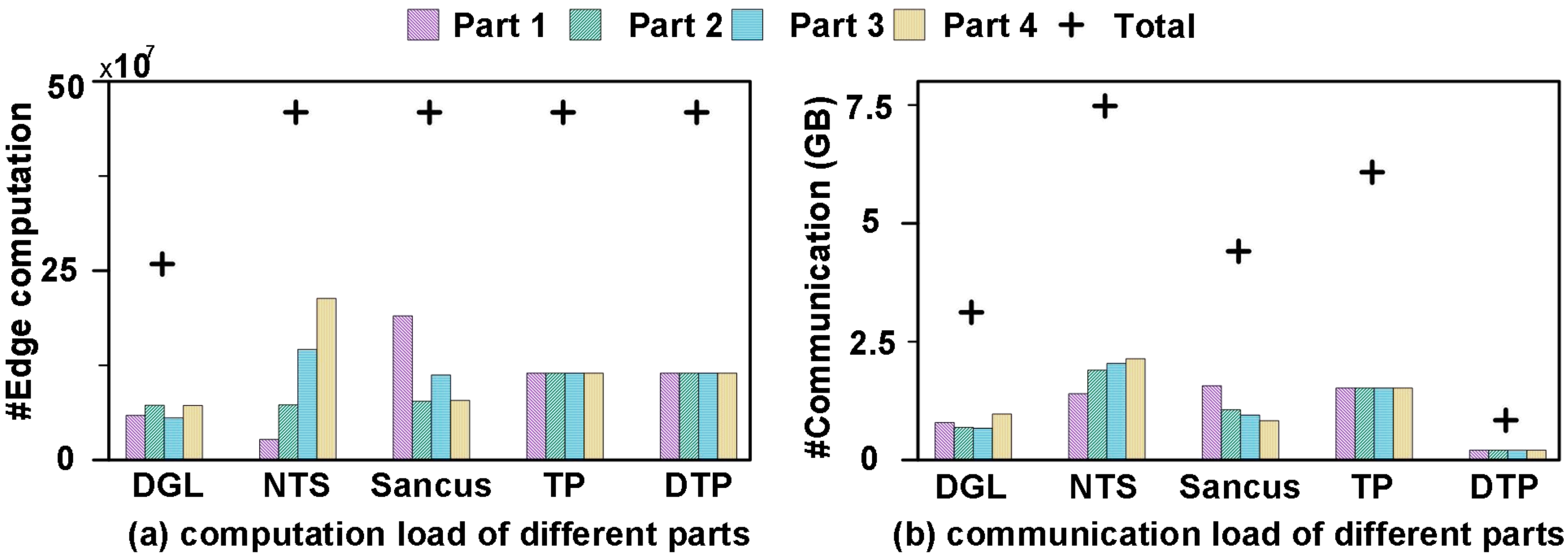
Overall Results

NeutronTP shows better performance than the competitors

- 1.29X-6.36X faster than **DistDGL**
- 4.68X-8.72X faster than **NeutronStar**
- 3.41X-4.81X faster than **Sancus**

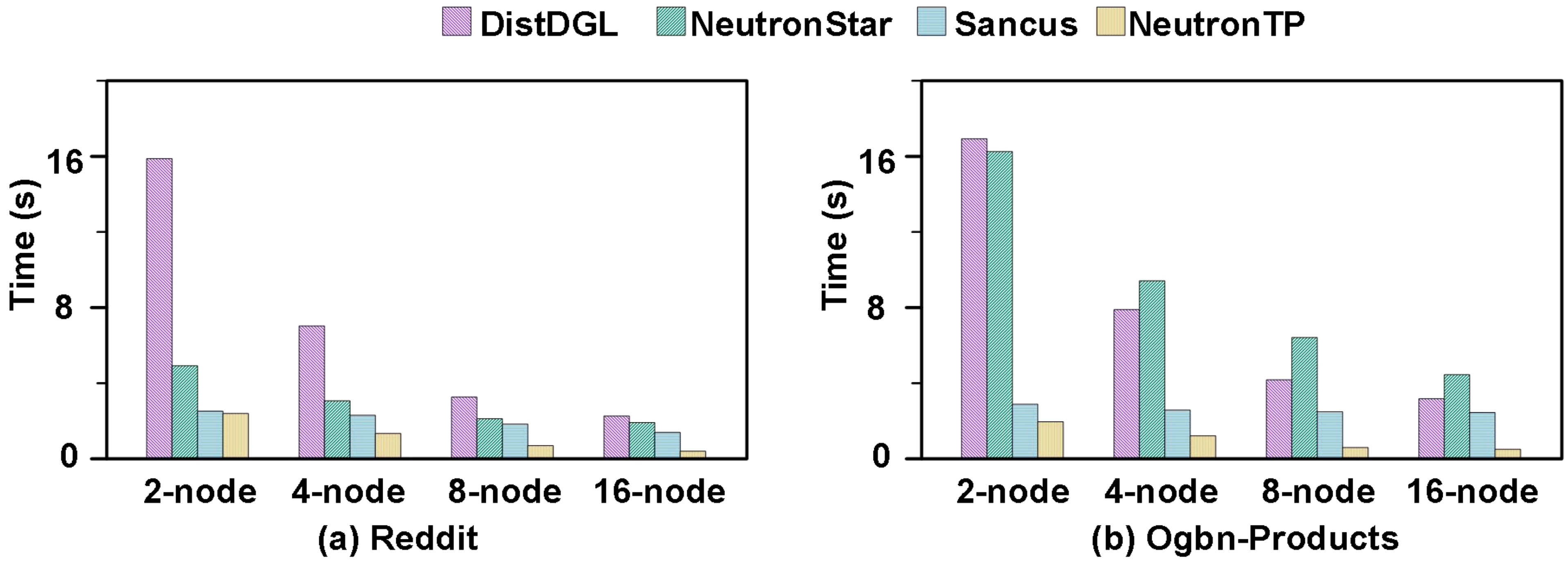
Model	Dataset	System	Runtime (s)				
			Computation		Communication		total
			max	**min**	**max**	**min**	
GCN	RDT	DistDGL	0.15	0.11	2.12	1.38	2.27
		NeutronStar	0.86	0.77	1.17	0.87	1.92
		Sancus	0.35	0.31	0.82	0.71	1.17
		NeutronTP	0.39	0.38	0.19	0.18	**0.40**
	OPT	DistDGL	0.26	0.16	2.82	1.28	3.18
		NeutronStar	2.71	1.42	2.89	1.78	4.45
		Sancus	0.86	0.36	1.59	1.22	2.45
		NeutronTP	0.46	0.44	0.24	0.22	**0.50**
	OPR	DistDGL	5.35	4.19	20.1	11.21	**25.4**
		NeutronStar	-	-	-	-	OOM
		Sancus	-	-	-	-	OOM
		NeutronTP	95.8	95.2	53.6	49.4	134.4
	FS	DistDGL	136.4	118.9	323.4	197.5	459.5
		NeutronStar	-	-	-	-	OOM
		Sancus	-	-	-	-	OOM
		NeutronTP	74.3	73.5	32.9	29.4	**90.5**
GAT	RDT	DistDGL	0.75	0.52	2.17	1.49	2.92
		NeutronStar	-	-	-	-	OOM
		Sancus	-	-	-	-	OOM
		NeutronTP	0.92	0.88	0.48	0.42	**1.29**
	OPT	DistDGL	1.17	0.94	2.76	1.29	3.93
		NeutronStar	8.72	5.98	15.9	8.29	22.4
		Sancus	-	-	-	-	OOM
		NeutronTP	2.17	1.94	1.06	0.95	**3.03**
	OPR	DistDGL	8.40	6.48	21.1	11.7	**29.5**
		NeutronStar	-	-	-	-	OOM
		Sancus	-	-	-	-	OOM
		NeutronTP	154.3	136.4	98.9	84.7	235.4
	FS	DistDGL	157.8	110.4	419.8	283.7	577.6
		NeutronStar	-	-	-	-	OOM
		Sancus	-	-	-	-	OOM
		NeutronTP	115.2	92.5	72.1	61.4	**167.9**

Workload Analysis



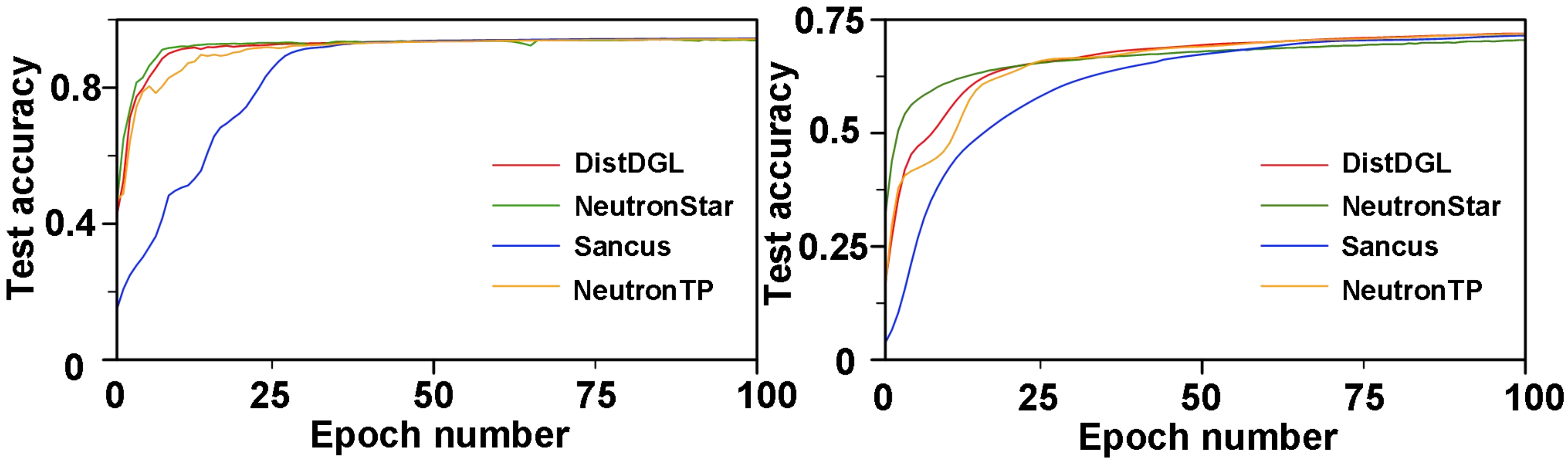
- Tensor Parallelism (TP) achieved a more balanced workload.
- Decouple training significantly reducing communication volume by up to 7.2 X.

Scalability Analysis



- **Across different cluster sizes, NeutronTP acheves an average speedup of 6.33X, 5.97X, and 2.69X compared to DistDGL, NeutronStar, and Sancus, respectively.**

Accuracy Comparison



(a) GCN on Reddit

(b) GCN on Product

- Decoupled training maintains comparable accuracy and convergence speed.

Summary

NeutronTP: Load-Balanced Distributed Full-Graph GNN Training with Tensor Parallelism.

- **GNN tensor parallelism**

  We propose a distributed GNN training method based on tensor parallelism, which eliminates cross-worker vertex dependencies and achieves complete load balancing.

Summary

NeutronTP: Load-Balanced Distributed Full-Graph GNN Training with Tensor Parallelism.

- **GNN tensor parallelism**
  We propose a distributed GNN training method based on tensor parallelism, which eliminates cross-worker vertex dependencies and achieves complete load balancing.
- **Generalized decoupling training method**
  We propose a generalized decoupling training method to separate NN operations from graph aggregation, significantly reducing communication frequency in GNN tensor parallelism.

Summary

NeutronTP: Load-Balanced Distributed Full-Graph GNN Training with Tensor Parallelism.

- □ **GNN tensor parallelism**
  We propose a distributed GNN training method based on tensor parallelism, which eliminates cross-worker vertex dependencies and achieves complete load balancing.
- □ **Generalized decoupling training method**
  We propose a generalized decoupling training method to separate NN operations from graph aggregation, significantly reducing communication frequency in GNN tensor parallelism.
- □ **Memory-efficient task scheduling strategy**
  We propose a memory-efficient task scheduling strategy to support large-scale graph processing and overlap the communication and computation.

Summary

NeutronTP: Load-Balanced Distributed Full-Graph GNN Training with Tensor Parallelism.

- **GNN tensor parallelism**
  We propose a distributed GNN training method based on tensor parallelism, which eliminates cross-worker vertex dependencies and achieves complete load balancing.
- **Generalized decoupling training method**
  We propose a generalized decoupling training method to separate NN operations from graph aggregation, significantly reducing communication frequency in GNN tensor parallelism.
- **Memory-efficient task scheduling strategy**
  We propose a memory-efficient task scheduling strategy to support large-scale graph processing and overlap the communication and computation.
- **Delivering a fast distributed GNN system**
  We develop NeutronTP, a distributed system for full-graph GNN training that utilizes tensor parallelism to achieve fully balanced workloads and integrates a series of optimizations to achieve high performance.

Summary

NeutronTP: Load-Balanced Distributed Full-Graph GNN Training with Tensor Parallelism.

- **GNN tensor parallelism**
  We propose a distributed GNN training method based on tensor parallelism, which eliminates cross-worker vertex dependencies and achieves complete load balancing.
- **Generalized decoupling training method**
  We propose a generalized decoupling training method to separate NN operations from graph aggregation, significantly reducing communication frequency in GNN tensor parallelism.
- **Memory-efficient task scheduling strategy**
  We propose a memory-efficient task scheduling strategy to support large-scale graph processing and overlap the communication and computation.
- **Delivering a fast distributed GNN system**
  We develop NeutronTP, a distributed system for full-graph GNN training that utilizes tensor parallelism to achieve fully balanced workloads and integrates a series of optimizations to achieve high performance.
- **The codes are publicly available on github**
  https://github.com/iDC-NEU/NeutronTP

Thanks for your listening