LeapGNN: Accelerating Distributed GNN Training Leveraging Feature-Centric Model Migration

Weijian Chen, Shuibing He, Haoyang Qu, Xuechen Zhang[#]

浙江大学 Zhejiang University

[#] Washington State University

USENIX FAST 2025

Graph Neural Network (GNN)

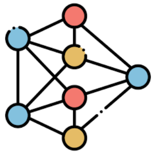
- GNNs are designed for learning from graph-structured data.

Image

Convolution Neural Network (CNN)

Text

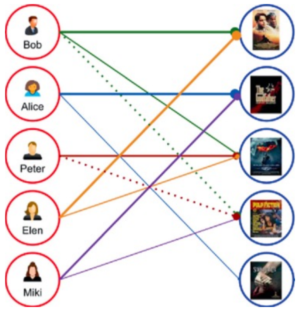
LSTM, Transformer



Graph

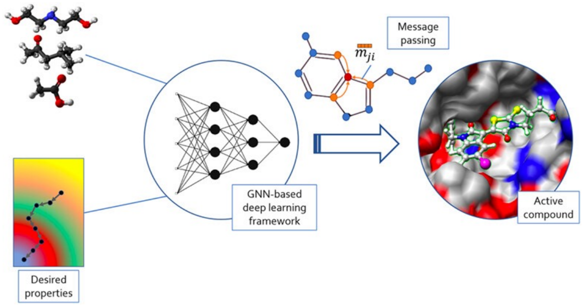
Graph Neural Network (GNN)

- GNN has been used for vertex/graph classification, edge prediction in many domains.



Recommendation Systems

Social Networks Analysis

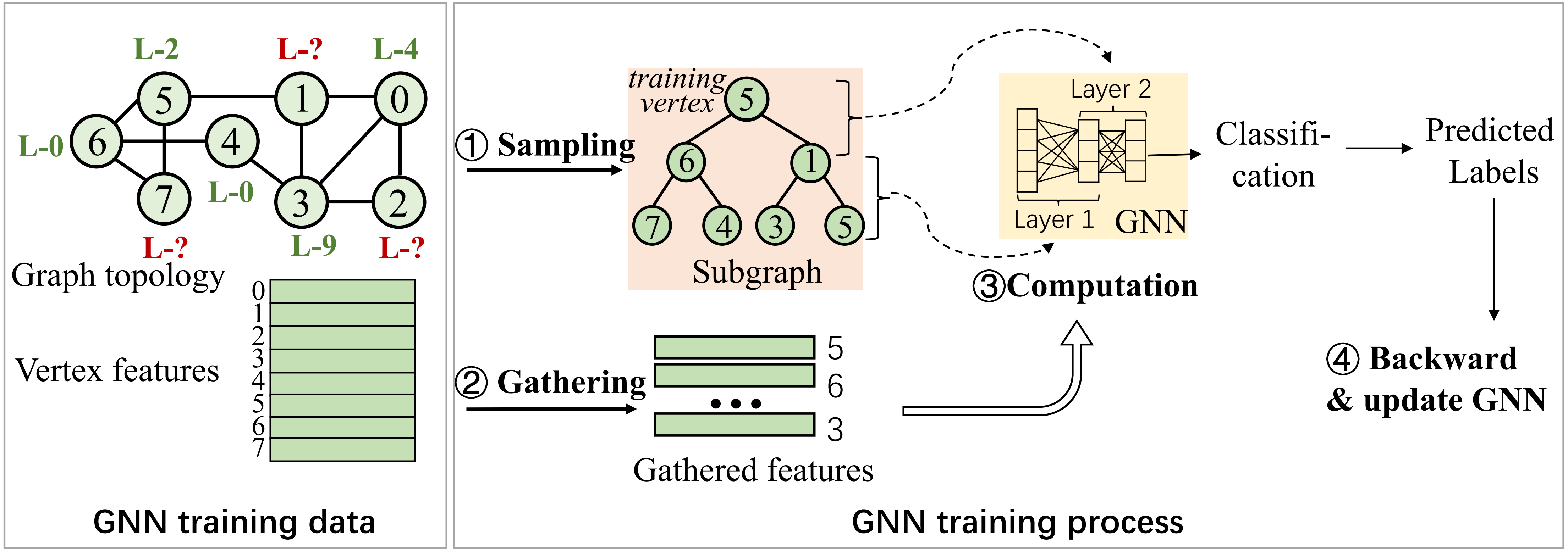


Drug Discovery

* *Image Source: Internet*

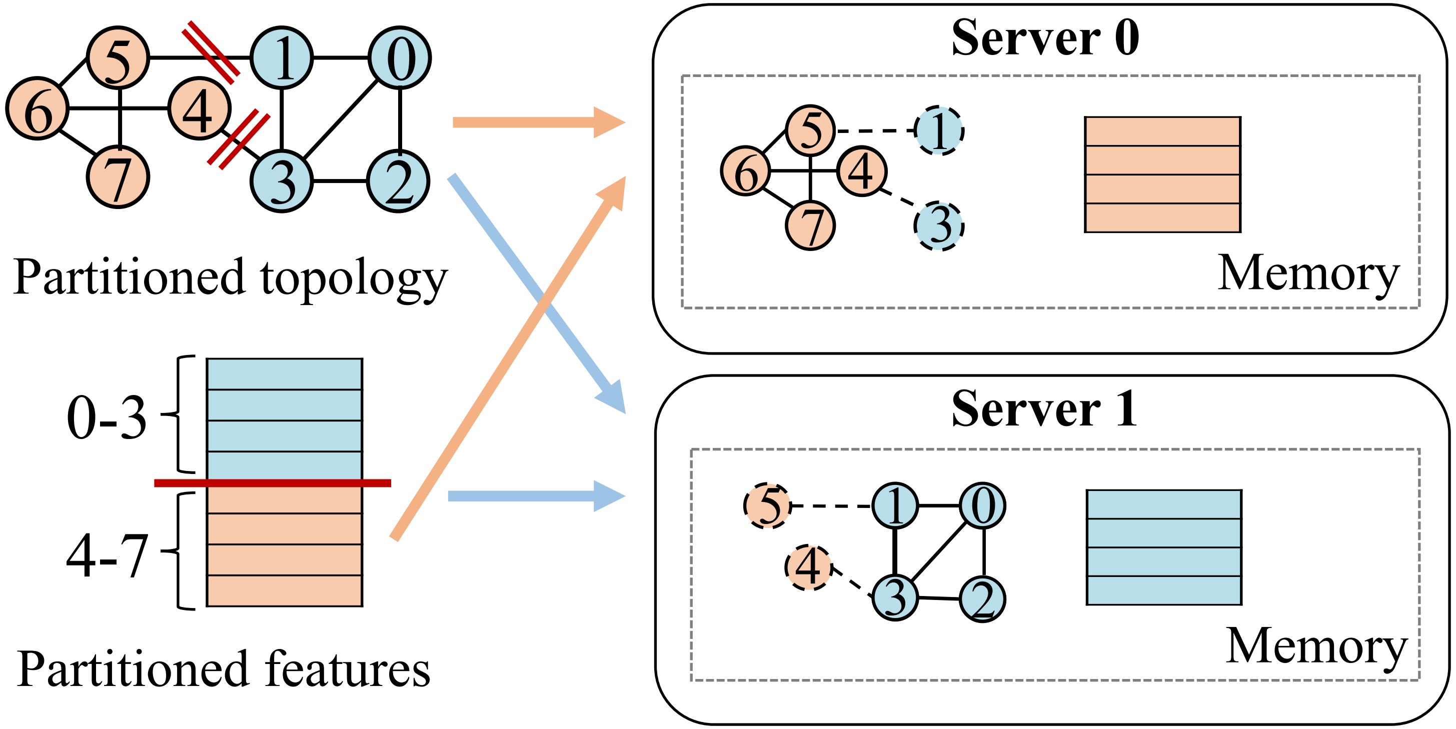
Graph Neural Network (GNN) Training

➢ **Sampling-based GNN training** is a standard approach for large-graph training.



Distributed GNN Training

➢ **Distributed sampling-based GNN training** on multiple servers.



Graph & features are partitioned

Distributed GNN Training

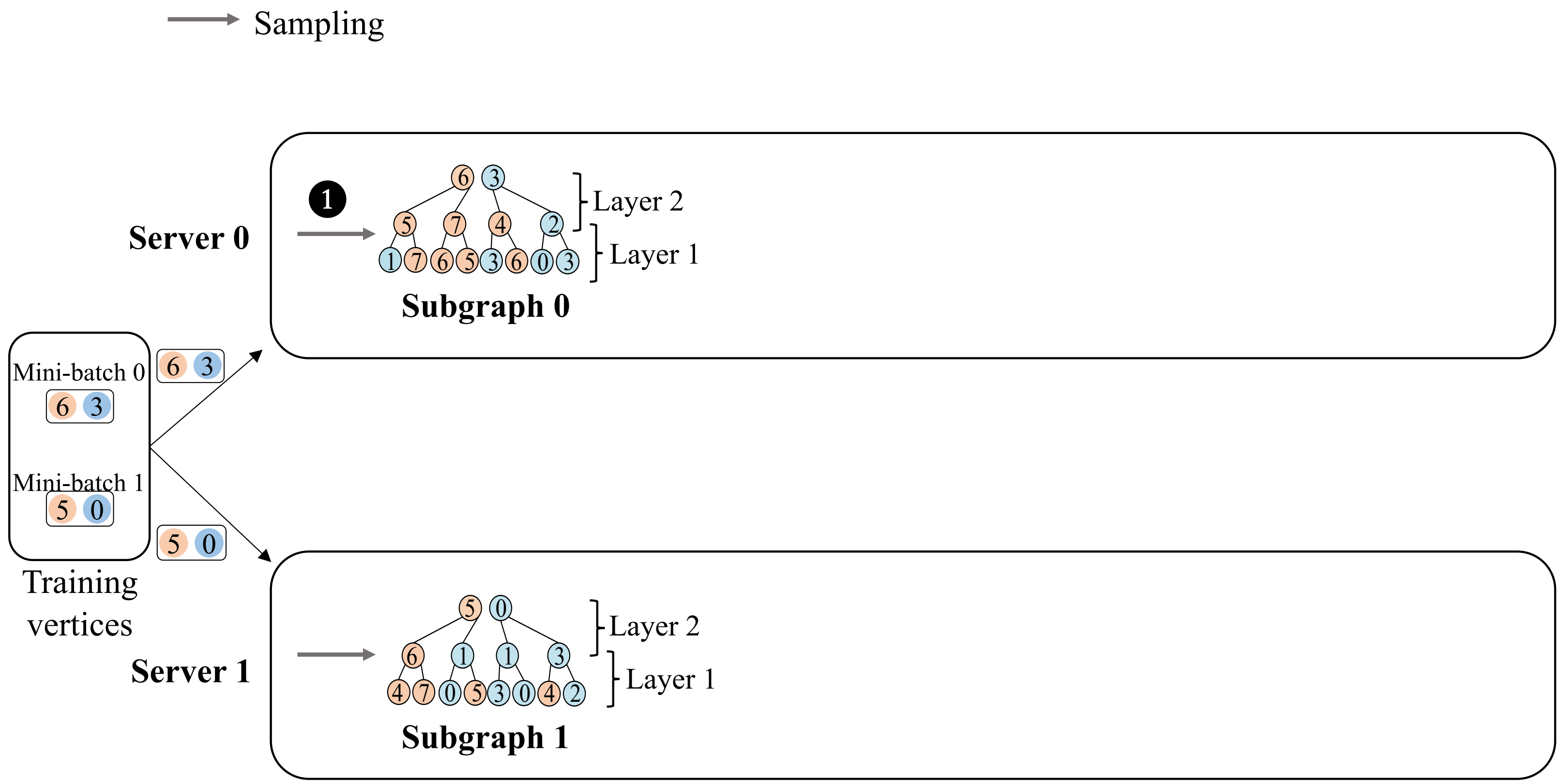
➢ **Distributed sampling-based GNN training** on multiple servers.

Vertex Color	Server ID
red	0
blue	1

Distributed GNN Training

- **Distributed sampling-based GNN training** on multiple servers.

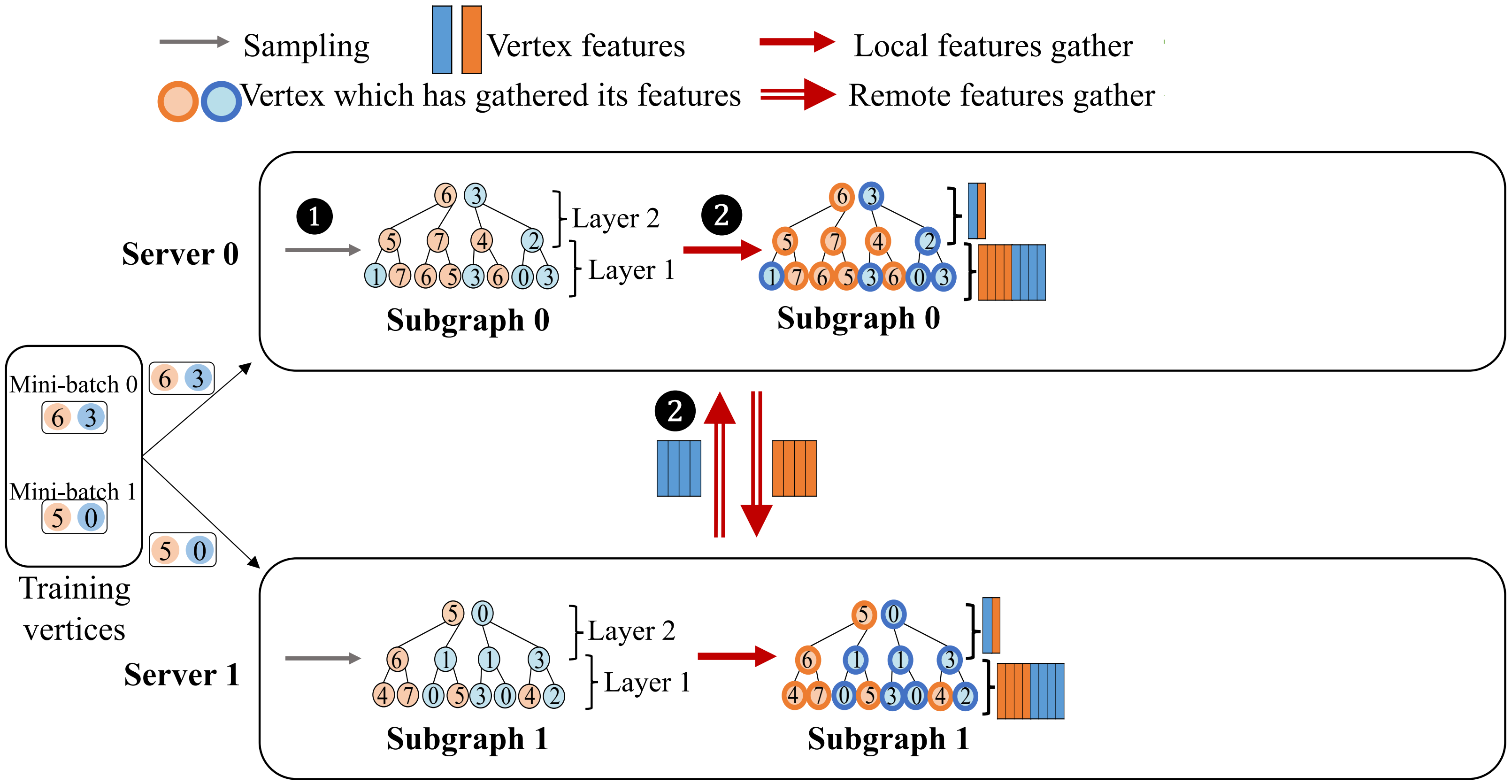
Vertex Color	Server ID
red	0
blue	1



Distributed GNN Training

- **Distributed sampling-based GNN training** on multiple servers.

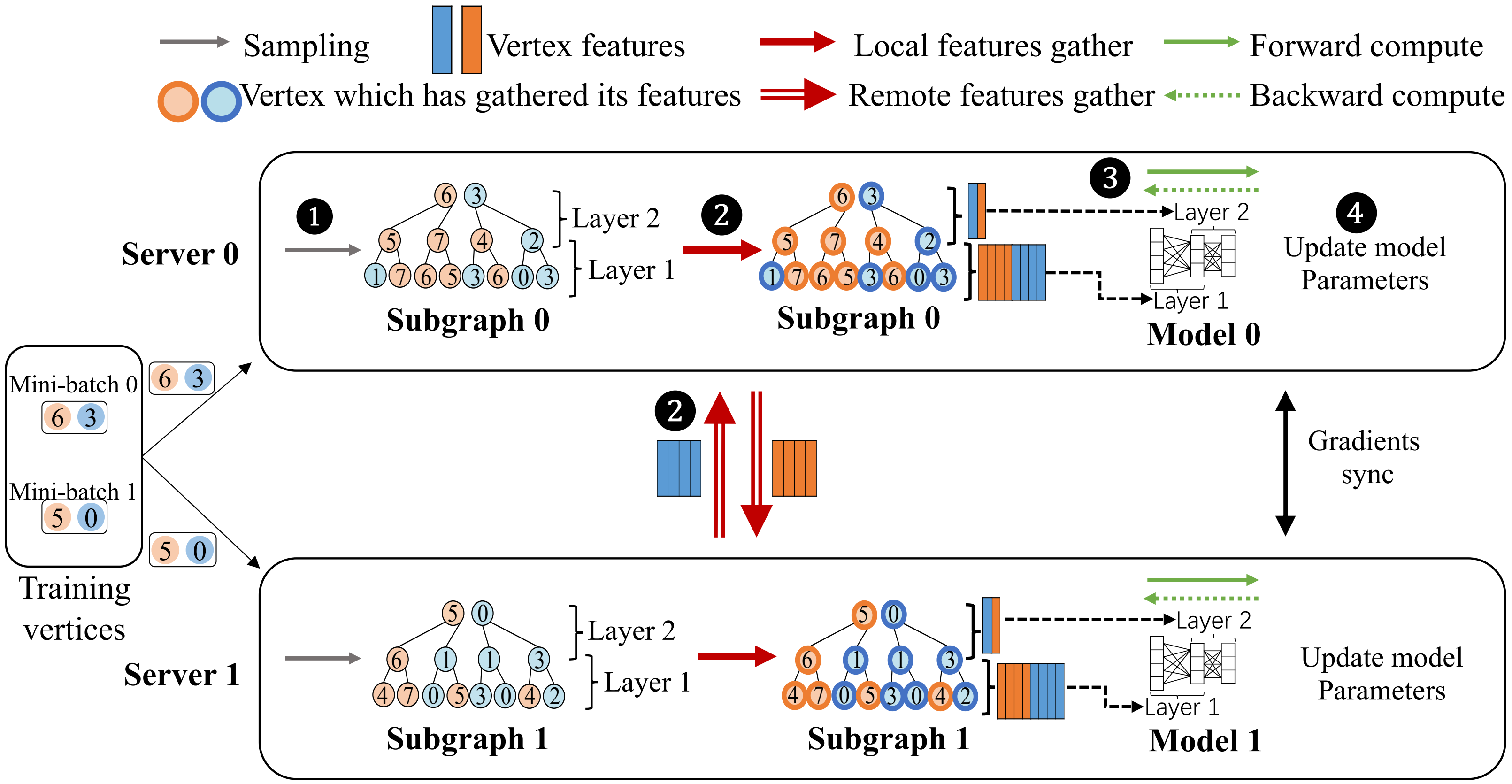
Vertex Color	Server ID
red	0
blue	1



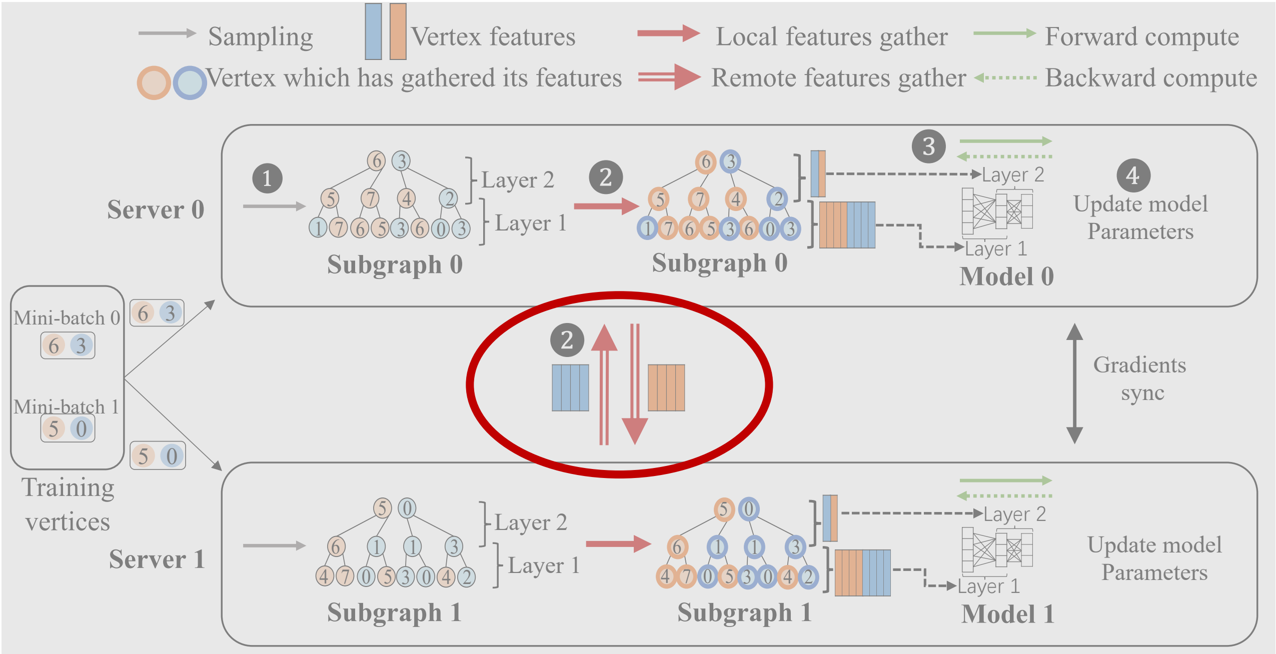
Distributed GNN Training

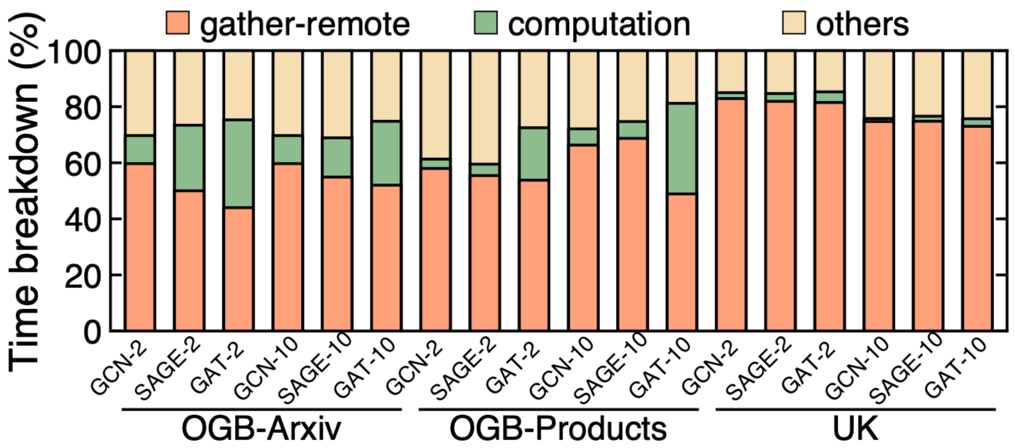
- **Distributed sampling-based GNN training** on multiple servers.

Vertex Color	Server ID
red	0
blue	1



Bottleneck of Distributed GNN Training





Remote vertex feature gathering causes the communication bottleneck! (44%~83%)

Existing Methods

Limitations

- **Partitioning Optimization**

GNN-aware graph partitioning to reduce cross-server feature transmission. [DistDGL-IA320, ROC-MLSys20, ByteDance-VLDB22, BGL-NSDI23]

Insufficient due to dynamic and random nature of sampling.

- **Sampling Optimization**

Locality-aware sampling to reduce the probability of being sampled for remote vertices. [Pagraph-SoCC20, DistGNN-SC21, LAS-ICS24]

Compromise model accuracy.

- **Cache Optimization**

Cache hot features in GPU to reduce the redundant feature gathering. [PaGraph-SoCC20, GNNLab-EuroSys22, BGL-NSDI23, Legion-ATC23]

Limited by the cache size.

- **New Training Schema**

Combine model parallelism and data parallelism to avoid original feature transfers. [P3-OSDI21]

Introduce additional intermediate feature transfer.

We name these methods as "model-centric" methods.

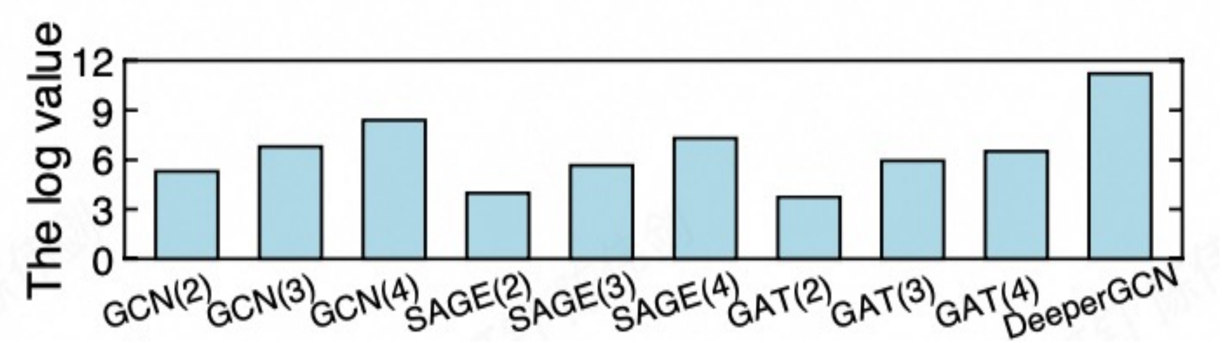
Outline

- Background & Related work
- **New observation**
- Our naïve method & Challenges
- Design
- Experimental Results

New Observation

➢ **The amount of data transferred for vertex feature gathering is significantly larger than the GNN model size.**

$$\alpha = \frac{training\ data\ transfer\ between\ servers}{model\ parameter\ size}$$

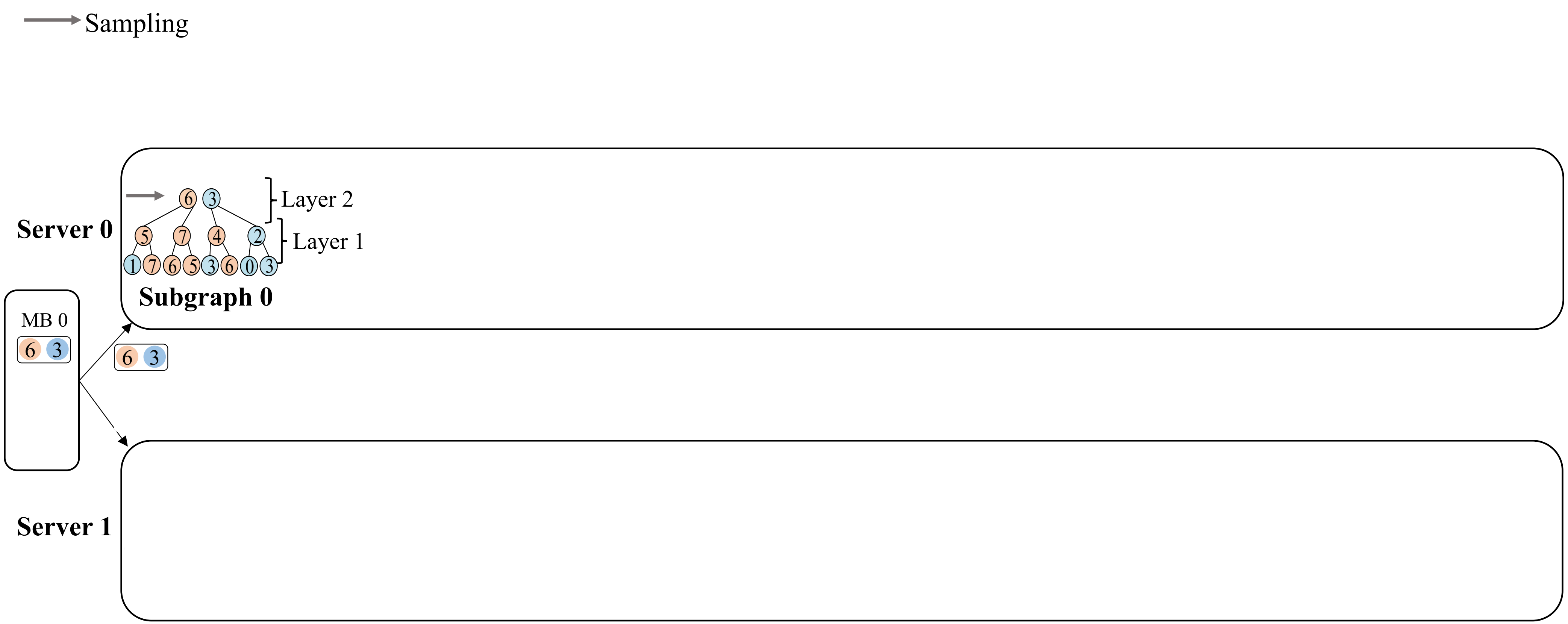


Feature-centric method.

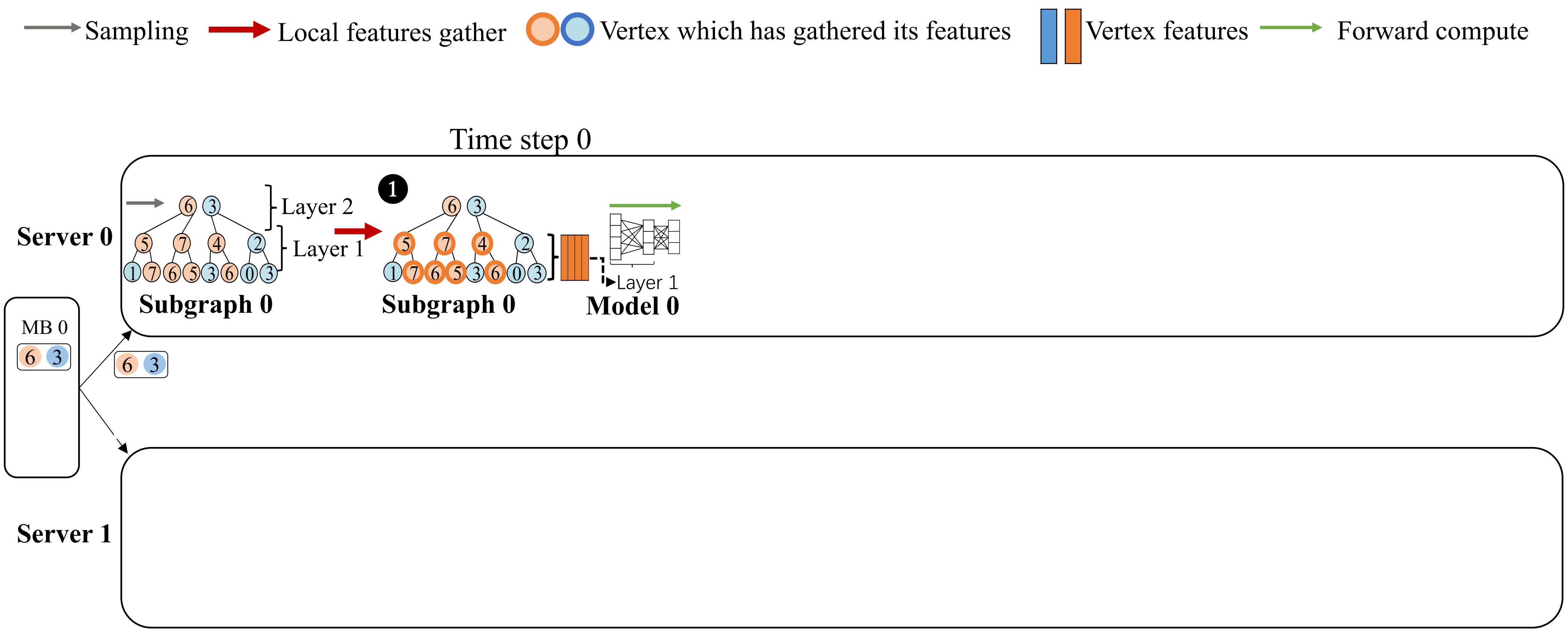
Move the model to the servers where the vertex features are located, **rather than fetching the features** from the remote servers.

(Denoted as ***Naïve model migration*** method)

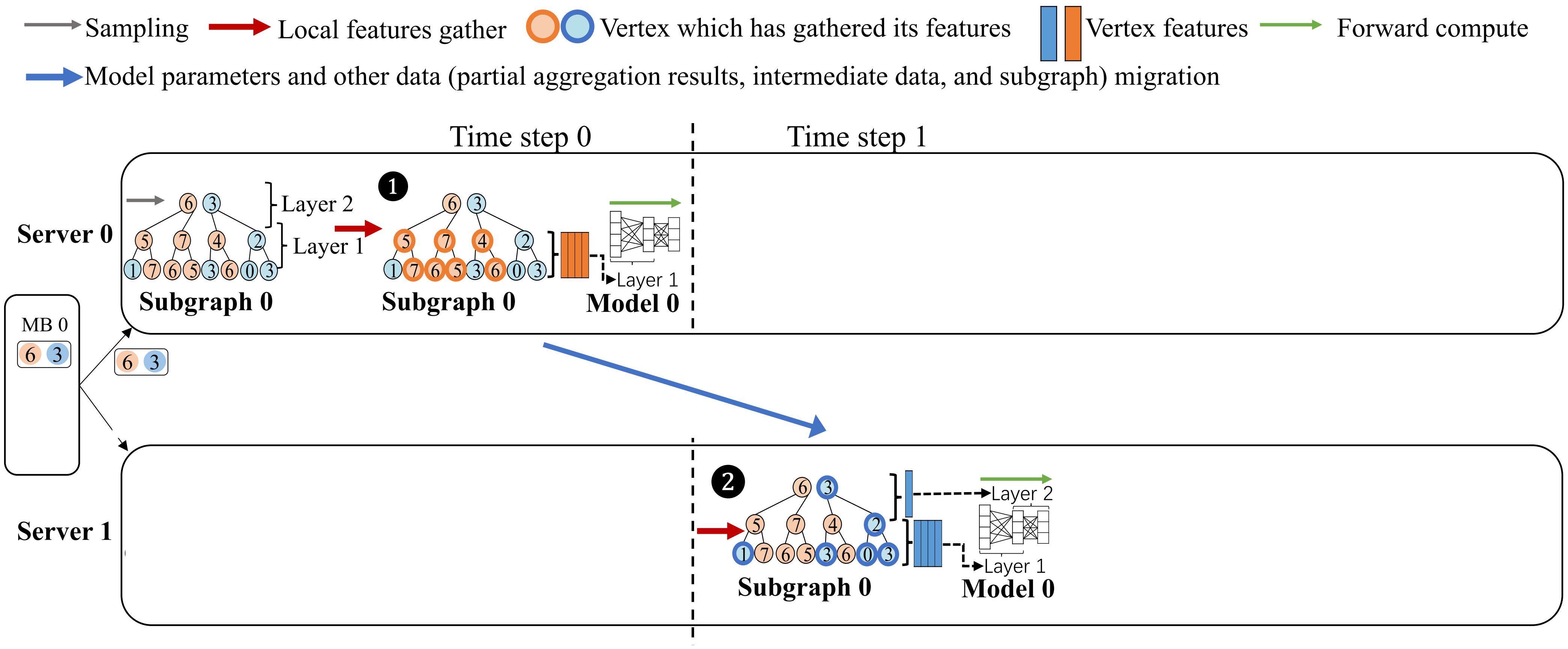
Naïve Model Migration



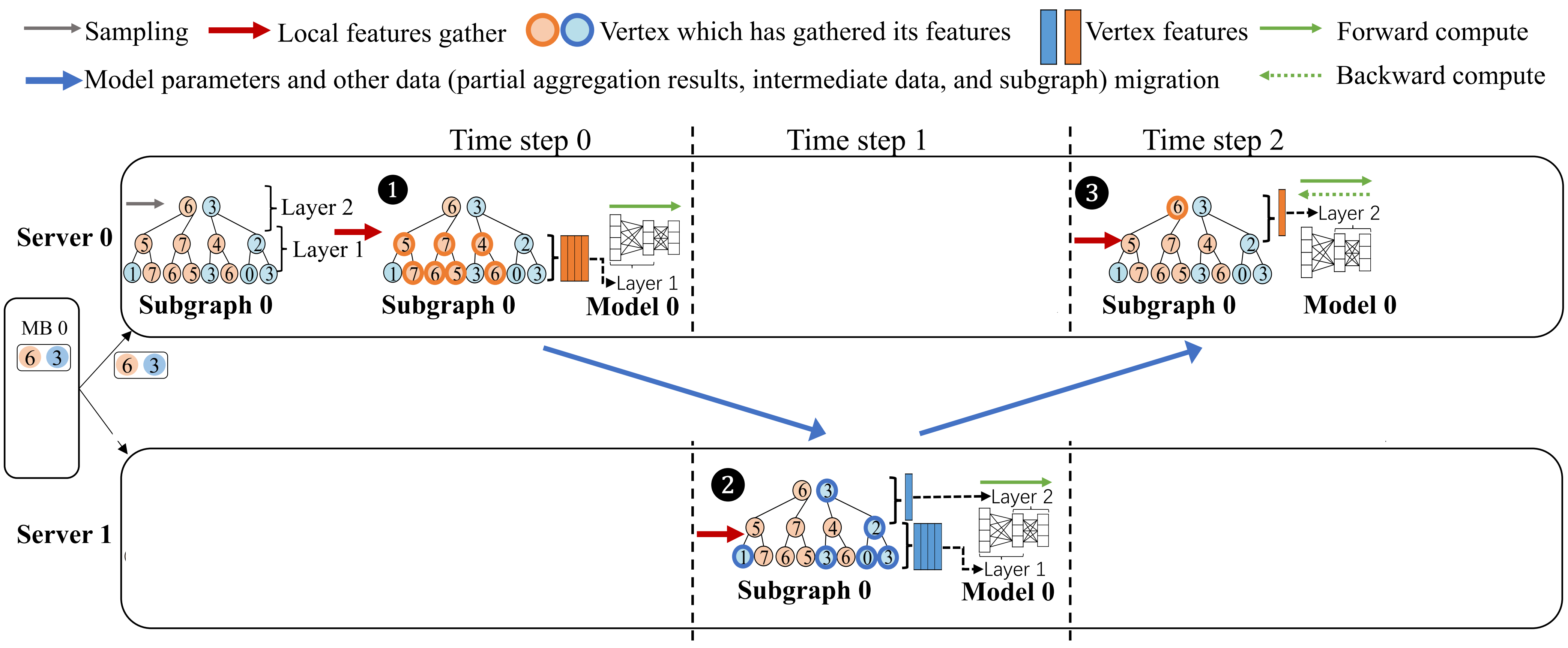
Naïve Model Migration



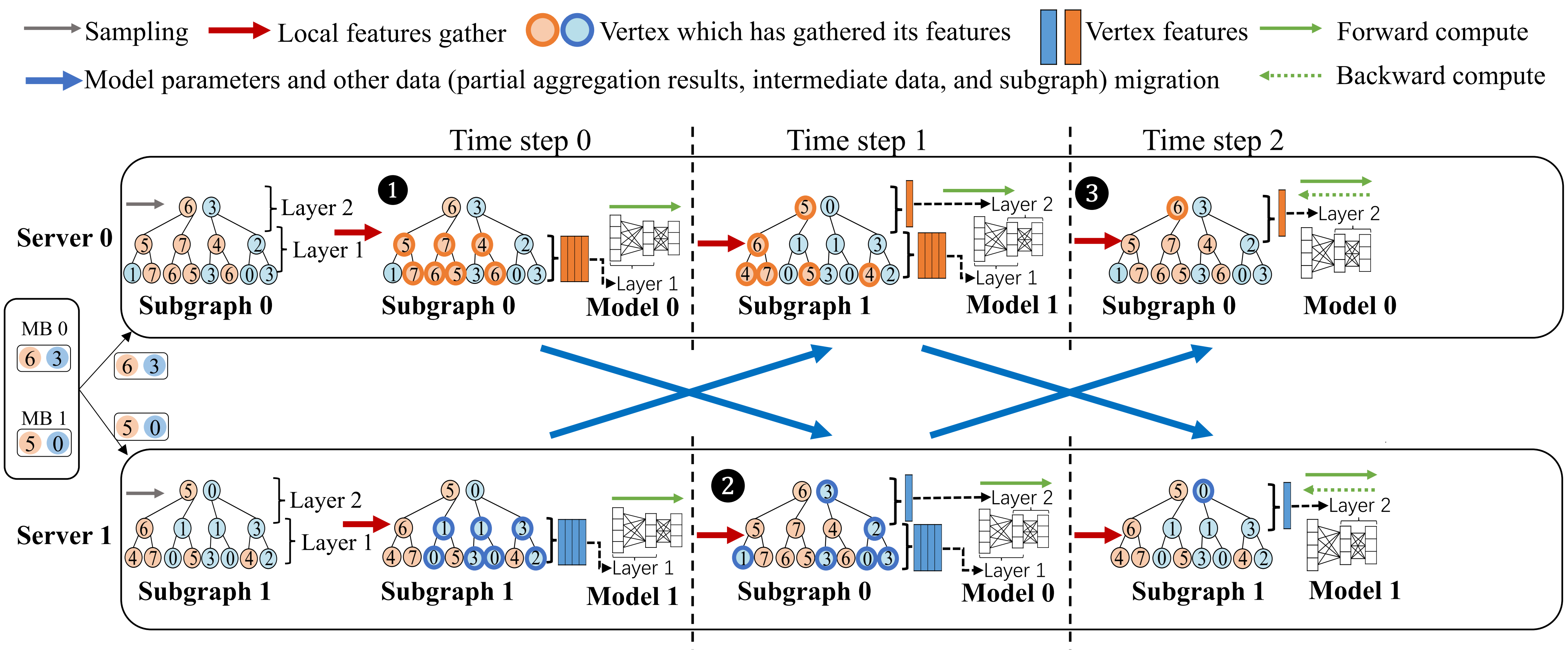
Naïve Model Migration



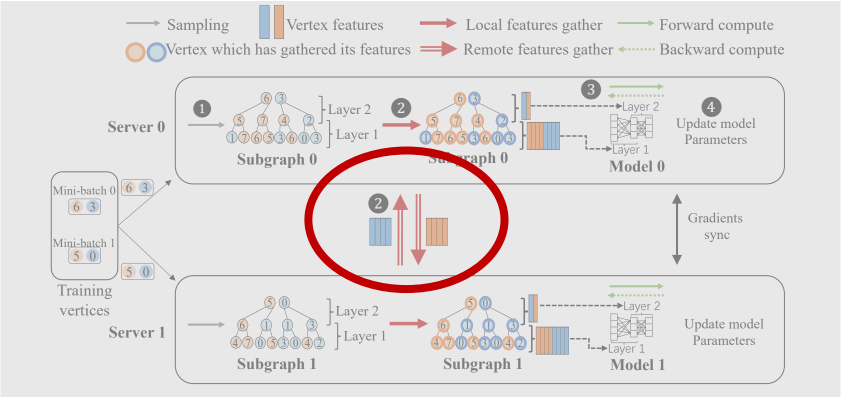
Naïve Model Migration



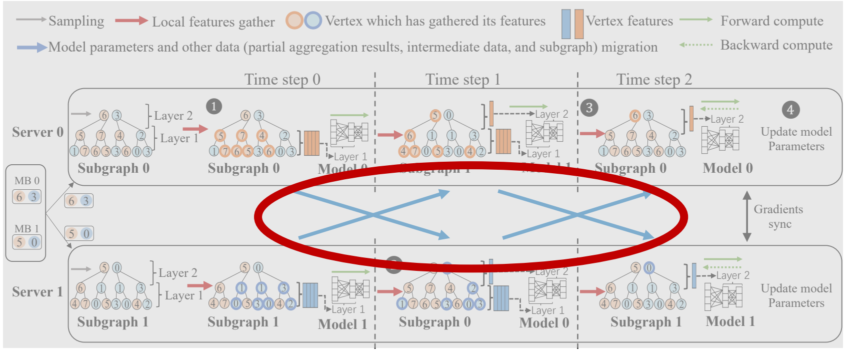
Naïve Model Migration



Naïve Model Migration



(a) Existing model-centric GNN training

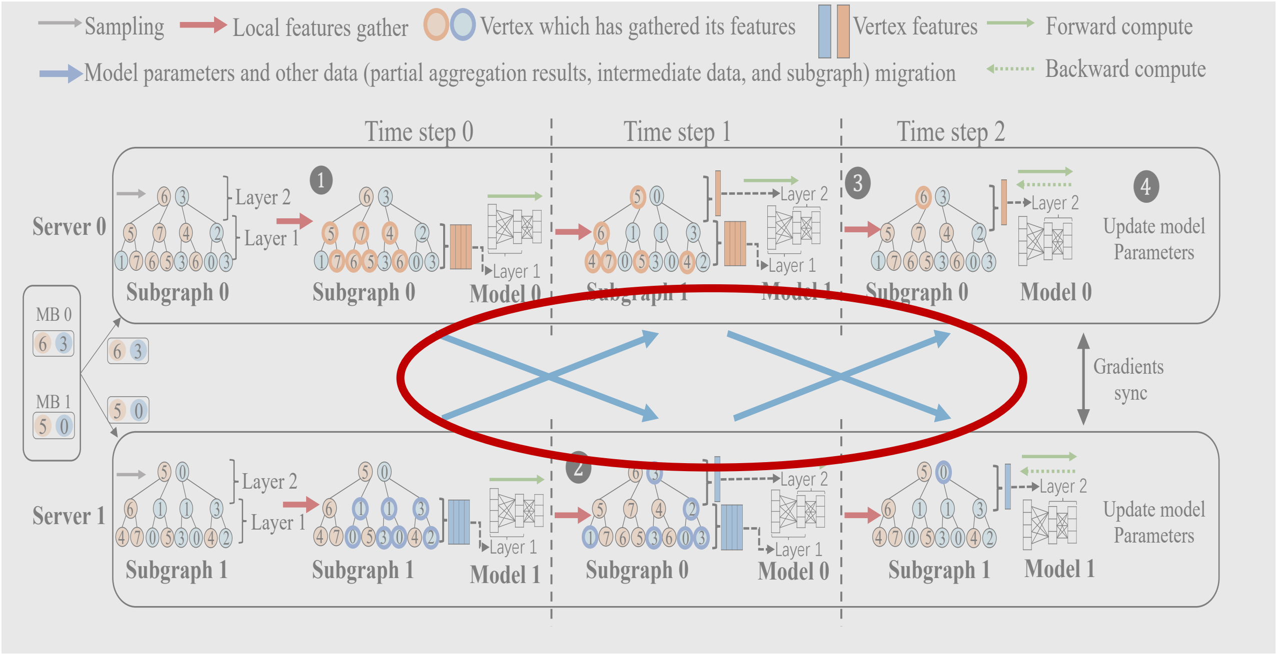


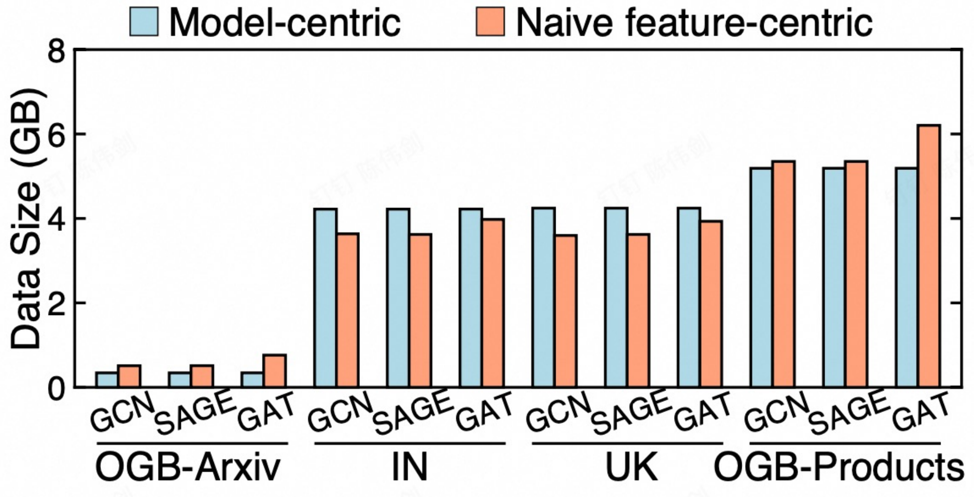
(b) Our preliminary solution: Naïve model migration

✓ **Compared to existing model-centric GNN training**

Totally eliminate cross-machine features transmission by model migration.

Challenges in our naïve solution





(y-values are log2 scaled)

Sometimes naïve method is advantageous, but may incur up to 2.6× model-centric data transmission.

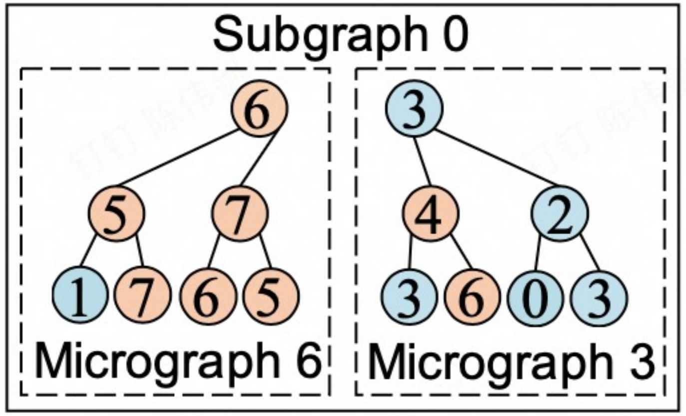
Avoid feature transfer, but incur other data transmission.

- partial aggregation results
- intermediate data for backward

Locality of Micrograph

➢ Micrograph

Definition. A micrograph G' is a computation graph derived from **a single mini-batch vertex *v*** via k-hop sampling in the original graph G.



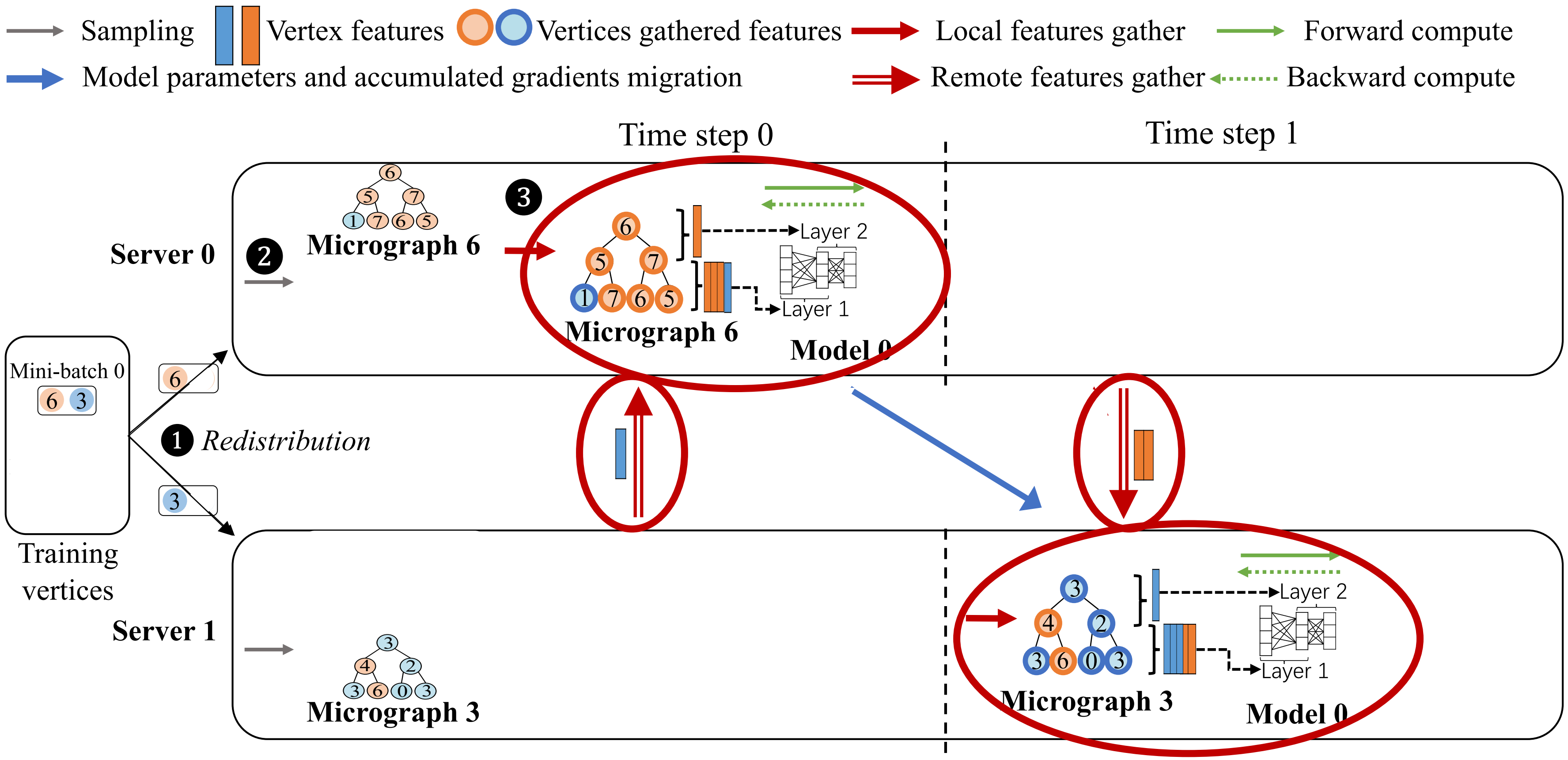
➢ Data locality in micrographs

Most fanout neighbors are located within the same partition (server) as the root vertex.

Sam-pling	#S	METIS (%)				Heuristic (%)				$R_{sub}(\%)$
		Arxiv		Products		Papers		IT		
		2L	10L	2L	10L	2L	10L	2L	10L	
Node-wise	2	75	73	95	88	93	61	66	64	50
	4	66	45	92	79	89	43	54	46	25
	8	59	27	88	68	84	35	48	36	12
	16	63	35	86	61	84	30	46	32	6
Layer-wise	2	79	54	55	52	85	58	80	53	50
	4	70	30	34	28	77	31	67	30	25
	8	65	18	25	14	56	24	63	18	12
	16	61	12	21	9	57	12	61	12	6

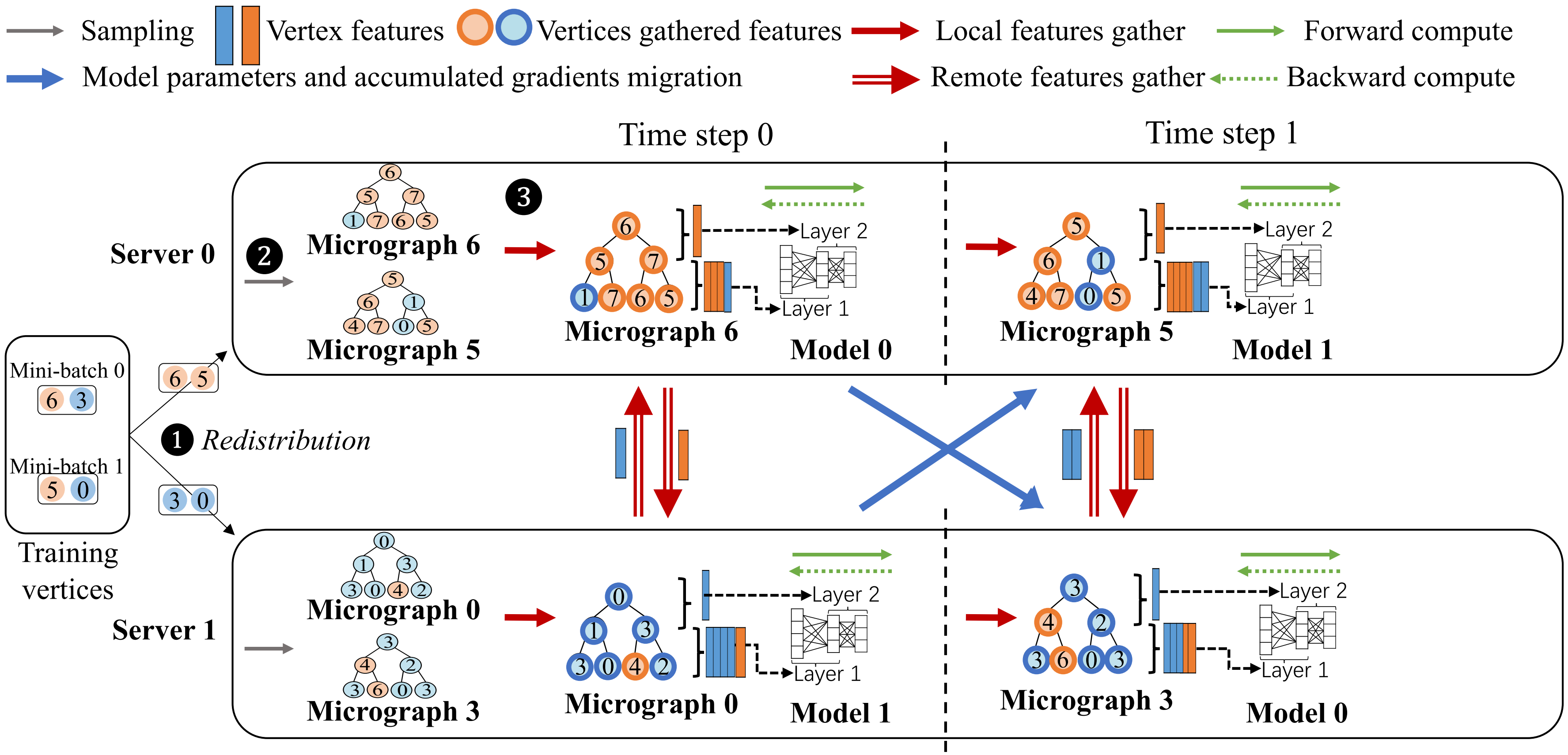
- #S: the number of distribute servers
- xL: the number of sampling layers is x
- R_{sub}: the locality of subgraph

Micrograph-Based GNN Training



Model migration + Locality of micrograph

Micrograph-Based GNN Training



✓ **Model accuracy fidelity.**

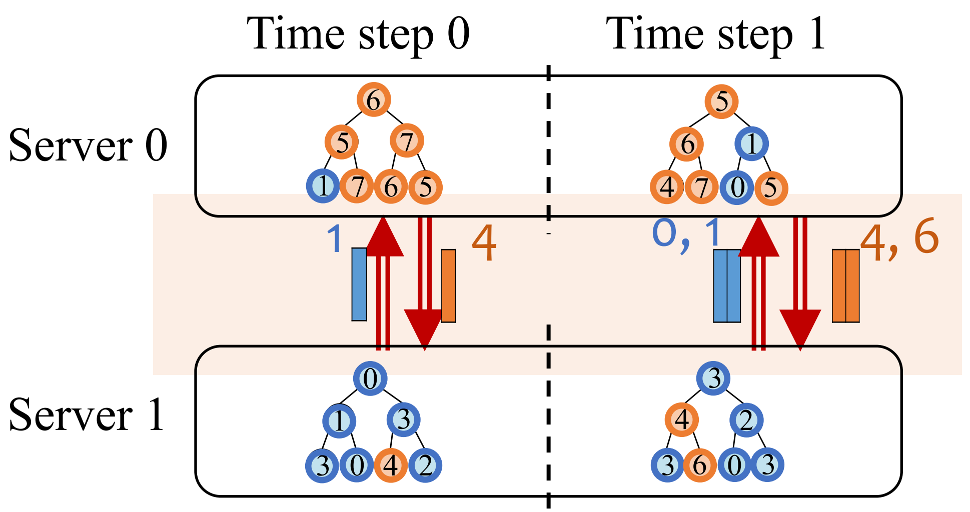
It does not compromise model accuracy as keep the randomly assigned training data unchanged.

Further Optimization 1

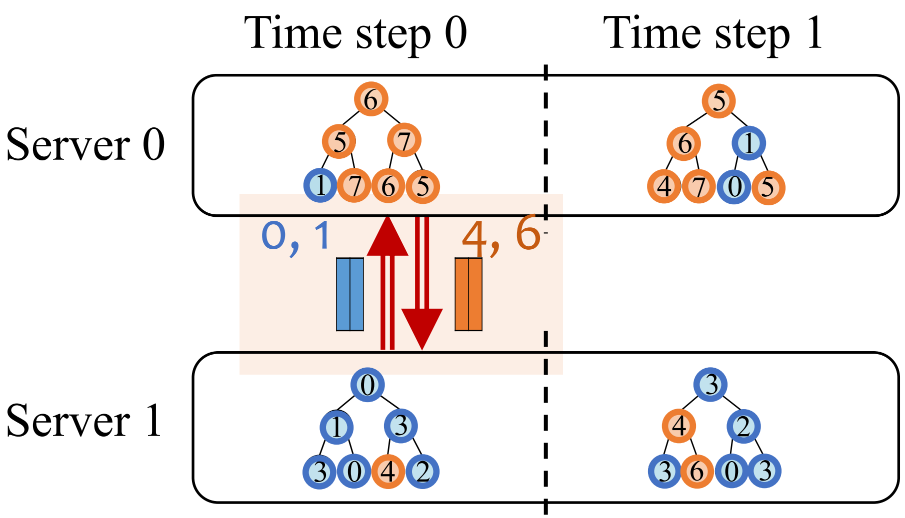
➢ Vertex Feature Pre-Gathering

- **Problem**
  Redundant vertex transmission across time steps.

- **Solution**
  Pre-gathering vertex features in the subsequent few time steps.



(a) without Pre-Gathering



(b) with Pre-Gathering

Further Optimization 2

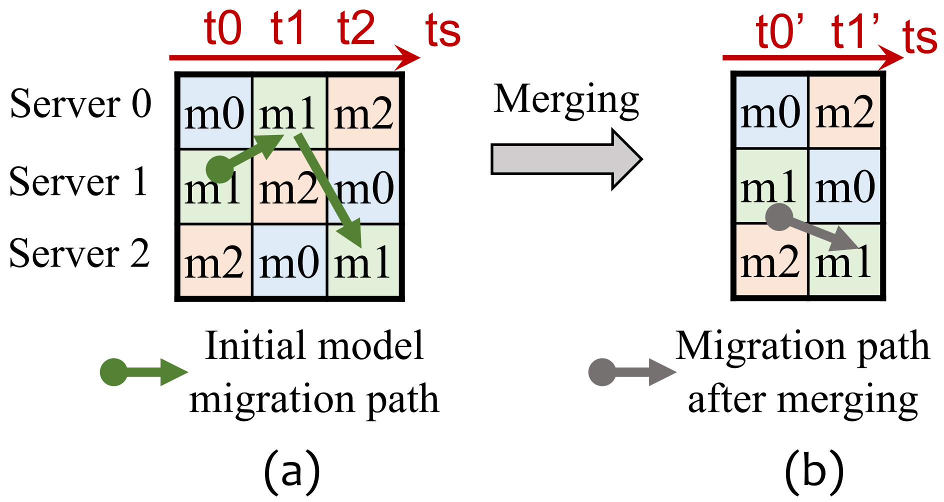
➢ Micrograph Merging in GNN Training

- **Problem**
  Time overhead increases with more distributed servers (N servers means N time steps in one iteration).

- **Solution**
  Merge micrographs, thus reducing the number of time steps.

  - **Which micrograph should be merged?**
  - **How many micrograph should be merged?**

More details: please checkout our paper.



One iteration training before (a) and after (b) micrograph merging.

Experimental Setup

➢ System configuration

- 4 GPU Servers, each equipped with:

CPU	2×Intel(R) Xeon(R) Gold 5318Y CPUs (48 cores)
Memory	128 GB CPU memory
GPU	One NVIDIA A100 GPU (40 GB)
Network	10Gb/s Ethernet

➢ Models and datasets

- Shallow models: GCN, GraphSAGE, GAT with hidden sizes of 16 and 128
- Deep models: DeepGCN (7), GNN-FiLM (10)

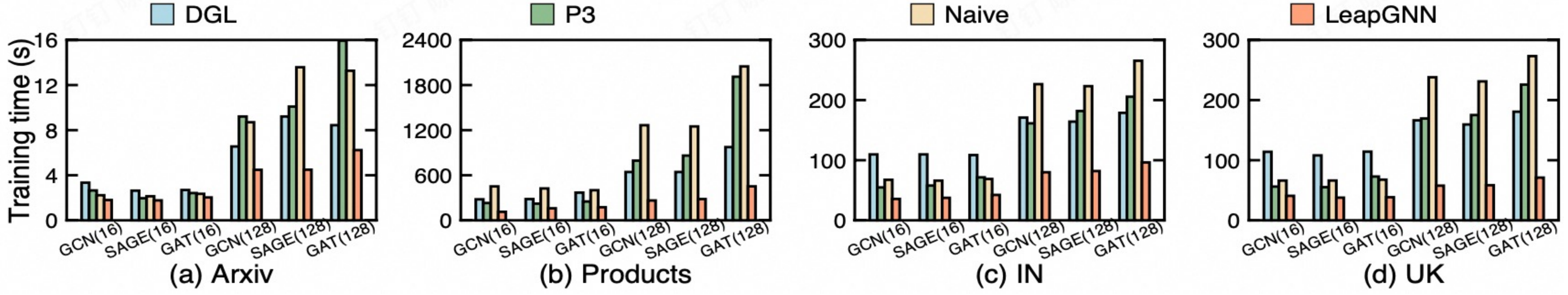
Dataset	**#Vertex**	**#Edge**	**Dim.**	**Vol**$_G$	**Vol**$_F$
Arxiv	169K	1.17M	128	3.3 MB	85 MB
Products	2.45M	61.9M	100	464 MB	980 MB
UK	1M	41.2M	600	12 MB	2.3 GB
IN	1.38M	16.9M	600	8.2 MB	3.2 GB
IT	41.3M	1.15B	600	363MB	92.3 GB

➢ Compared systems

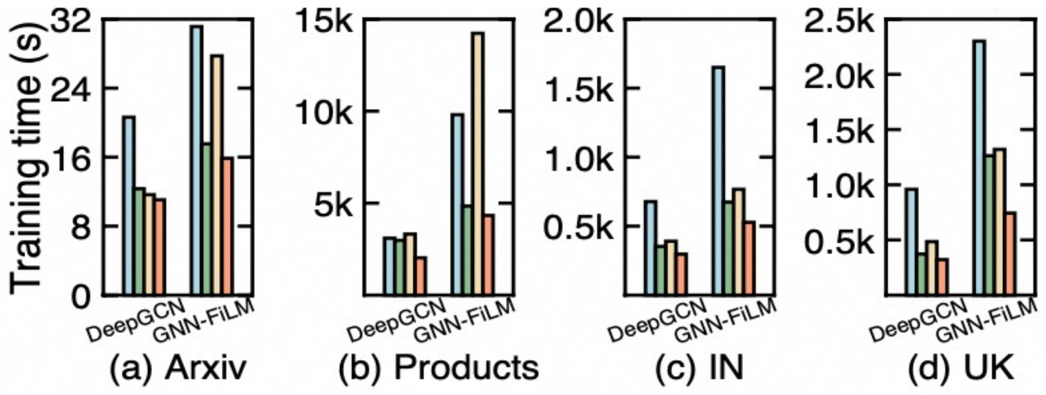
DGL [IA3'20]	Fetches all the required features locally or remotely
P3 [OSDI'21]	Combined model-parallel and data-parallel
NeutronStar [SIGMOD'22]	Balance the redundant computation and communication time
Naïve	Naïve model migration methods

Overall Training Time

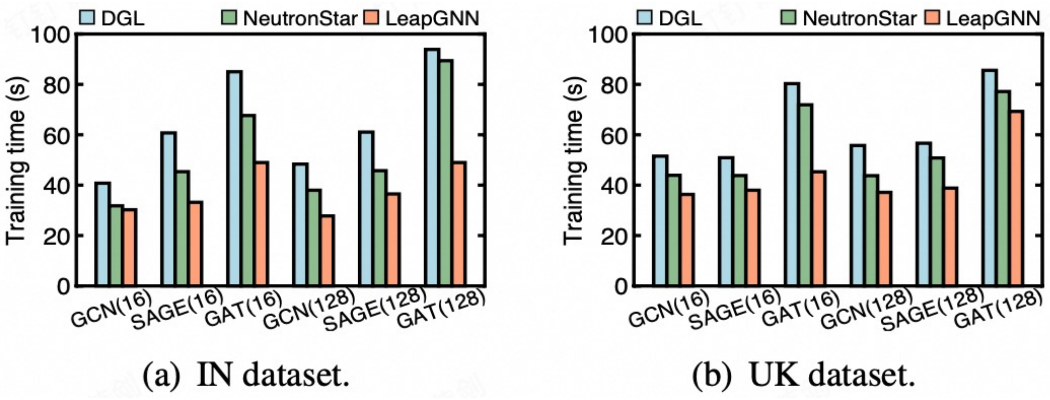
Mini-batch training with shallow models



Mini-batch training with deep models

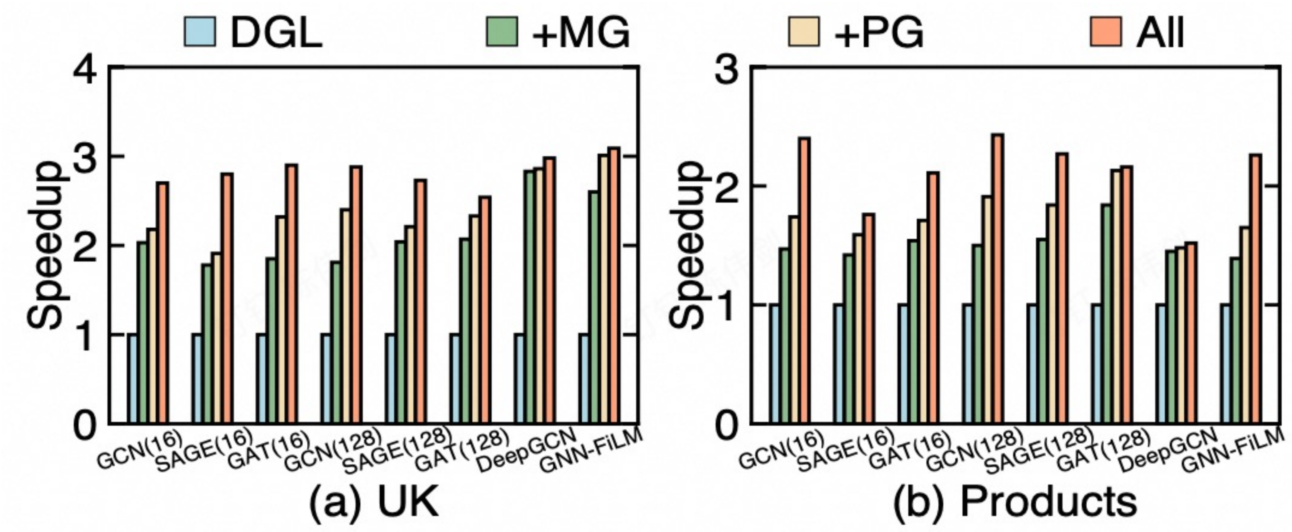


Full-batch training



LeapGNN achieves 1.3-3.1x over DGL, 1.2-4.2x over P^3, 1.1-4.8x over Naïve, and 1.1-1.8x over NeutronStar.

Impact of Individual Techniques

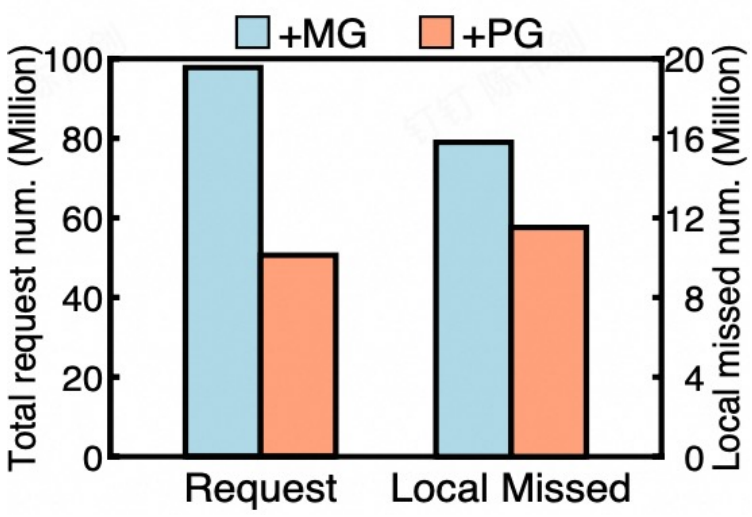


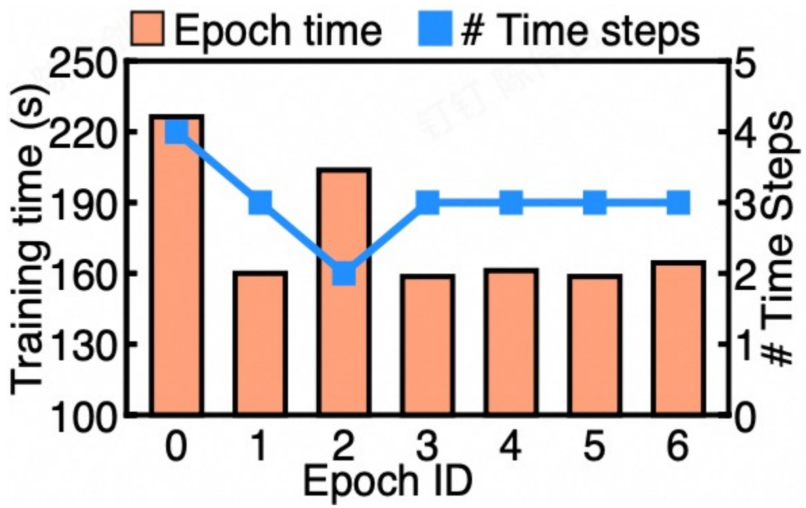
- *+MG* denotes the version where micrograph-based GNN training is turned on.
- *+PG* denotes the version where pre-gathering is added based on +MG.
- *All* means the micrograph merging is also enabled.

Each technique enhances performance.
The most impactful technique varies across scenarios.

Impact of Individual Techniques

		Miss Rate
Arxiv	DGL	74%
	+MG	43%
Products	DGL	77%
	+MG	22%
UK	DGL	78%
	+MG	19%
IN	DGL	77%
	+MG	9.2%



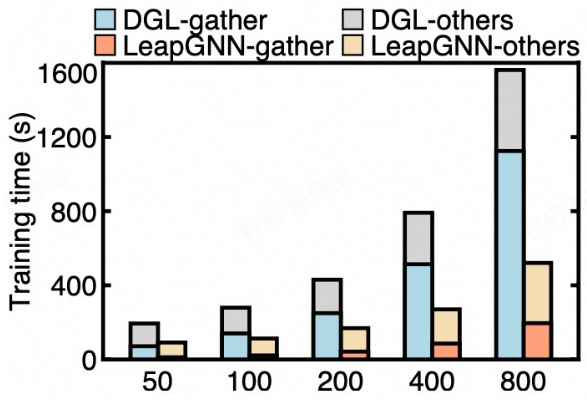


+MG decreases the cache miss rate across four datasets.

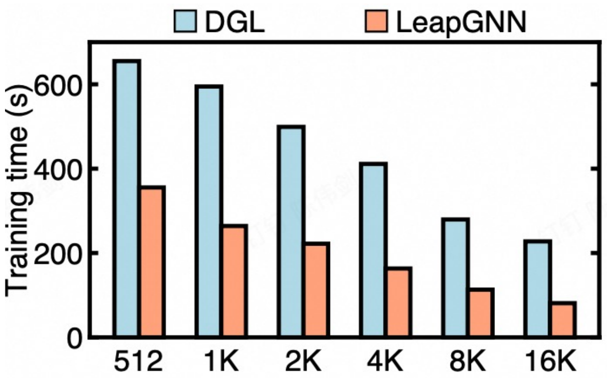
+PG decreases the number of request vertex and local missed vertex.

LeapGNN automatically adjusts the number of time steps to 3 which shows the best performance.

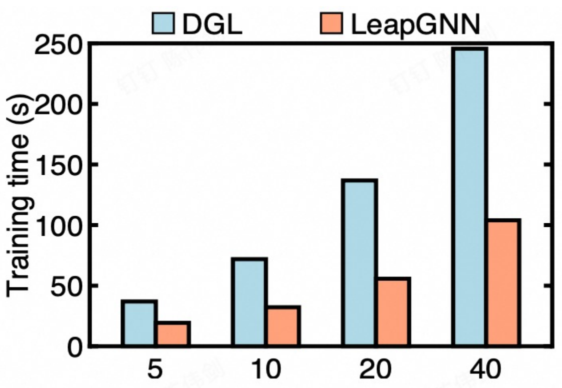
Sensitivity Analysis



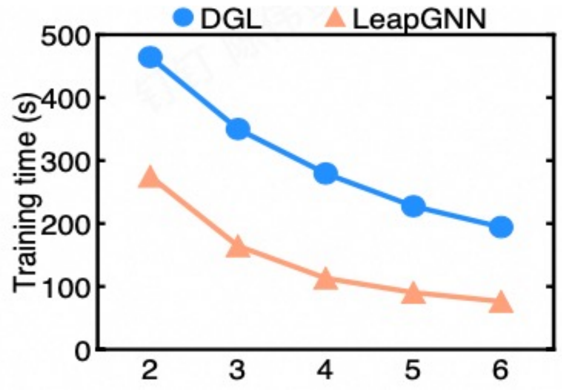
(a) Feature dimension



(b) Batch size



(c) Fanout size



(d) Number of servers

LeapGNN consistently outperforms the comparisons under various conditions.

More evaluations: checkout our paper.

Summary & Conclusion

- **Problem**
  - Feature transmission becomes the bottleneck in distributed GNN training.
- **Key idea**
  - Introduce feature-centric model migration to reduce features transmission.
- **Challenges**
  - Naïve model migration avoids feature transfer, but incurs intermediate data transmission.
- **Techniques in LeapGNN**
  - Micrograph-Based GNN Training
  - Vertex Feature Pre-Gathering
  - Micrograph Merging in GNN Training
- **Results**
  - LeapGNN achieves up to 4.2× speedup compared to the state-of-the-art counterpart P^3.

Thanks & QA

LeapGNN: Accelerating Distributed GNN Training Leveraging Feature-Centric Model Migration

浙江大学 Zhejiang University

Washington State University

Source Code: https://github.com/ISCS-ZJU/LeapGNN-AE

Contact Email: weijianchen@zju.edu.cn

ISCS Lab: https://shuibing9420.github.io

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED