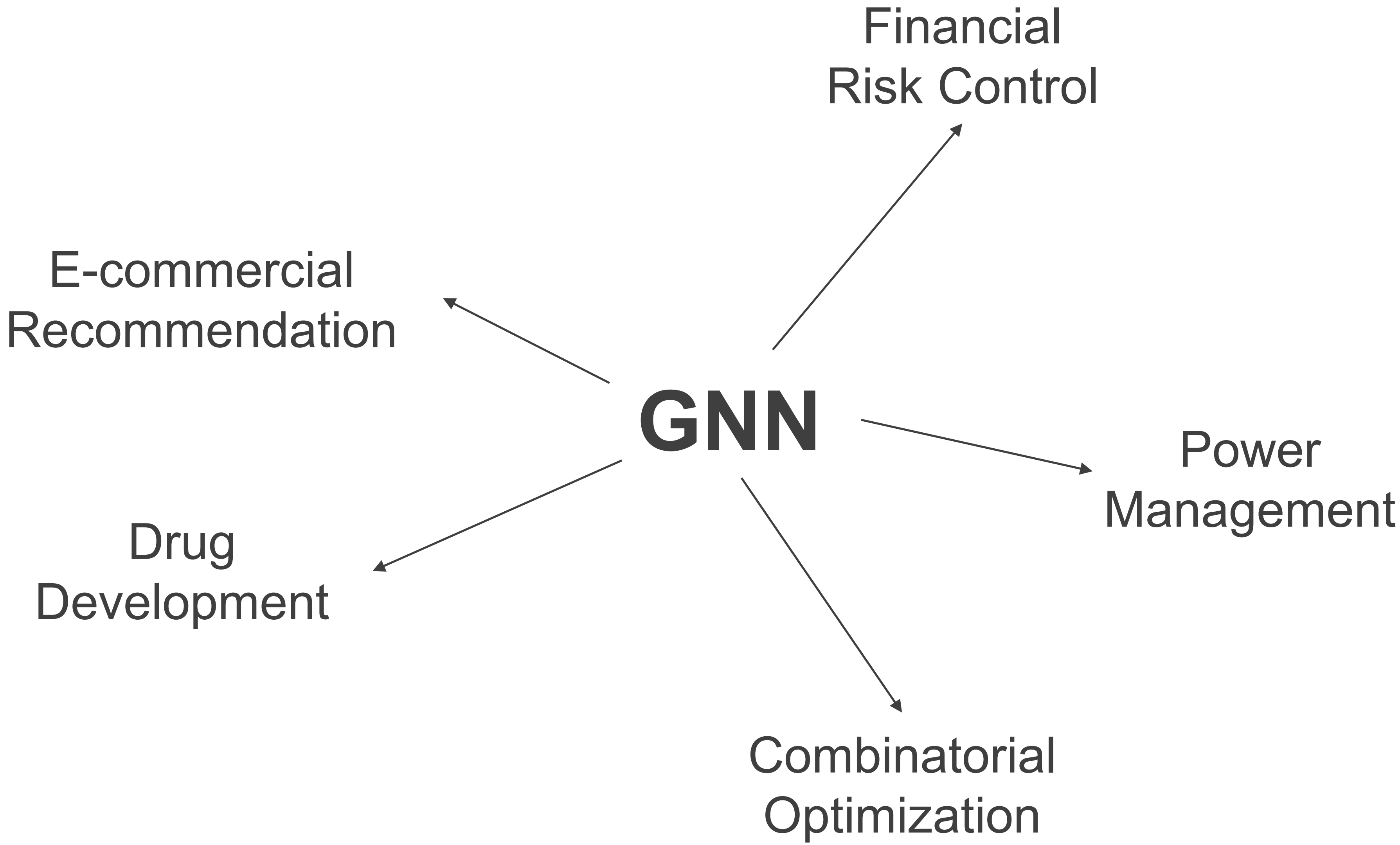
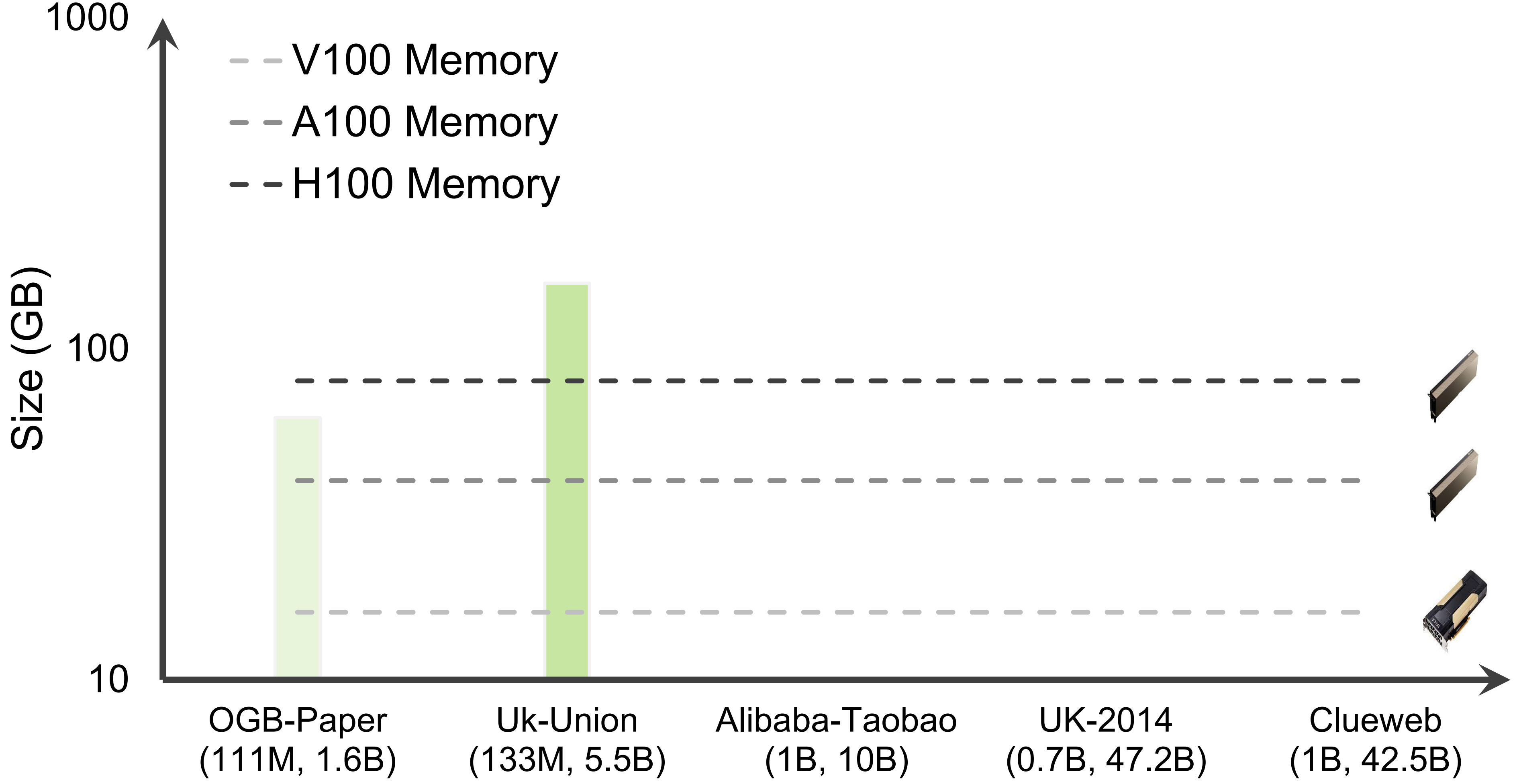
Legion:
Automatically Pushing the Envelope of Multi-GPU System for Billion-Scale GNN Training

Jie Sun, Li Su, Zuocheng Shi, Wenting Shen, Zeke Wang
Lei Wang, Jie Zhang, Yong Li, Wenyuan Yu, Jingren Zhou, Fei Wu

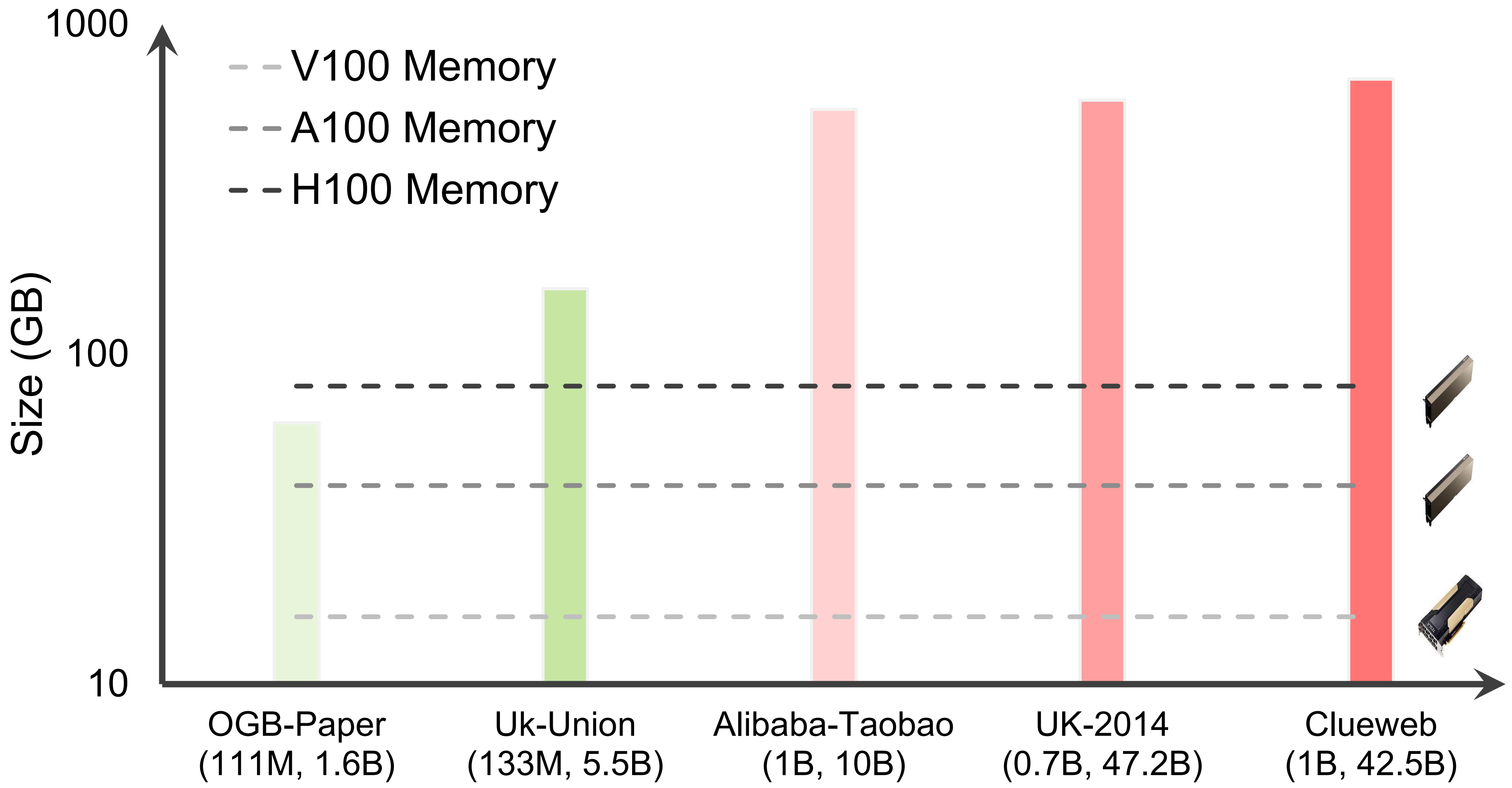
Graph Neural Network (GNN)



Billion-scale Graphs



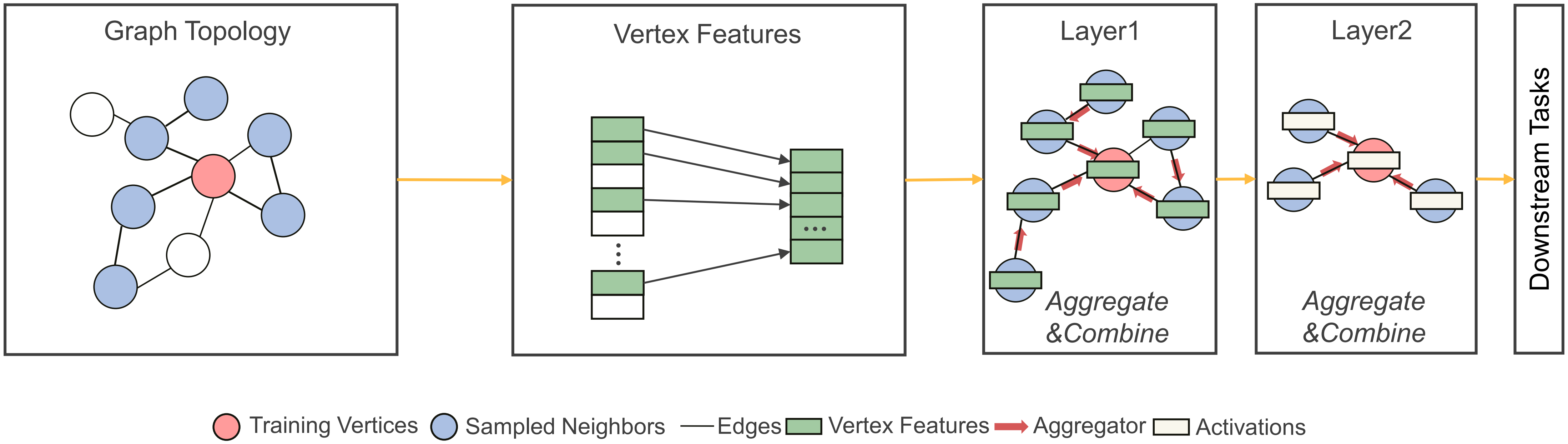
Challenge from Industry



Sampling-based GNN

- **Three Key Stages:**

1. Graph Sampling
2. Feature Extraction
3. Model Training

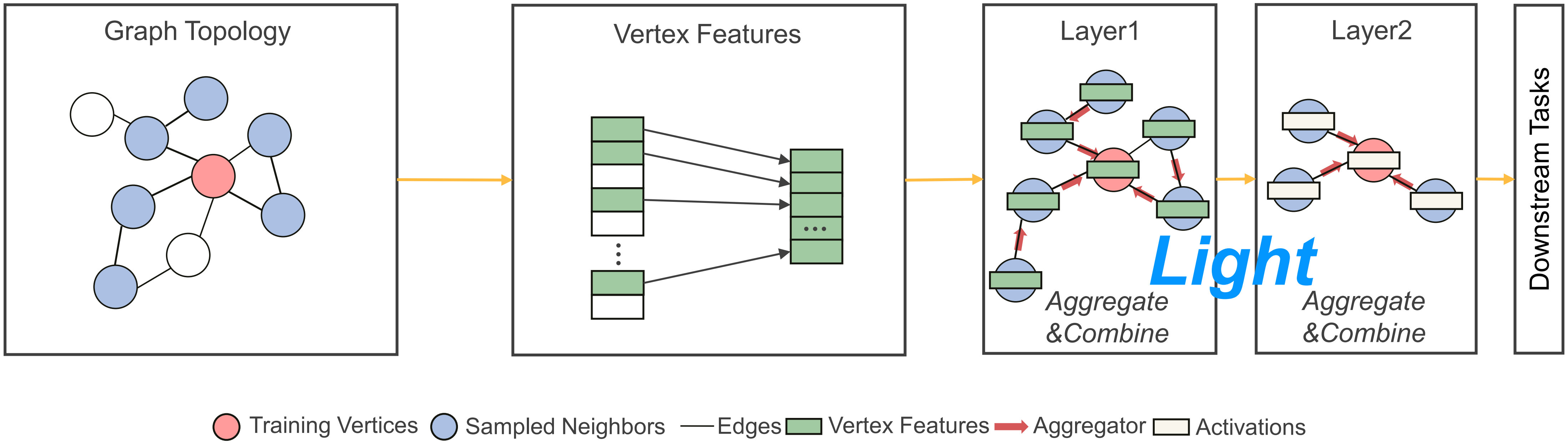


GraphSAGE [NeurIPS 2017]

Sampling-based GNN

- **Three Key Stages:**

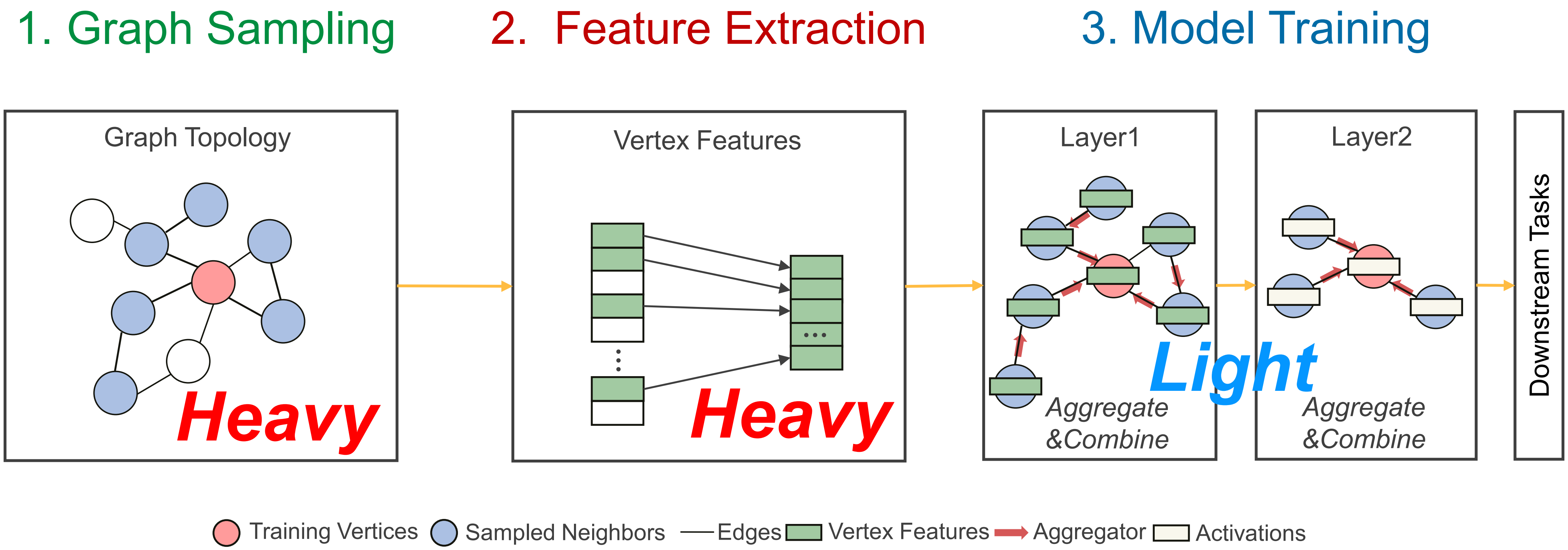
1. Graph Sampling
2. Feature Extraction
3. Model Training



GraphSAGE [NeurIPS 2017]

Sampling-based GNN

- **Three Key Stages:**



GraphSAGE [NeurIPS 2017]

Traditional GNN Systems

DGL DEEP GRAPH LIBRARY

PyG

graph-learn

Traditional GNN Systems

- **Properties:**
  - GPU model training
  - Storing graph in CPU memory
  - CPU graph sampling
  - CPU feature extraction

Traditional GNN Systems

- **Properties:**
  - ➢ GPU model training
  - ➢ Storing graph in CPU memory
  - ➢ CPU graph sampling
  - ➢ CPU feature extraction
- **Issues:**
  - PCIe communication becomes major bottleneck!
  - CPU sampling can not catch up with GPU training!

Traditional GNN Systems

- **Properties:**
  - GPU model training
  - Storing graph in CPU memory
  - CPU graph sampling
  - CPU feature extraction
- **Issues:**
  - PCIe communication becomes major bottleneck!
  - CPU sampling can not catch up with GPU training!

Cache-based GNN Systems

- **Existing Works:**
  - PaGraph [SoCC 2020]
  - Quiver [2022]
  - GNNLab [Eurosys 2022]

- **Optimizations:**
  - GPU Feature Cache

Cache-based GNN Systems

- **Existing Works:**
  - PaGraph [SoCC 2020]
  - Quiver [2022]
  - GNNLab [Eurosys 2022]

- **Optimizations:**
  - GPU Feature Cache
  - GPU Sampling

Cache-based GNN Systems

- **Existing Works:**
  - PaGraph [SoCC 2020]
  - Quiver [2022]
  - GNNLab [Eurosys 2022]

- **Optimizations:**
  - GPU Feature Cache
  - GPU Sampling

- **They are not optimized for billion-scale GNN training:**

- **Two Issues:**

  I_1: Poor Multi-GPU Cache Scalability

  I_2: Coarse-grained Topology Management

Legion

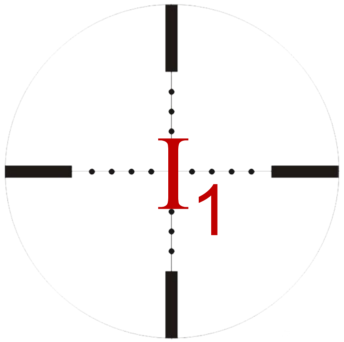
Goal:

- Fully explore the hardware capabilities of modern multi-GPU systems for training billion-scale graphs

Legion

Contributions:

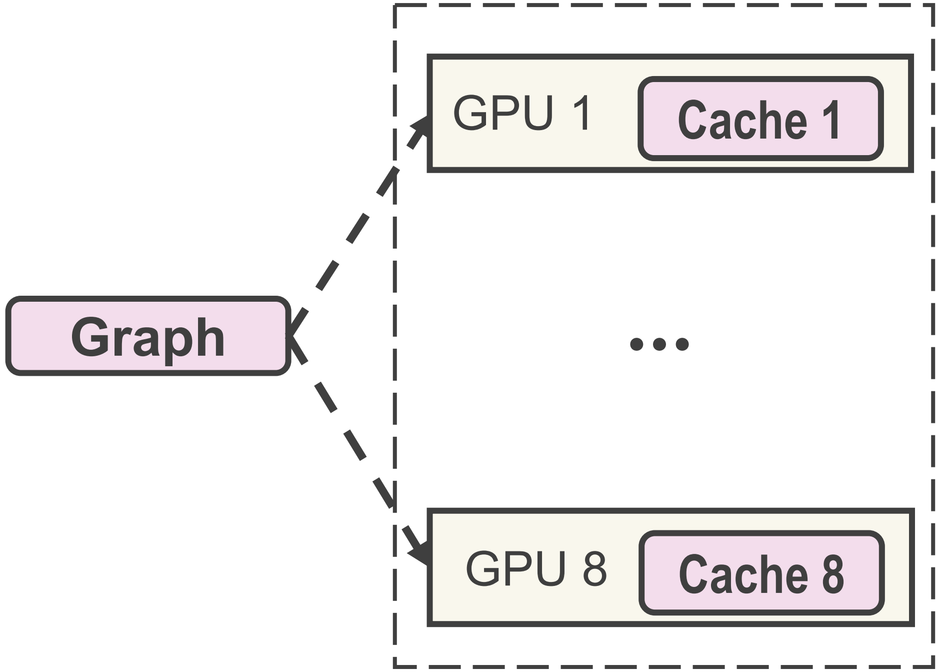
1. Hierarchical Graph Partitioning
2. Hotness-aware Unified Cache
3. Automatic Cache Management



I_1: Poor Multi-GPU Cache Scalability

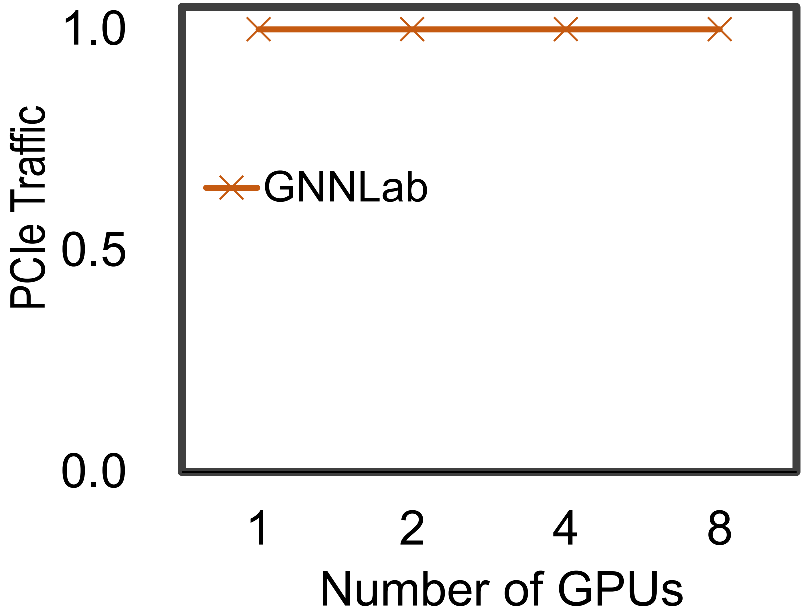
➢ GNNLab Design

No Partitioning   Replicate cache in all GPUs



➢ Cache Scalability Evaluation

Platform: 4 NVLink cliques, 2 GPUs per clique

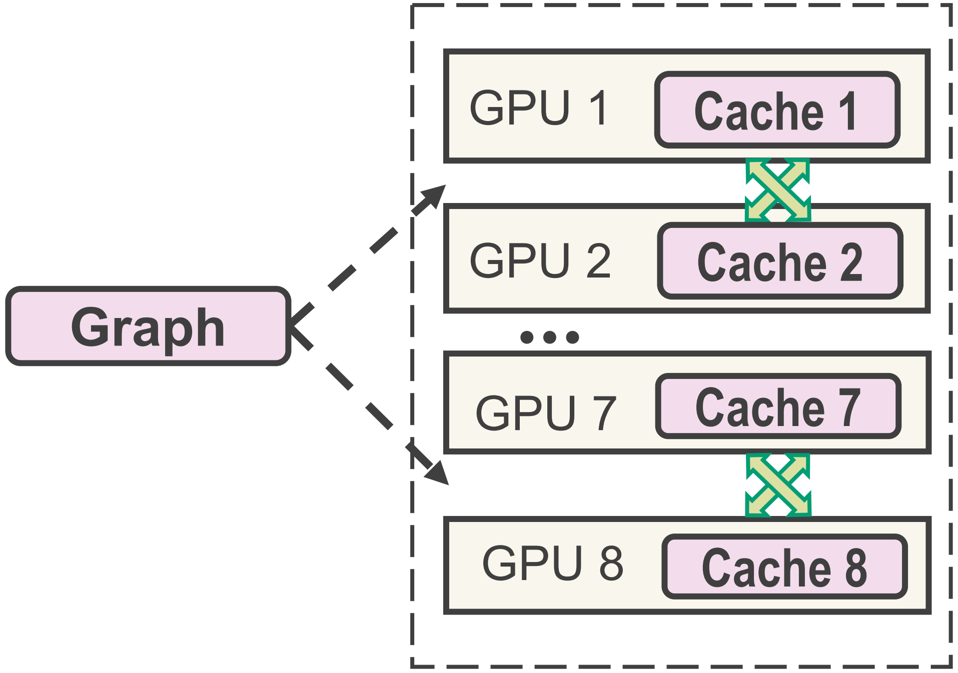


☹ PCIe traffic does not decrease with more GPUs

I_1: Poor Multi-GPU Cache Scalability

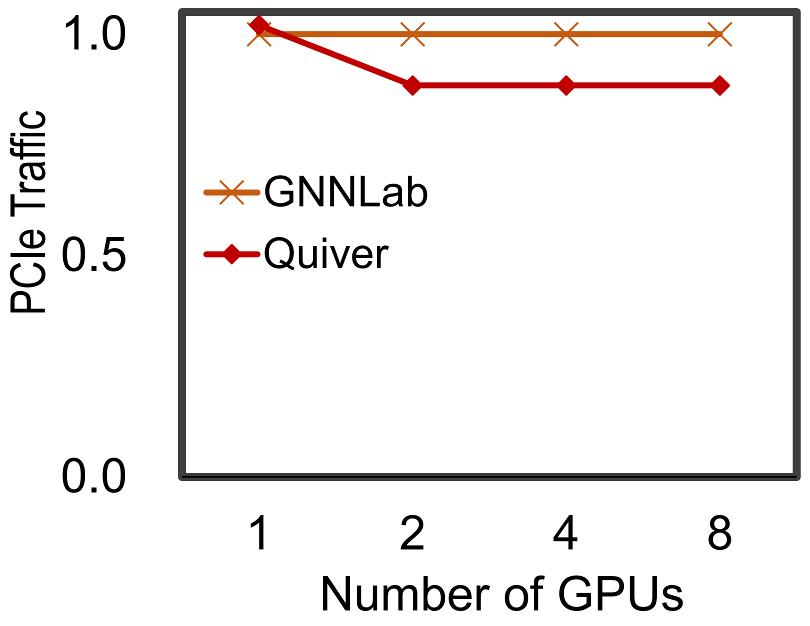
- **Quiver Design**

No Partitioning   Replicate cache in all cliques



- **Cache Scalability Evaluation**

Platform: 4 NVLink cliques, 2 GPUs per clique

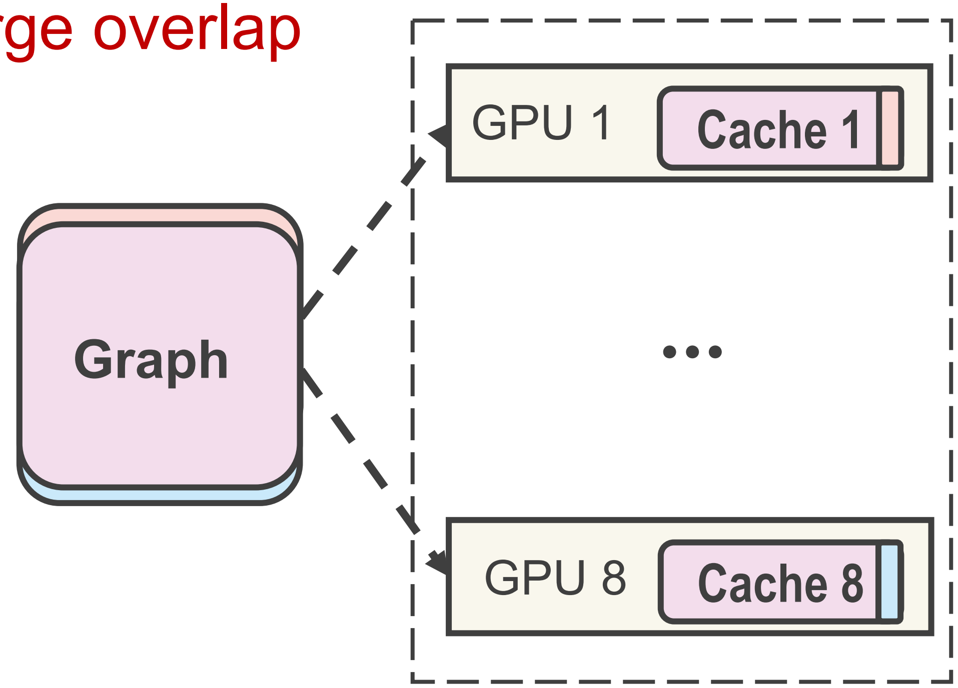


☹ PCIe traffic does not decrease with more NVLink cliques

I_1: Poor Multi-GPU Cache Scalability

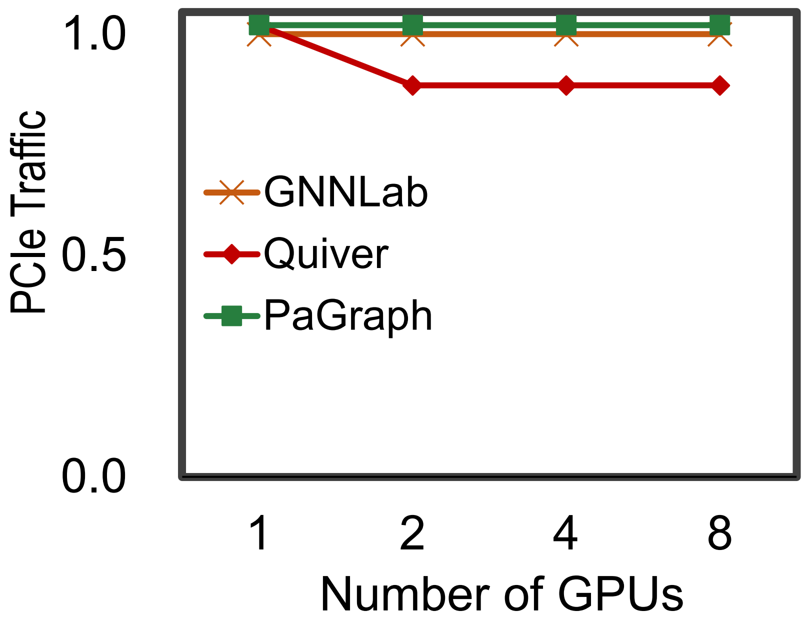
➢ PaGraph Design

Partitioning with large overlap



➢ Cache Scalability Evaluation

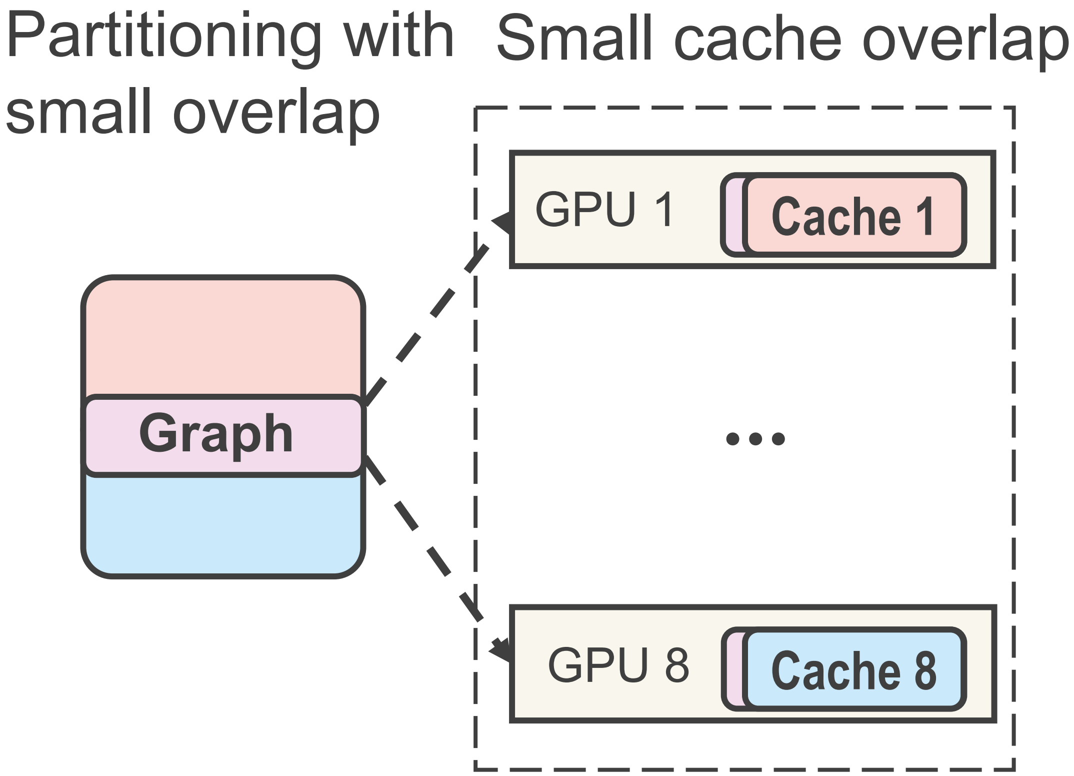
Platform: 4 NVLink cliques, 2 GPUs per clique



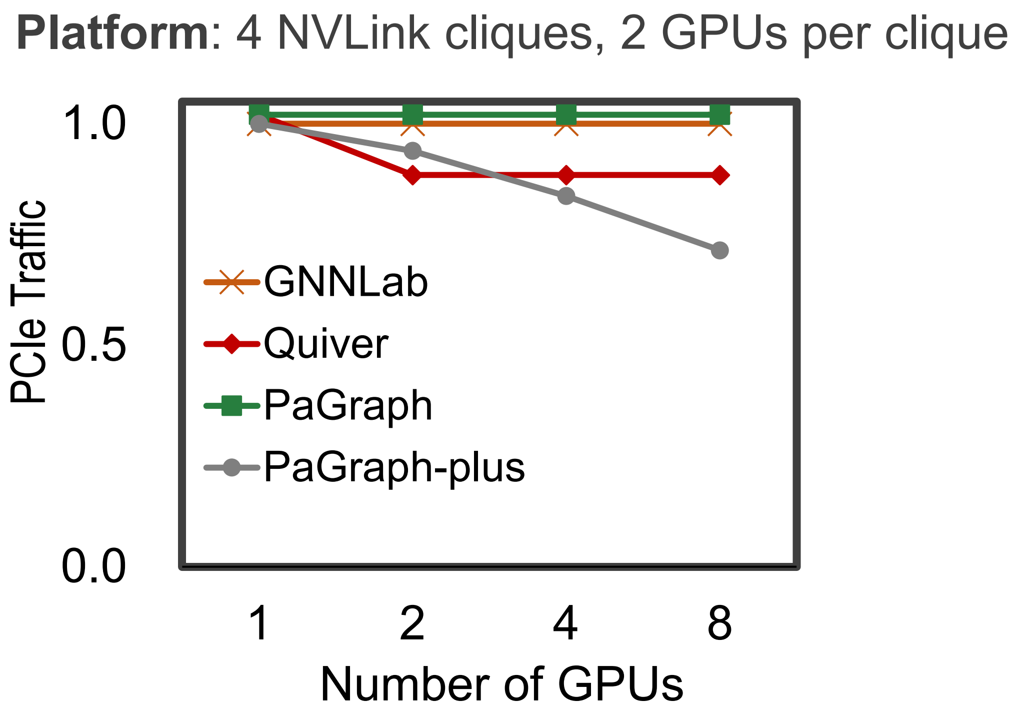
☹ PCIe traffic decreases very little with more GPUs

I_1: Poor Multi-GPU Cache Scalability

➢ PaGraph-plus Design



➢ Cache Scalability Evaluation



☹ PCIe traffic still decreases very little with more GPUs

☹ Unbalanced cache hit among GPUs

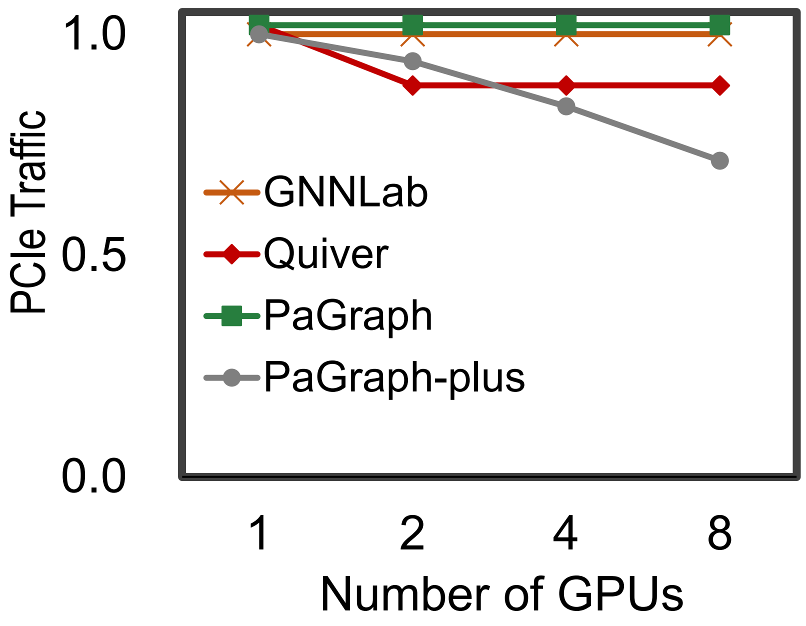
I_1: Poor Multi-GPU Cache Scalability

➢ ? Design

?

➢ Cache Scalability Evaluation

Platform: 4 NVLink cliques, 2 GPUs per clique



How to improve multi-GPU cache scalability?

Hierarchical Graph Partitioning

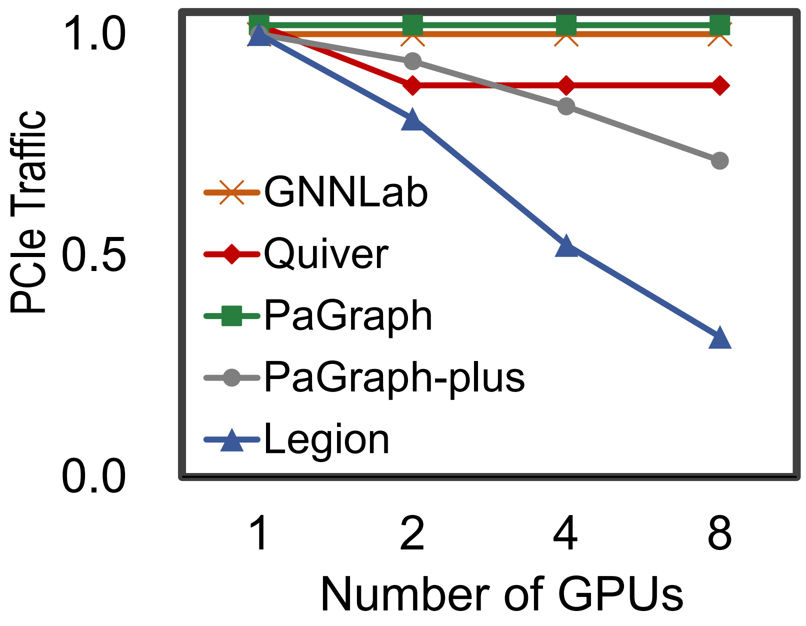
➢ Legion Design

Hierarchical graph partitioning

NVLink-enhanced multi-GPU cache

➢ Cache Scalability Evaluation

Platform: 4 NVLink cliques, 2 GPUs per clique

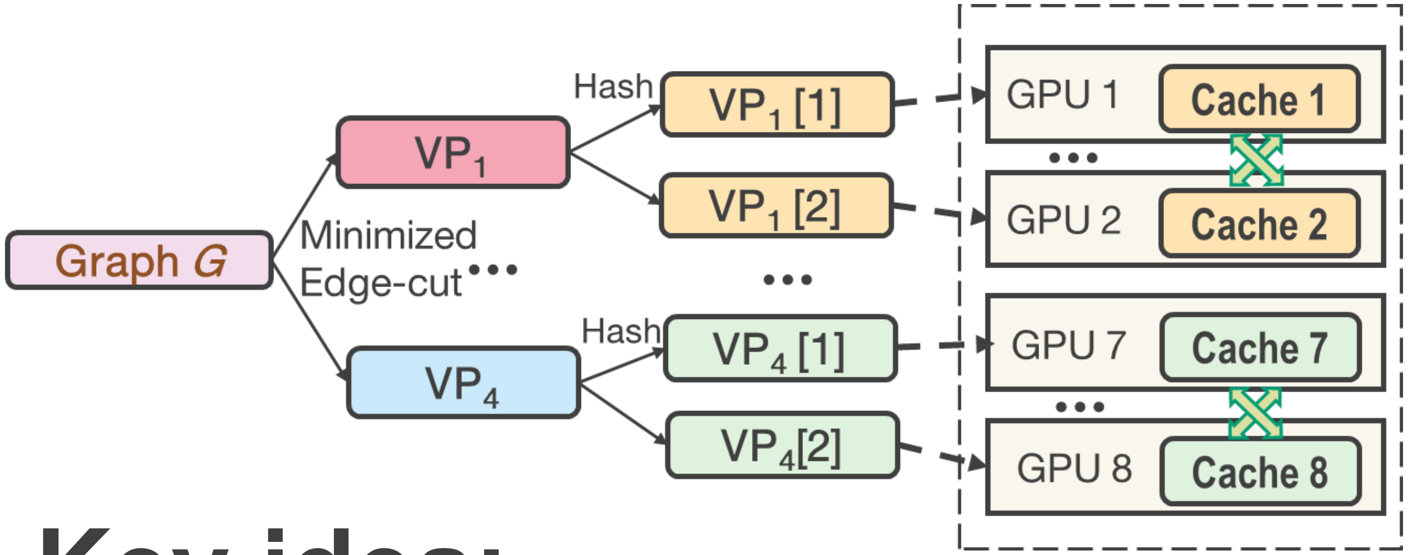


Key idea:

➢ Co-design hierarchical graph partitioning with NVLink-enhanced multi-GPU cache

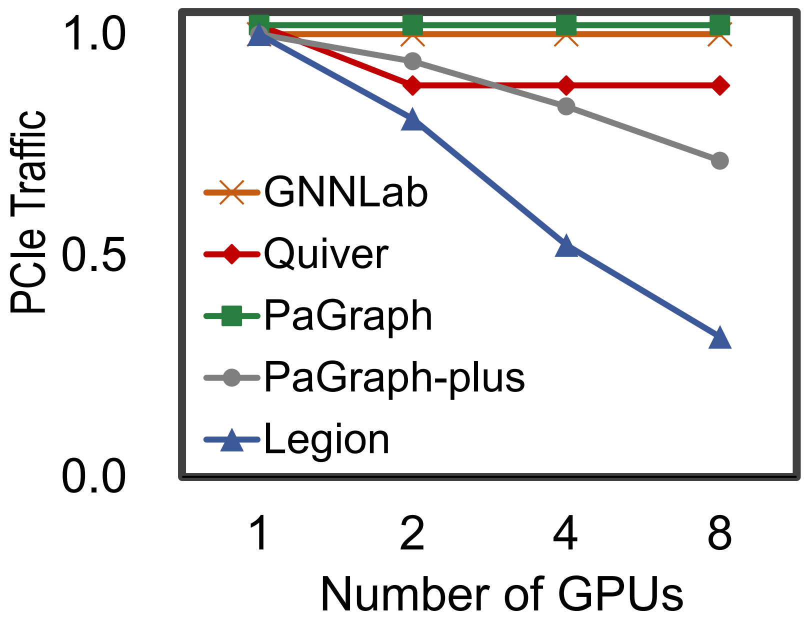
Hierarchical Graph Partitioning

➢ Legion Design



➢ Cache Scalability Evaluation

Platform: 4 NVLink cliques, 2 GPUs per clique



Key idea:

➢ Co-design hierarchical graph partitioning with NVLink-enhanced multi-GPU cache

Hierarchical Graph Partitioning

- **Goal**: Improve multi-GPU cache scalability

Hierarchical Graph Partitioning

- **Goal**: Improve multi-GPU cache scalability

- **Principles:**
  - ➢ Between NVLink cliques:
    - ➢ Maintain different caches for different partitions
      => Minimize cache replication

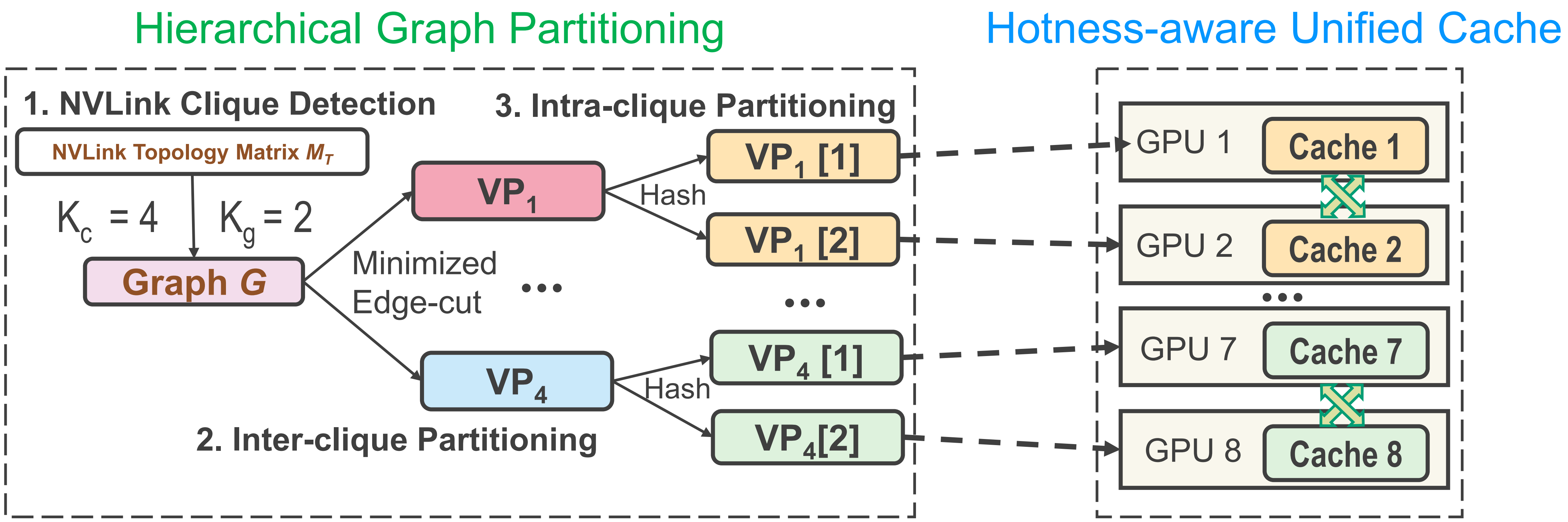
Hierarchical Graph Partitioning

- **Goal**: Improve multi-GPU cache scalability

- **Principles:**
  - ➢ Between NVLink cliques:
    - ➢ Maintain different caches for different partitions
      => Minimize cache replication
  - ➢ Within NVLink cliques:
    - ➢ Split cache exclusively and uniformly
      => Eliminate cache replication & improve load balance

Hierarchical Graph Partitioning

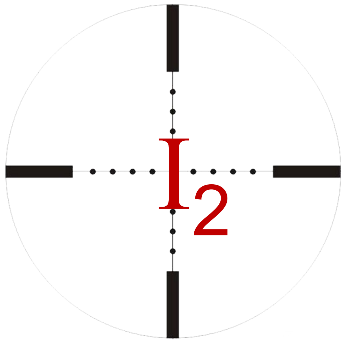
- **Goal**: Improve multi-GPU cache scalability



Legion

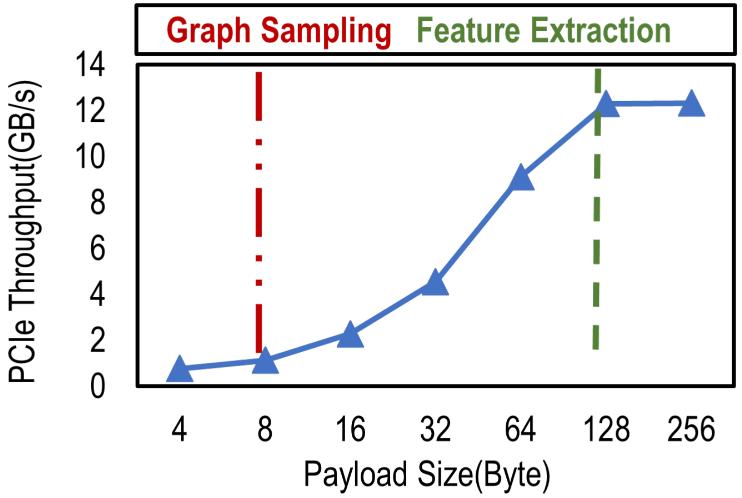
Contributions:

1. Hierarchical Graph Partitioning
2. Hotness-aware Unified Cache
3. Automatic Cache Management



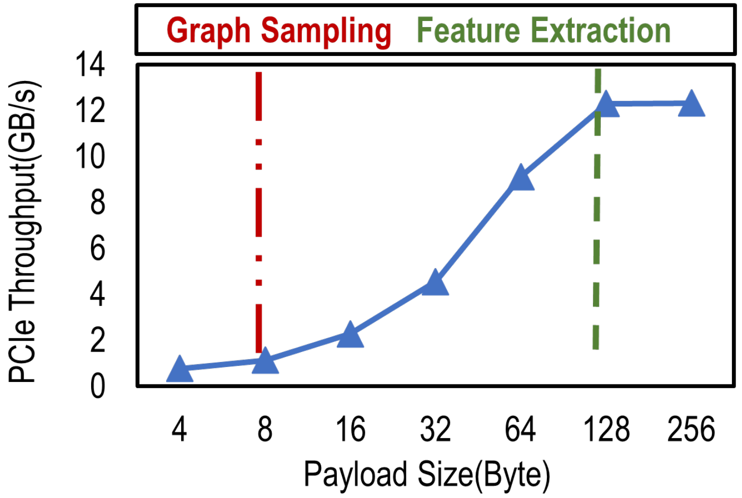
I$_2$: Coarse-grained Topology Management

- **DGL** [ICLR 2019]
- **Quiver** [2022]
  - Design:
    - All topology in CPU memory
  - Issue:
    - Low PCIe utilization



I_2: Coarse-grained Topology Management

- **DGL** [ICLR 2019]
- **Quiver** [2022]
  - Design:
    - All topology in CPU memory
  - Issue:
    - Low PCIe utilization



- **GNNLab** [Eurosys 2022]
  - Design:
    - All topology in GPU memory
  - Issue:
    - Limited graph topology size

Examples	16 GB V100
UK-Union	OOM
Alibaba-Taobao	OOM
Clueweb	OOM

How to Manage Graph Topology?

- All topology in CPU memory

☹ Low PCIe utilization

→ ? ←

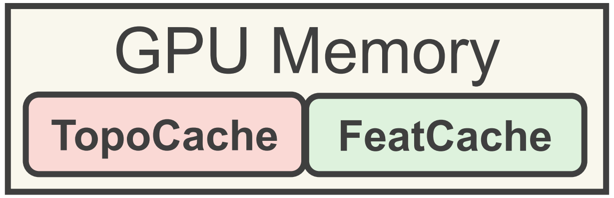
- All topology in GPU memory

☹ Limited graph topology size

=> Hotness-aware Unified Cache

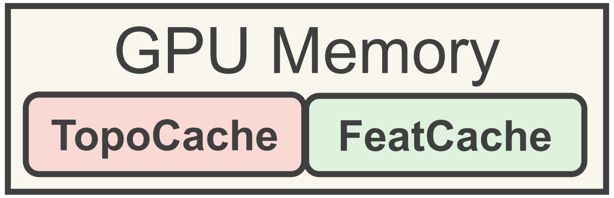
Hotness-aware Unified Cache

- **Goal:**
  - Minimize PCIe traffic generated by both graph sampling and feature extraction



Hotness-aware Unified Cache

- **Goal:**
  - Minimize PCIe traffic generated by both graph sampling and feature extraction

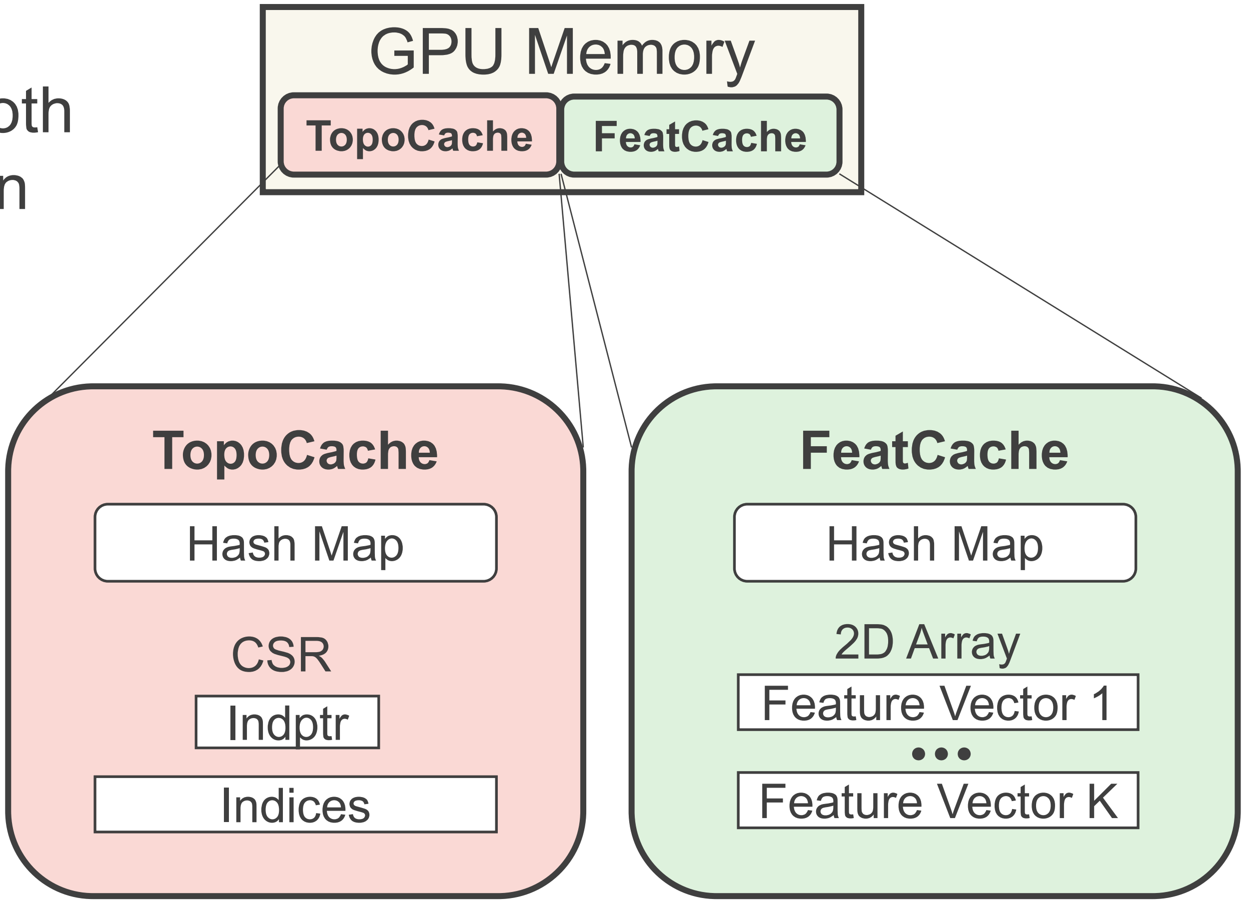


- **Principle:**
  - Fill the hottest graph topology and feature into TopoCache and FeatCache

Hotness-aware Unified Cache

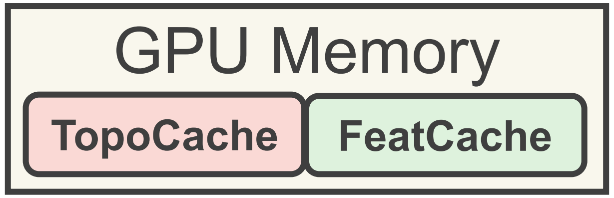
- **Goal:**
  - Minimize PCIe traffic generated by both graph sampling and feature extraction

- **Vertex-centric Data Structure**
  - ✓ TopoCache: CSR
  - ✓ FeatCache: 2D Array

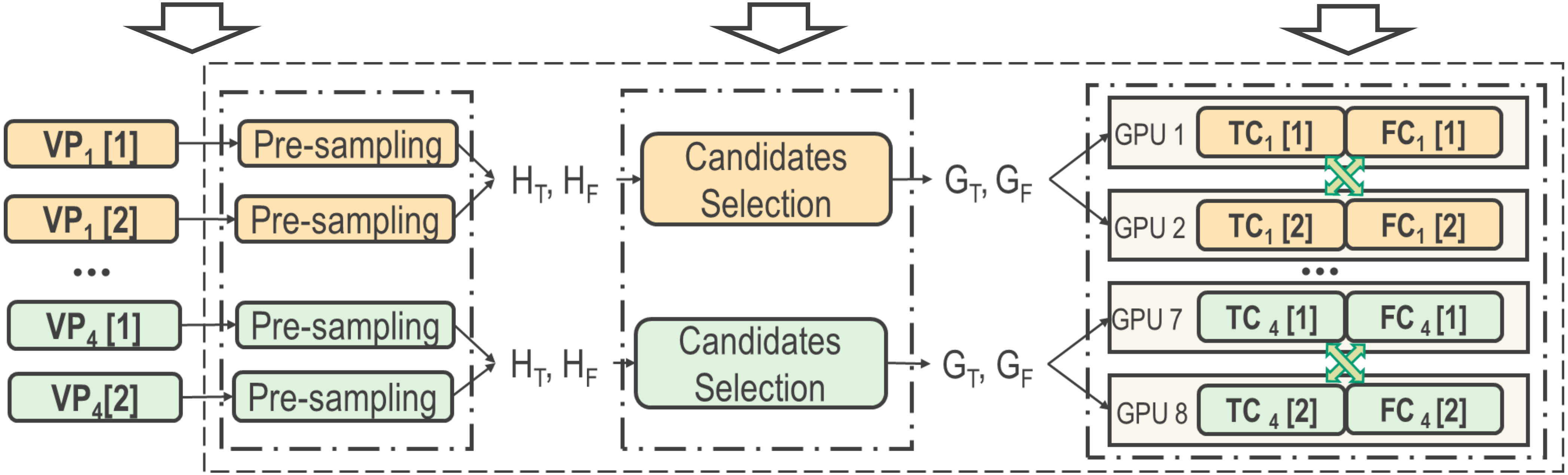


Hotness-aware Unified Cache

- **Goal:**
  - Minimize PCIe traffic generated by both graph sampling and feature extraction

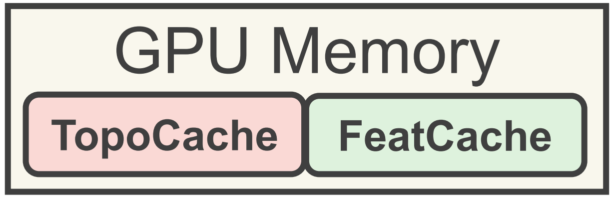


- **Step 1.** Pre-Sampling
- **Step 2.** Cache Candidate Selection
- **Step 3.** Cache Initialization and Fill-up



Pre-sampling

- **Goal:**
  - Count the hotness (access frequency) of vertices on every GPU



After 1 epoch of pre-sampling:

- Vertices Hotness of Topology

H_T [1]:

Vertex ID	Hotness
0	11
1	12
2	8
3	7
4	5
5	2
6	3
7	1

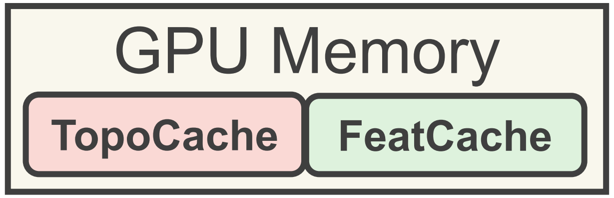
- Vertices Hotness of Feature

H_F [1]:

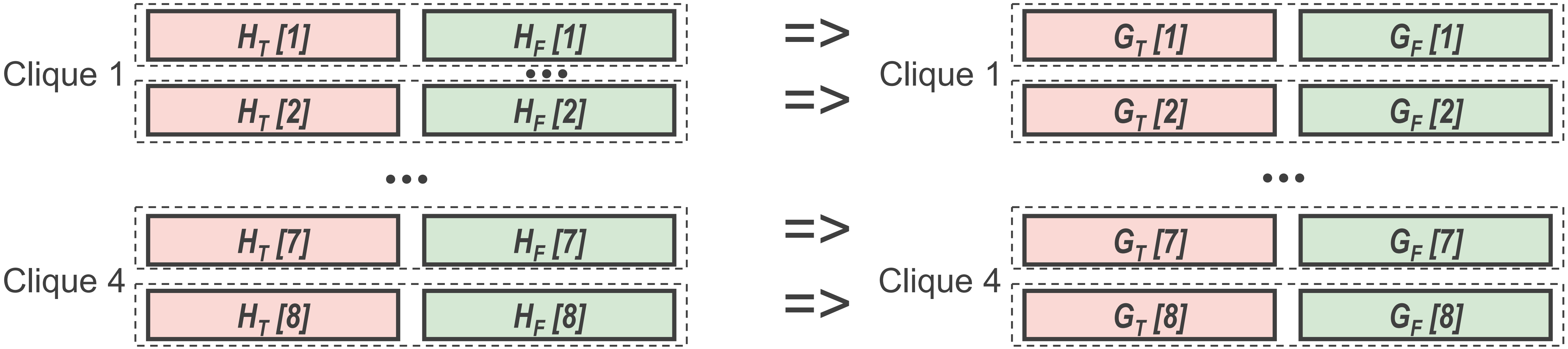
Vertex ID	Hotness
0	10
1	8
2	7
3	6
4	5
5	5
6	1
7	1

Cache Candidate Selection

- **Goal:**
  - Sort the vertices with high hotness to get the candidate queues on every GPU

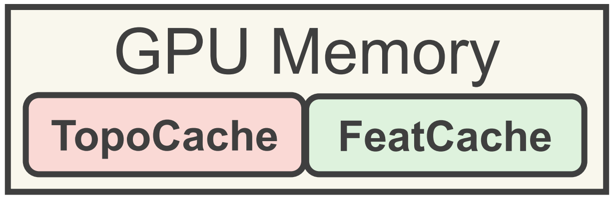


- Hotness
- Candidate Queues

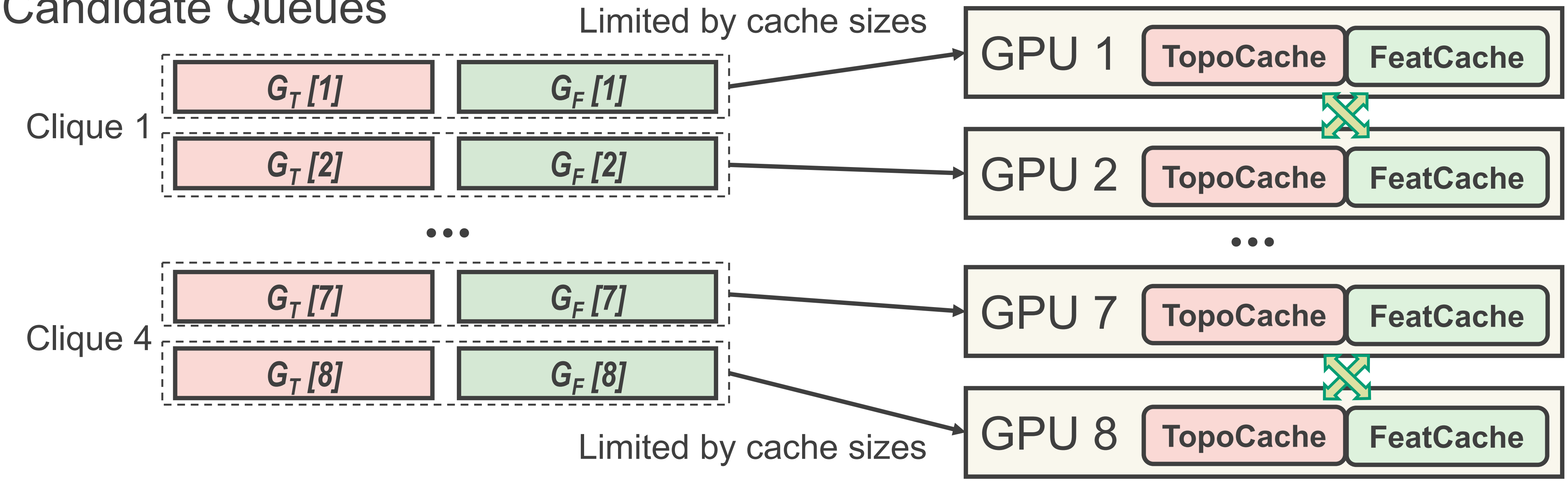


Cache Initialization and Fill-up

- **Goal:**
  - Load the topology & feature data from CPU to GPU memory



- Candidate Queues



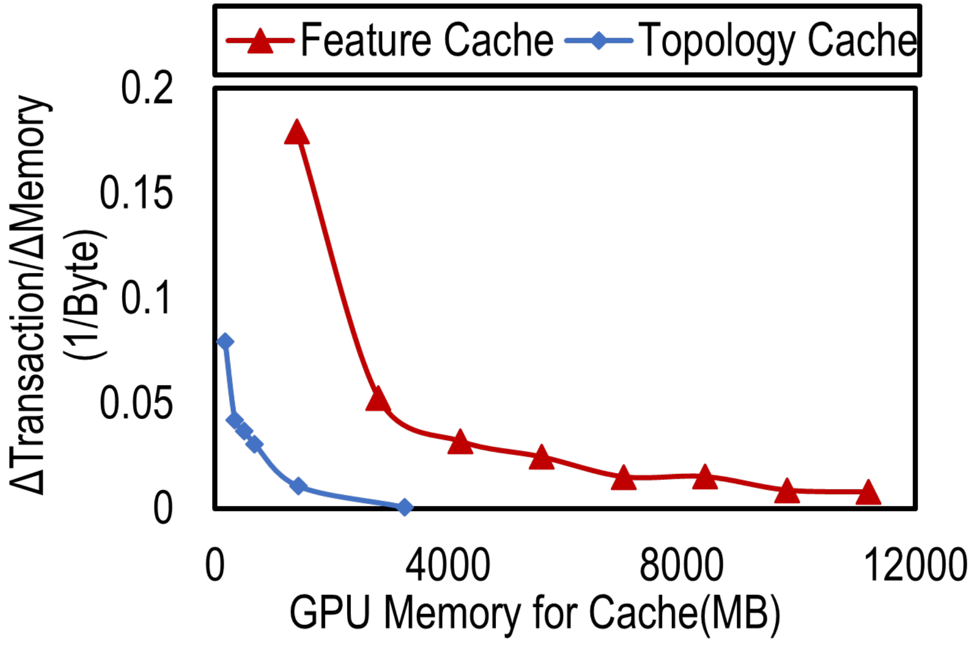
Legion

Contributions:

1. Hierarchical Graph Partitioning
2. Hotness-aware Unified Cache
3. Automatic Cache Management

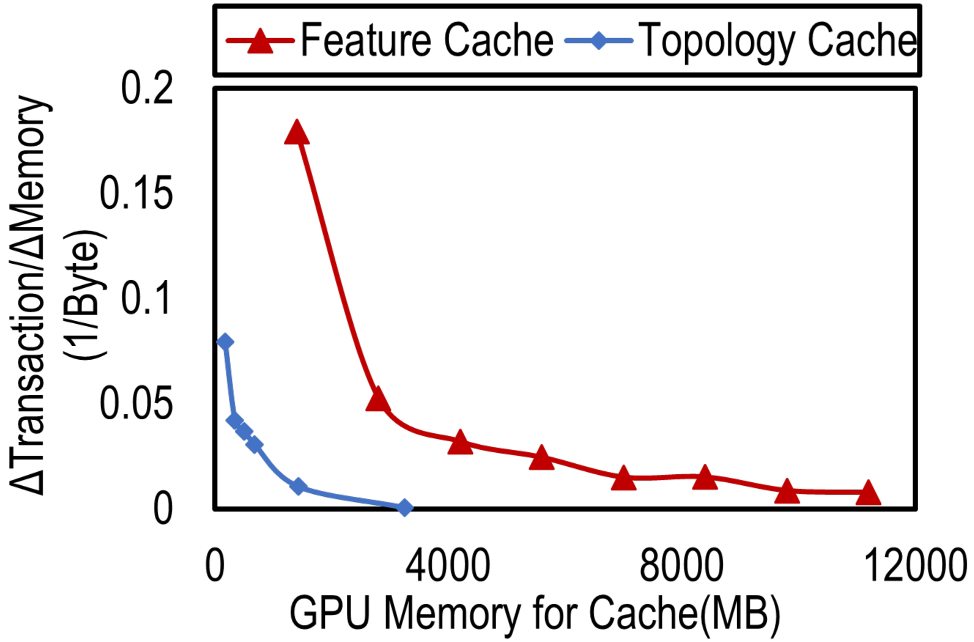
New Challenge

- **Trade-off**: Topology Cache vs Feature Cache



New Challenge

- **Trade-off**: Topology Cache vs Feature Cache



- How to find the optimal size of topology and feature cache **automatically**?

Automatic Cache Management

- **Goal**: Automatically decide topology & feature cache size to maximize the overall training throughput

Automatic Cache Management

- **Goal**: Automatically decide topology & feature cache size to maximize the overall training throughput

- Use the overall PCIe traffic to estimate overall throughput

Reasons:

◆ PCIe traffic is the system bottleneck
◆ Larger topology cache size => Lower PCIe traffic of graph sampling
◆ Larger feature cache size => Lower PCIe traffic of feature extraction

Automatic Cache Management

- **Goal**: Automatically decide topology & feature cache size to maximize the overall training throughput

- Use the overall PCIe traffic to estimate overall throughput

- Build cost model to estimate the overall PCIe traffic

Automatic Cache Management

- **Goal**: Automatically decide topology & feature cache size to maximize the overall training throughput

↑

- Use the overall PCIe traffic to estimate overall throughput

↑

- Build cost model to estimate the overall PCIe traffic

↓

- **Method:**
  - Build the cost model at the NVLink-clique granularity
  - One GPU in a clique calculates cost model and search for optimal cache plan

Experimental Settings

- **Datasets:**
  - Billion-scale real-world graphs

Dataset	PR	PA	CO	UKS	UKL	CL
Vertices	2.4M	111M	65M	133M	0.79B	1B
Edges	120M	1.6B	1.8B	5.5B	47.2B	42.5B
Topology Storage	640M	6.4GB	7.2GB	22GB	189GB	170GB
Feature Size	100	128	256	256	128	128
Feature Storage	960M	56GB	65GB	136GB	400GB	512GB

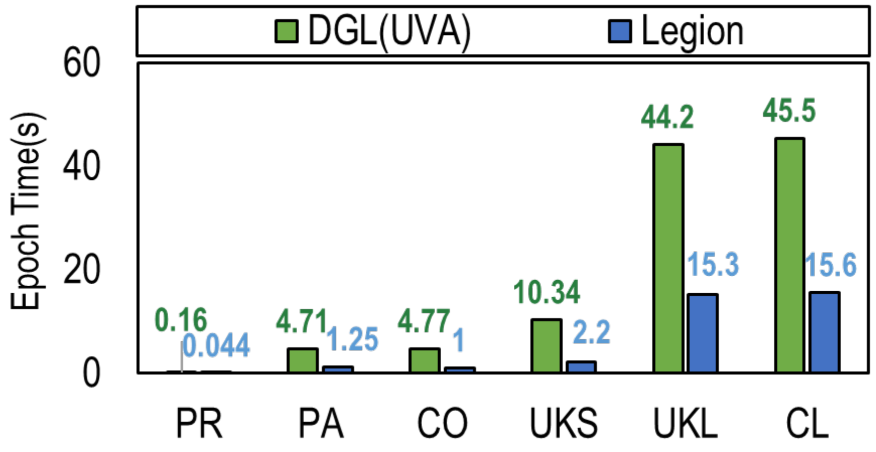
- **Platforms:**
  - Three multi-GPU platforms with different NVLink topologies

Server	DGX-V100	Siton	DGX-A100
GPU Type	16GB-V100x8	40GB-A100x8	80GB-A100x8
NVLink Topo.	$K_c = 2, K_g = 4$	$K_c = 4, K_g = 2$	$K_c = 1, K_g = 8$
PCIe	3.0x16	4.0x16	4.0x16
CPU Mem.	384GB	1TB	1TB

- **Models:**
  - Two popular GNN models: GraphSAGE, GCN

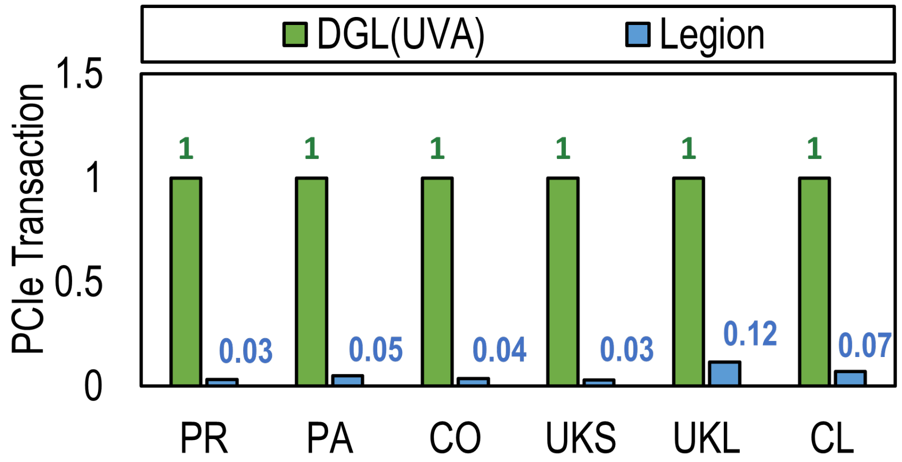
Evaluation

- **Train billion-scale graphs**
  - Existing cache-based system cannot scale well



DGX-A100, GraphSAGE

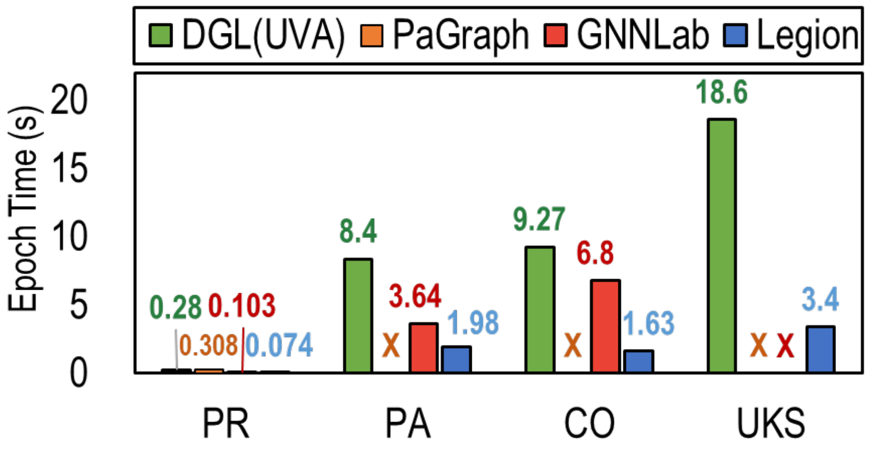
- **Minimize PCIe traffic**
  - Significantly reduce the traffic comparing to baseline



DGX-A100, GraphSAGE

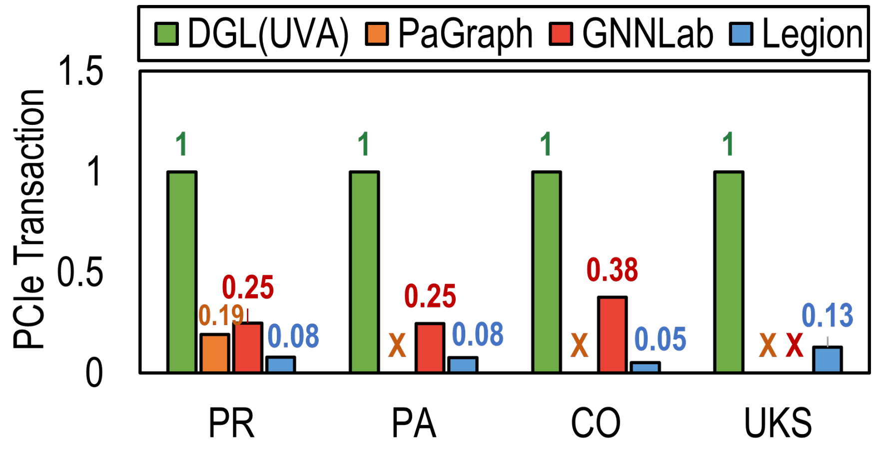
Evaluation

- **Train small graphs**
  - Outperform SOTA systems by up to 4.32x



DGX-V100, GraphSAGE

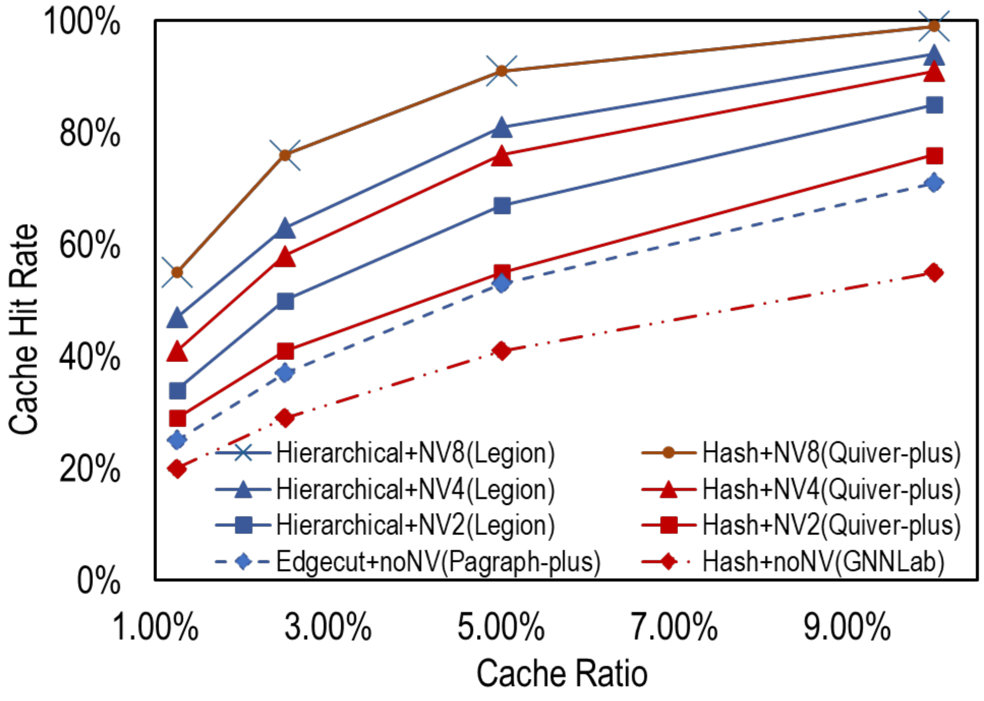
- **Minimize PCIe traffic**
  - Significantly reduce the traffic comparing to baselines



DGX-V100, GraphSAGE

Evaluation

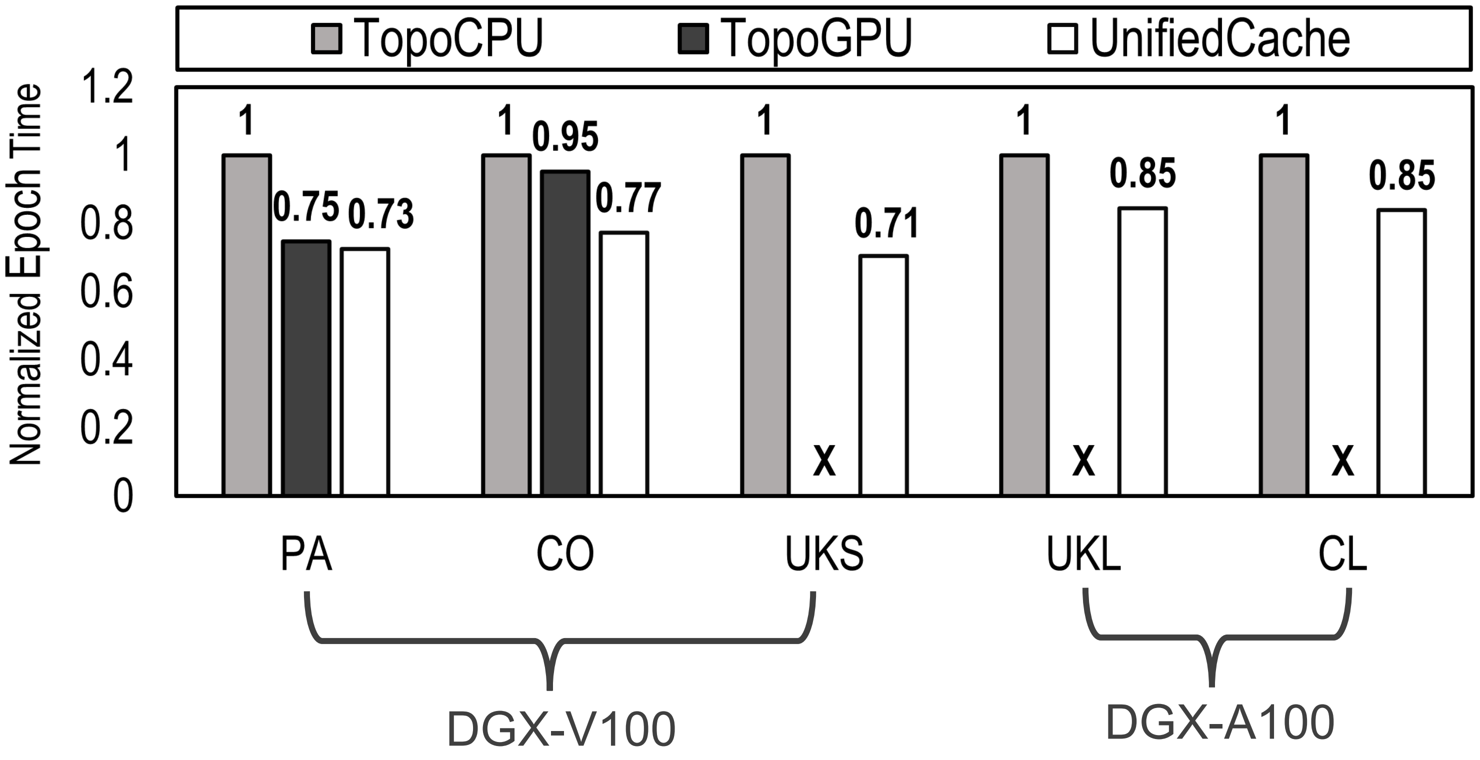
- **Impact of Hierarchical Graph Partitioning**
  - In all platforms, Legion has a higher cache hit rate than baselines



Dataset: CO

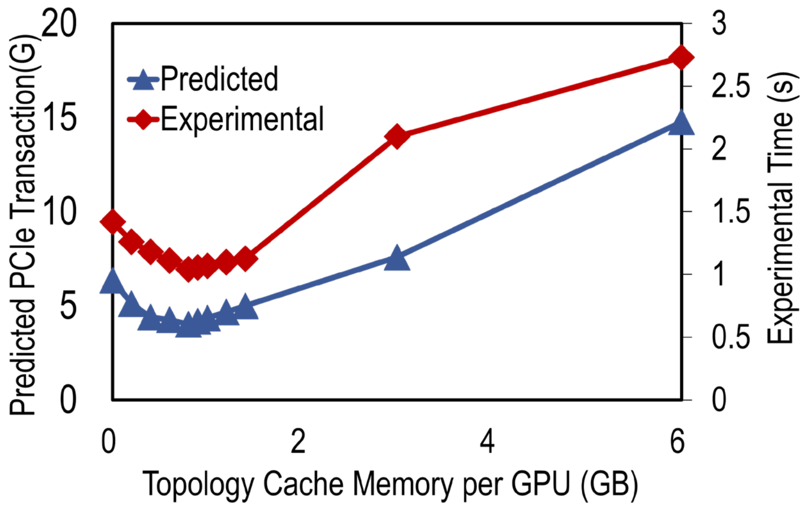
Evaluation

- **Impact of Unified Cache**
  - Unified cache outperforms all baselines in all datasets
  - All topology in GPU meet OOM in UKS, UKL, and CL

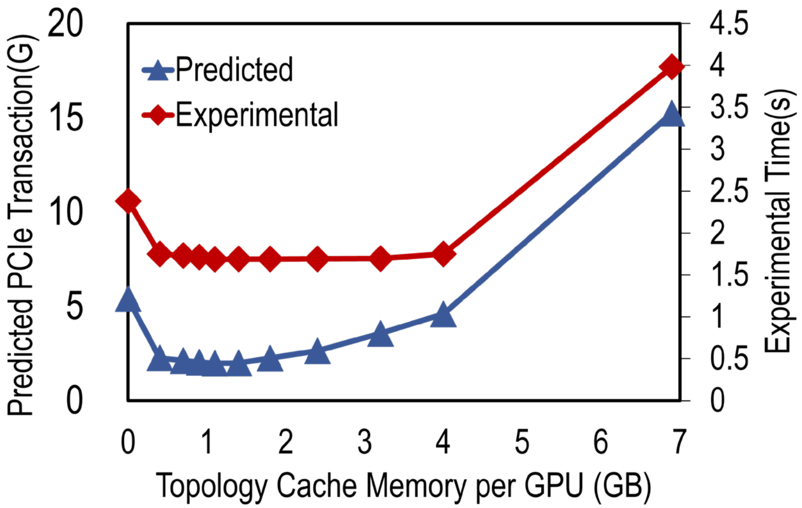


Evaluation

- **Impact of Automatic Cache Management**
  - Legion precisely predicts the trend of per-epoch execution time without manual interference



Single GPU



DGX-V100

Q & A

Thanks!

Q & A