DiskGNN: Bridging I/O Efficiency and Model Accuracy for Out-of-Core GNN Training

Renjie Liu[*1], **Yichuan Wang**[*2], Xiao Yan[3], Haitian Jiang[4],

Zhenkun Cai[5], Minjie Wang[5], Bo Tang[1], Jinyang Li[4]

1SUSTech, **2**UC Berkeley, **3**CPII HK, **4**New York University, **5**AWS

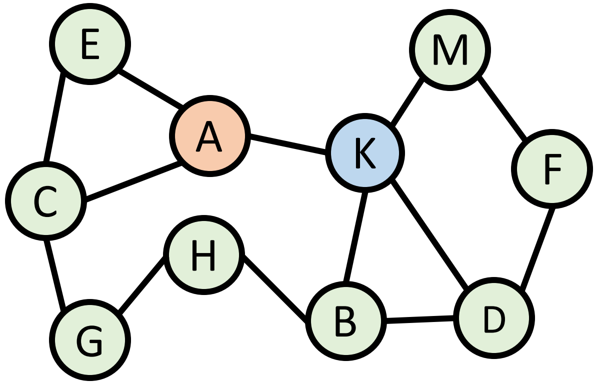
*Equal Contribution

Graph Neural Network (GNN)

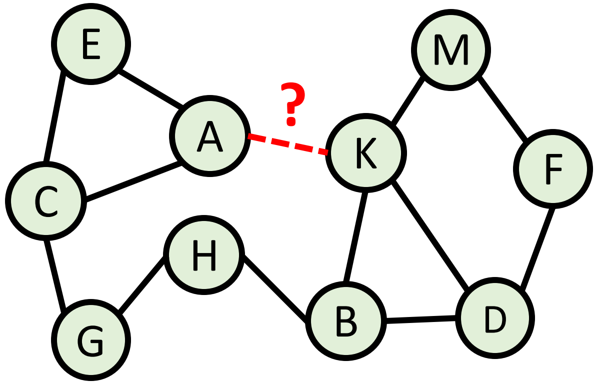
- Learn representation by neighbor aggregation and message passing:

$$h_v^k = \sigma[AGG^{\mathrm{k}}(\{W^{\mathrm{k}} h_{\mathrm{u}}{}^{\mathrm{k-1}}, \forall u \in N(v)\})]$$

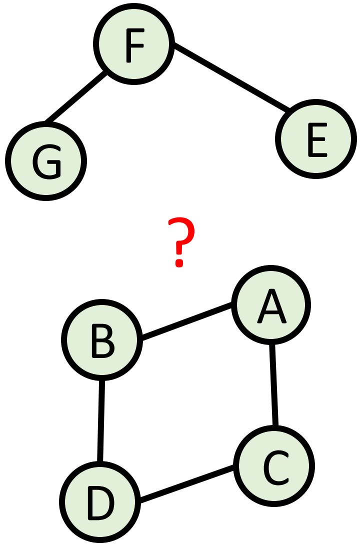
- Down-stream applications:



Node Classification



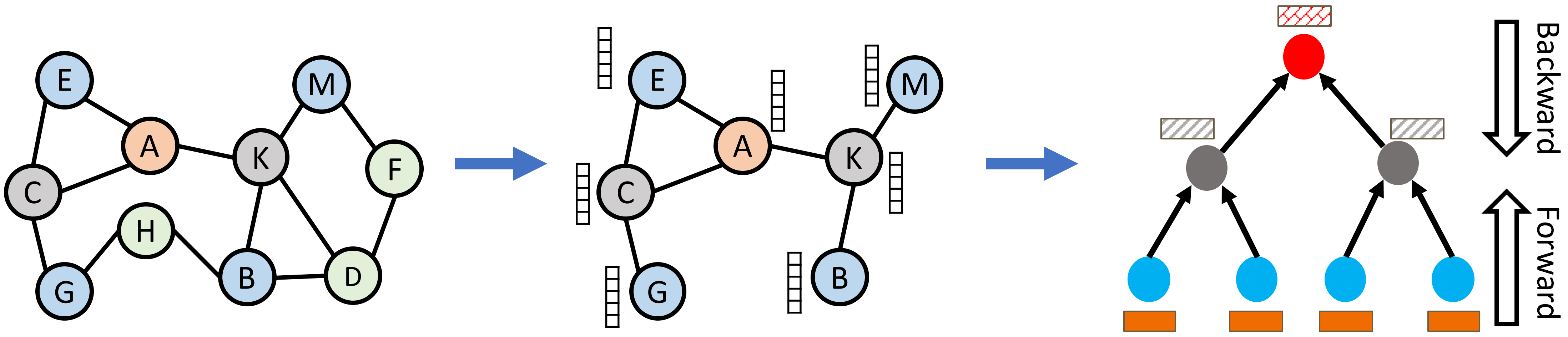
Link Prediction



Graph Classification

Training Graph Neural Network

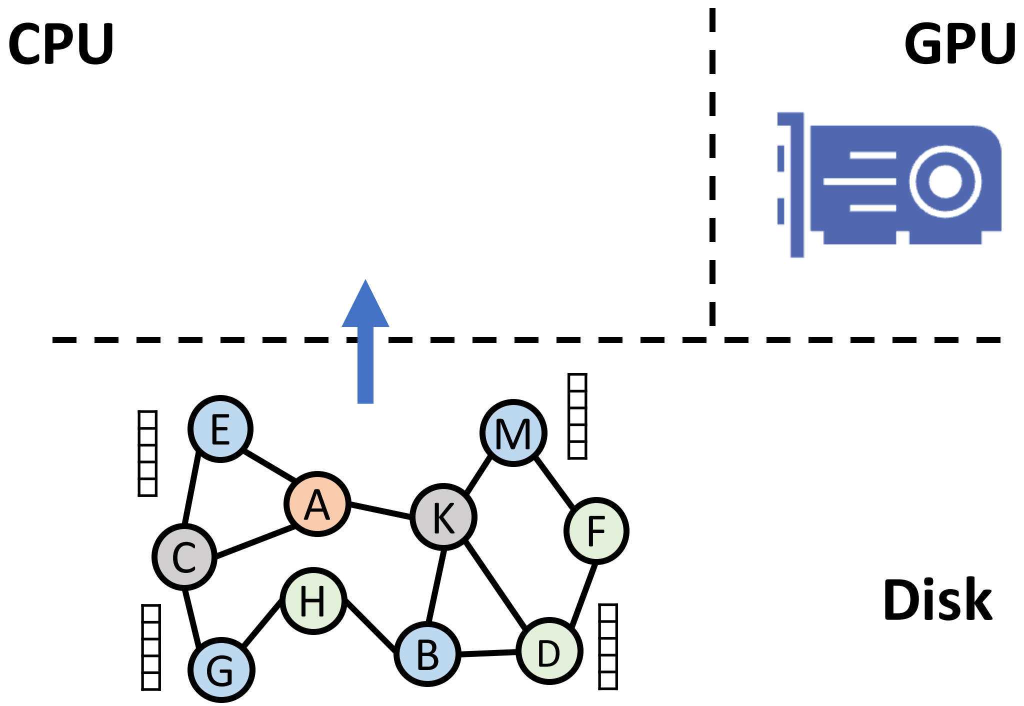
- **Graph Sampling**: Sample multi-hop neighbors of the seed node.
- **Feature Loading**: Load features of the sampled nodes.
- **Model Training**: Compute GNN forward/backward traces.



Graph Sampling **Feature Loading** **Model Training**

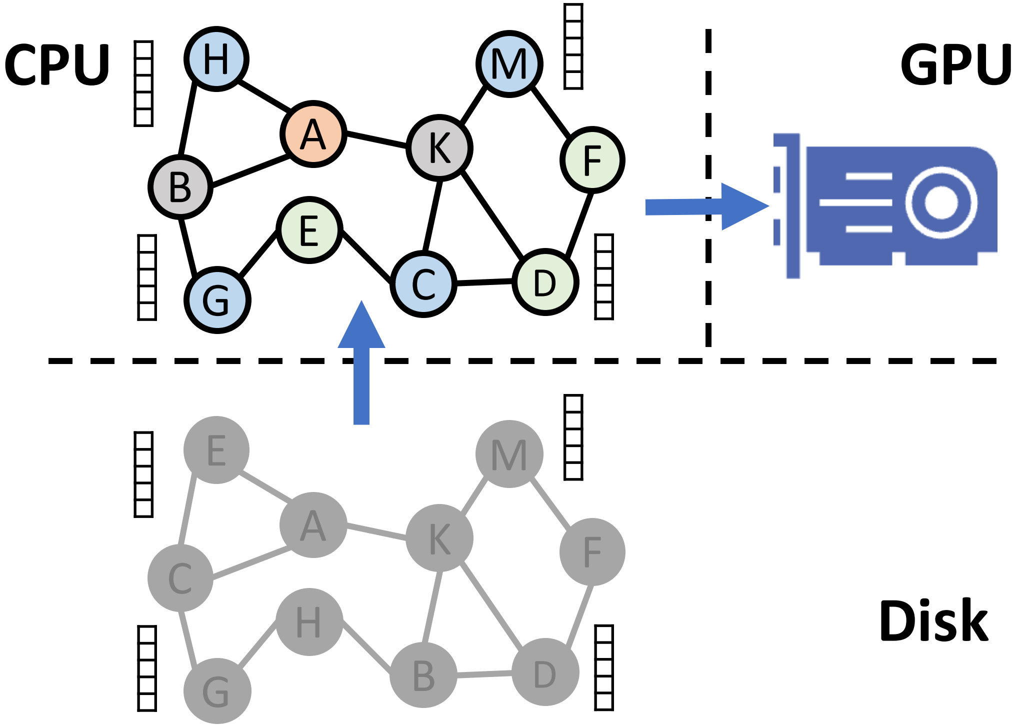
Before Training GNNs…

- The whole graph will be loaded from disk to memory.



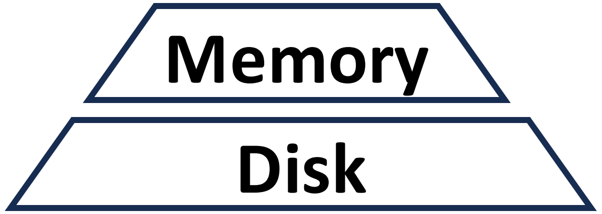
Before Training GNNs…

- The whole graph will be loaded from disk to memory.
- Not feasible for large graphs in resource-constrained environments.



Memory (DDR4)

Size: GBs, Tput: ~25GB/s, IOPS: >10M IOPS

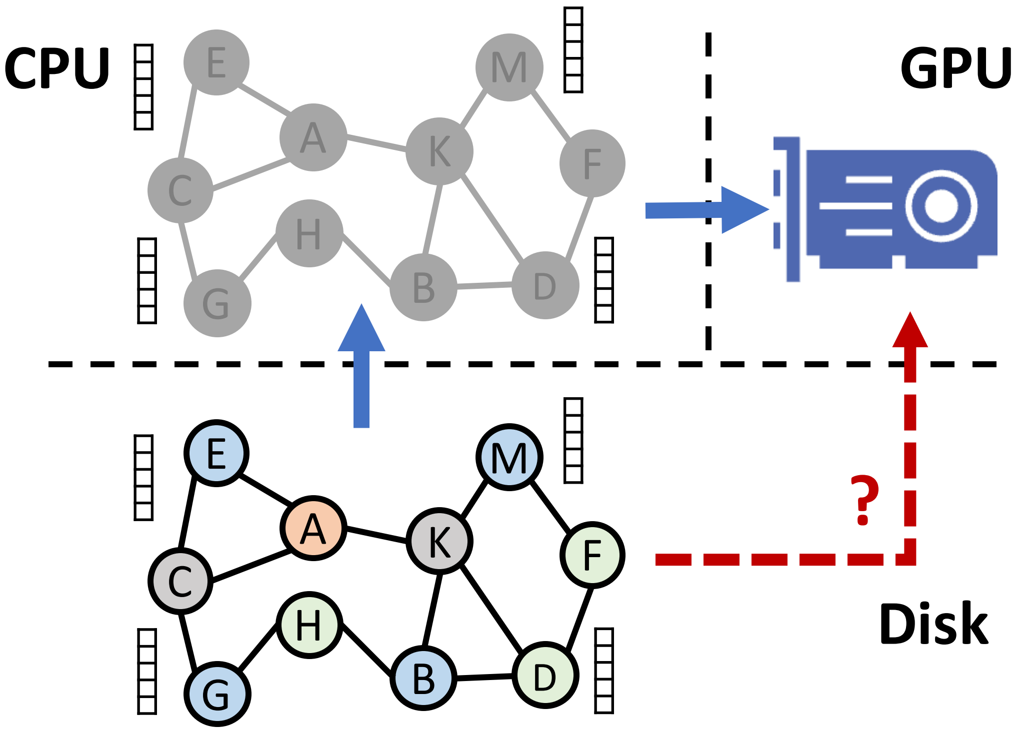


Disk (NVMe):

Size: TBs, Tput: ~3GB/s, IOPS: ~1M IOPS

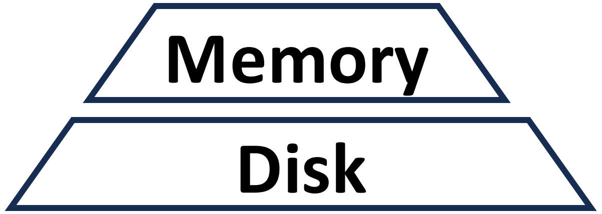
Before Training GNNs…

- We can only store the whole large graphs on disk.
- How to train GNNs in reasonable time on slower storage?



Memory (DDR4)

Size: GBs, Tput: ~25GB/s, IOPS: >10M IOPS



Disk (NVMe):

Size: TBs, Tput: ~3GB/s, IOPS: ~1M IOPS

Storage Type	RD IOPS (4KB)	Bandwidth	Price ($/GB)
DRAM (DDR4)	>10M	25 GB/s	11.13
AWS NVMe	625k	3 GB/s	0.125
AWS gp3	16k	2.5 GB/s	0.08

Out-of-core GNN Training

Problem: Existing systems either lack efficiency or degrade model accuracy.

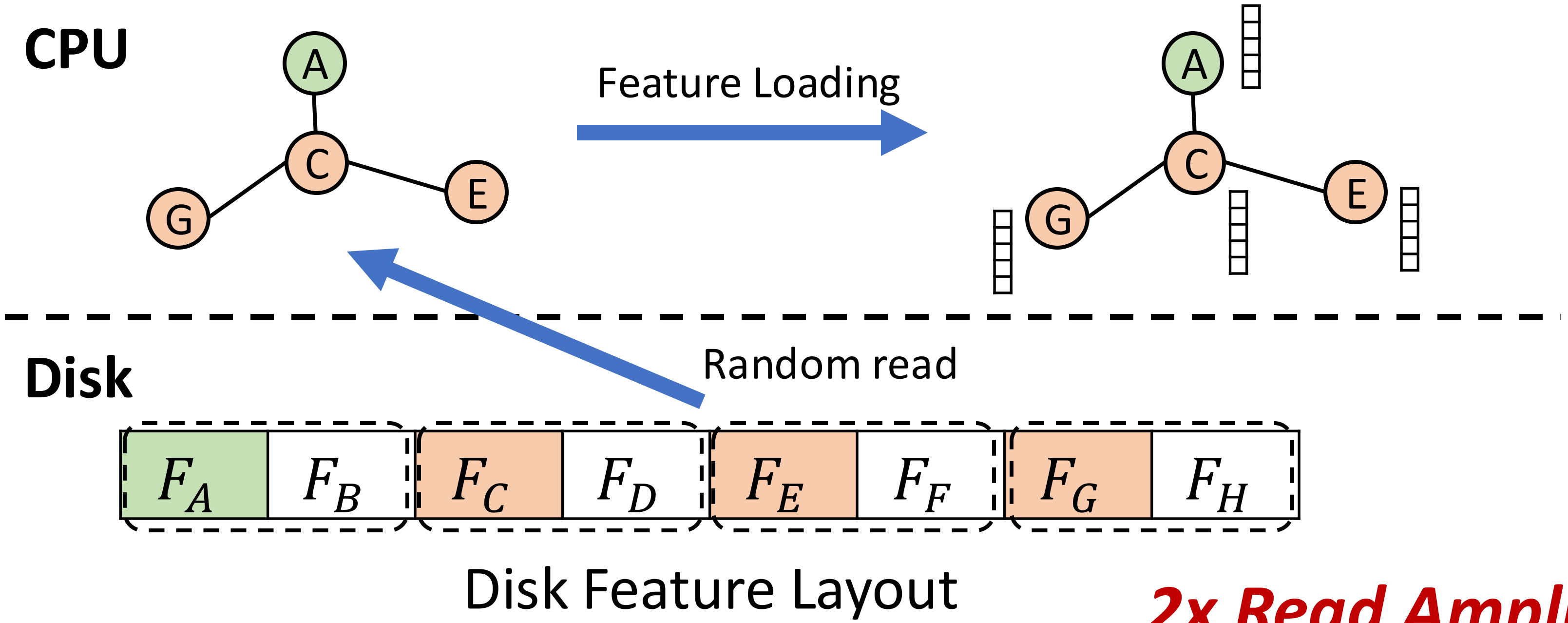
Node-wise disk access: Suffer from **Disk Read Amplification**.

Block-wise disk access: Suffer from **Degraded Model Accuracy**.

Out-of-core GNN Training

Node-wise disk access (Ginex, GIDS, Helios):

- Fine-grained accesses are smaller than a disk page (4KB).

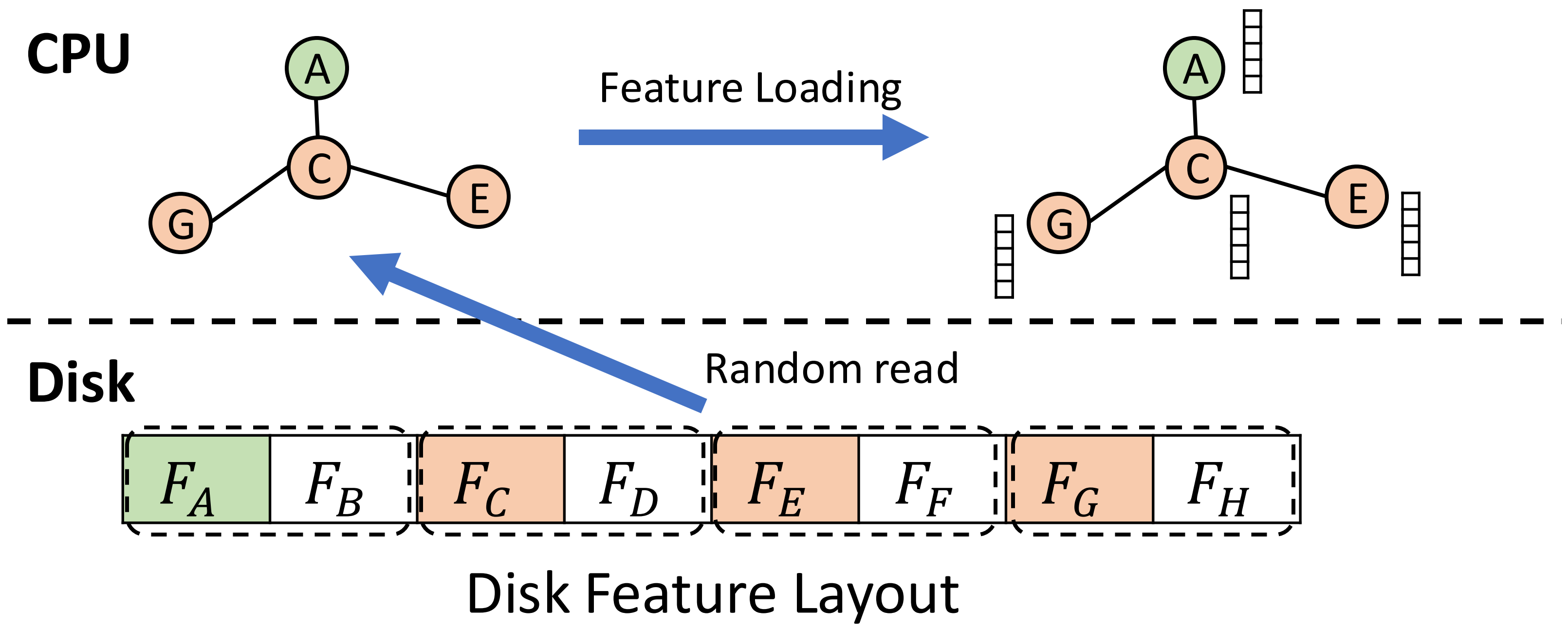


2x Read Amplification!

Out-of-core GNN Training

Node-wise disk access (Ginex, GIDS, Helios):

- Fine-grained accesses are smaller than a disk page (4KB).

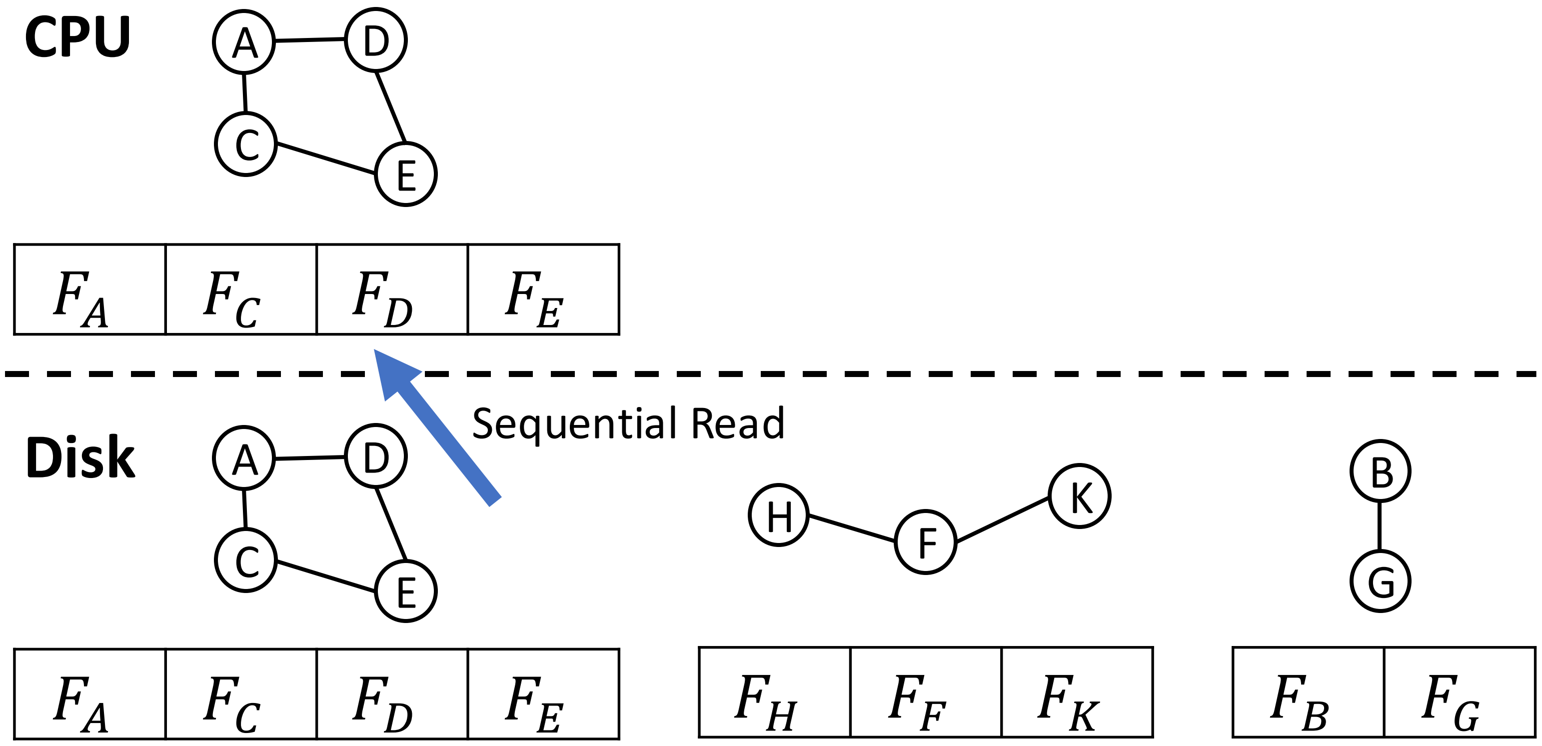


8x read amplification when feature dimension is 128 (FP32)!

Out-of-core GNN Training

Block-wise disk access (MariusGNN):

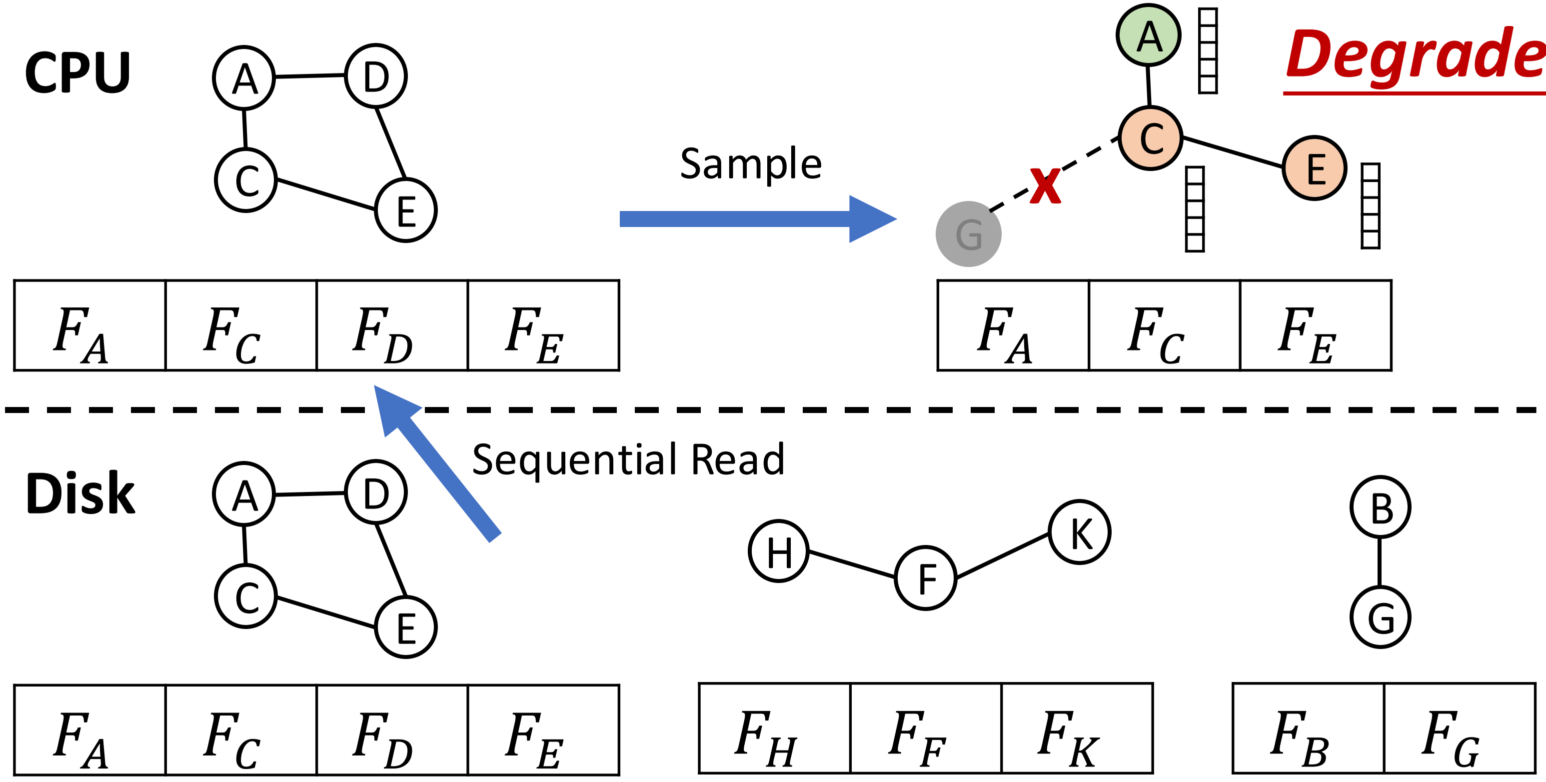
- Cross-partition edges are ignored during sampling.



Out-of-core GNN Training

<u>Block-wise disk access (MariusGNN):</u>

- Cross-partition edges are ignored during sampling.



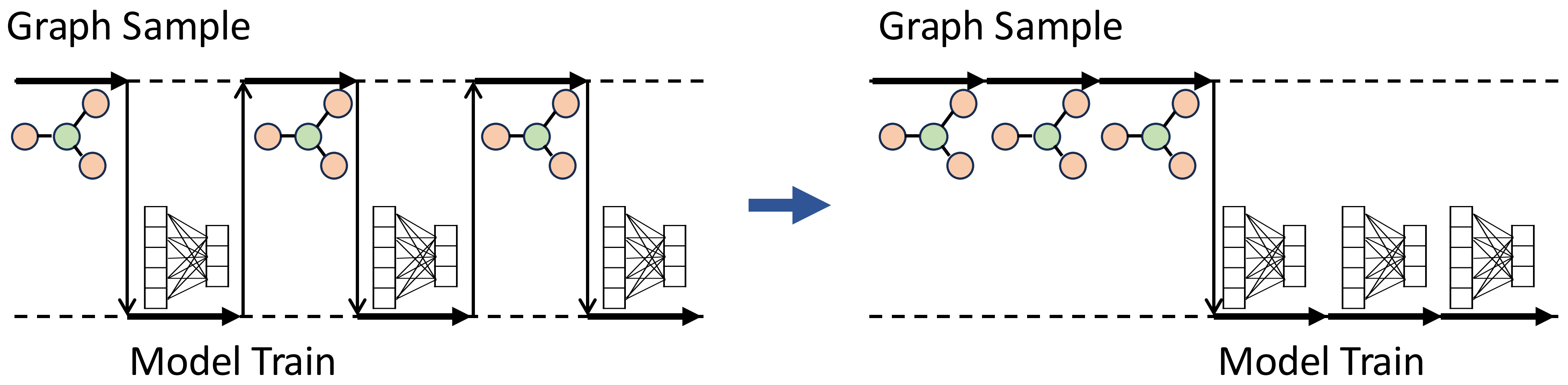
Degrade Model Accuracy!

(~2% in practice)

DiskGNN: Offline Sampling

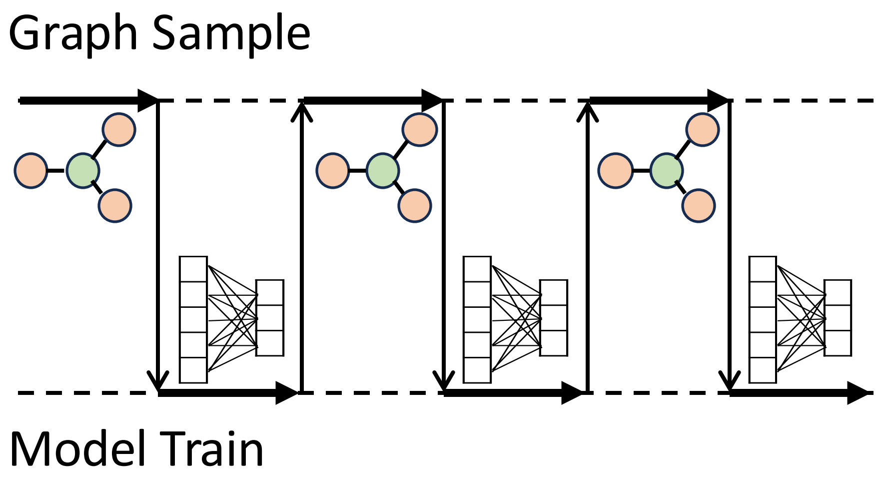
Sampling & training can be decoupled:

- Observation: accuracy is not harmed with sufficient minibatches.

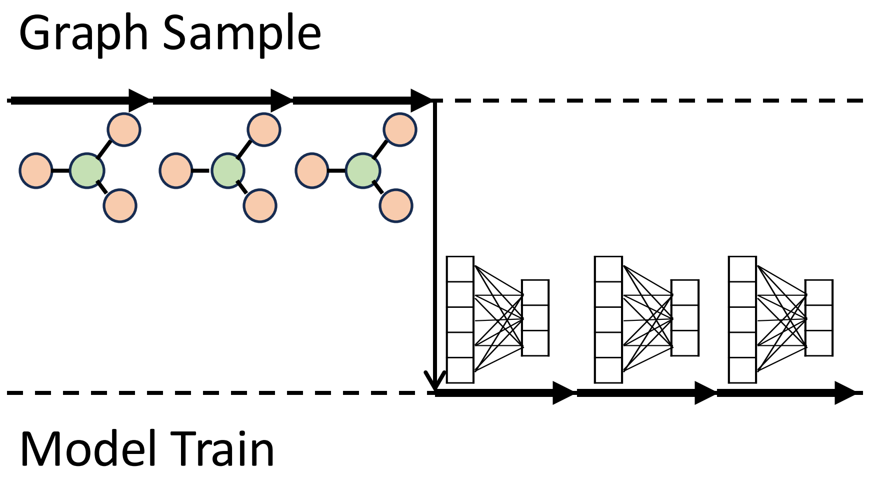


Interleaved Execution ***Offline Sampling***

DiskGNN: Offline Sampling



Interleaved Execution



Offline Sampling

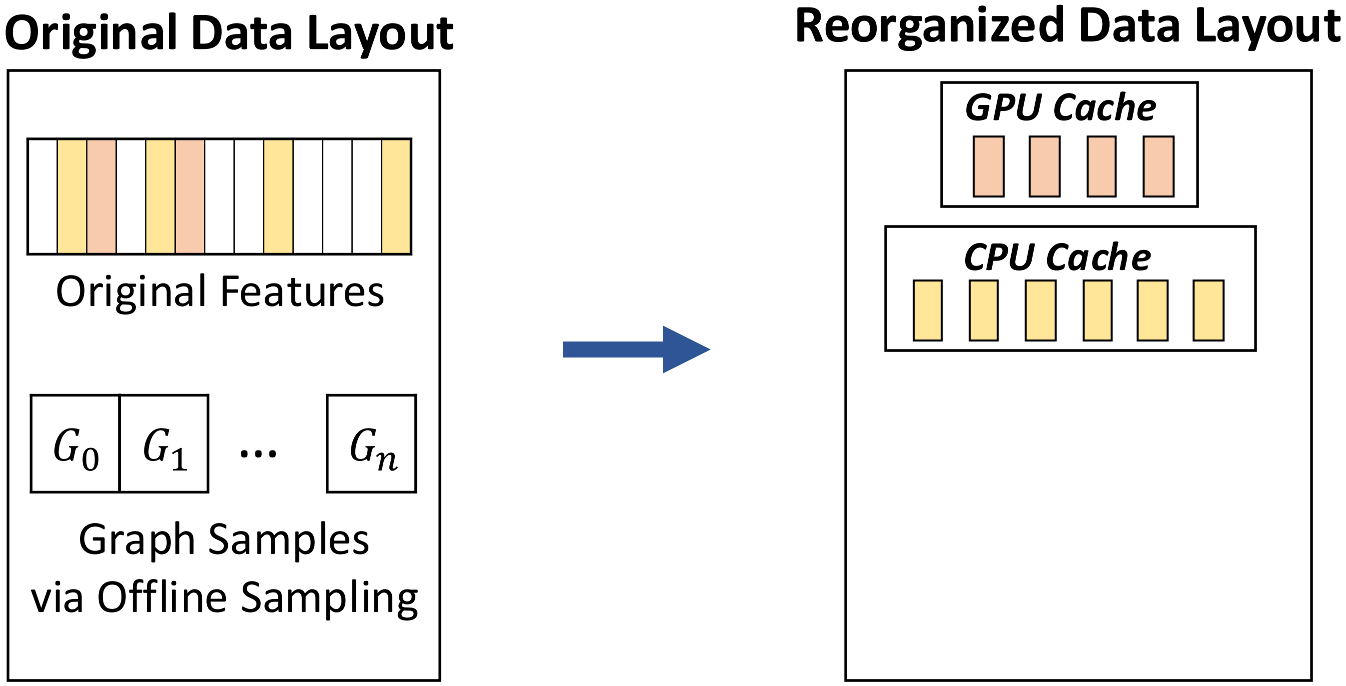
<u>Benefits to out-of-core training:</u>

- Better feature cache strategy: less disk I/O volume.
- Better disk data layout: lower read amplification.

DiskGNN: System Design

Built on top of offline sampling:

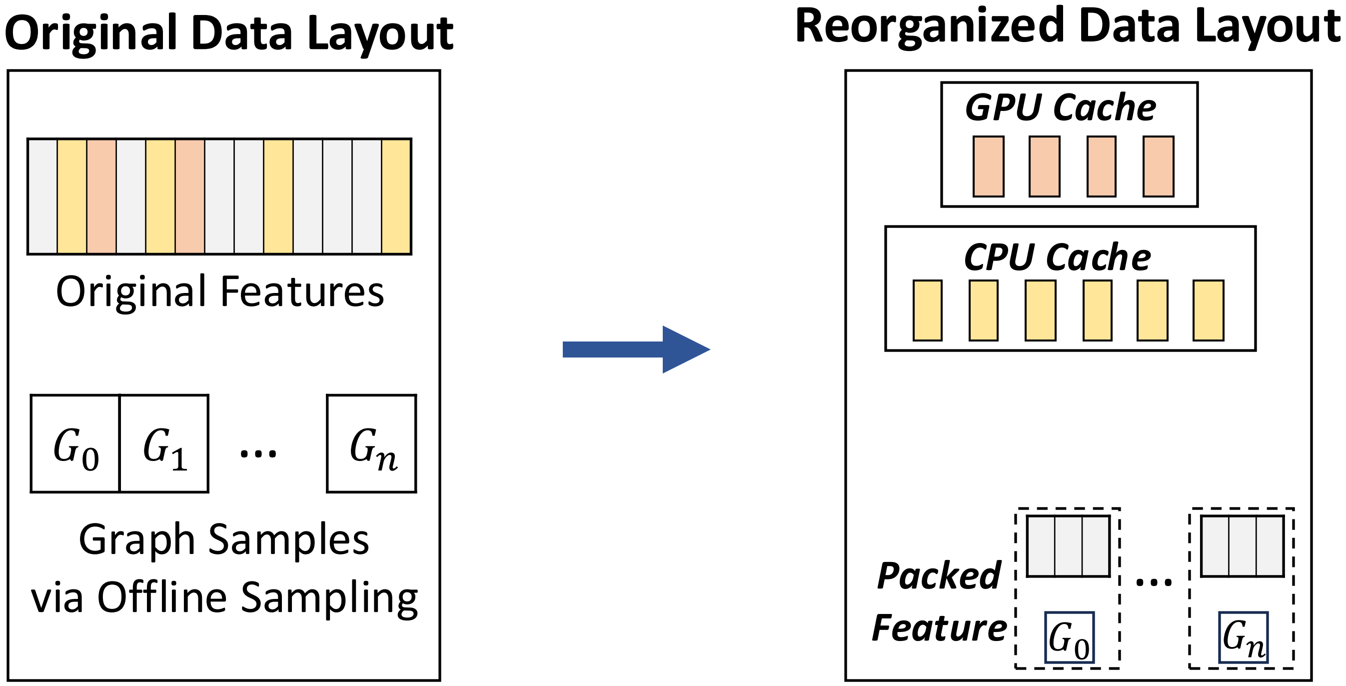
- Cache node features in GPU and CPU memory by global hotness.



DiskGNN: System Design

Built on top of offline sampling:

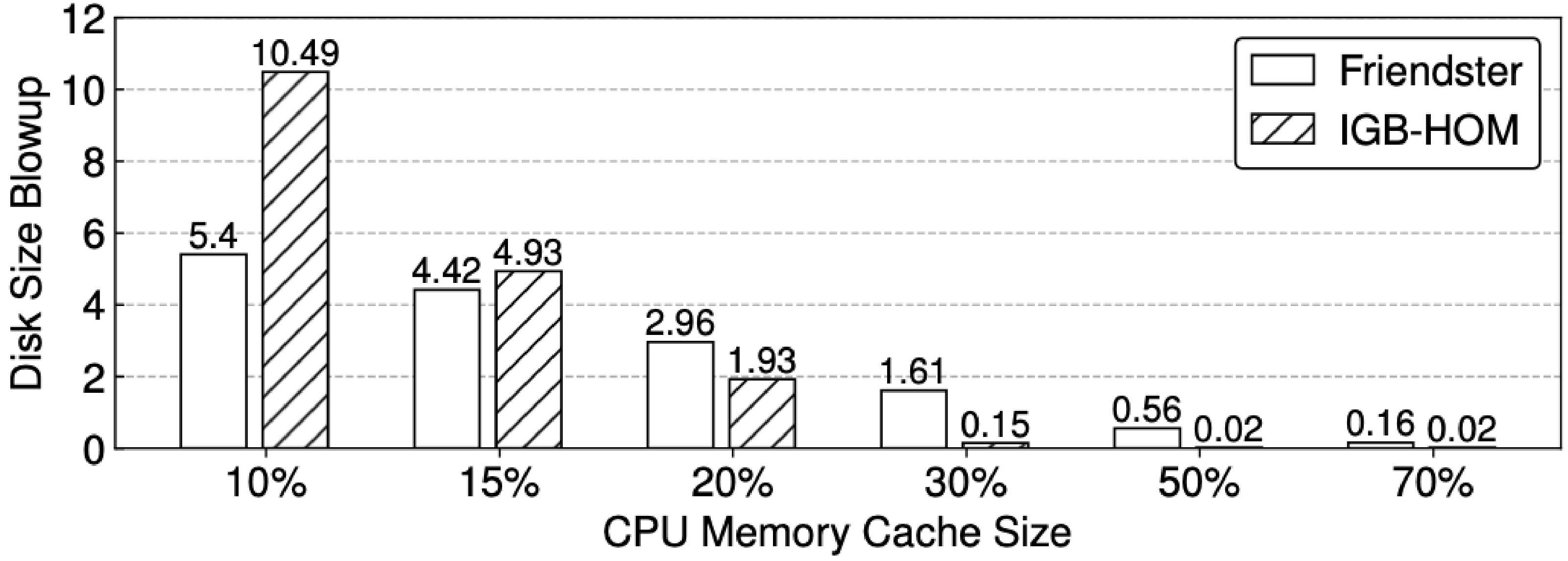
- Cache node features in GPU and CPU memory by global hotness.
- Pack cache-missed features in contiguous disk storage.



DiskGNN: Challenges & Solutions

<u>Challenge 1: Feature Packing introduces replication of data.</u>

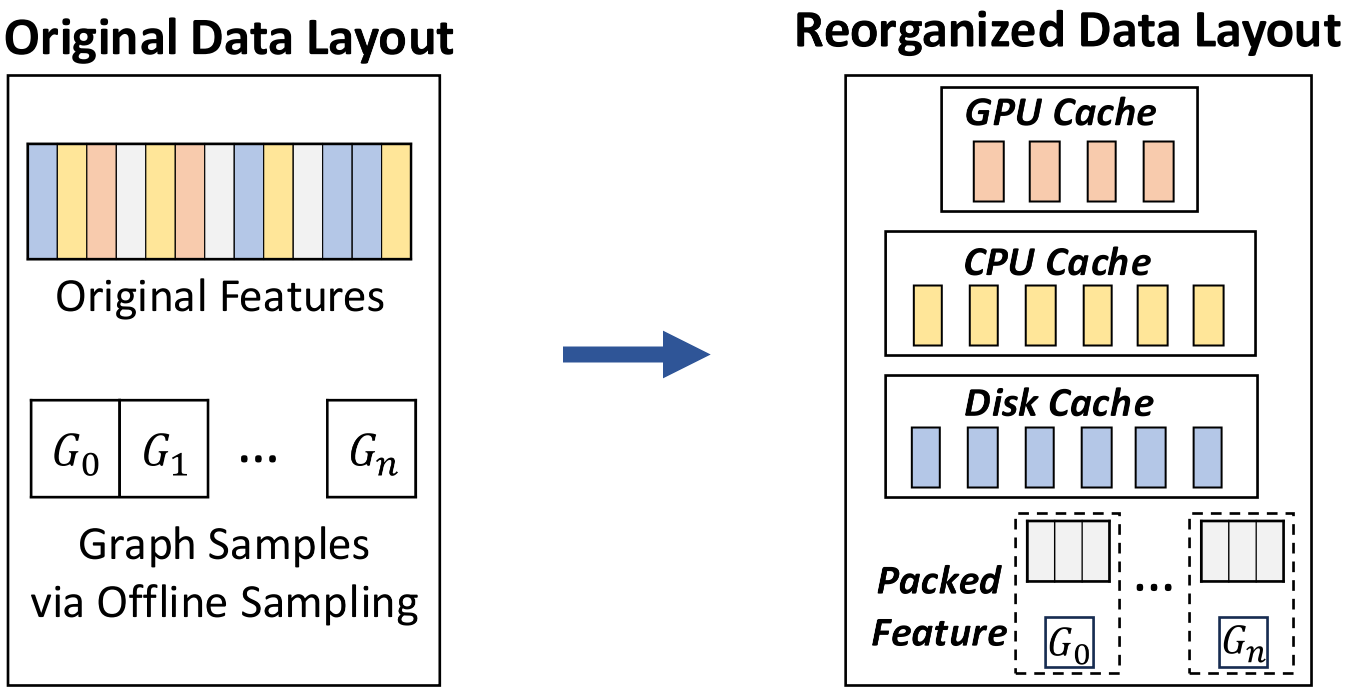
- Might consume too much disk storage (e.g., 10x on IGB-HOM).



DiskGNN: Challenges & Solutions

Challenge 1: Feature Packing introduces replication of data.

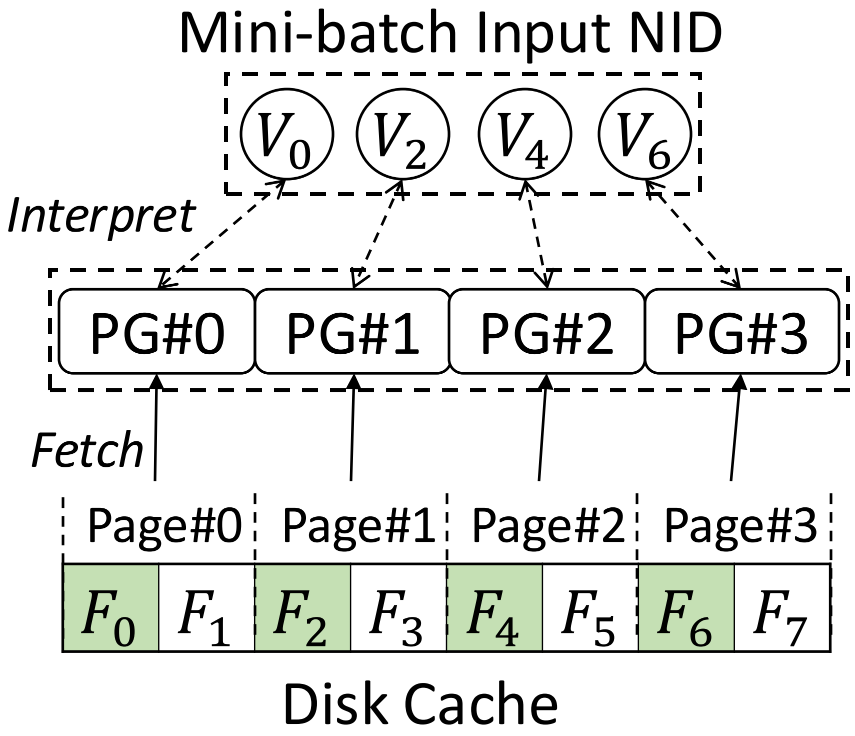
- **Solution: introduce another cache on disk to reduce replication.**



DiskGNN: Challenges & Solutions

<u>Challenge 2: Fetching data from disk cache is still random read.</u>

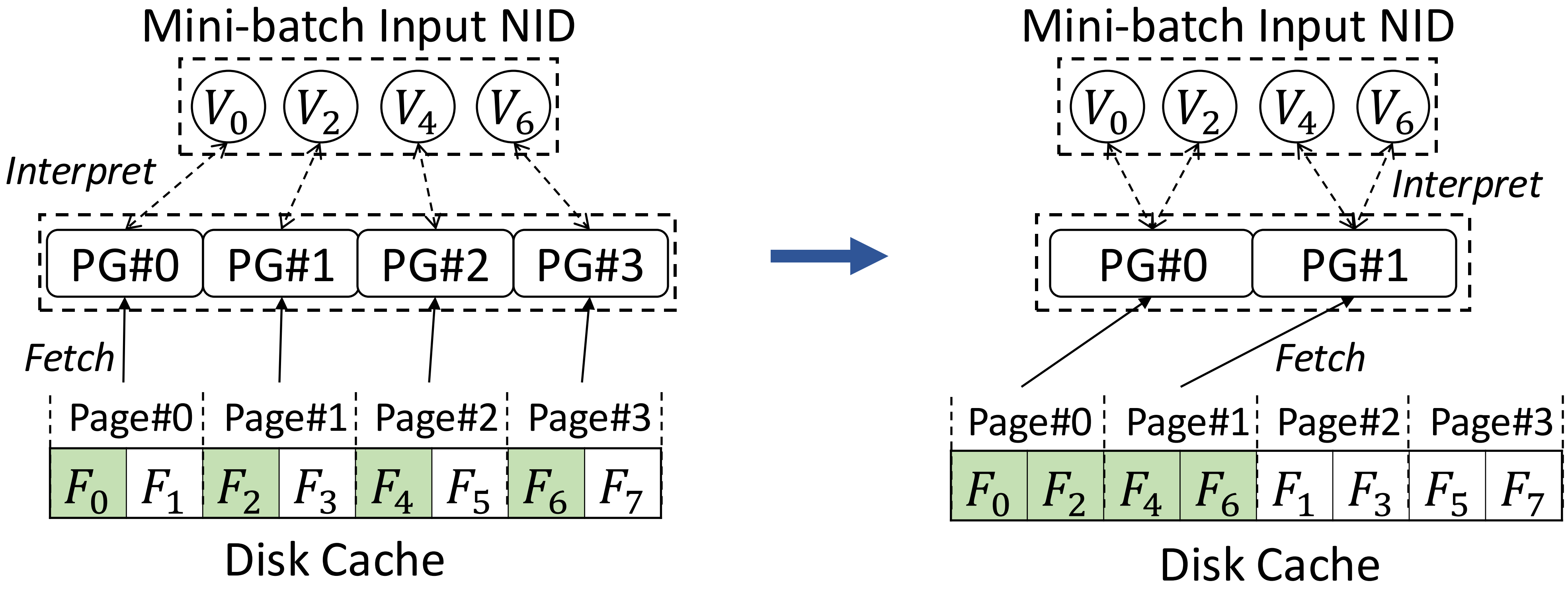
- Random read from disk cache involves read amplification.



DiskGNN: Challenges & Solutions

Challenge 2: Fetching data from disk cache is still random read.

- **Solution: reorder disk cache (*MinHash*) to reduce read amplification.**

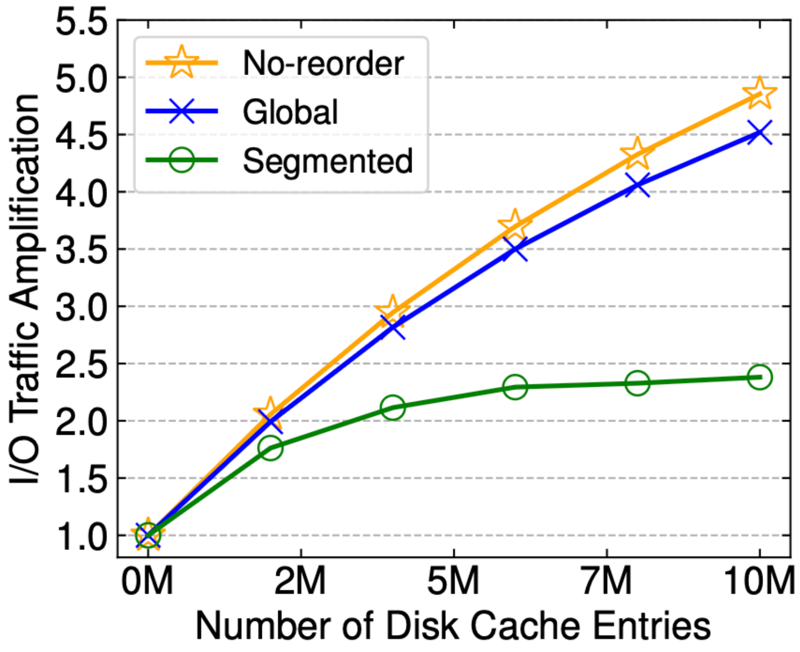


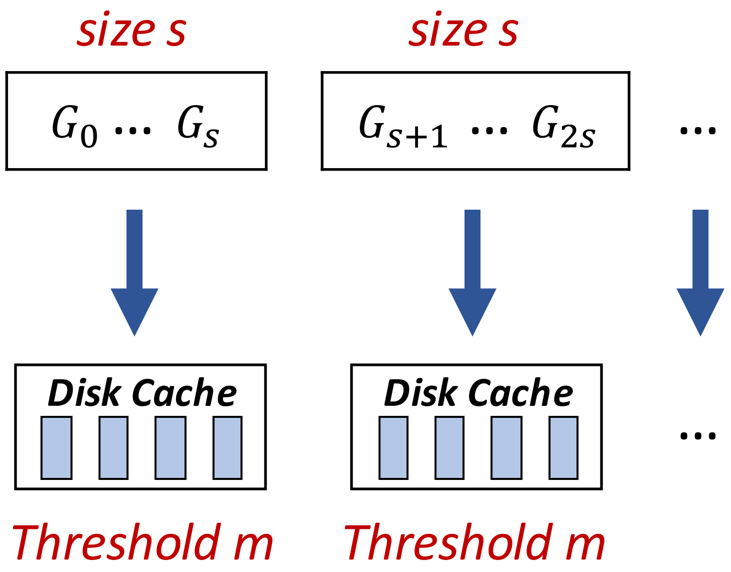
DiskGNN: Challenges & Solutions

Challenge 2: Fetching data from disk cache is still random read.

- **Solution: reorder disk cache (*MinHash*) to reduce read amplification.**

Segmented Disk Cache with MinHash:

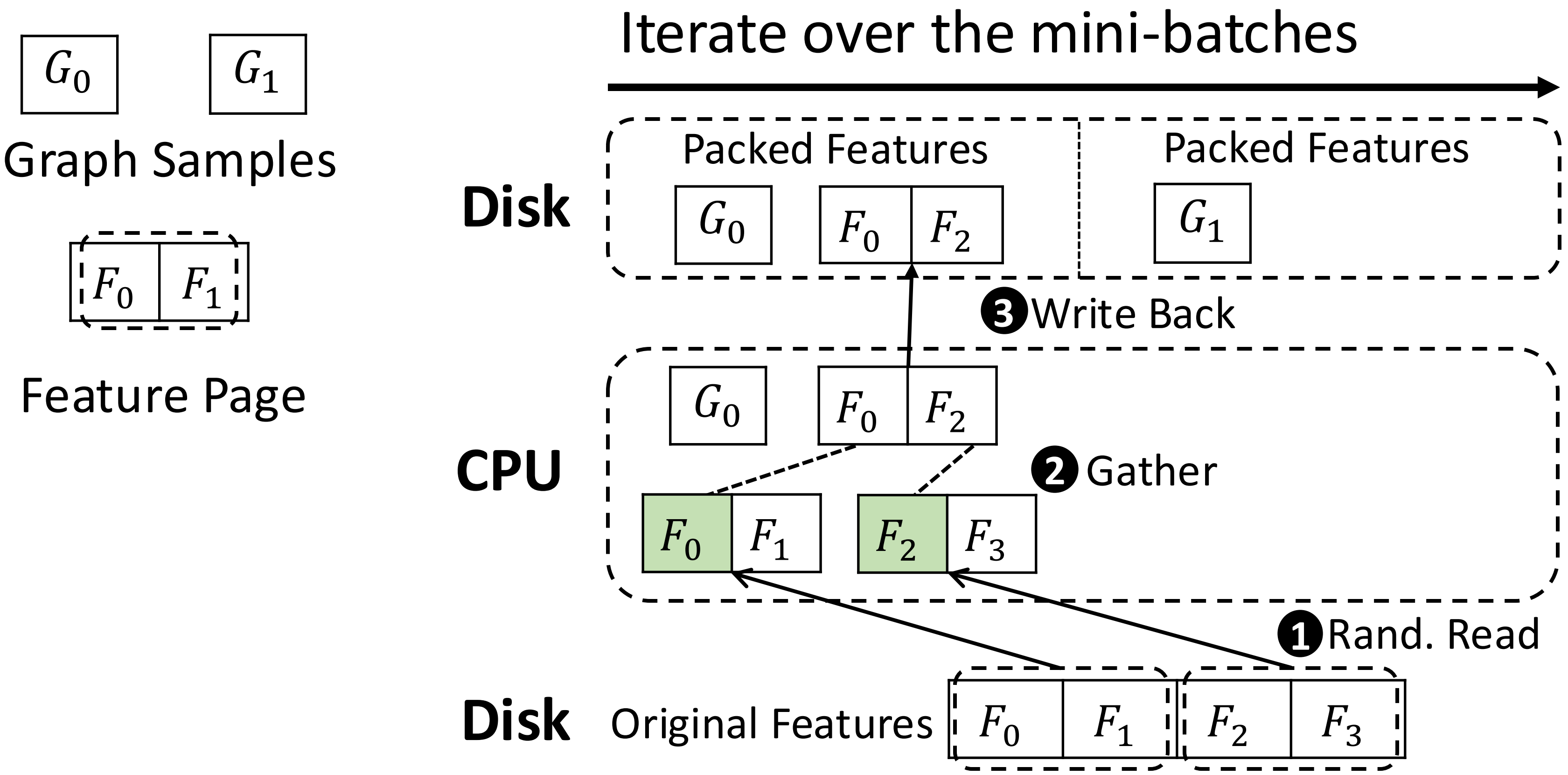




DiskGNN: Challenges & Solutions

Challenge 3: Feature Packing could also take a long time.

- Feature packing requires to read all feature replicas to memory.

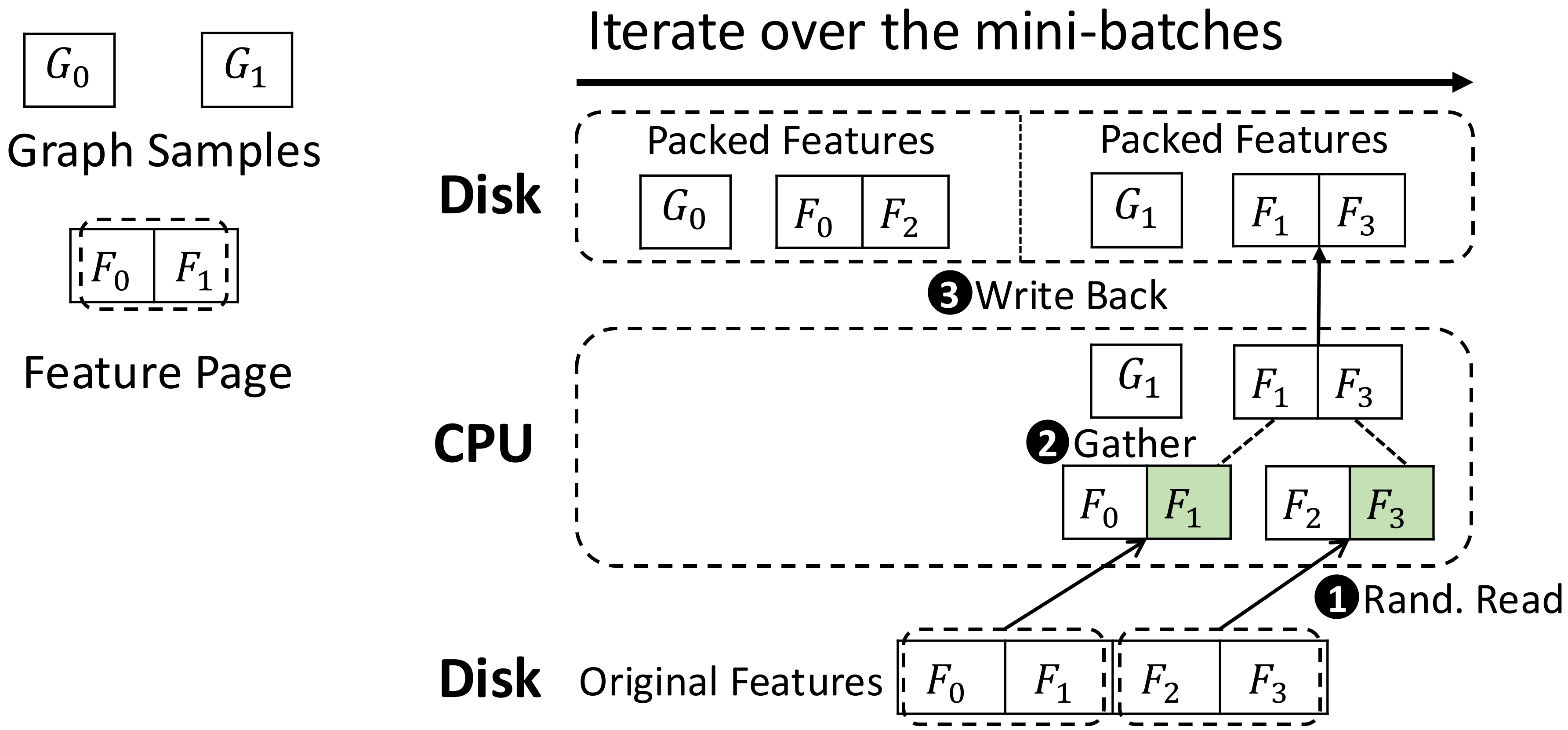


Individual Packing

DiskGNN: Challenges & Solutions

Challenge 3: Feature Packing could also take a long time.

- Feature packing requires to read all feature replicas to memory.

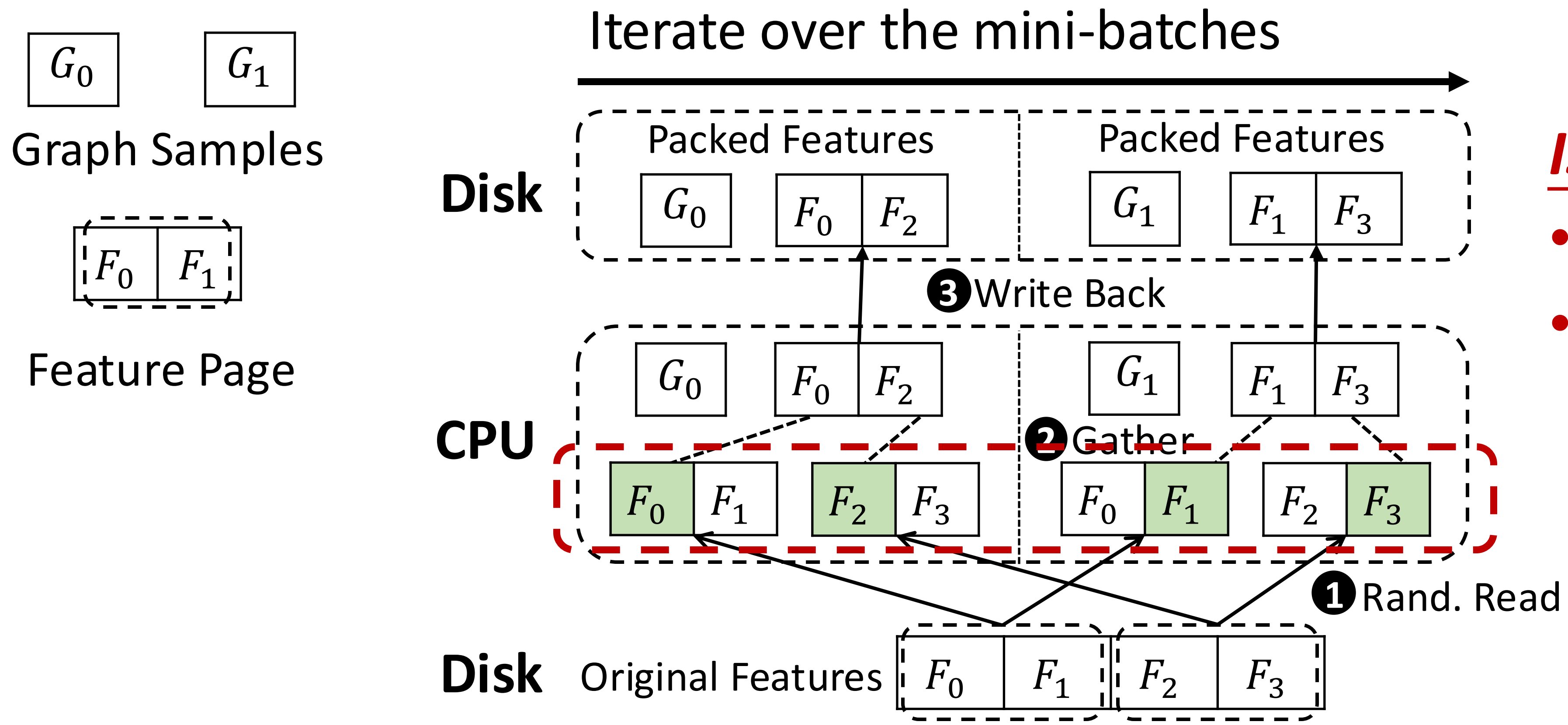


Individual Packing

DiskGNN: Challenges & Solutions

Challenge 3: Feature Packing could also take a long time.

- Feature packing requires to read all feature replicas to memory.



Issue:

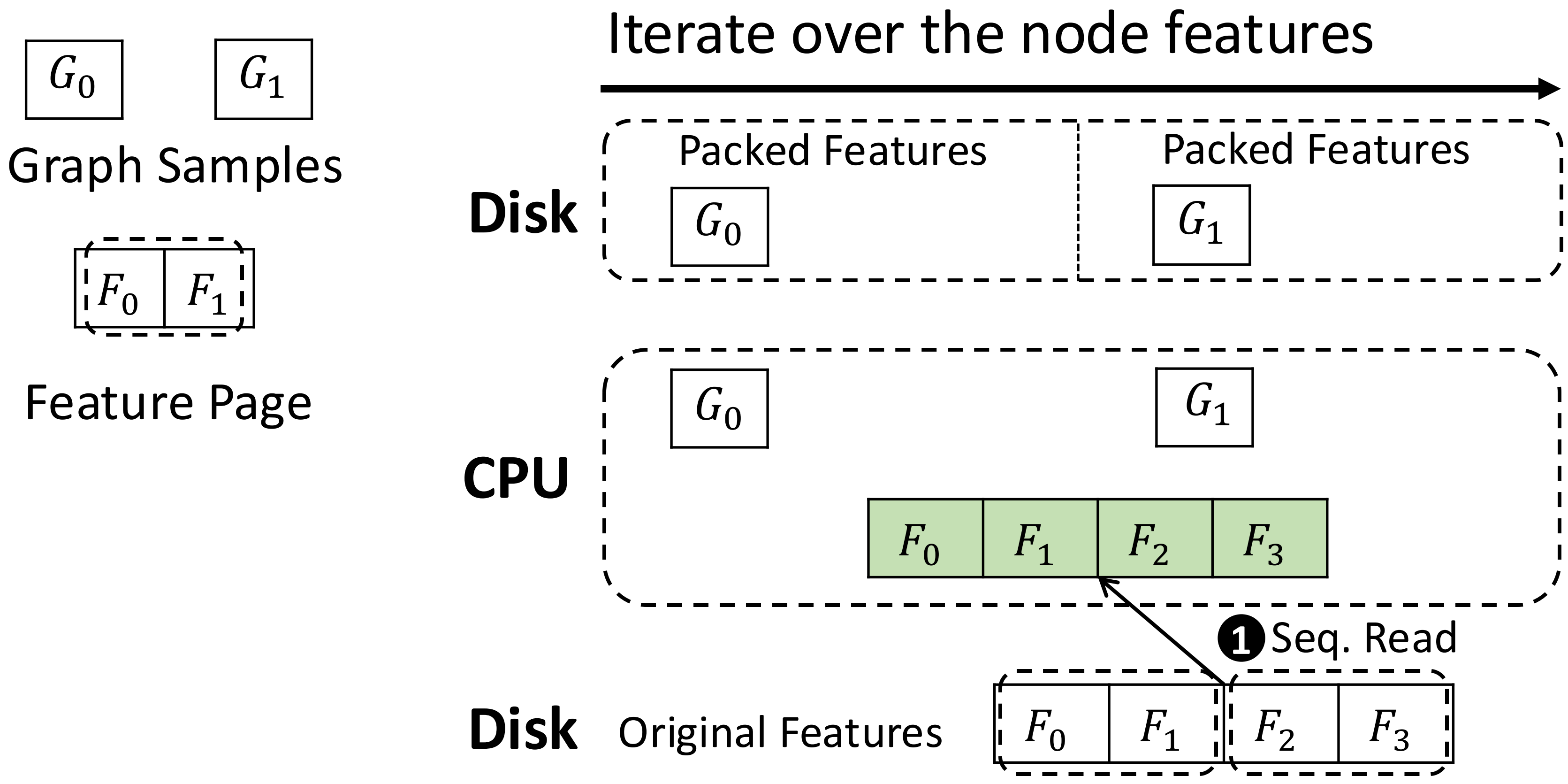
- Read replication.
- Read amplification.

Individual Packing

DiskGNN: Challenges & Solutions

Challenge 3: Feature Packing could also take a long time.

- **Solution: change iteration order from mini-batches to node features.**

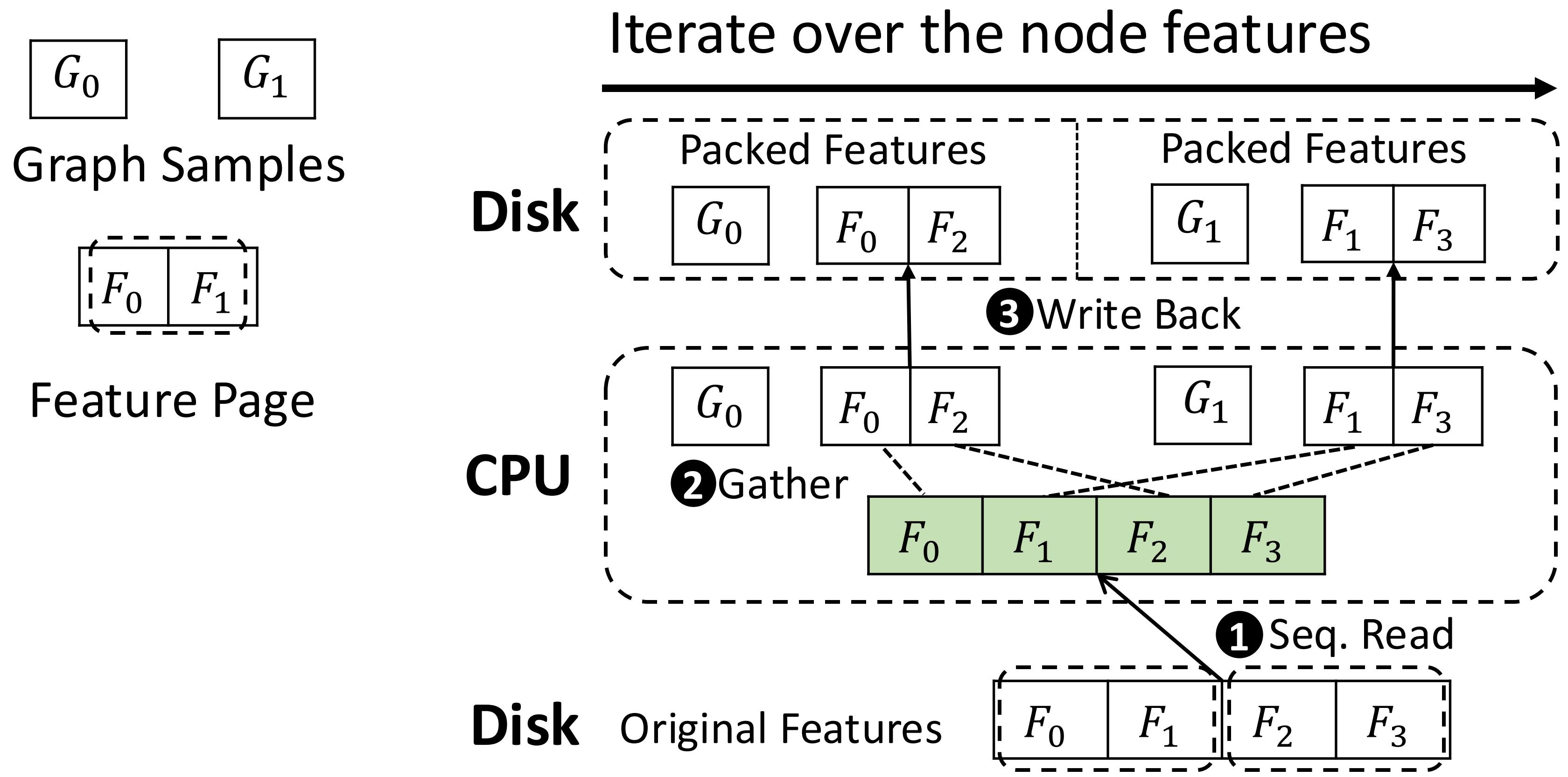


batched Packing

DiskGNN: Challenges & Solutions

Challenge 3: Feature Packing could also take a long time.

- **Solution: change iteration order from mini-batches to node features.**

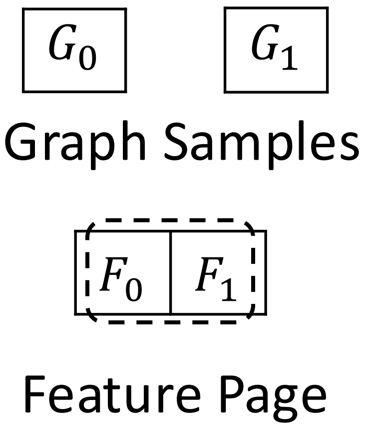


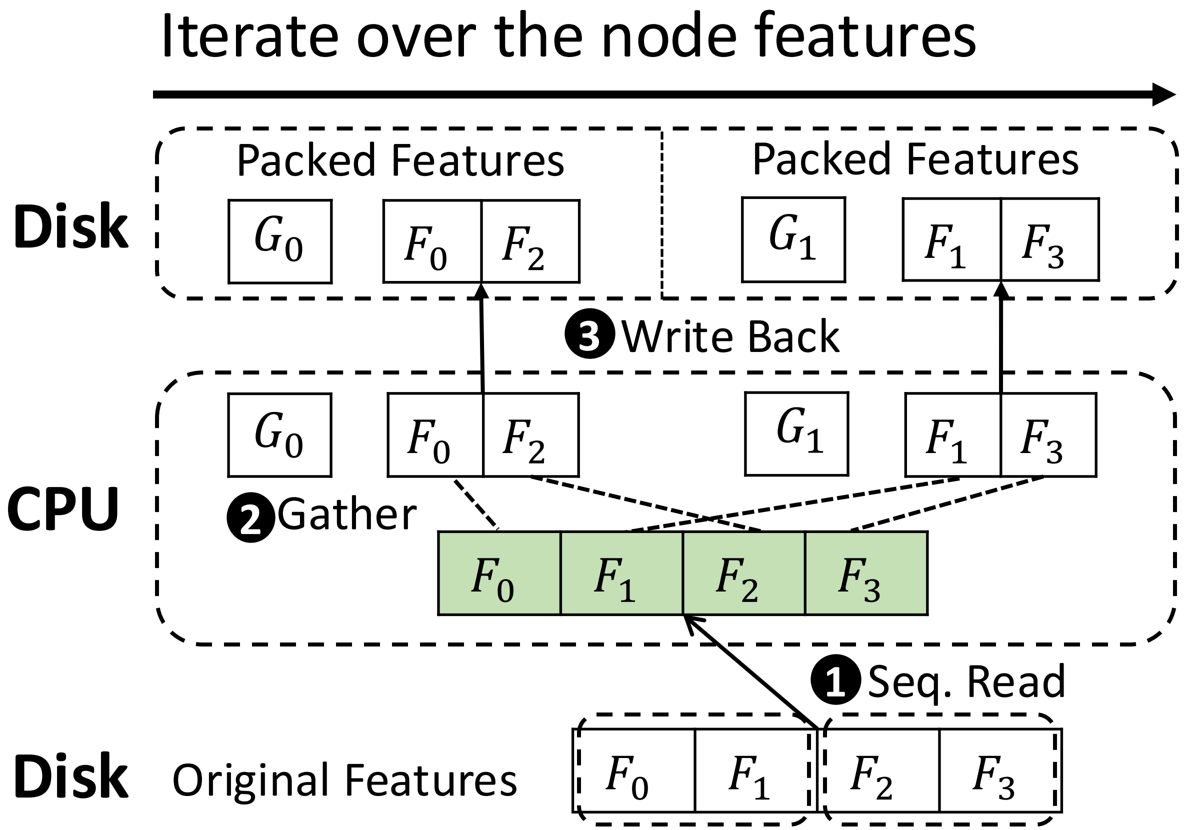
batched Packing

DiskGNN: Challenges & Solutions

Challenge 3: Feature Packing could also take a long time.

- **Solution: change iteration order from mini-batches to node features.**





Pros:

- No read replication.
- No read amplification.

batched Packing

DiskGNN: Other Techniques

- Multi-layer feature assembling with low overhead.
- Asynchronous I/O interface (io_uring).
- Overlapping model training with data movement.
- …

Evaluation

Hardware:

- A single machine with 1 NVIDIA A10 GPU of 24GB memory on AWS EC2.
- DDR4 memory with 25GB/s bandwidth and >10M IOPS.
- 1 NVMe SSD with 3GB/s bandwidth and 625k IOPS.

Baselines:

- Node-wise disk access system: Ginex [VLDB'22].
- Block-wise disk access system: MariusGNN [Eurosys'23].

Datasets:

- Ogbn-Papers100M (78GB), Friendster (90GB).
- MAG240M (145GB), IGB-HOM (158GB).

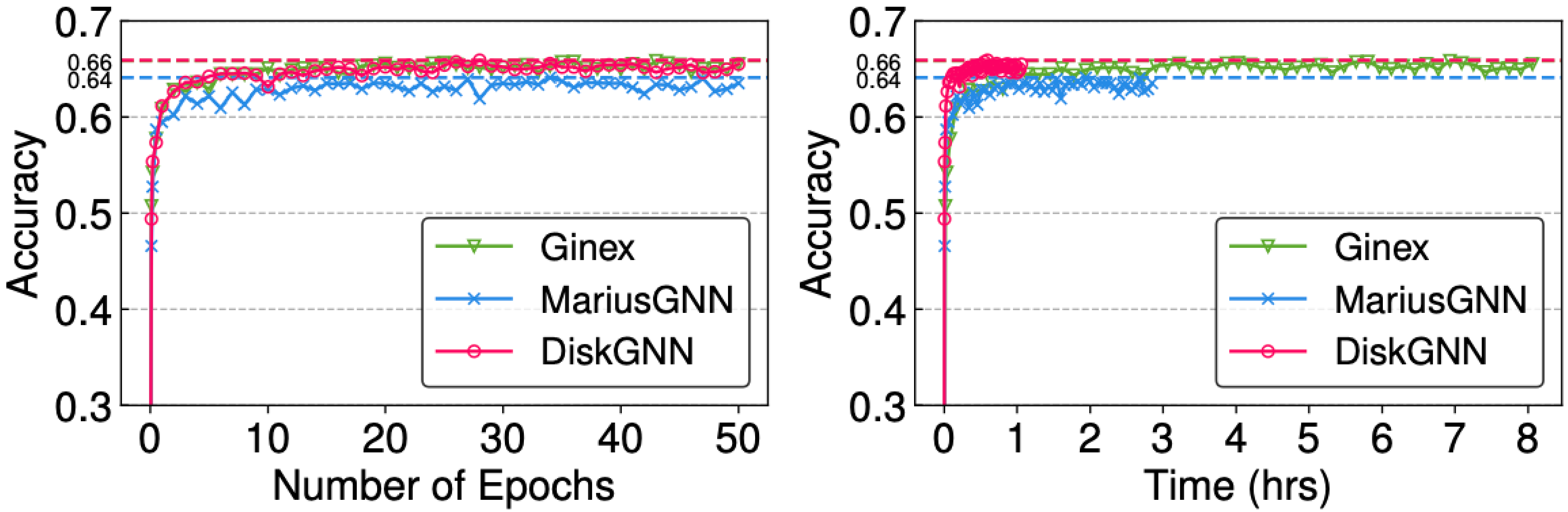
Models:

- GraphSAGE [NeurIPS'17], GAT [ICLR'18].

Evaluation: Model Accuracy & E2E Time

- DiskGNN shows similar convergence rate with Ginex.
- DiskGNN achieves the shortest training time.

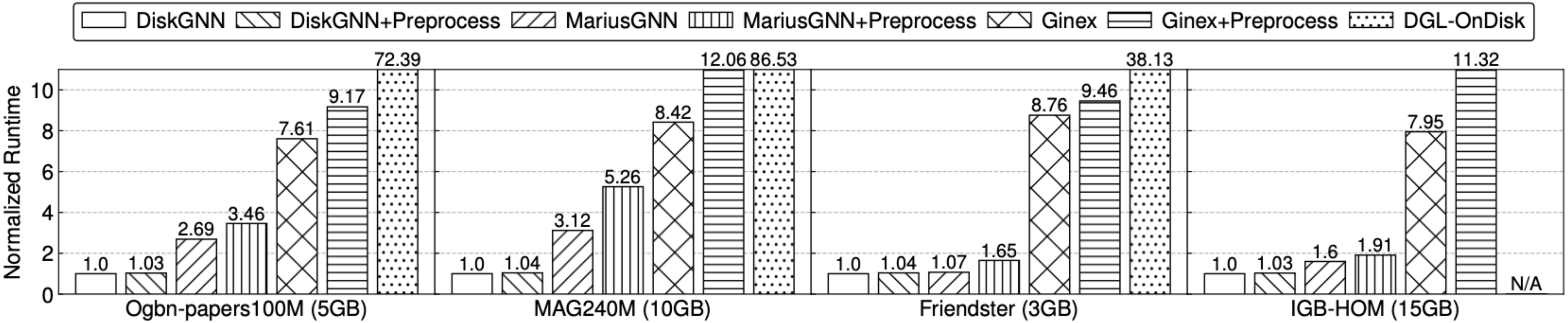
Using 1-epoch of pre-sampled subgraphs



Evaluation: Epoch Time Comparison

- DiskGNN outperforms Ginex by ~8x and MariusGNN by ~2x.
- By batched packing, DiskGNN has a low preprocessing overhead (<5%).

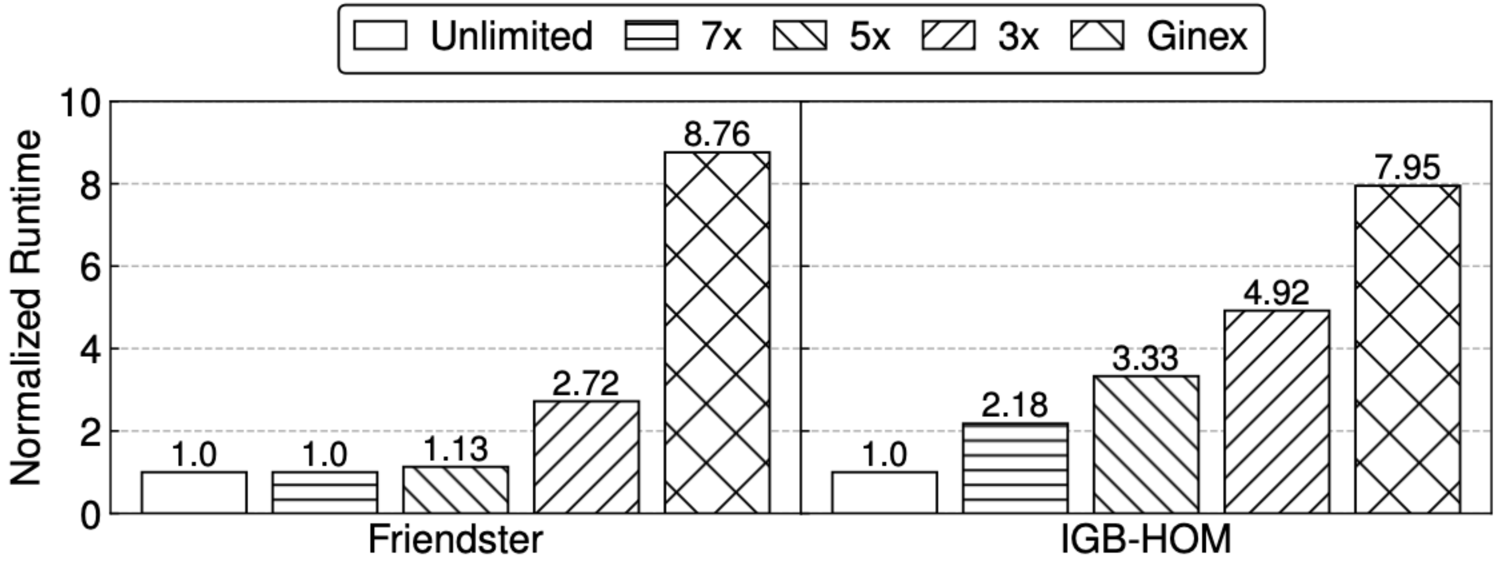
Cache ratio: 10% of whole graph



(a) GraphSAGE

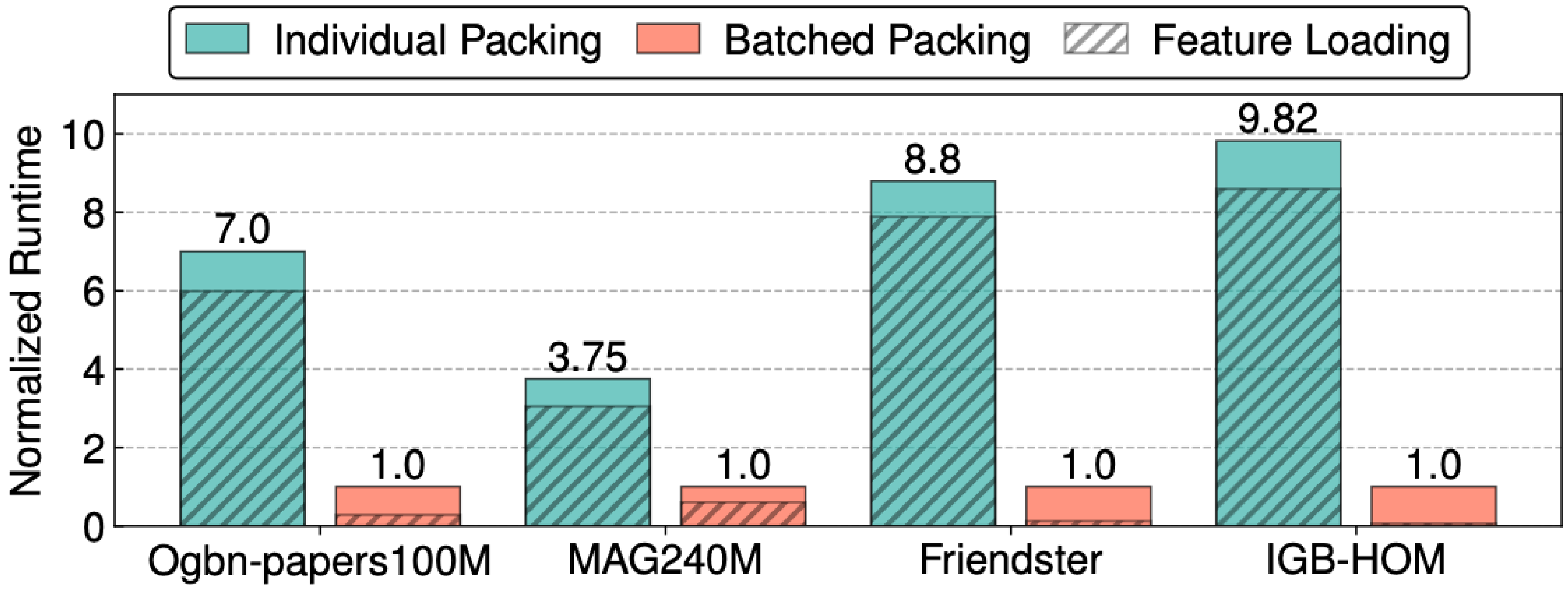
Evaluation: Disk Storage Budgets

- DiskGNN can adapt to different disk storage budgets with adjustable disk cache configuration.



Evaluation: Batched Packing

- Batched packing speeds up preprocessing time by up to 10x.



Takeaway

- **Offline sampling** does not harm accuracy with sufficient mini-batches.
  - Empirically 1-epoch mini-batches are enough for node classification.
- By **observing all mini-batches**, data accesses can be largely optimized.
  - Better cache management to reduce I/O volume.
  - Aligned disk data placement to mitigate read amplification.

Code available at https://github.com/Liu-rj/DiskGNN.

Personal Homepage: https://liu-rj.github.io/ (Renjie Liu), https://yichuan520030910320.github.io/ (Yichuan Wang).