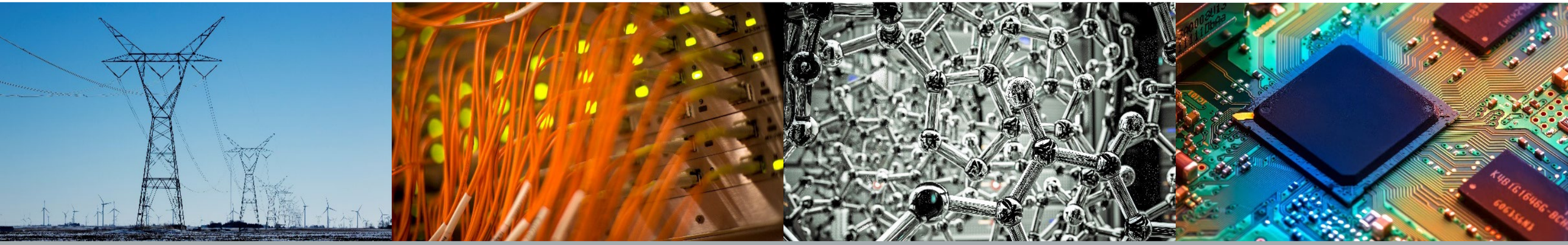
Large Graph Convolutional Network Training with GPU-Oriented Data Communication Architecture

Seung Won Min[1], Kun Wu[1], Sitao Huang[1], Mert Hidayetoğlu[1], Jinjun Xiong[2], Eiman Ebrahimi[3], Deming Chen[1], Wen-mei Hwu[13]

[1]University of Illinois at Urbana-Champaign, [2]IBM Research, [3]NVIDIA Research

August 19th, 2021



ILLINOIS
Electrical & Computer Engineering
GRAINGER COLLEGE OF ENGINEERING

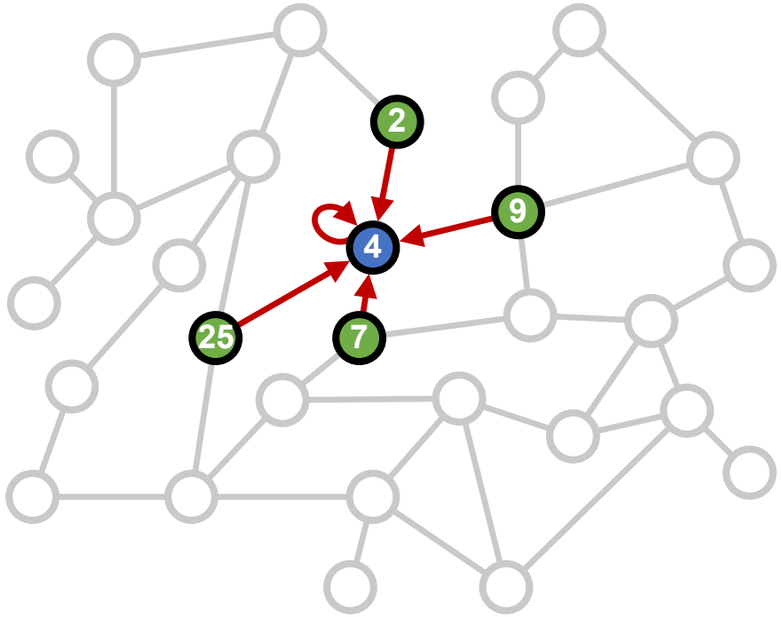
C3SR center for cognitive computing systems research

The IMPACT Research Group

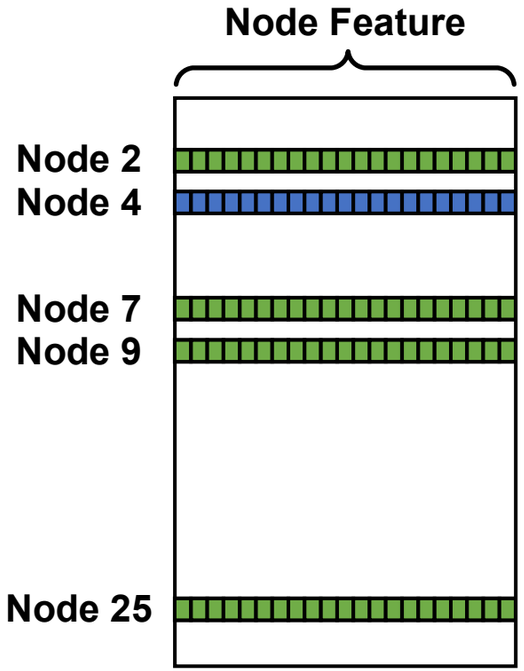
VLDB 2021

Graph Convolutional Network (GCN) Example

- Identify neighboring nodes and aggregate their features
- Problem statement: Transferring sparse data is not simple



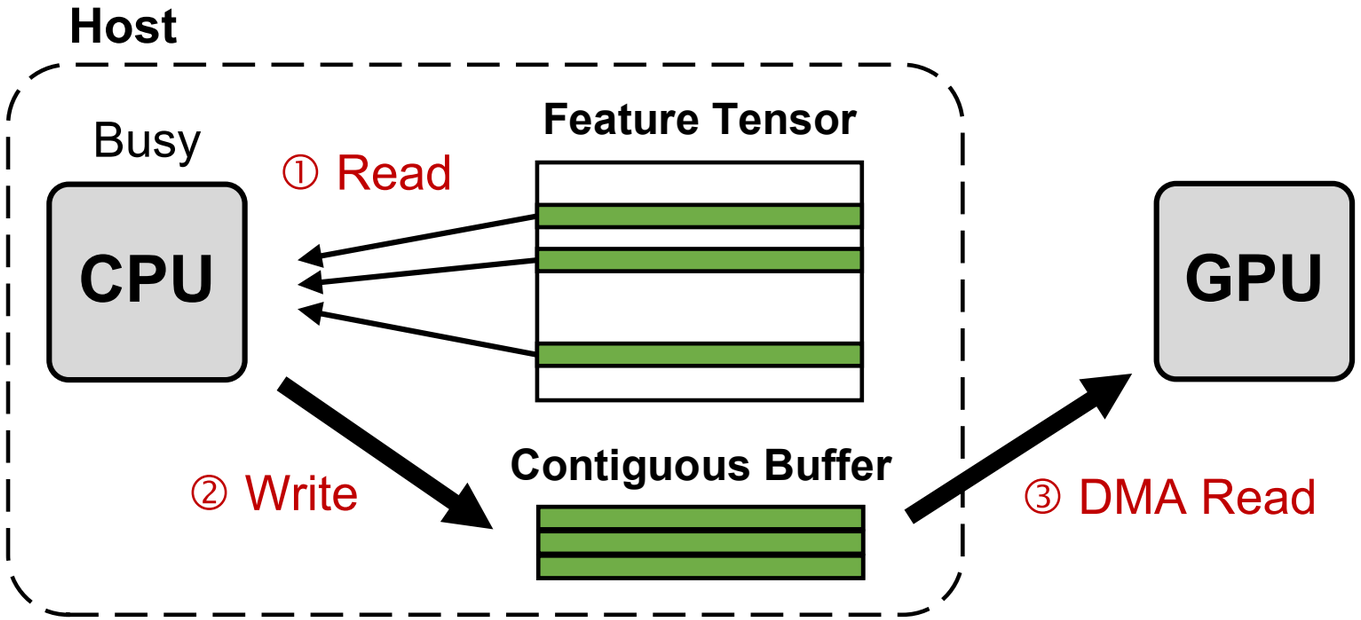
Graph



Node Feature Tensor

Current Method

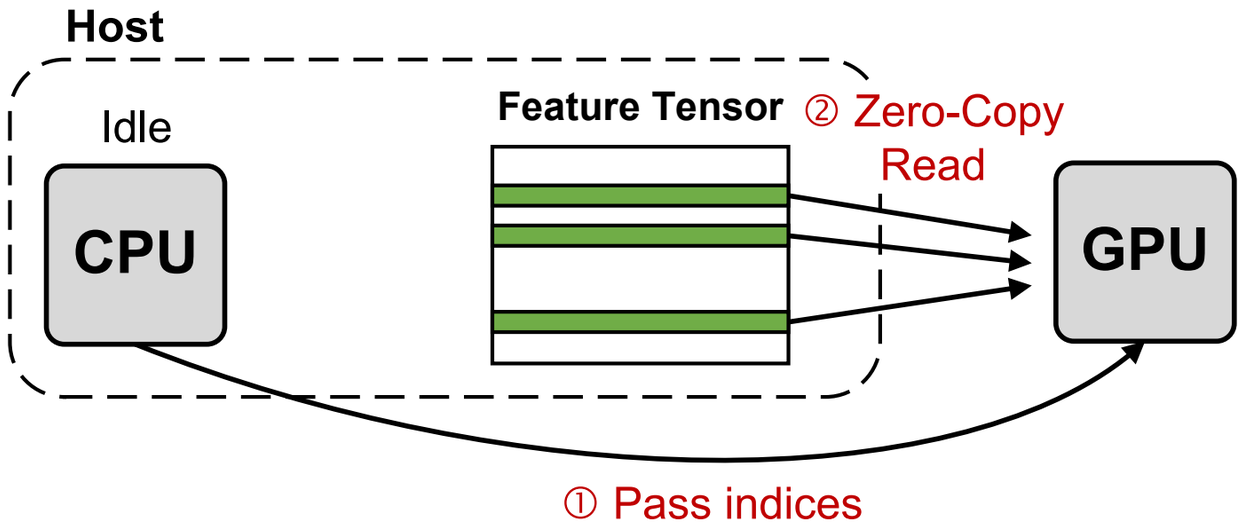
- **Transform sparse data → dense data with CPU [1]**
  - Drawbacks: Long data access latency and waste of host resource
  - To sustain a single PCIe 4.0 x16 link, you consume 25×3=75GB/s



[1] Ying et al., *Graph Convolutional Neural Networks for Web-Scale Recommender Systems,* KDD 2018

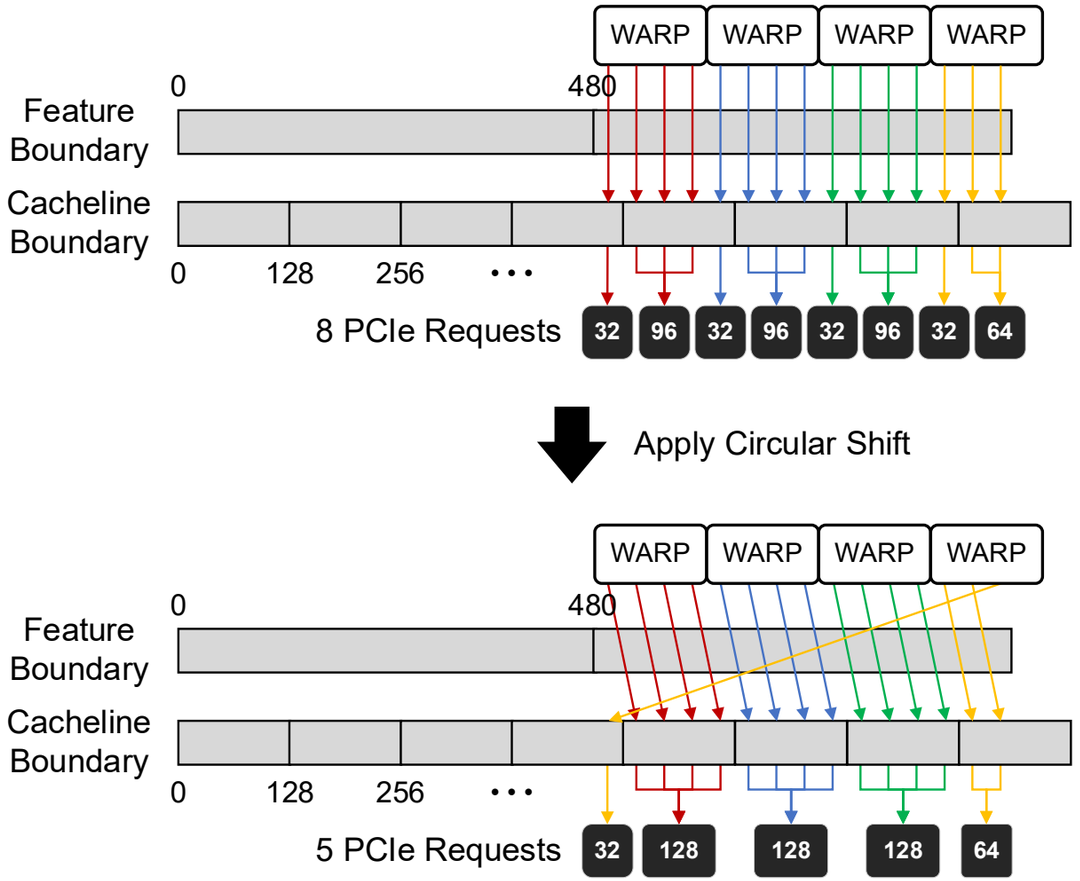
Proposed Method: Zero-Copy

- **GPUs directly access to the targeted node features**
  - Benefits: Less host resource usage (1/3 memory access) and reduced data access latency
  - Questions: Unclear actual performance benefits

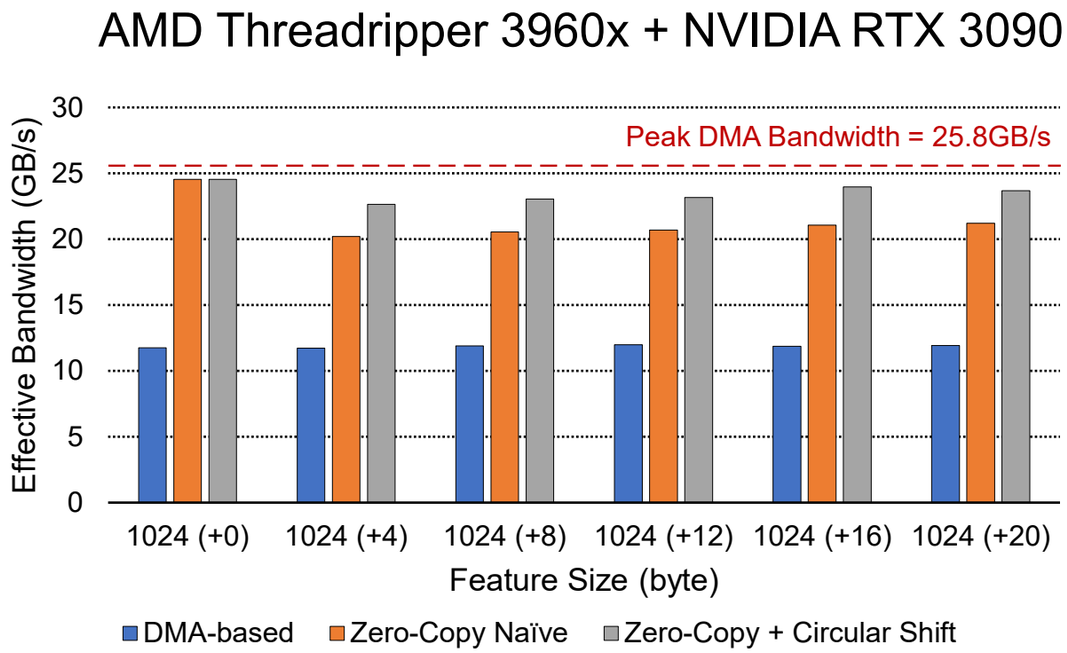


Zero-Copy Performance Optimization 1

- **Aligned PCIe memory read request**



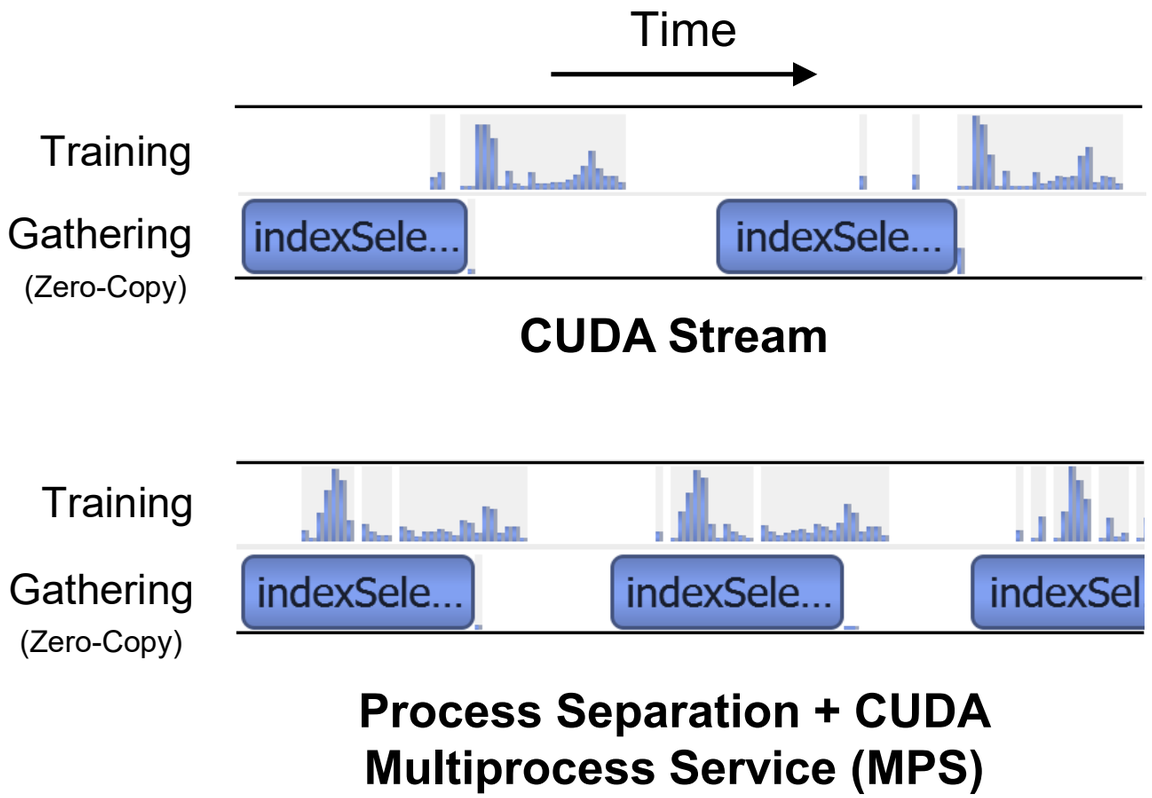
Automatic Alignment Adjustment



Sparse Feature Access Bandwidth

Zero-Copy Performance Optimization 2

- **Asynchronous zero-copy kernel operation**



Set default GPU active thread to X%

↓

Launch main program

↓

Launch zero-copy gathering process

↓

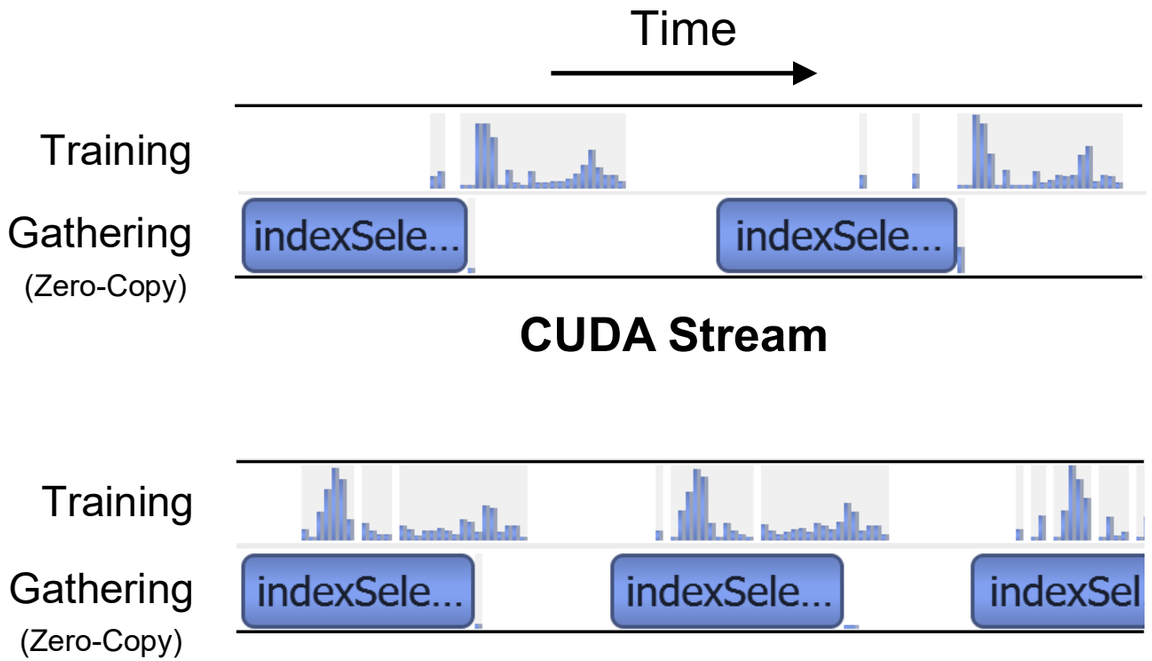
Set default GPU active thread to (100-X)%

↓

Launch training process

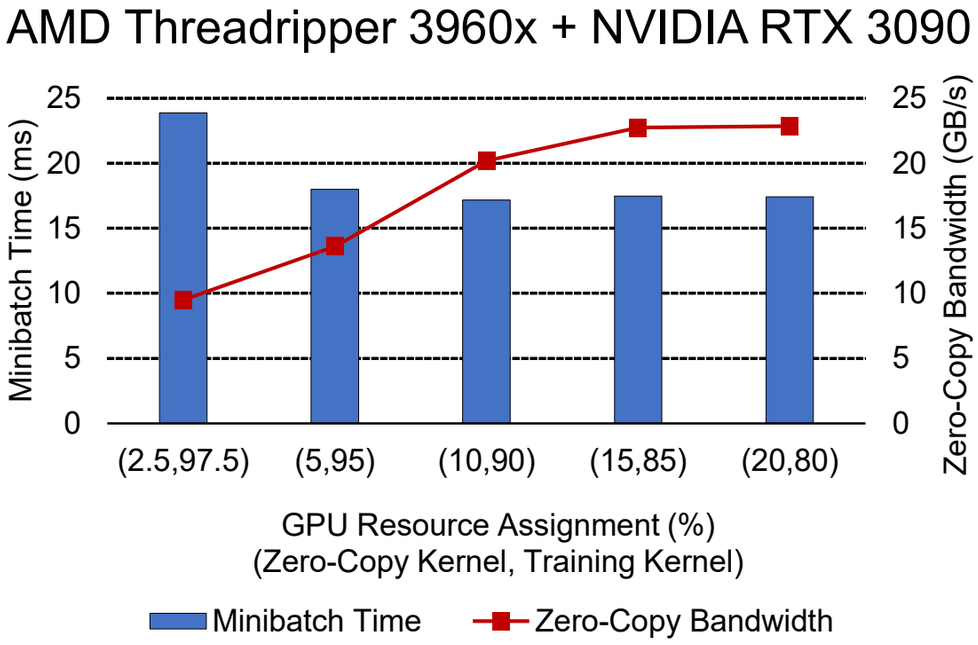
Zero-Copy Performance Optimization 2

- **Asynchronous zero-copy kernel operation**



CUDA Stream

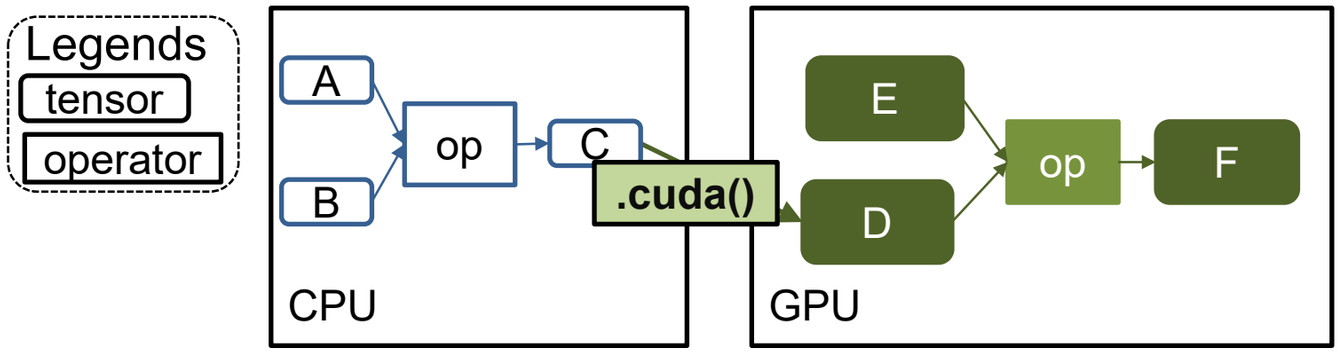
Process Separation + CUDA Multiprocess Service (MPS)



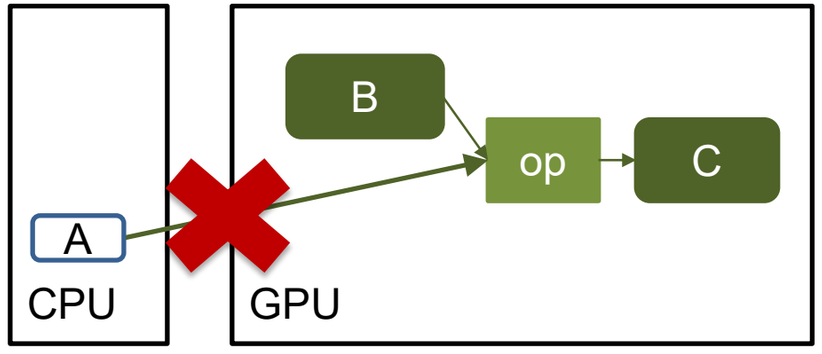
GraphSAGE Training Evaluation

Implementation: Major Challenges

- PyTorch ‘always’ keep data right next to the processing unit
  - Ex) CPU tensor in host memory and GPU tensor in GPU memory
  - No ways for GPU operators to directly access host memory
- Zero-copy tensor is transient
  - Result tensor should be GPU tensor in order to be stored in device memory.



Native PyTorch tensors are bound to devices.



Cross-device access not allowed

Implementation: Unified Tensor

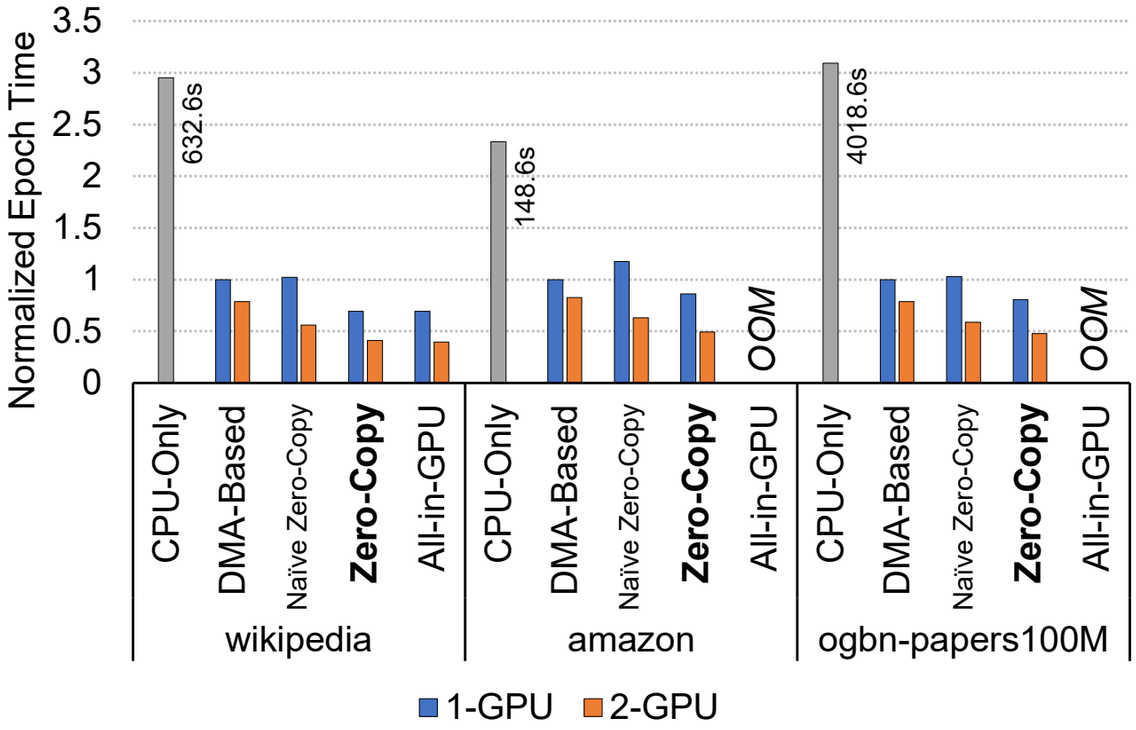
- We want a full-fledged new tensor type, to specify custom optimizations for zero-copy accesses
- Design of the "**Unified Tensor**"
  - Two affinity modes: GPU-affinitive and CPU-affinitive

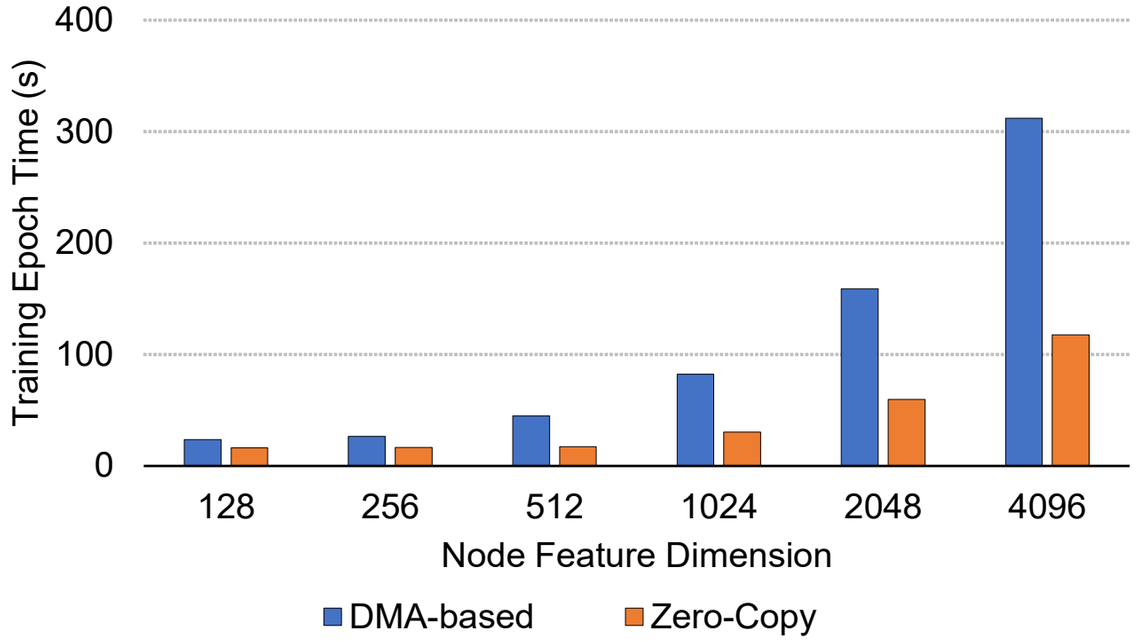
Tensor type	Executor	Storage
CPU	CPU	CPU Memory
CUDA	GPU	GPU Memory
GPU-Affinitive Unified	GPU	GPU Memory
Host-Affinitive Unified	CPU	CPU Memory

  - No actual data movements when the affinity is switched

Evaluations

- Deep Graph Library (DGL) used
- AMD Threadripper 3960x + NVIDIA RTX 3090 x2





Thank You!

Email: min16@illinois.edu
Github: https://github.com/K-Wu/pytorch-direct_dgl
DGL: https://docs.dgl.ai/en/latest/api/python/dgl.contrib.UnifiedTensor.html

ILLINOIS
Electrical & Computer Engineering
GRAINGER COLLEGE OF ENGINEERING