THE UNIVERSITY of EDINBURGH
informatics

Quiver: A distributed graph learning system with workload-aware data and task management

Zeyuan Tan[1], Xiulong Yuan[1,4], Guo Li[2], Peter Pietzuch[2], Kai Zeng[3], Luo Mai[1]

University of Edinburgh[1], Imperial College London[2], Alibaba[3], Tsinghua University[4]

Graph Learning (GL)

Graph data

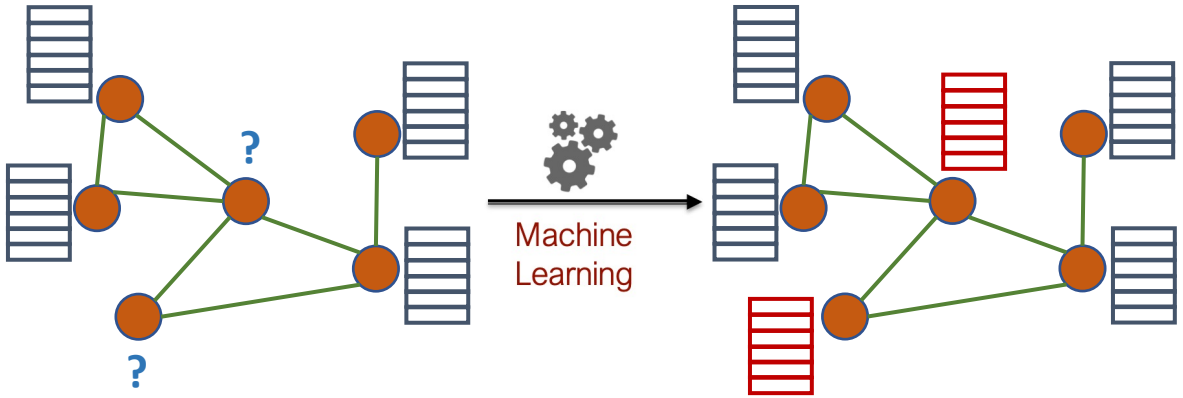
- Topology (nodes + edges)
- Features

Graph Learning (GL) models

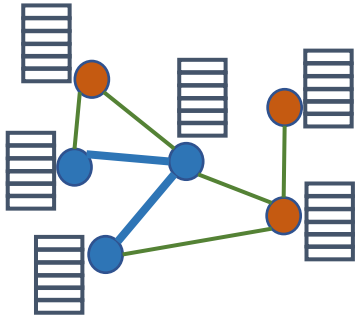
- Graph data + deep learning models
- Recommendation, CV, NLP

Large-scale deployments

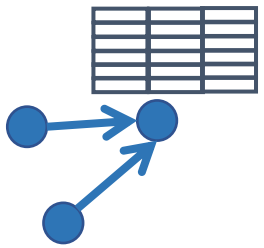
- Model zoo: PyG, Amazon DGL
- Pinterest, Google, Alibaba, …



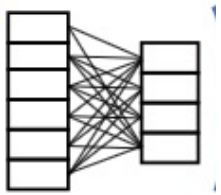
Prediction made by GL models



Sampling



Feature aggregation



Training

Distributed GL systems

Data management

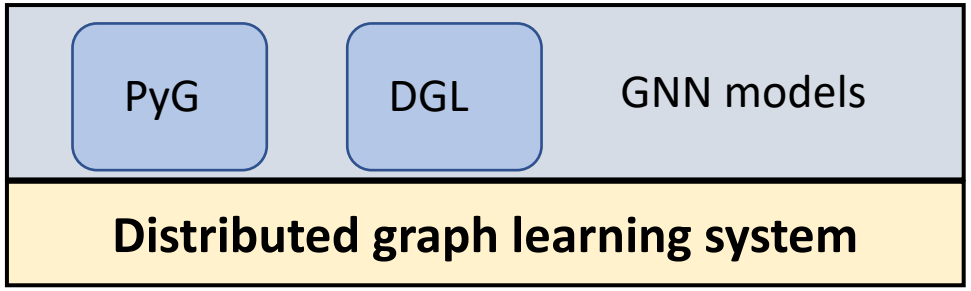
- Graph topology, features, models
- Decision: Partitioning, replication

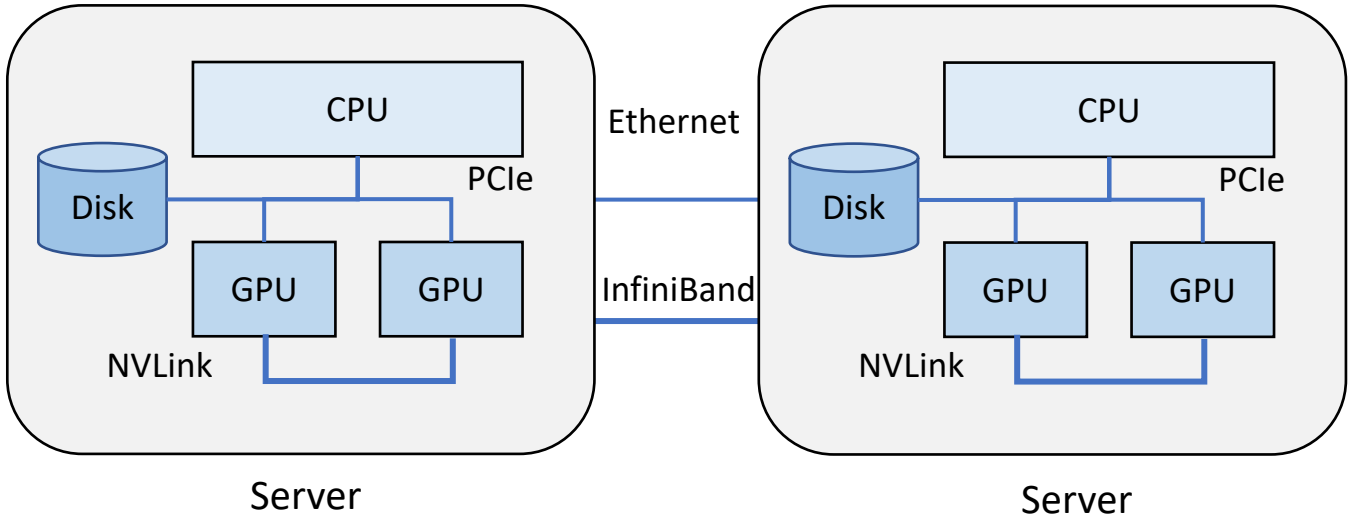
Task management

- Sampling, feature aggregation, training
- Decision: CPU, GPU or both?

Decisions are workload-dependent

- Model (e.g., NN depth)
- Data (e.g., feature size)
- Hardware (e.g., NVLink)





Problems of existing distributed GL systems

Production systems: Amazon DistDGL and AliGraph [VLDB'20]

- **Fixed** data management (e.g., Metis)
- **Fixed** task management (e.g., sampling on CPU, training on GPU)

Research prototype: P3 [OSDI'21]

- Specific to certain GL models
- Under-utilise GPUs (e.g. <30%)

Issues:

- Data access bottleneck
- Computation bottleneck

How to design a **generic workload-aware** distributed GL system?

[1] DistDGL: Distributed Graph Neural Network Training for Billion-Scale Graphs, 2020
[2] AliGraph: A Comprehensive Graph Neural Network Platform, VLDB 2020
[3] P3: Distributed Deep Graph Learning at Scale, OSDI 2021

Quiver overview

Workload-aware data management

- **GPU-based distributed computation** of data access-probability (i.e., hot/cold vertices)
- Data partitioning based on **access-probability**
- Data replication based on **available networks and memory**

Workload-aware task management

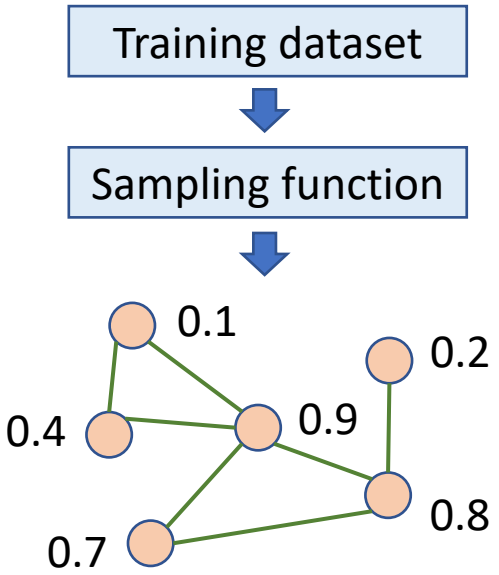
- Partition a training batch into **micro-batches**
- Assign micro-batches to **all processors** (i.e., CPUs + GPUs)
- **Auto-tune** micro-batch sizes and assignment

Generality and compatibility

- Provide **unified tensors** to support PyG and DGL models

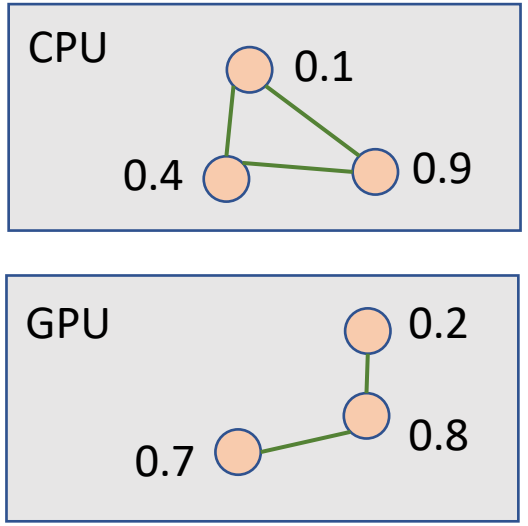
Workload-aware data management

Compute access probability with multi-GPU



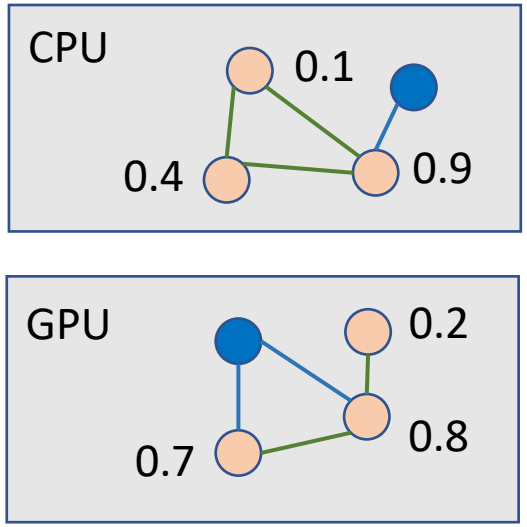
GL data with **access-probability**

Assign graph partitions to CPUs/GPUs



Balanced aggregated access probability

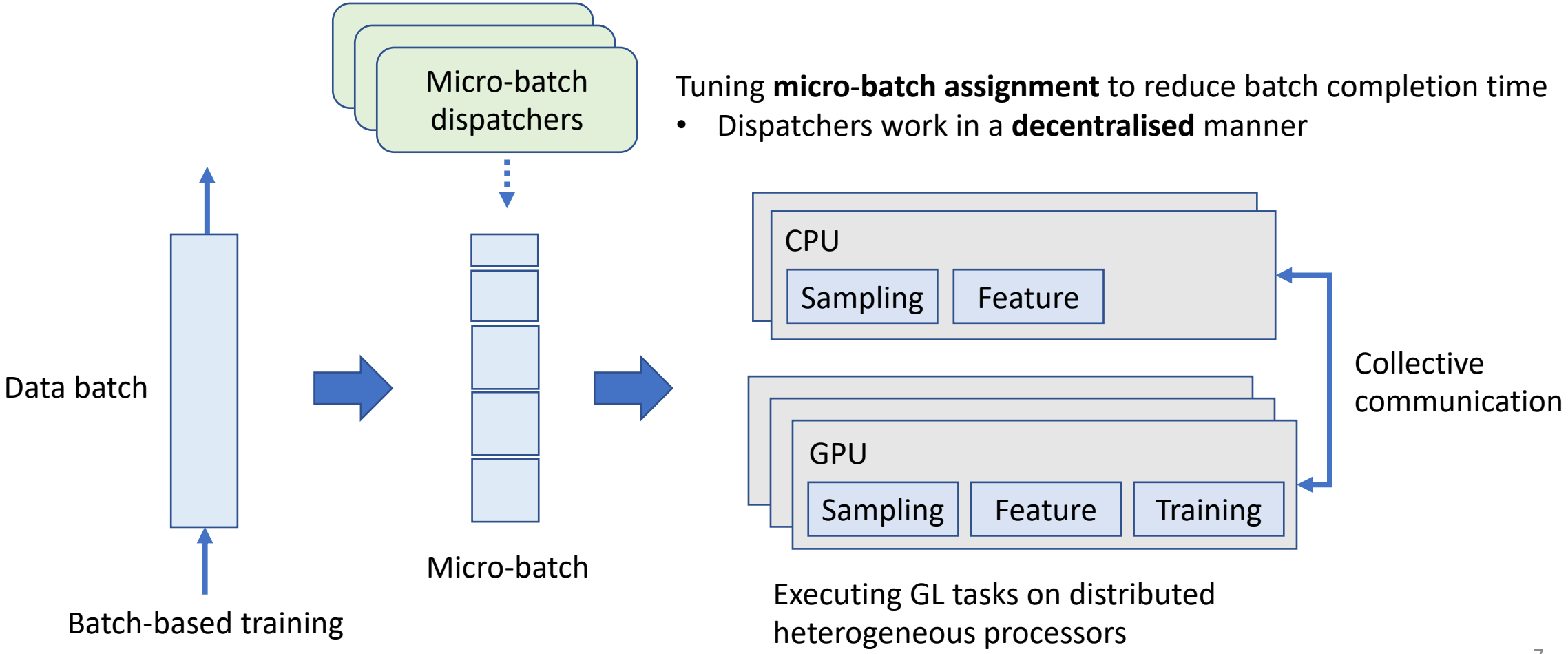
Replicate graph data based on available networks and memory



Network and memory:

- NVLink, InfiniBand, NUMA
- CPU memory, GPU memory

Workload-aware task management

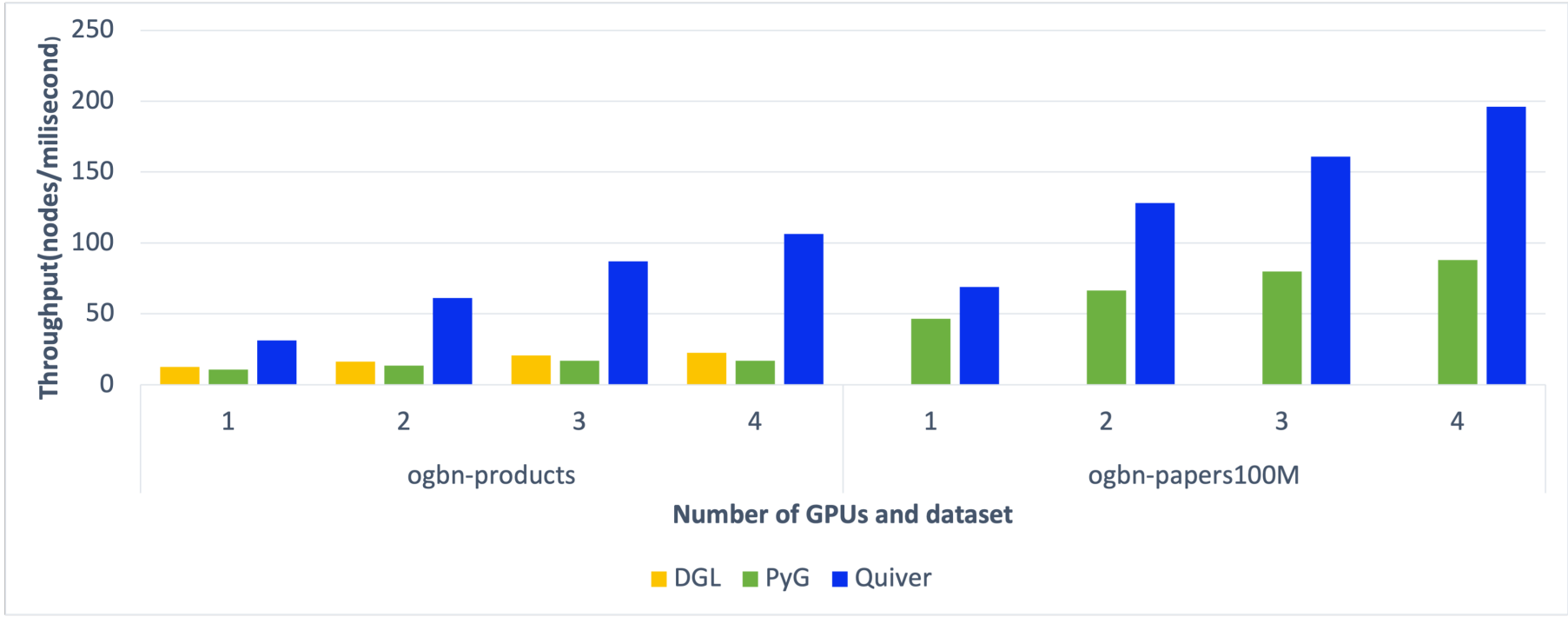


Tuning **micro-batch assignment** to reduce batch completion time

- Dispatchers work in a **decentralised** manner

Multi-GPU Experiments

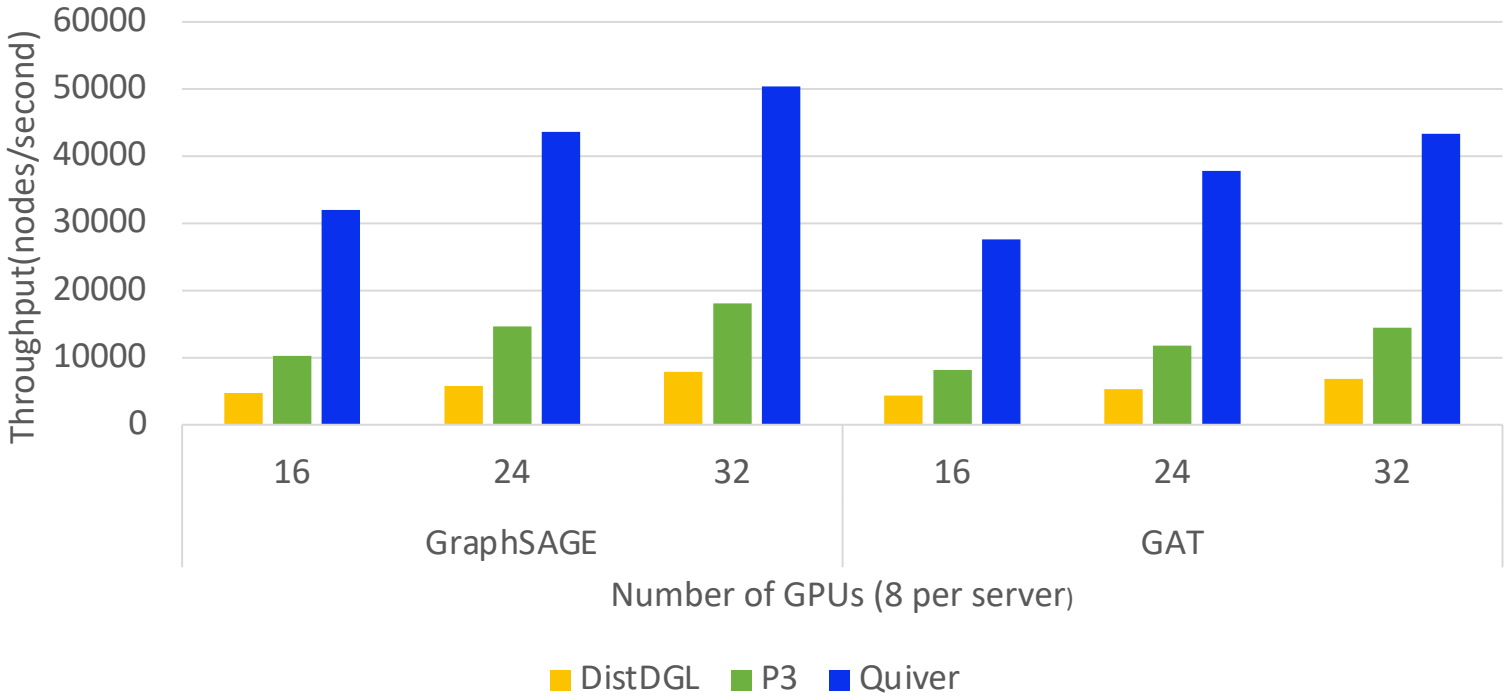
Dataset: (1) ogbn-products: 2.5M nodes, 62M edges; (2) ogbn-papers100M: 110M nodes, 1.6B edges



Quiver incurs **low overhead** in analysing workload, and out-performs DGL and PyG by **up to 5x**

Cluster Experiment

Meg240M dataset: 240M nodes, 1.7B edges (The biggest GL workload in open benchmarks)



Quiver can effectively **minimise communication cost** and **utilise distributed heterogeneous processors**

Summary

- Quiver: A workload-aware distributed GL system
- Superior performance (Up to 10x over state-of-the-art)
- Open-source GitHub project: **quiver-team/torch-quiver**
- Fast growing community Quiver

Project link:

Thank You — Any Questions?

Zeyuan Tan
zeyuan.tan@ed.ac.uk