Motivation

Mitigate the memory bottleneck, and enable large-scale GNN training within a single GPU

- **Challenge**: Easily exceeding GPU memory capacity.

  ![Memory Capacity](image1.png)

  To work around the memory capacity bottleneck, prior work explored both algorithmic (sampling[1]) and system optimizations (DGL[2], PyTorch Geometric[3], and NeuGraph[4]).

- **Sampling**
  - **Pros**: Sample neighbors to compute the feature for a given node/subgraph.
    - reduce the sampling rate
    - reduce the number of neighbors participating in aggregation
  - **Cons**: May cause loss of important neighbor information that hurts the final model accuracy.

- **System optimizations**
  - **Pros**: Support convenient and highly efficient graph operation primitives (e.g., aggregators) in terms of compute and memory efficiency.
  - **Cons**: GNN training can still run out of memory as more advanced configuration, especially when using more memory intensive aggregators.

Overview

Betty introduces two novel techniques, redundancy-embedded graph (REG) partitioning and memory-aware partitioning.

![Overview Diagram](image2.png)

**Design**

- **Batch-Level Partitioning**: reduces the memory consumption via the batch-level partitioning and using both CPU and GPU memory to enable training of advanced GNNs on single GPU.
- **Traditional mini-batch training**: reduces the number of redundant node introduced by the partition of multi-level bipartite structure.
- **Micro-batch training in Betty**: reduces the number of redundant node introduced by the partition of multi-level bipartite structure.

**Partitioning the Multi-Level Bipartite for Micro-batch GNN Training**

Dividing each batch into $K$ micro-batches, each micro-batch is a hierarchical bipartite that is a subgraph of the original bipartite.

![Partitioning Diagram](image3.png)

**Redundancy Reduction**

Reduce the number of redundant node introduced by the partition of multi-level bipartite structure.

**Redundancy-Embedded Graph (REG) Construction and Partition**

- **In REG**
  - output node $u$
  - edge weight $= \#$ of shared neighbors
  - output node $v$

**Reducing Maximal Memory Footprint**

- Memory-aware Partitioning
- Partition memory estimation

Experiment Results

Betty breaks the memory capacity constraint, reduce the peak memory consumption up to 48.3%.

- **Dataset**: Cora, Pubmed, Reddit, ogbn-arxiv and ogbn-products
- **Baselines**
  - We evaluate the scalability of GNN training, Aggregator, Hidden size, Number of model layers, Fanout
  - We use three common graph partition algorithms: range partition, random partition, and Metis[5]. (The partition is applied on the IDs of output nodes.)
- **Enable advanced and efficient GNN training with hybrid CPU-GPU memory**
- **A transparent solution that does not require any hyperparameter tuning and preserve model convergence.**

![Experiment Results Diagram](image4.png)

- **Compared with other graph partition methods, Betty can:**
  - reduce max memory consumption by 48.3% and 37.7% on average.
  - reduce the node redundancy by up to 49.2% and 28.4% on average.
  - improve computation efficiency by 20.6%, 21.1%, and 22.9%, when the number of batches increases (number of redundant nodes increases).

**References**

[1] Da Zheng, Xiang Song, Shuangyan Yang, Minjia Zhang, Wenqian Dong, and Dong Li. Betsey: Enabling Large-Scale GNN Training with Batch-Level Graph Partitioning and Tiered Memory. 2020. (preprint)


