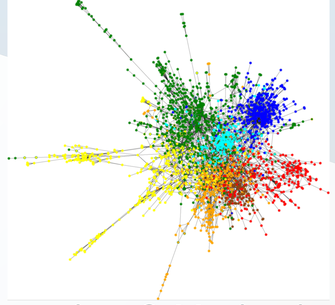


Analyzing the Performance of Graph Neural Networks with Pipe Parallelism

Matthew T. Dearing & Xiaoyan (Angela) Wang

torchgpipe

Overview & Motivation

- Many datasets in the real world can be structured as *graphs* or *networks*, such as social groups, chemical interactions, protein models, and phase transitions.
- Graph neural networks (GNN) generalize neural networks for deep learning on arbitrary irregular graphs.
- Because current GNN models do not scale well and datasets of interest continue to increase in complexity and size, improving computational efficiency is important to further research applicability.
- We study parallelizing GNNs using *existing* tools and frameworks known to be successful for traditional deep learning.

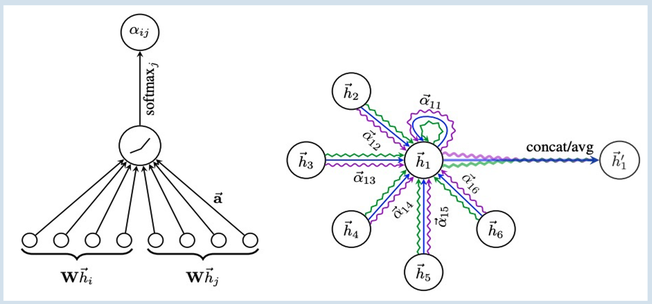
Graph Neural Networks

Input is a graph composed of *nodes* with features, such as class labels. One type of task is to classify each node using the information from the connecting *edges*.

- GNNs build a neural network atop a simultaneous message passing paradigm that aggregates neighborhood information.
- Each NN layer is a function of the previous layer *and* the input graph.

Graph Attention Network (GAT)

- GAT (Veličković et al. 2017) is a GNN with *attention layers* on top of graph convolutions.
- The entire network consists of two GAT layers with a multi-head attention layer to compute attention coefficients for the neighboring nodes, which is used to propagate information through the network.



The PubMed Dataset

- A citation network dataset with 19,717 nodes and 44,338 edges. Each node has 500 features, representing unique words in a single paper (node).
- Three node classes: Diabetes Mellitus Experimental, Diabetes Mellitus Type 1, Diabetes Mellitus Type 2.

Graph Learning Frameworks

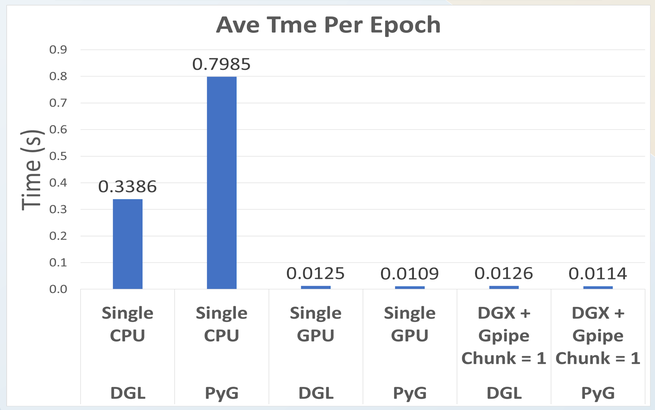
Deep Graph Library (DGL): Tensor framework-agnostic Python implementation of graph neural network model families.

PyTorch Geometric (PyG): PyTorch extension library for geometric deep learning.

GPipe for Parallelizing DNNs

- Introduced by Google Brain in 2018 to enable efficient training of large, memory-consuming models on current accelerator architectures.
- Splits NN layers across devices and microbatches the dataset.
- *torchgpipe*: A GPipe implementation in PyTorch.

Benchmark Training: Single Devices



As expected, single GPU performs much faster than single CPU for both PyG and DGL frameworks.

No significant performance improvement using DGX (4 GPUs) and GPipe with "chunk size" = 1 (i.e., no micro-batching) compared to single GPU.

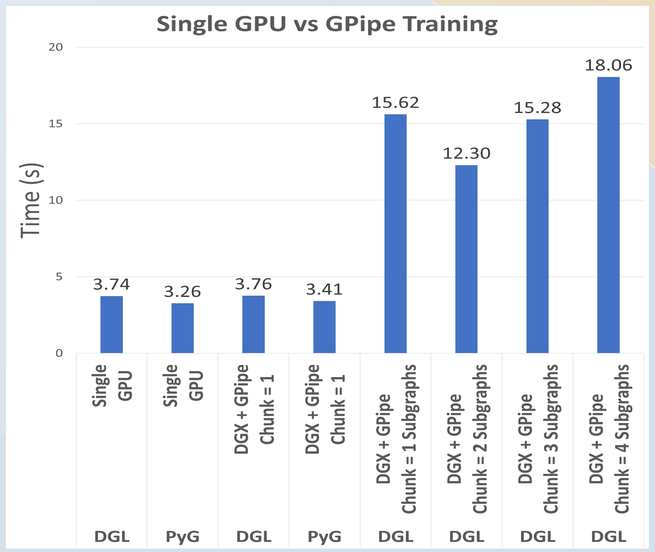
Refactoring the GNN into a Sequential container that only passes a single argument required extracting the graph nodes and embedding features into a tuple of tensors so that the graph could be re-built in the convolutional layers.

Results: Training time increased

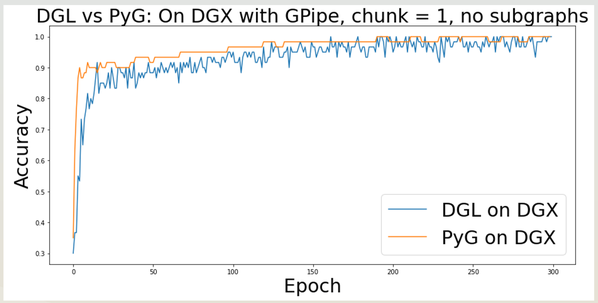
As we increase the number of GPipe chunks to enable micro-batching, the training time increases.

The implementation of GPipe for GNN through a Sequential model required re-constructing (on the CPU!) the micro-batched data as *subgraphs* for each GNN layer.

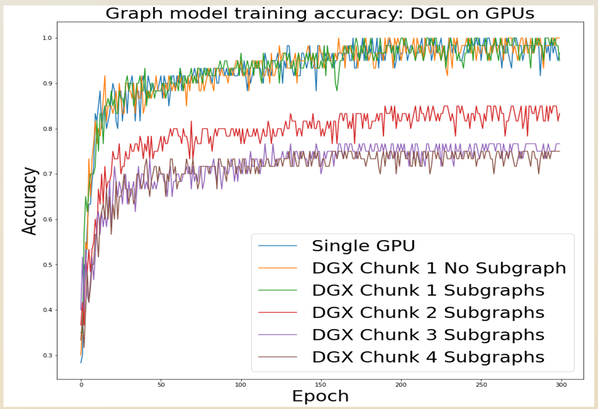
The increased time is attributed to re-constructing the graphs within each chunk and moving data across devices.



Results: Accuracy degraded



DGL and PyG implemented on DGX with GPipe without micro-batching offer similar performance.



GPipe micro-batches the graph nodes *sequentially*, from which the subgraphs are generated. Significant information loss resulted from ignored edge relationships across the batches.

DGL and PyG provide sophisticated single graph micro-batching solutions, but GPipe would not use these batches.

Future work

- Implement other GNN layer models, such as GraphConv, with these frameworks.
- PubMed is too small. Experiment with massive datasets: *Can performance over single GPU increase on extremely large graphs?*

Acknowledgements

Thank you to Prof. Ian Foster, Prof. Rick Stevens, and Peng Ding for the guidance and feedback on this paper. We also thank the DGX team, Daniel Murphy-Olson and Ryan Aydelott, and the Computing, Environment, and Life Sciences directorate at Argonne National Laboratory.