Adaptive Load Balancing for Parallel GNN Training

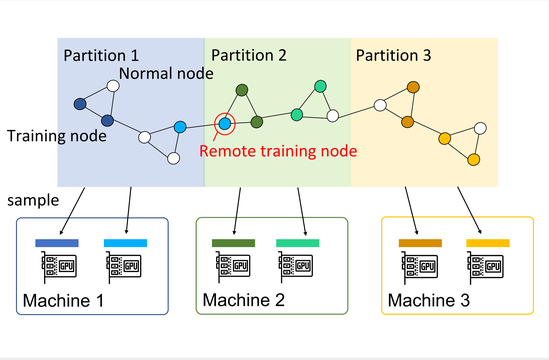
Qidong Su, Minjie Wang, Da Zheng, Zheng Zhang

Abstract

The recent emergence of demand for running Graph Neural Networks (GNNs) on giant real world graphs requires more scalable system designs. Due to the sparse and irregular connections a graph has, parallel GNN training encounters the problem of load imbalance among workers. In this paper, we show that previous techniques based on graph partitioning is insufficient to address the load imbalance caused by GNN sampling algorithms. We thus propose a two-stage strategy to balance the workload adaptively during training. Our evaluation shows that the strategy effectively produces more balanced workloads which accelerates the training by 25%.

Background

As real world graphs can be gigantic, i.e., consisting of billions of nodes or edges, it is critical to support training GNNs at scale. To parallelize the training procedure, current GNN systems adopt a synchronous data parallelism approach: each worker performs sampling and training in parallel and synchronizes model parameters before the next iteration.

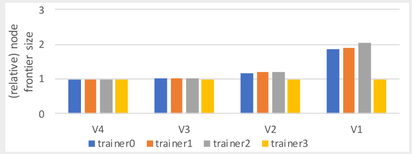


Parallel GNN Training

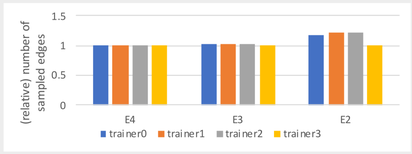
Challenges

However, due to the sparse and irregular connections a graph has, the sample sizes among workers can vary drastically, causing severe workload imbalance. Existing systems such as DistDGL leverage off-the-shelf graph partitioning algorithms to address the problem.

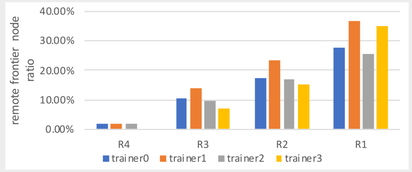
The approach based on graph partitioning algorithms are not sufficient to balance the GNN workload among trainers. Many graph partitioning algorithms aim at minimizing the number of cross-partition edges or the number of replicated nodes while balancing the number of nodes and edges in each partition. However, the workload balance of parallel GNN training is characterized by a different set of metrics including the number of nodes/edges and the remote nodes ratio of each sampled subgraph and thus demands different solutions.



Imbalanced number of nodes in each layer among trainers



Imbalanced number of edges in each layer among trainers



Imbalanced remote nodes ratio in each layer among trainers

Proposed Method

Two-stage load balancing strategy

Stage 1: Before training, we employ the graph partitioning algorithm in DistDGL to generate static graph partition scheme that minimizes the cut edge set and meanwhile roughly balances the number of nodes and edges among partitions.

Stage2: During training, we keep track of the training time of all trainers and adjust their workload accordingly. The basic Idea is to increase the workload of trainers that finish one iteration faster while do the opposite for those that are slower.

Algorithm 3 Training with adaptive batch size tuning

input total batch size B, number of trainers N, rank of the current trainer i, tuning period P

$\mathbf{W} \leftarrow [1, 1, \ldots, 1]$, $B_i \leftarrow \frac{B}{N}$

loop

Perform P training iterations and record iteration time T_i

Synchronize across trainers and get their recorded time

$\mathbf{T} \leftarrow [T_1^{-1}, T_2^{-1}, \ldots, T_N^{-1}]$

$\mathbf{W} \leftarrow \mathbf{W} \odot \mathbf{T}$

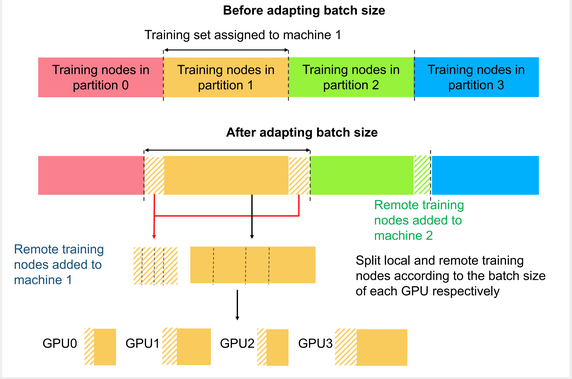
$\mathbf{W} \leftarrow \frac{1}{\sum_{j=1}^{N} W_j} \cdot \mathbf{W}$

$B_i \leftarrow W_i \cdot B$

Re-assign training nodes.

end loop

After adjusting the batch size of each trainer, we then need to decide the assignments of training node set. Because training nodes that belong to partitions on other machines require network requests to perform neighbor sampling, it is important to balance them among trainers.



Locality-aware training set re-assignment

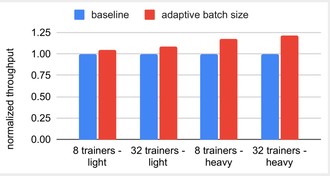
Evaluation

Setup

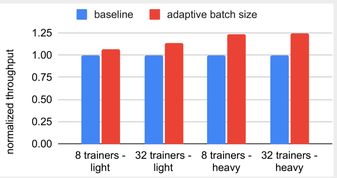
Dataset: Open Graph Benchmark (ogbn-products, ogbn-papers100M)
Hardware: 4 * AWS EC2 p3dn.24xlarge instances with 8 V100 GPUs on each instance
Model: GraphSAGE
Hyper-parameters:

Dataset	OGBN-Products		OGBN-Papers100M	
	Light	Heavy	Light	Heavy
Fan-out	15,10,5	10,10,5,5	15,10,5	10,10,5,5
Batch size	2000	2000	2000	5000

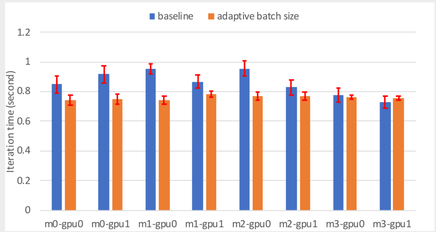
Results



Speed-up on OGBN-products



Speed-up on OGBN-papers100M



Average iteration time of each trainer. It is well balanced with the adaptive load balancing strategy.