POSTER: ParGNN: Efficient Training for Large-Scale Graph Neural Network on GPU Clusters

Shunde Li[1,2,*], Junyu Gu[1,2,*], Jue Wang[1,2,†], Tiechui Yao[1,2,5], Zhiqiang Liang[1,2], Yumeng Shi[1,2], Shigang Li[3], Weiting Xi[4], Shushen Li[4], Chunbao Zhou[1,2], Yangang Wang[1,2], Xuebin Chi[1,2]
[1]Computer Network Information Center, Chinese Academy of Sciences,[2]University of Chinese Academy of Sciences
[3]School of Computer Science, Beijing University of Posts and Telecommunications
[4]North China Electric Power University,[5]State Grid Smart Grid Research Institute Co., LTD
{lishunde,gujunyu,wangjue,yaotiechui,zqliang,shiyumeng}@cnic.cn,shigangli.cs@gmail.com,
{120201080705,120212227121}@ncepu.edu.cn,{zhoucb,wangyg,chi}@cnic.cn

Abstract

Full-batch graph neural network (GNN) training is essential for interdisciplinary applications. Large-scale graph data is usually divided into subgraphs and distributed across multiple compute units to train GNN. The state-of-the-art load balancing method based on direct graph partition is too rough to effectively achieve true load balancing on GPU clusters. We propose ParGNN, which employs a profiler-guided load balance workflow in conjunction with graph repartition to alleviate load imbalance and minimize communication traffic. Experiments have verified that ParGNN has the capability to scale to larger clusters.

CCS Concepts: • **Computing methodologies** → **Parallel computing methodologies**; **Machine learning**.

Keywords: Graph neural network, Load balancing, Data transfer hiding, Distributed training

ACM Reference Format:
Shunde Li[1,2,*], Junyu Gu[1,2,*], Jue Wang[1,2,†], Tiechui Yao[1,2,5], Zhiqiang Liang[1,2], Yumeng Shi[1,2], Shigang Li[3], Weiting Xi[4], Shushen Li[4], Chunbao Zhou[1,2], Yangang Wang[1,2], Xuebin Chi[1,2]. 2024. POSTER: ParGNN: Efficient Training for Large-Scale Graph Neural Network on GPU Clusters. In *The 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP '24), March 2–6, 2024, Edinburgh, United Kingdom.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3627535.3638488

[*]Both authors contributed equally to this research.
[†]Corresponding author.

PPoPP '24, March 2–6, 2024, Edinburgh, United Kingdom

ACM ISBN 979-8-4007-0435-2/24/03
https://doi.org/10.1145/3627535.3638488

1 Introduction

Graph Neural Network (GNN) [6] based on graph theory and graph data structure has developed rapidly in recent years. As a classic data structure, graphs are widely used in many scientific fields. Many real-world data can be naturally represented as large-scale graphs with over 100 million edges. The computation core of GNN training is the Sparse-Matrix dense-matrix Multiplication (SpMM) operation and General Matrix Multiplication (GeMM) operation, where the sparse matrix represents the topology in the large-scale graph, and the dense matrix represents the vertex features in the graph and the weights of the model.

To alleviate the load imbalance between subgraphs, existing GNN training systems commonly employ graph partition to divide the origin graph into multiple small subgraphs [1, 3–5, 7, 9]. The partition process typically aims to minimize the number of edges between subgraphs while balancing the sizes or weights of the subgraphs. However, this method is relatively crude, as the performance of SpMM on GPU is not only dependent on the dimension size and the number of nonzero elements but also on many other features, such as skew and locality [2, 8]. Previous studies only focused on the performance improvements without demonstrating actual changes in load balancing.

In this paper, we propose ParGNN, a dedicated solution that leverages hardware resources to efficiently train GNNs on multi-GPU clusters, with a focus on achieving load balancing among compute units. We employ a profiler-guided load balancing workflow in conjunction with two-level graph partition to alleviate load imbalance and minimize communication traffic. An extensive experimental study has been conducted to demonstrate the parallel efficiency. To the best of our knowledge, ParGNN is the first to achieve distributed full-batch training on the full-precision papers100M dataset using GPU-based on-device systems.

2 PROFILER-GUIDED REPARTITION

The sparsity of graph data introduces significant load imbalance in the execution process of distributed GNN. Directly representing the computation load based on the number of

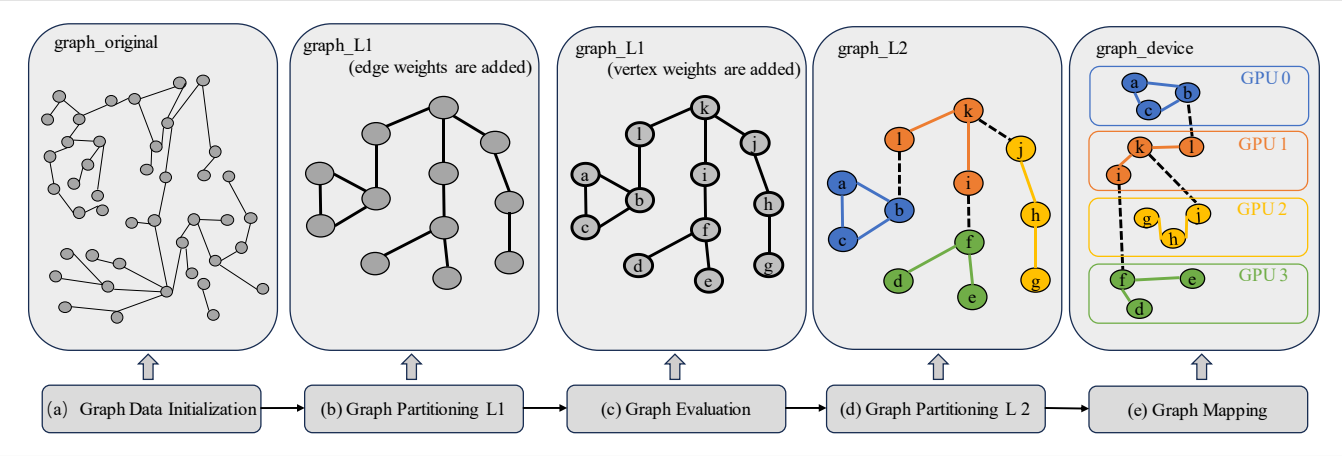

Figure 1. Profiler-guided load balance workflow. The workflow of two-level load balance involves two graph partition steps. In step (b), the Graph Partition initially divides the original data graph graph_original in step (a) into multiple subgraphs, i.e. graph_L1, where the number of subgraphs is several times the number of GPUs used. In step (c), profiling is done on graph_L1, followed by the addition of weights of edge and vertex. In step (d), the Graph Repartition then divides graph_L1 into multiple subgraphs, i.e. graph_L2, where the number of subgraphs is equal to the number of GPUs used.

vertices is coarse, as the computational complexity of sparse matrices largely depends on their memory access density. To mitigate this severe load imbalance, it is necessary to partition the original graph data effectively, ensuring that computation tasks are distributed as evenly as possible among different computing units. This paper presents a novel two-level partition approach guided by profiler to achieve load balancing in distributed GNN training.

By employing the workflow shown in Figure 1, we can alleviate the load imbalance caused by the sparsity of graph data in distributed GNN training. We achieve optimal utilization of computational resources, thereby enhancing system efficiency and performance. Compared to other frameworks that adopt one-time graph partition, they require a time-consuming new partition of the origin graph if there is a slight change in the origin graph or a change in the number of underlying hardware processing units. Our strategy is to only perform graph partition once, and graph repartitions are performed on smaller coarsened graphs. The first three steps can be computationally expensive and may take several hours. However, it is worth mentioning that the preprocessing step is a one-time requirement for a specific dataset. Moreover, the last two steps of the process can be completed within a few seconds. This implies that the load balancing method can be employed with lower overhead for various scale training tasks on supercomputers.

Additionally, our observations indicate that multiple subgraphs which are mapped to the same GPU may have common neighboring subgraphs. The proposed load balance workflow presents an opportunity for eliminating redundant communication and improving training performance through the pipelined execution of multiple subgraphs.

3 Parallel Performance Evaluation

To evaluate the scalability of ParGNN on larger-scale clusters, we conducted experiments using the products dataset. We employed the adaptive load balancing method to partition the products dataset into 320 subgraphs and mapped them onto 8, 16, 32, 64, and 128 GPUs, respectively. We calculated the average execution time for 300 epochs during the training process and presented the experimental results in Figure 2. The experimental results indicate that when scaled to 128 GPUs, compared to training on 8 GPUs, the training speed improved by 7.4 times, and maintaining a parallel efficiency of 46.29%. Through this experiment, we have validated the ability of ParGNN to accelerate training on larger-scale hardware.

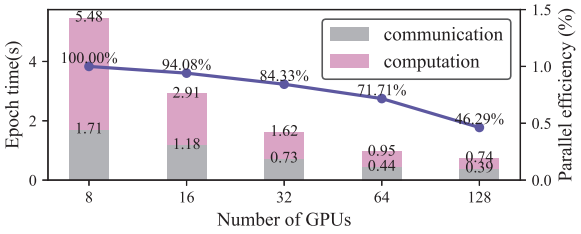

Figure 2. Strong scalability of products

Acknowledgments

This work is supported by the the National Key Research & Development Program of China (2023YFB4502303),National Natural Science Foundation of China under Grant No. 62372055 and the Fundamental Research Funds for the Central Universities.

References

[1] Zhenkun Cai, Xiao Yan, Yidi Wu, Kaihao Ma, James Cheng, and Fan Yu. 2021. DGCL: An Efficient Communication Library for Distributed GNN Training. In *Proceedings of the Sixteenth European Conference on Computer Systems* (Online Event, United Kingdom) *(EuroSys '21)*. Association for Computing Machinery, New York, NY, USA, 130–144. https://doi.org/10.1145/3447786.3456233

[2] Zhen Du, Jiajia Li, Yinshan Wang, Xueqi Li, Guangming Tan, and N. Sun. 2022. AlphaSparse: Generating High Performance SpMV Codes Directly from Sparse Matrices. *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis* (2022), 1–15. https://api.semanticscholar.org/CorpusID:254877267

[3] Matthias Fey, Jan E. Lenssen, Frank Weichert, and Jure Leskovec. 2021. GNNAutoScale: Scalable and Expressive Graph Neural Networks via Historical Embeddings. arXiv:2106.05609 [cs.LG]

[4] Lingxiao Ma, Zhi Yang, Youshan Miao, Jilong Xue, Ming Wu, Lidong Zhou, and Yafei Dai. 2019. NeuGraph: Parallel Deep Neural Network Computation on Large Graphs. In *2019 USENIX Annual Technical Conference*. https://www.microsoft.com/en-us/research/publication/neugraph-parallel-deep-neural-network-computation-on-large-graphs/

[5] Hesham Mostafa. 2021. Sequential Aggregation and Rematerialization: Distributed Full-batch Training of Graph Neural Networks on Large Graphs. *ArXiv* abs/2111.06483 (2021). https://api.semanticscholar.org/CorpusID:244103039

[6] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20, 1 (2009), 61–80. https://doi.org/10.1109/TNN.2008.2005605

[7] Cheng Wan, Youjie Li, Cameron R. Wolfe, Anastasios Kyrillidis, Nam Sung Kim, and Yingyan Lin. 2022. PipeGCN: Efficient Full-Graph Training of Graph Convolutional Networks with Pipelined Feature Communication. arXiv:2203.10428 [cs.LG]

[8] Serif Yesil, Azin Heidarshenas, Adam Morrison, and Josep Torrellas. 2023. WISE: Predicting the Performance of Sparse Matrix Vector Multiplication with Machine Learning. In *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming* (Montreal, QC, Canada) *(PPoPP '23)*. Association for Computing Machinery, New York, NY, USA, 329–341. https://doi.org/10.1145/3572848.3577506

[9] Da Zheng, Xiang Song, Chengru Yang, Dominique LaSalle, and George Karypis. 2022. Distributed Hybrid CPU and GPU training for Graph Neural Networks on Billion-Scale Graphs. arXiv:2112.15345 [cs.DC]