Plexus-v2: Extreme-Scale GNN Training with Communication Free Sampling and 4D Hybrid Parallelism

Cunyang Wei*, Siddharth Singh†, Aishwarya Sarkar‡, Daniel Nichols§, Aditya K. Ranjan¶, Sayan Ghosh‖, Ali Jannesari‡, Nathan R. Tallent‖, Abhinav Bhatele*

*Department of Computer Science, University of Maryland, College Park, USA
†NVIDIA, Inc., Santa Clara, USA
‡Department of Computer Science, Iowa State University, Ames, USA
§Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, USA
¶Google, Inc., Mountain View, USA
‖Pacific Northwest National Laboratory, Richland, USA
E-mail: cunyang@umd.edu, bhatele@cs.umd.edu

***Abstract*—Mini-batch training with sampling is the dominant approach for scaling graph neural network (GNN) training to large graphs, but existing distributed frameworks are limited by expensive sampling and by vanilla data parallelism. We present Plexus-v2, a four-dimensional (4D) hybrid parallel framework that combines communication-free distributed sampling, three-dimensional (3D) parallel matrix multiplication (PMM), and data parallelism. Plexus-v2 introduces a uniform vertex sampling algorithm that enables every GPU to construct its local shard of each mini-batch subgraph without any inter-process communication, and adds optimizations including sampling–training overlap, lower-precision communication, and kernel fusion. On five graph datasets, Plexus-v2 scales to 2048 GPUs on Perlmutter, 2048 GCDs on Frontier, and 1024 APUs on Tuolumne, achieves a 3.5× end-to-end training speedup over the best state-of-the-art framework on ogbn-products, and reaches higher test accuracy than GraphSAINT and GraphSAGE.**

I. Introduction and Motivation

Graph neural networks (GNNs) power recommender systems, fraud detection, and scientific discovery on real-world graphs. Learning from graphs with hundreds of millions of vertices requires distributed training, and mini-batch training with sampling is the most popular approach because it can converge faster than full-graph training. However, existing distributed frameworks face two fundamental bottlenecks: 1) sampling pipelines typically execute on CPUs and require frequent communication for neighbor and feature accesses, and 2) scaling out relies on vanilla data parallelism, which improves throughput but often fails to reduce end-to-end training time. Plexus-v2 solves both bottlenecks and scales to the largest GPU-based supercomputers.

II. Communication-Free Uniform Vertex Sampling

Plexus-v2 is based on a GCN [1] architecture augmented with RMS normalization, ReLU, and dropout. At every training step, it samples B vertices uniformly from the vertex set of size N and constructs the vertex-induced subgraph

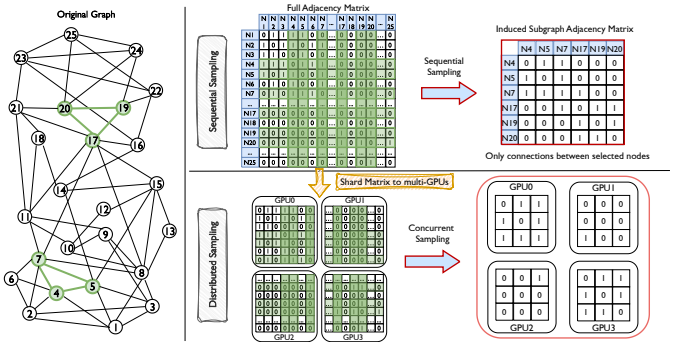


Fig. 1. Plexus-v2 uniform vertex sampling. (Left) Sampled vertices (green) on the original graph. (Upper right) Sequential sampling: the induced subgraph retaining only edges between sampled vertices. (Lower right) Distributed sampling: each GPU independently extracts its local shard.

over the sampled set (Fig. 1). To correct the bias introduced by edges dropped during subgraph construction, Plexus-v2 rescales off-diagonal edge weights by the inverse of the conditional inclusion probability $p = (B-1)/(N-1)$, which makes mini-batch aggregation an unbiased estimator of full-graph aggregation in expectation. Since the rescaling factor depends only on the global constants B and N, and the sample itself is derived from a random seed shared within each group, every GPU independently computes the identical sample and extracts its local shard of the induced subgraph from its local partition of the adjacency matrix. The extraction is fully vectorized on GPUs using binary-search range location, prefix-sum-based row gathering, and a step-tagged $O(B)$ index remap. The entire sampling and subgraph construction pipeline therefore requires no inter-GPU communication, unlike GraphSAINT [2] and GraphSAGE-based [3] distributed systems.

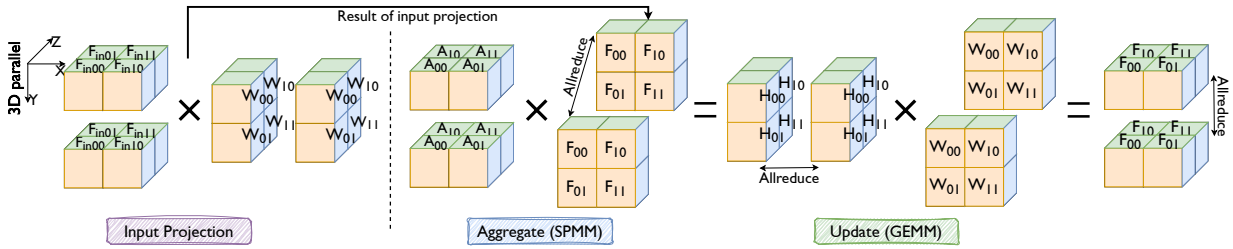


Fig. 2. 3D PMM forward pass in Plexus-v2 with eight GPUs in a 2×2×2 grid ($X \times Y \times Z$). Matrices are distributed across orthogonal planes (shown in different colors). Each operation performs a local multiplication followed by an all-reduce along the corresponding axis.

III. 4D Hybrid Parallel Training

For distributed training, Plexus-v2 organizes GPUs into a virtual $G_d \times G_x \times G_y \times G_z$ grid that combines data parallelism with 3D parallel matrix multiplication (PMM) [4], [5]. Each tensor-parallel group of $G_x \times G_y \times G_z$ GPUs holds a complete copy of the model and parallelizes the sparse (SpMM) and dense (GEMM) matrix products of each GCN layer across orthogonal planes of the 3D grid, combining partial results with all-reduces along the grid axes (Fig. 2). Adjacency shards for consecutive layers rotate across the three planes in round-robin order so that matrix layouts remain aligned without additional communication. The G_d data-parallel groups train on independent mini-batches and synchronize weight gradients with a single all-reduce per step. We further apply four optimizations: prefetching next-step sampling on a dedicated CUDA stream to fully overlap it with training (24% lower epoch time); running the dominant 3D PMM collectives in BF16 while keeping numerically sensitive reductions in FP32 (17%); fusing element-wise operators via `torch.compile` (6%); and overlapping backward-pass all-reduces on orthogonal communicator groups with local computation (3%). Together these deliver a 1.75× epoch-time speedup.

IV. Results

We evaluate Plexus-v2 on five graph datasets (Reddit, ogbn-products, Isolate-3-8M, Products-14M, and ogbn-papers100M [6]) on three HPE Cray EX systems: Perlmutter (NVIDIA A100), Frontier (AMD MI250X), and Tuolumne (AMD MI300A). Uniform vertex sampling preserves model quality: Plexus-v2 reaches 81.3% test accuracy on ogbn-products, outperforming GraphSAINT (80.2%) and GraphSAGE (79.6%), and 96.3% on Reddit. Compared against four state-of-the-art frameworks (DistDGL [7], MassiveGNN [8], SALIENT++ [9], and BNS-GCN [10]) using time-to-target-accuracy, Plexus-v2 achieves the lowest end-to-end training time in all configurations (Fig. 4). On ogbn-products at 64 GPUs on Perlmutter, it is 3.5× faster than the best baseline (SALIENT++) and 78.2× faster than DistDGL. On Frontier, the gap widens to over 300×: profiling shows that DistDGL and MassiveGNN use RPC-based remote feature fetching, which dominates per-epoch time there. Several baselines even become slower at larger GPU counts. In contrast, Plexus-v2 continues to reduce end-to-end training time as it scales out, underscoring the scalability of its communication-free sampling and 4D hybrid parallel design. Its strong performance on Frontier and Tuolumne also demonstrates performance portability: RPC-based feature fetching fails to exploit Frontier's Slingshot interconnect, whereas Plexus-v2 communicates only through the vendor's collective communication library (NCCL or RCCL) and therefore runs efficiently on both NVIDIA and AMD systems. Finally, Plexus-v2 shows consistent strong scaling (Fig. 5). On ogbn-papers100M, it achieves a 21.7× speedup scaling from 64 to 2048 GPUs on Perlmutter (epoch time 4.1 s to 189 ms), and 20.3× and 15.9× speedups on Frontier and Tuolumne, respectively.

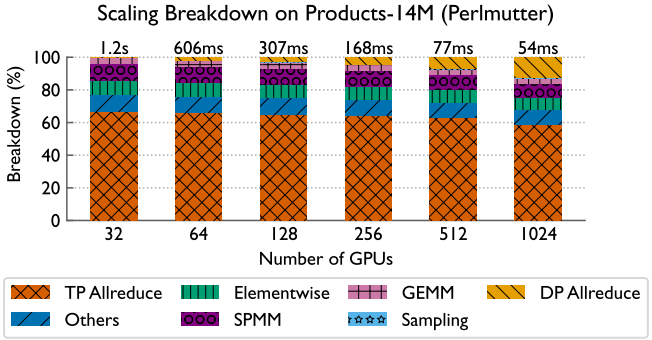


Fig. 3. Epoch time breakdown on Products-14M on Perlmutter.

To understand the scaling behavior, we profile the epoch time breakdown. As shown in Figure 3, at $G_d = 1$, tensor-parallel all-reduces and compute kernels dominate the epoch time on Products-14M. As G_d grows, the data-parallel all-reduce rises from negligible to a growing fraction of epoch time, reflecting gradient synchronization across more replica groups, while the fractions of tensor-parallel operations and sampling stay roughly constant.

We also investigate why epoch times are higher on the two AMD systems, and find that the sparse kernel performance of ROCm is the main cause: GEMM performance on Frontier and Tuolumne is comparable to (or even better than) that on Perlmutter, but SpMM runs several times slower, and this gap widens as the graphs grow larger. In addition, RCCL delivers lower all-reduce throughput than NCCL at large GPU counts. We therefore call for optimized sparse kernels for irregular computations on AMD GPUs and for improved RCCL performance at scale.

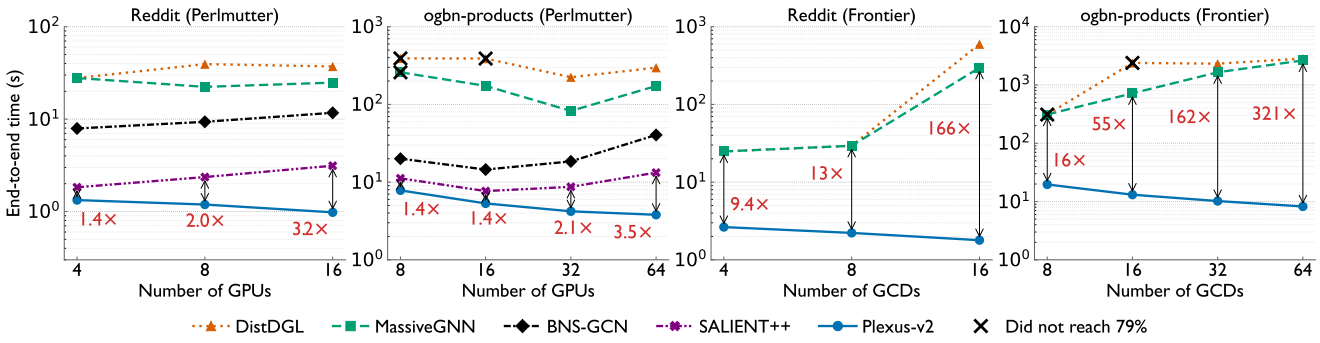


Fig. 4. End-to-end training time to target accuracy on Perlmutter and Frontier (log scale; lower is better). Points marked × did not reach the target accuracy. Annotations compare Plexus-v2 to SALIENT++ (Perlmutter) and MassiveGNN (Frontier) for each dataset.

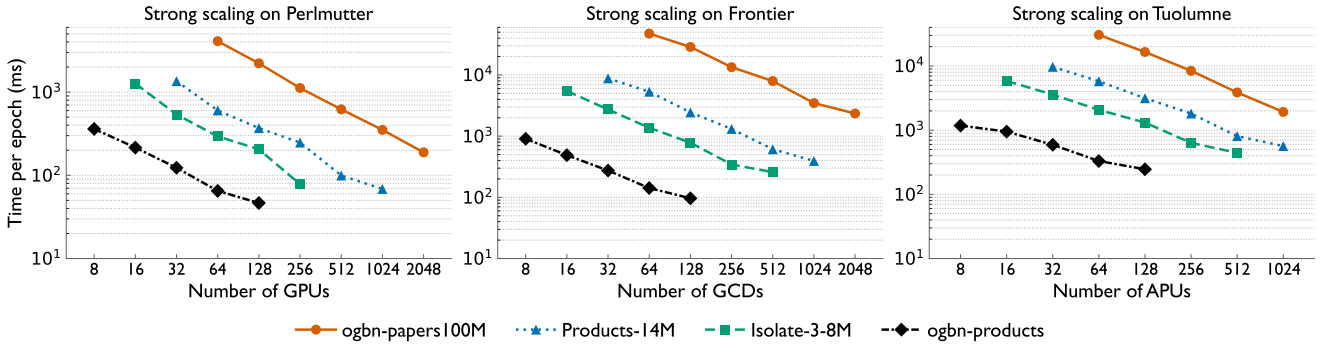


Fig. 5. Strong scaling of Plexus-v2 on Perlmutter (left), Frontier (center), and Tuolumne (right). Each curve starts at the smallest 3D PMM configuration (G_d=1) and scales out by increasing data-parallel replicas G_d.

Acknowledgment

This material is based upon work supported in part by the National Science Foundation under Grant No. 2047120. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy (DOE) Office of Science User Facility, operated under Contract No. DE-AC02-05CH11231 using NERSC awards DDR-ERCAP0034262 and ALCC-ERCAP0034775, and that of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. DOE under Contract No. DE-AC05-00OR22725. This work was performed in part under the auspices of the U.S. Department of Energy (DOE) by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-CONF-2017867).

References

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016. [Online]. Available: https://doi.org/10.48550/arXiv.1609.02907

[2] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graphsaint: Graph sampling based inductive learning method," *arXiv preprint arXiv:1907.04931*, 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1907.04931

[3] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2018. [Online]. Available: https://doi.org/10.48550/arXiv.1706.02216

[4] R. C. Agarwal, S. M. Balle, F. G. Gustavson, M. Joshi, and P. Palkar, "A three-dimensional approach to parallel matrix multiplication," *IBM Journal of Research and Development*, vol. 39, no. 5, pp. 575–582, 1995. [Online]. Available: https://doi.org/10.1147/rd.395.0575

[5] A. K. Ranjan, S. Singh, C. Wei, and A. Bhatele, "Plexus: Taming billion-edge graphs with 3D parallel full-graph GNN training," in *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '25. ACM, Nov. 2025. [Online]. Available: https://doi.acm.org/10.1145/3712285.3759890

[6] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2005.00687

[7] D. Zheng, C. Ma, M. Wang, J. Zhou, Q. Su, X. Song, Q. Gan, Z. Zhang, and G. Karypis, "Distdgl: Distributed graph neural network training for billion-scale graphs," in *2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3)*. IEEE, 2020, pp. 36–44. [Online]. Available: https://doi.org/10.1109/IA351965.2020.00011

[8] A. Sarkar, S. Ghosh, N. R. Tallent, and A. Jannesari, "Massivegnn: Efficient training via prefetching for massively connected distributed graphs," in *2024 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2024, pp. 62–73. [Online]. Available: https://doi.org/10.1109/Cluster59578.2024.00013

[9] T. Kaler, A. Iliopoulos, P. Murzynowski, T. Schardl, C. E. Leiserson, and J. Chen, "Communication-efficient graph neural networks with probabilistic neighborhood expansion analysis and caching," *Proceedings of Machine Learning and Systems*, vol. 5, pp. 477–494, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.03152

[10] C. Wan, Y. Li, A. Li, N. S. Kim, and Y. Lin, "Bns-gcn: Efficient full-graph training of graph convolutional networks with partition-parallelism and random boundary node sampling," 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2203.10983