Poster ID: 08

Fahao Chen, 2nd Year Ph. D Student
Computer Organization Lab., The University of Aizu

Poster Session at Graduate School Information Fair

DGC: Training Dynamic Graphs with Spatio-Temporal Non-Uniformity using Graph Partitioning by Chunks

Background and Motivation

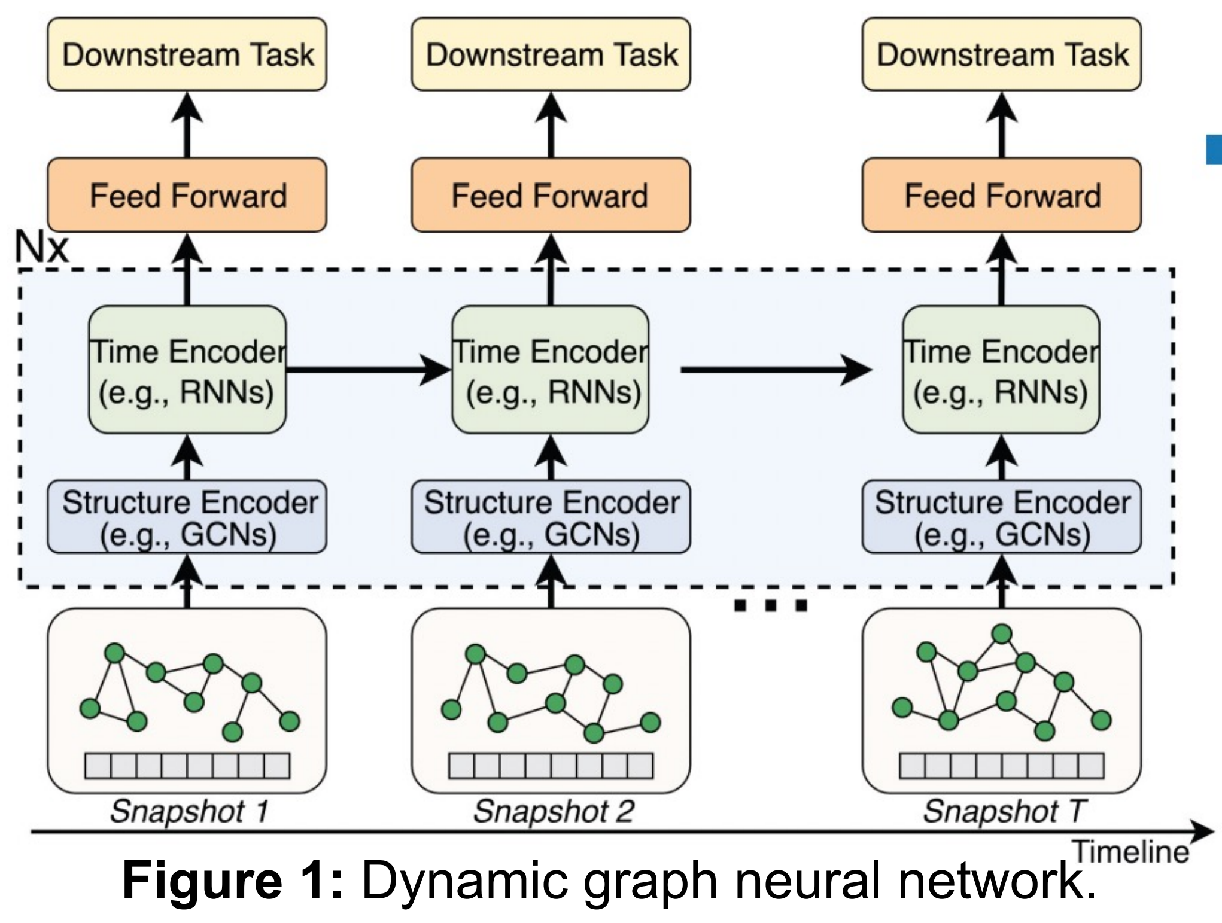
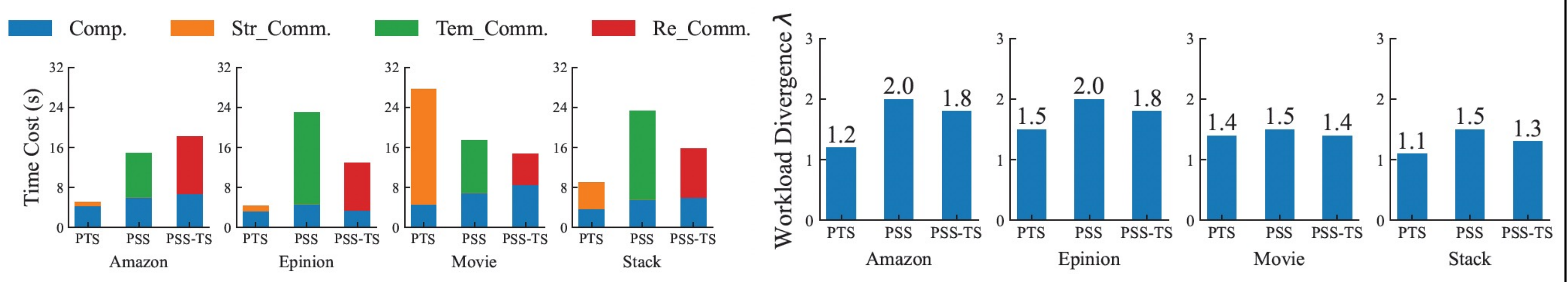


Figure 1: Dynamic graph neural network.

A dynamic graph neural network (DGNN) is composed of multiple blocks, where each block consists of a structure encoder and a time encoder, as illustrated in Figure 1. The structure encoder extracts hidden information for each vertex by aggregating information from its structural neighbors. Meanwhile, the time encoder accumulates information for each vertex from its temporal neighbors.



(a) Average epoch time. (b) GPU workload divergence.

Figure 2: Performance of dynamic graph partitioning methods on different datasets

(1) Partitioning by Spatial Snapshots (PSS): it treats a snapshot as the partition unit and always keeps spatial dependencies within the same GPU. **(2) Partitioning by Temporal Sequences (PTS) [1]:** A temporal sequence records the states of the same vertex in different time. This approach eliminates communication overhead for vertices when aggregating their temporal neighborhoods. However, high communication overhead may arise when vertices aggregate their spatial neighborhoods, as spatial dependencies are broken down across GPUs. **(3) Partitioning by Snapshots and Sequences (PSS-TS) [2]:** a joint method adopts PSS for the structure encoder while transitioning to PTS for the time encoder.

DGC: A Distributed System for Efficient DGNN Training

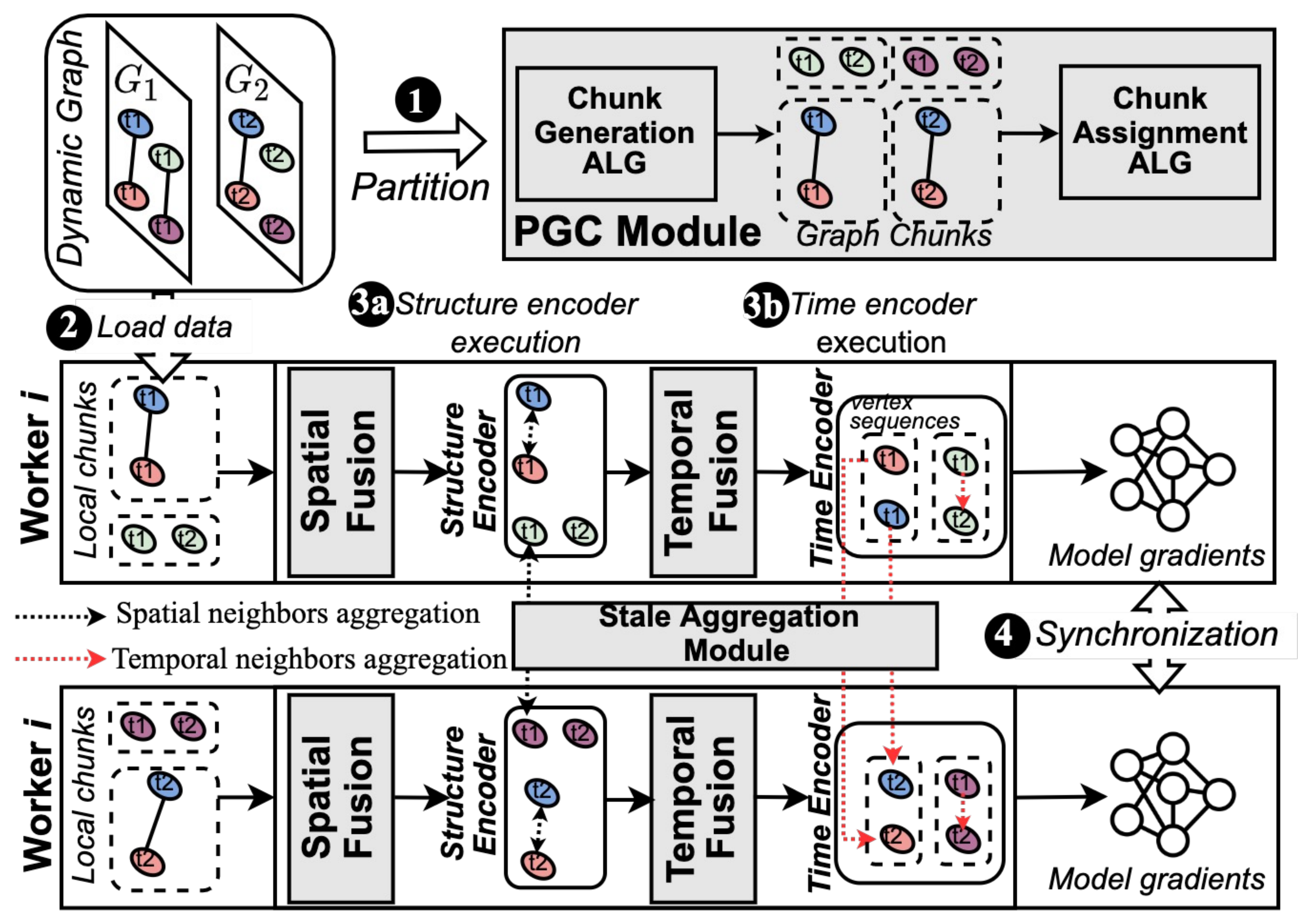
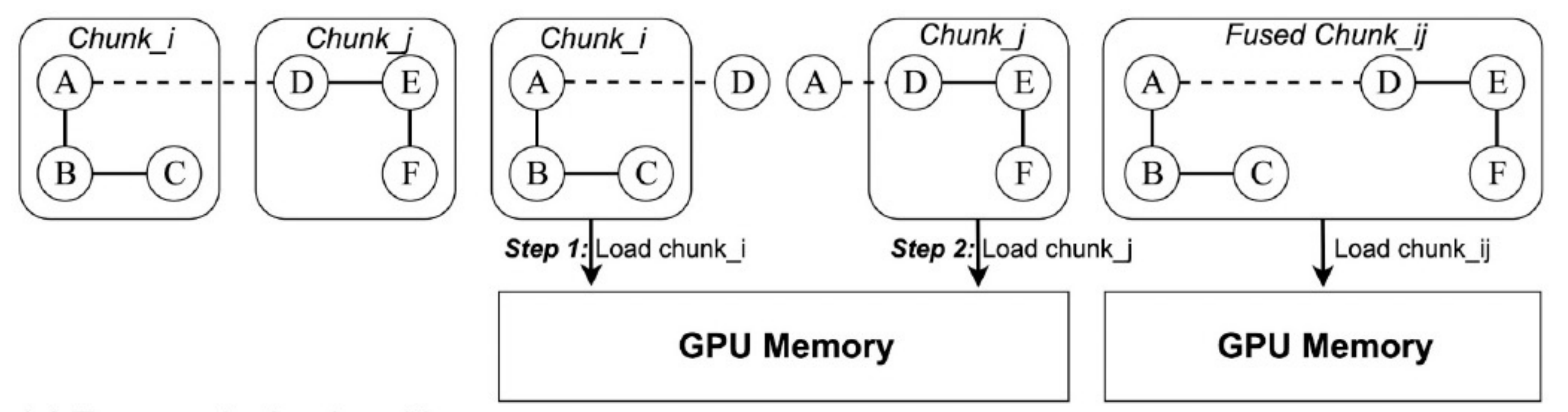


Figure 3: System Overview



(a) Two graph chunks with spatial dependencies (b) Load graph chunks sequentially (c) Load a fused graph chunk

Figure 4: An illustration of spatial fusion.

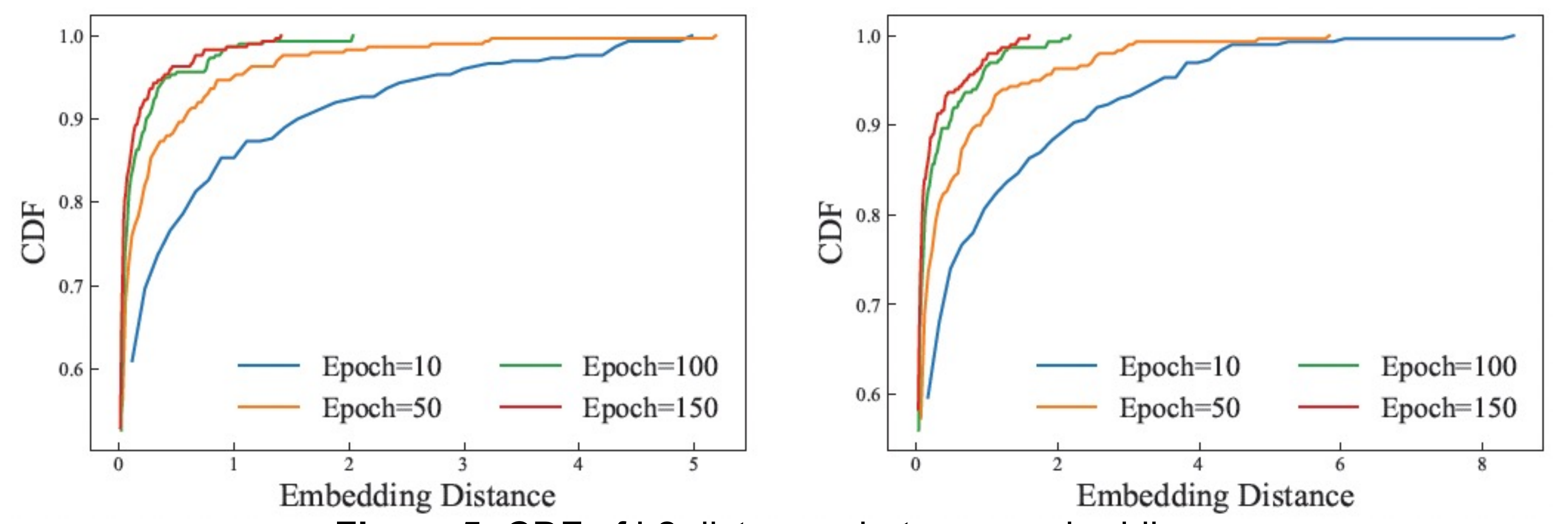


Figure 5: CDF of L2 distances between embeddings.

The design goals of DGC:

High training efficiency. Due to massive data dependencies (including spatial and temporal dependencies) among vertices in dynamic graphs, distributed DGNN training suffers from high communication cost that would be the performance bottleneck. DGC needs to reduce communication cost to accelerate training process.

High GPU utilization. Multiple GPUs are used to train DGNNs for handling large dynamic graphs. GPU utilization is a crucial metric for efficient resource management. DGC should ensure high GPU utilization during DGNN training.

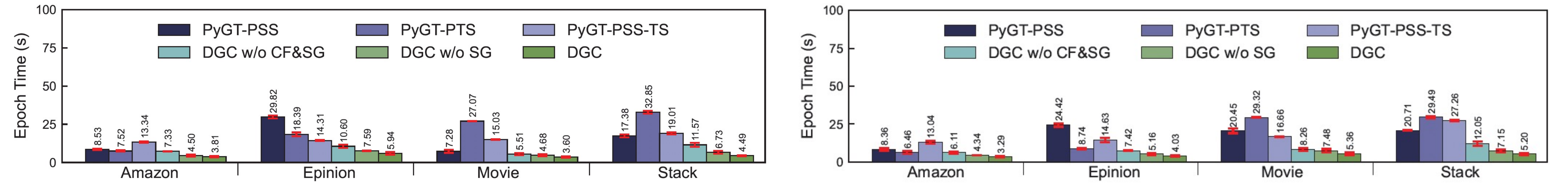
Consistent training convergence. While introducing various optimizations to accelerate DGNN training, DGC needs to ensure that these designs do not compromise training convergence, preserving the quality of the final model.

Partition by Graph Chunks (PGC): As shown in Figure 3, each graph chunk may contain vertices and edge belonging to different snapshots and temporal sequences. We design a graph chunk generation algorithm based on the graph coarsening technique with a full consideration of spatio-temporal non-uniformity, so that each graph chunk has modest training workload and few edge connections to other chunks. By a simple heuristic to assign these chunks to GPUs, DGC can achieve better workload balance and reduced communication cost, to significantly improve DGNN training efficiency.

Chunk Fusion (CF): The fuse spatial and temporal chunks into larger ones before loading, while considering the GPU memory constraint, to reduce redundant data loading and improve GPU utilization.

Adaptive stale embedding aggregation (SG): Motivated by the observation that vertices may generate similar embeddings in different training epochs (Figure 5). DGC allows GPUs to reuse stale embeddings from previous epochs if they are sufficiently similar, to reduce data traffic between GPUs.

Evaluation



(a) T-GCN. (b) DySAT.

Figure 6: Epch time of different methods.

Figure 12 shows the overall speedup over PyGT when using different models. DGC outperforms all baselines by 1.25× - 7.52× (on average 3.95×, 3.97×, and 3.77× for T-GCN, DySAT, and MPNN-LSTM, respectively). Different baselines exhibit varying performance on these datasets. DGC with only PGC module, denoted by the bar of “DGC w/o CF&SG”, can still accelerate the training process by 1.03× to 4.92×, compared to other methods. When chunk fusion (CF) is enabled, the epoch time can be further reduced by 1.39×. (For more experiments, please refer to out paper [3].)

Conclusion

We introduces DGC, a distributed training framework designed to optimize DGNN training efficiency. By incorporating a novel dynamic graph partitioning method (PGC) and run-time optimizations, DGC effectively tackles the challenges of high communication costs and low GPU utilization in distributed DGNN training. Experimental results demonstrate that DGC achieves a 1.25×-7.52× speedup compared to state-of-the-art DGNN training frameworks.

Reference

[1] Guan, Mingyu, Anand Padmanabha Iyer, and Taesoo Kim. "DynaGraph: dynamic graph neural networks at scale." Proceedings of SIGMOD Joint International Workshop on GRADES and and NDA. 2022.
[2] Chakaravarthy, Venkatesan T., Shivmaran S. Pandian, Saurabh Raje, Yogish Sabharwal, Toyotaro Suzumura, and Shashanka Ubaru. "Efficient scaling of dynamic graph neural networks." In Proceedings of SC, pp. 1-15. 2021.
[3] Chen, Fahao, Peng Li, and Celimuge Wu. "DGC: Training Dynamic Graphs with Spatio-Temporal Non-Uniformity using Graph Partitioning by Chunks." Proceedings of SIGMOD. 2024