TC-GNN: Accelerating Sparse Graph Neural Network Computation Via GPU Dense Tensor Core

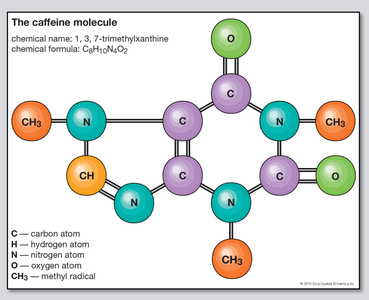
Yuke Wang, Boyuan Feng, Zheng Wang, and Yufei Ding.
Department of Computer Science, UC at Santa Barbara

OSDI 22

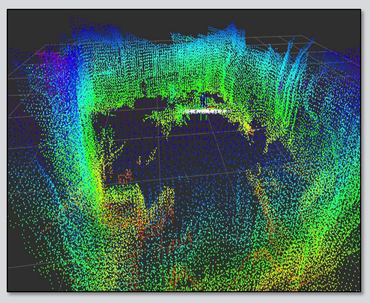
Graphs are everywhere...

Social Networks

Financial Services



Molecular chemistry



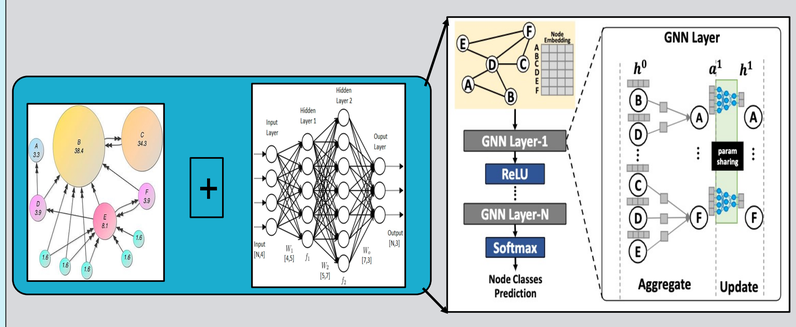
Point Cloud

Power Grid

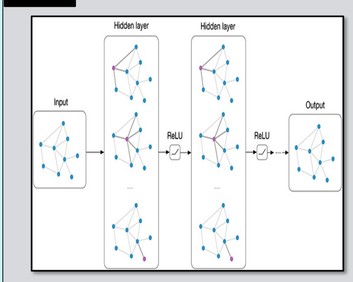
Molecular Biology

Credit: Google Image

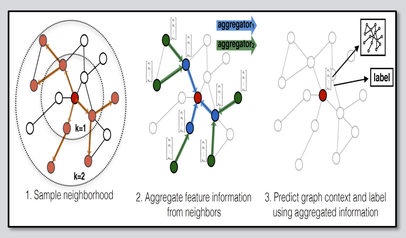
GNN is the key for graph learning



GCN



GraphSAGE



Credit: Google Image

Challenges

- Existing deep-learning frameworks are optimized for dense neural network operations.

Lack of efficient support for sparse graph neural network computation.

- Existing sparse computation kernel can only leverage CUDA core on GPUs.

Underutilize the latest GPU with new hardware feature that offer high -performance compuation.

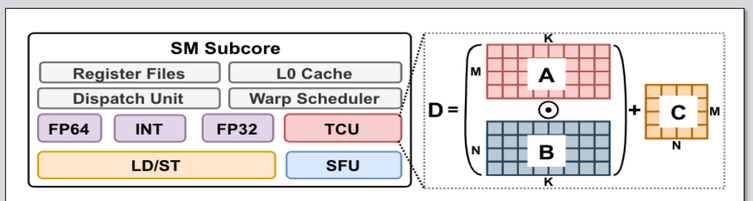
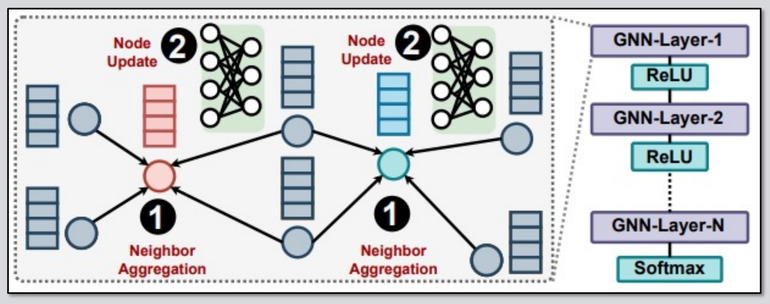


Figure 3. A Subcore of GPU SM with TCU.

GNN Background

- **Graph Neural Network Basics.**

$$a_v^{(k+1)} = \textbf{\textit{Aggregate}}^{(k+1)}(h_u^{(k)} | u \in \mathbf{N}(v) \cup h_v^{(k)})$$
$$h_v^{(k+1)} = \textbf{\textit{Update}}^{(k+1)}(a_v^{(k+1)})$$



- **Basic computation in GNNs.**

- Neighbor aggregation (SpMM-like).

$$\hat{\mathbf{X}} = (\mathbf{F}_{N\times N} \odot \mathbf{A}_{N\times N}) \cdot X_{N\times D})$$

- Edge feature computation (SDDMM-like).

$$\mathbf{F} = (\mathbf{X}_{N\times D} \cdot \mathbf{X}^T_{N\times D}) \odot \mathbf{A}_{N\times N}$$

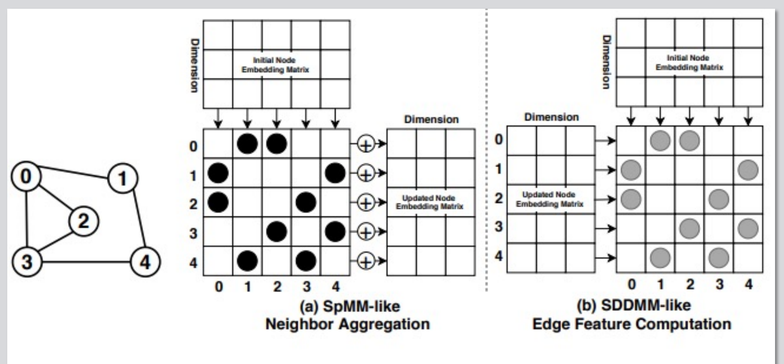


Figure 2. SpMM-like and SDDMM-like Operation in GNNs. Note that "→" indicates loading data; "⊕" indicates neighbor embedding accumulation.

Motivation

- **Sparse MM on CUDA core**

GPUs fit GNNs

Table 1. Profiling of GCN Sparse Operations.

Dataset	Aggr. (%)	Update (%)	Cache(%)	Occ.(%)
Cora	88.56	11.44	37.22	15.06
Citeseer	86.52	13.47	38.18	15.19
Pubmed	94.39	5.55	37.22	16.24

- **Dense MM on CUDA core**

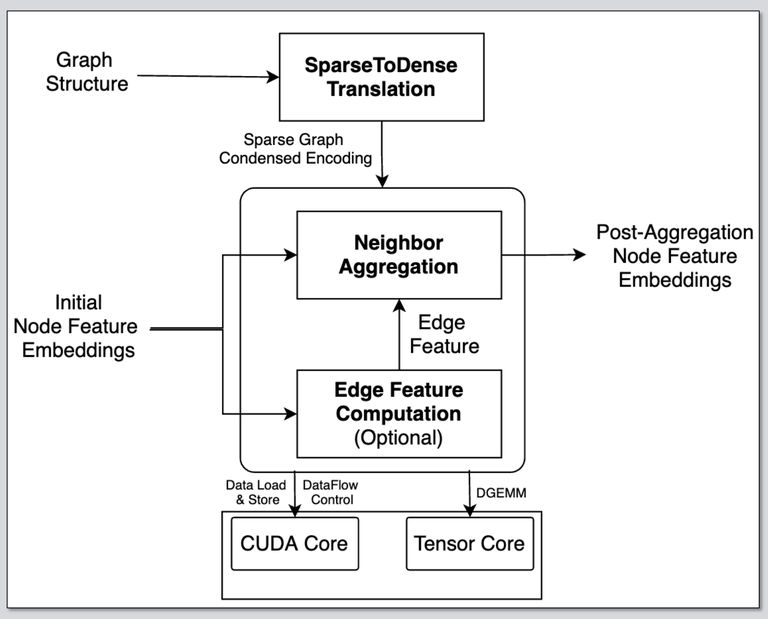
GNNs fit GPUs

Table 3. Medium-size Graphs in GNNs.

Dataset	# Nodes	# Edges	Memory	Eff.Comp
OVCR-8H	1,890,931	3,946,402	14302.48 GB	0.36%
Yeast	1,714,644	3,636,546	11760.02 GB	0.32%
DD	334,925	1,686,092	448.70 GB	0.03%

☹ Apply separate optimization on one direction only would hardly work

Overall Design



TC-GNN Design

- The *first* TCU-based GNN acceleration design on GPUs.

Lack of efficient support for sparse graph neural network computation

- At the input level technique.

Sparse graph translation (SGT) technique condense non-zero elements from sparse tiles into a fewer "dense" tiles

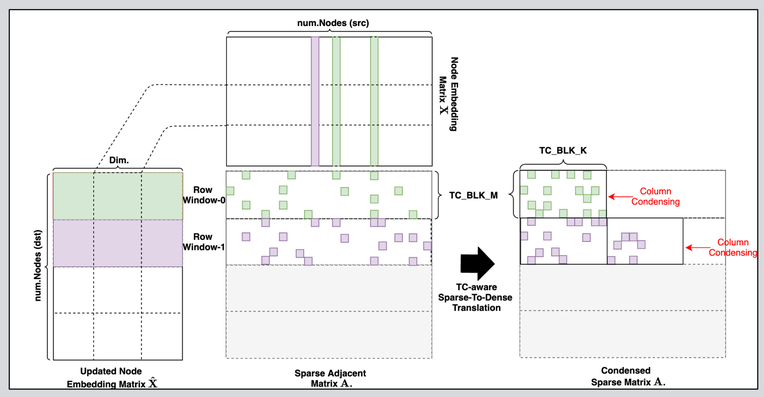
- At the kernel level innovation.

TC-GNN exploits the benefits of CUDA core and TCU collaboration.

- At the framework level design.

TC-GNN integrates with the popular Pytorch framework.

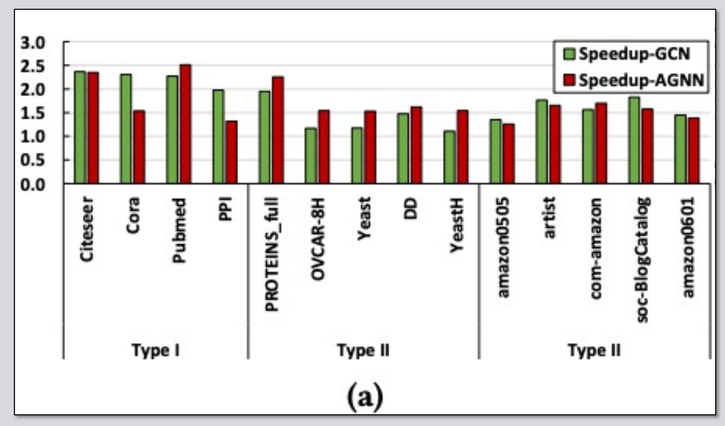
Sparse Graph Translation

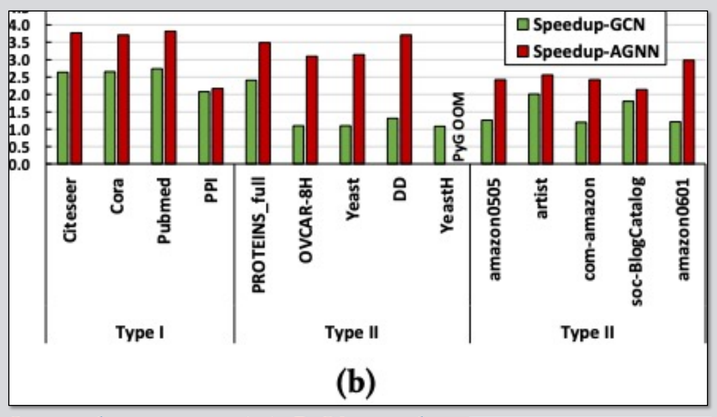


1. Fewer number of iterations for Calling TC WMMA primitives.
2. Fewer number of dense row access for node embedding vector.
3. Lower Shared Memory Usage due to more condensed tiles loading.

Evaluation

- **Baseline:** 1) Deep Graph Library (DGL); 2) PyTorch Geometric (PyG).
- **GNN model:** 1) GCN (Graph Convolutional Network); 2) AGNN (Attention-based GNN).
- **Platform:** A desktop server with 8-core 16-thread. Intel Xeon Silver 4110 CPU (64GB host memory) and NVIDIA RTX3090 GPU (24GB device memory)





Speedup over (a) DGL and (b) PyG on GCN and AGNN.

Avg: 1.70X

UCSB