1. Motivation & Challenges

- **Graph Neural Network (GNN) has wide application.**
- **GNN training is time-consuming.**
  - GAT [Velickovic et al., 2018] on a 2-million-node graph: 1.86 hrs on V100.
  - A recommender model on 3-billion-node graph: up to 78 hrs on 16 Tesla K80 GPUs [Ying et al., KDD 2018].

2. Feature Aggregation as SpMM

- **Feature aggregation can be treated as Sparse-Dense Matrix Multiplication (SpMM) with custom aggregator.**

3. Solution Overview

4. SpMM Kernel Design

5. Experimental Results

- **1. Sparse matrix re-use**
  - **Challenge:** implementing SpMM with concurrent SpMVs causes parallel threads to load the same sparse matrix element.
  - **Method:** Sparse Row Caching (SRC). Parallel threads first load a tile of the sparse matrix row into shared memory, and then traverse the cached elements one-by-one.

- **2. Workload balance**
  - **Challenge:** Irregular distribution of non-zeros among rows causes workload imbalance in row-wise parallel algorithm.
  - **Method:** Choose between row-wise parallel (Rbalance) or element-wise parallel (Ebalance) algorithm, according to row-length variance.

- **2. Model speedup**
  - **Compare with PyG (Fey & Lenssen, 2019) on GCN**

- **2. Model speedup**
  - **Compare with DGL (Zhang et al., 2019) on RTX2080 GPU**

- **Ablation Study**
  - Training speedup summary:
    - Against message-passing (not SpMM kernel) implementation in PyG, 0.99x-2.89 GPU time reduction.
    - Against DGL’s cuSPARSE-based implementation: 0.86x-1.29 GPU time reduction.
    - Against DGL’s custom-aggertor SpMM (not supported in cuSPARSE): up to 3.51x kernel speedup.