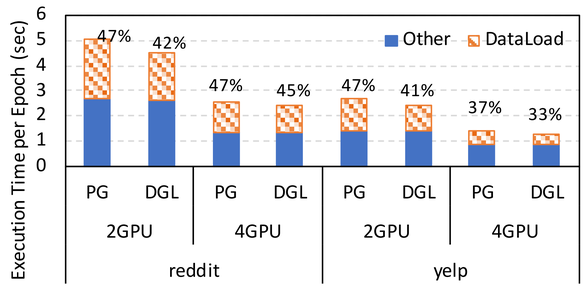
HighLights: Rethinking Graph Data Placement for Graph Neural Network Training on Multiple GPUs

Motivation

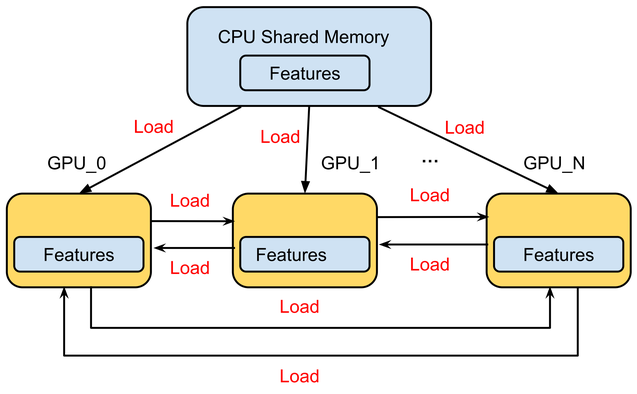
- **Partition is commonly used for large graph training on multiple GPUs**
  - Feature data exceeds memory capacity of one GPU
  - Distribute features onto multiple GPUs
  - Loading features is a bottleneck



Background on Graph Partition

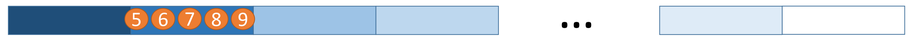
- **Distdgl**
  - Adopts METIS graph partitioning
  - Assumes that the graph can be entirely stored on multiple GPUs
  - Not work for large graphs that exceed the memory capacity (Zheng et al. IA3'20,)
- **PaGraph**
  - Uses a heuristic partitioning method
  - For large graphs that exceed the memory capacity of multiple GPUs, it stores the graph on CPU and buffers the most frequently accessed nodes of each partition on GPU (Lin et al. SoCC'20)

Performance Model



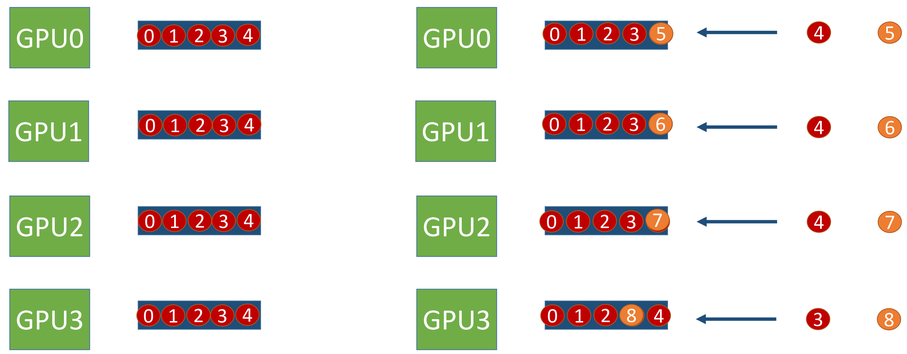
- **Cost Function of GPU_i**
- **Goal**
- **Goal**
  - Minimize the maximum cost across all GPUs

Minimizing The Data Movement Overhead



Sample probability high->low

- **T cpu < T gpu**
- **T cpu > T gpu**

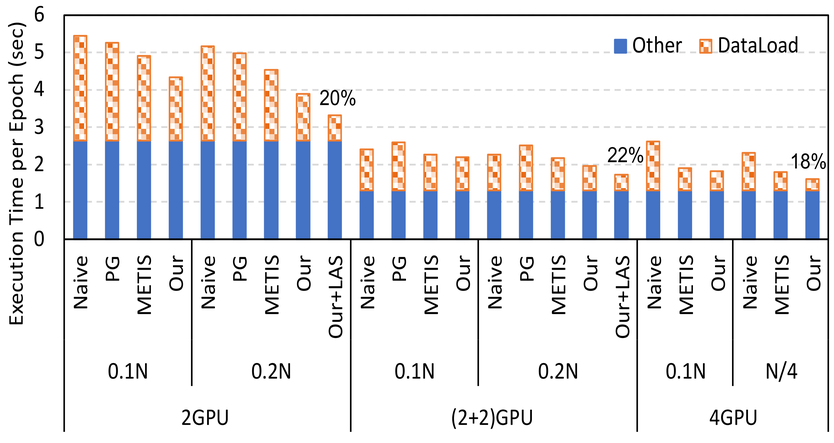


Rules

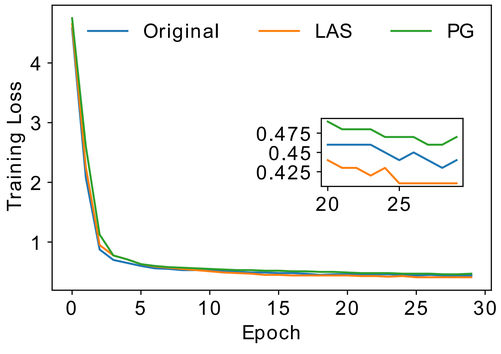
- Keep at least one copy of the node on GPUs
- Select the GPU with the lowest sum of sampling probabilities of new nodes in each round of replacement

Evaluation

- **Speedup**



- **Overhead**



- **Observation**
  - Loading the input features is a performance bottleneck when the GPU buffer is small
  - Our data placement strategy achieves smaller data loading time than both PaGraph and DGL without losing accuracy

Shihui Song and Peng Jiang