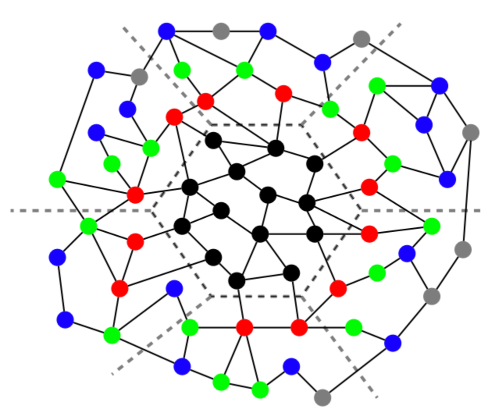
Reducing Communication in Graph Neural Network Training

Alok Tripathy, Katherine Yelick, Aydın Buluç
University of California, Berkeley
Lawrence Berkeley National Laboratory

Introduction

- GNN models are too big to fit on one GPU for large graphs.
- Minibatching could help, but neighborhood explosion causes space issues even for shallow networks
  - For L-layer GNN, need L-hop neighborhood of minibatch of vertices



[1]

- Lots of work subsample the aggregated L-hop neighborhood
- **We distribute L-hop neighborhood computations using communication-avoiding matrix multiplication algorithms**
- Contributions
  - Formulate GNN training as a series of sparse-dense matrix multiplication (both forward and backward propagation)
  - Use distributed matrix multiplication algorithms that provably reduce communication with increasing process counts
- Focus on GCNs, full-batch training, and node classification, but techniques generalize

GCN Training as Matrix Multiplication

- Forward propagation

$$\mathbf{Z}^l \leftarrow \mathbf{A}^\mathsf{T}\mathbf{H}^{l-1}\mathbf{W}^l$$

$$\mathbf{H}^l \leftarrow \sigma(\mathbf{Z}^l)$$

Backward propagation

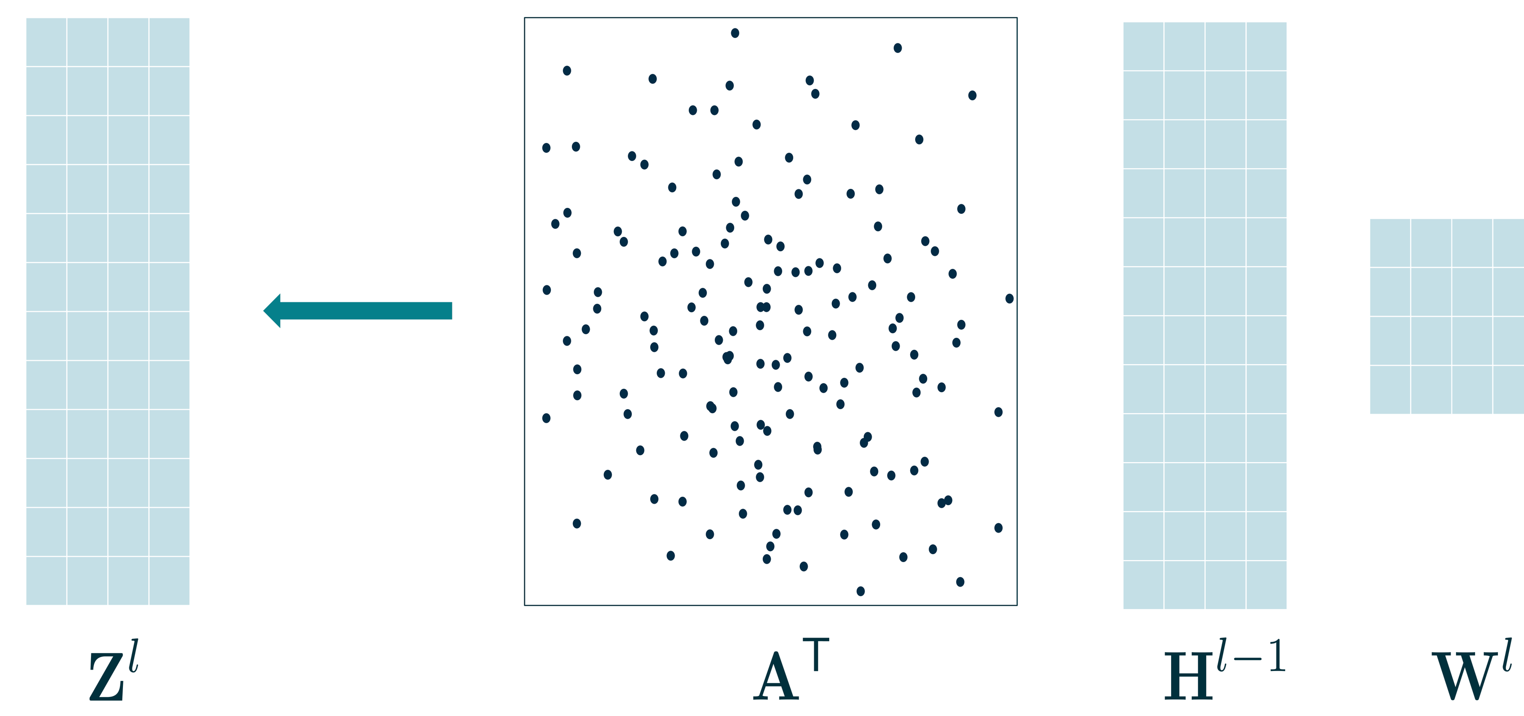
$$\mathbf{G}^{l-1} \leftarrow \mathbf{A}\mathbf{G}^l(\mathbf{W}^l)^\mathsf{T} \odot \sigma'(\mathbf{Z}^{l-1})$$

$$\mathbf{Y}^{l-1} \leftarrow (\mathbf{H}^{l-1})^\mathsf{T}\mathbf{A}\mathbf{G}^l$$

- $\mathbf{A}$ is stored in sparse format, everything else dense
- **All operations are either SpMM or DGEMM**

Symbols and Notations	
Symbol	Description
$\mathbf{A}$	Modified adjacency matrix of graph $(n \times n)$
$\mathbf{H}^l$	Embedding matrix in layer l $(n \times f)$
$\mathbf{W}^l$	Weight matrix in layer l $(f \times f)$
$\mathbf{Y}^l$	Matrix form of $\frac{\partial \mathcal{L}}{\partial W^l_{ij}}$ $(f \times f)$
$\mathbf{Z}^l$	Input matrix to activation function $(n \times f)$
$\mathbf{G}^l$	Matrix form of $\frac{\partial \mathcal{L}}{\partial Z^l_{ij}}$ $(n \times f)$
σ	Activation function
f	Length of feature vector per vertex
f_u	Feature vector for vertex u
L	Total layers in GNN
P	Total number of processes
α	Latency
β	Reciprocal bandwidth

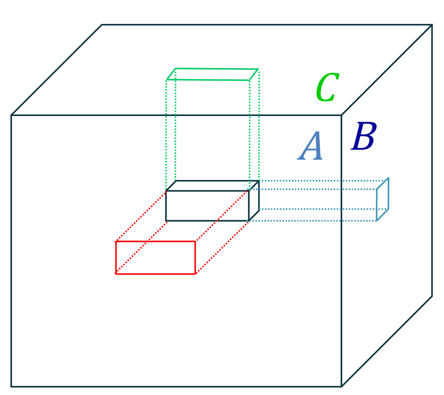
Bottleneck of GCN Training

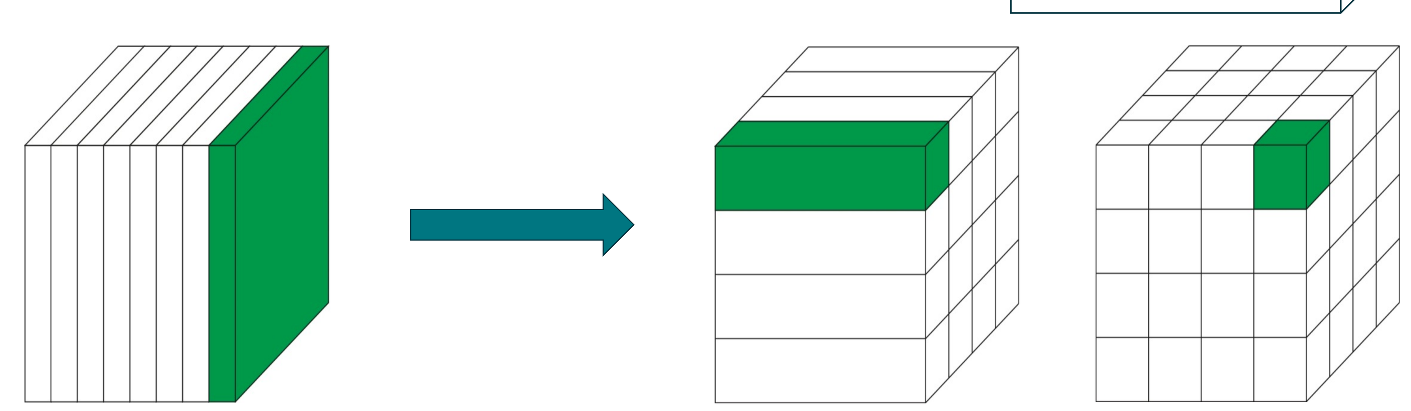


- $\mathbf{A}^\mathsf{T}\mathbf{H}^{l-1}$ --- sparse-dense matmul (SpMM)
- $(\mathbf{A}^\mathsf{T}\mathbf{H}^{l-1})\mathbf{W}^l$ --- dense-dense matmul (DGEMM)
- **SpMM is the bottleneck with much more work than DGEMM**
- Can use distributed SpMM algorithms to accelerate workload

Distributed Matrix Multiplication Algorithms

- Can view matrix multiplication as a computation cube
- Distribute computation by slicing cube
- 4 types of distributed SpMM algorithms
  - (1D, 1.5D, 2D, 3D algorithms)





1D algorithms 1.5D algorithms 2D algorithms 3D algorithms

- We implement GCN training which each of these distributed SpMM algorithms.

Communication Analyses			
Algorithm	Latency	Bandwidth	Memory
1D	$\lg P + 2P$	$2nf + f^2$	$\frac{nnz(\mathbf{A})L}{P} + \frac{nf}{P}$
1.5D	$2\frac{P}{c^2}\lg\frac{P}{c^2}$	$\frac{2nf}{c} + \frac{2nfc}{P}$	$\frac{nnz(\mathbf{A})L}{P} + \frac{nfc}{P}$
2D	$5\sqrt{P} + 3\lg P$	$\frac{8nf}{\sqrt{P}} + \frac{2nnz(\mathbf{A})}{\sqrt{P}}$	$\frac{nnz(\mathbf{A})L}{P} + \frac{nf}{P}$
3D	$4P^{1/3}$	$\frac{2nnz(\mathbf{A})}{P^{2/3}} + \frac{12nf}{P^{2/3}}$	$\frac{nnz(\mathbf{A})L}{P} + \frac{nfc}{P}$

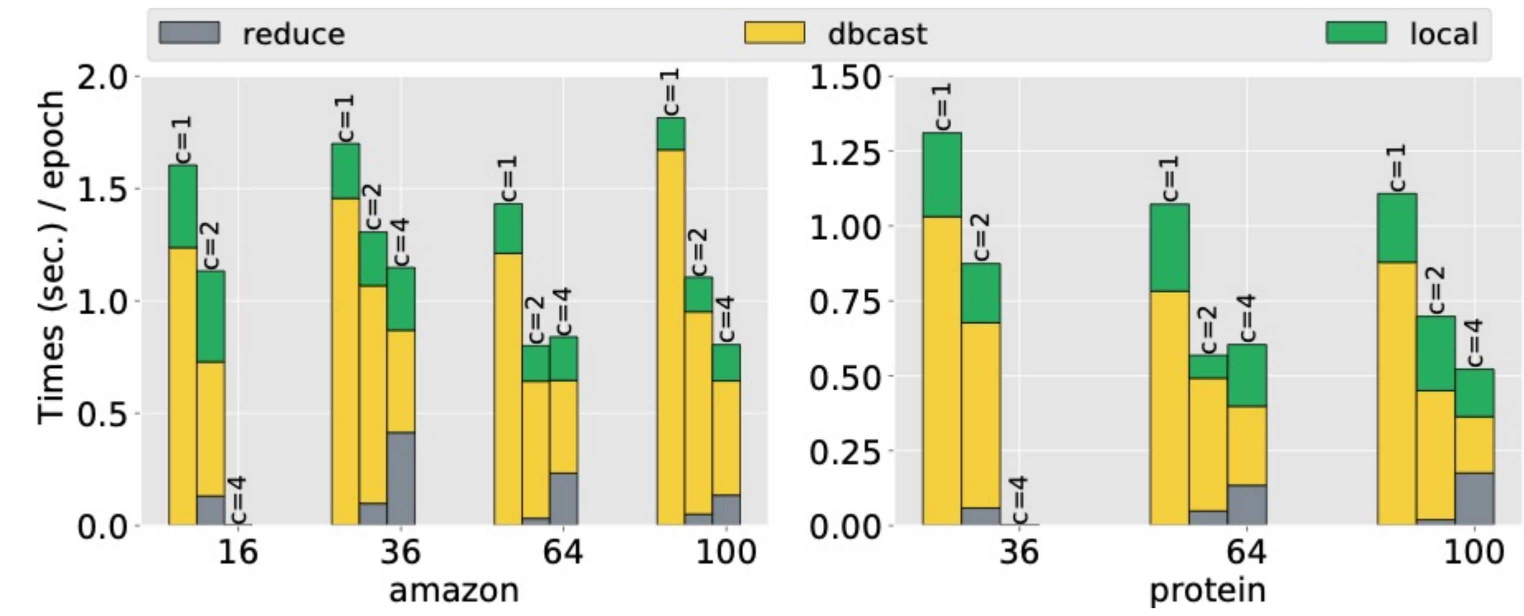
- **All algorithms except 1D provably reduce communication volume with process counts**

Implementation Details

- PyTorch 1.3 with NCCL 2.0 Backend
  - Kipf-Welling GCN model (3-layers, 16 hidden activations)
- System
  - Summit supercomputer at Oak Ridge Computing Facility
  - 6 NVIDIA V100s per node
  - NVLink 2.0, EDR Infiniband
- Datasets

Name	Vertices	Edges	Features	Labels
Amazon	14M	231M	300	24
Reddit	233K	114M	602	41
Protein	8M	2B	128	256

Results (1.5D GCN Training)



- Scales with both P and c (replication factor) with 1GPU/node
  - Full 6GPU/node results in paper
  - Expect to scale with all GPUs/node on future architectures (e.g. Perlmutter)
- Full results (including 1D, 2D, 3D) in paper, though 1.5D performed best

Conclusions

- GNN models can't fit on one device for large graphs
- Most work subsamples computation, but we distribute the computation
- **We formulate GCN training with matrix multiplication, and use communication-avoiding matrix multiplication algorithms to distribute bottlenecks in GCN training**
- Code: https://github.com/PASSIONLab/CAGNET
- Paper: https://arxiv.org/pdf/2005.03300.pdf [2]

References

[1] Marghoob Mohiyuddin, Mark Hoemmen, James Demmel, and Katherine Yelick. Minimizing Communication in Sparse Matrix Solvers. In Proceedings of the 2009 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis(SC), 2009.

[2] Alok Tripathy, Katherine Yelick, Aydın Buluç. Minimizing Communication in Sparse Matrix Solvers. In Proceedings of the 2020 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis(SC), 2020.