Towards Efficient Large-Scale Graph Neural Network Computing
Lingxiao Ma, Zhi Yang, Youshan Miao, Jilong Xue, Ming Wu, Lidong Zhou, Yafei Dai
Peking University, Microsoft Research Beijing
2018

Presenter : Derrick Blakely

https://qdata.github.io/deep2Read
Outline

1. Motivation
2. N Gra Programming Abstraction
3. N Gra System
4. Parallel Processing with Multiple GPUs
5. Evaluation
6. Conclusion
Outline

1. Motivation
2. NGra Programming Abstraction
3. NGra System
4. Parallel Processing with Multiple GPUs
5. Evaluation
6. Conclusion
GNNs and GPUs

- Graphs too large to fit in GPU memory
- Irregular inputs make SIMD difficult
- Sparse matrices
GNNs and DL Frameworks

- No simple programming interface for training GNNs
- Lots of time spent allocating computation graphs
- Bad job mapping computation graphs to GPU
- Simple data-parallelism isn’t the best way to parallelize GNNs
GNNs and DL Frameworks
Older Graph Libraries

- Pregel, PowerGraph, GraphLab, GraphX, etc
- Define vertex program, use Gather-Apply-Scatter (GAS)
- Don’t allow users to define the “dataflow”
- Don’t use efficient tensor libraries
- Don’t support NN architectures
- Use scalar vertex features, not feature vectors
Outline

1. Motivation
2. NGra Programming Abstraction
3. NGra System
4. Parallel Processing with Multiple GPUs
5. Evaluation
6. Conclusion
Example: Gated GCN

\[ h_u^{l+1} = \text{ReLU}(W^l(\sum_{v \rightarrow u} \sigma(W_H^l h_u^l + W_C^l h_v^l) \odot h_v^l)) \]

- Main observation: GNN layers can be split into edge functions and vertex functions
- Edge function: inside the summation
- Vertex function: outside the summation
SAGA - four stages of computation

1. Scatter
2. ApplyEdge
3. Gather
4. ApplyVertex
Scatter

\[ [h_{v0}, h_{v1}, data] \]

\[ [h_{v2}, h_{v1}, data] \]

Presenter: Derrick Blakely (University of Virginia)

Towards Efficient Large-Scale Graph Neural Networks

https://qdata.github.io/deep2Read
ApplyEdge
Gather
ApplyVertex
Outline

1. Motivation
2. NGra Programming Abstraction
3. NGra System
4. Parallel Processing with Multiple GPUs
5. Evaluation
6. Conclusion
Chunk-Based Streaming Dataflow

- Partition graph into chunks that fit in GPU memory
- $C_{ij}$: edge chunk connecting vertex chunks $V_i$ and $V_j$

NGra System

Presenter: Derrick Blakely (University of Virginia)
Chunk-Based Streaming Dataflow

2D Graph Partition

Vertex Feature

Edge Chunk

V0
C00
C10
C20

V1

V2
C10

ApplyEdge

scatter

Dest

W_c

mat mul

add

sig

mul

Gather

A0

A0
Dataflow Scheduling
Outline

1. Motivation
2. NGra Programming Abstraction
3. NGra System
4. Parallel Processing with Multiple GPUs
5. Evaluation
6. Conclusion
Ring-Based Parallel Streaming

- CPU/DRAM
- PCIe Host Bridge
- PCIe Switch
- GPU

Step 1: Load 1
Step 2: Load 2
Step 3: Load 3
Step 4: Load 4
Step 5: Load 5
Step 6: Load 6

maximal bandwidth fat-tree
original bandwidth tree

Presenter: Derrick Blakely (University of Virginia)
Towards Efficient Large-Scale Graph Neural Networks
https://qdata.github.io/deep2Read
Outline

1 Motivation
2 NGra Programming Abstraction
3 NGra System
4 Parallel Processing with Multiple GPUs
5 Evaluation
6 Conclusion
## Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>vertex#</th>
<th>edge#</th>
<th>feature</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>pubmed</td>
<td>19.7K</td>
<td>108.4K</td>
<td>500</td>
<td>3</td>
</tr>
<tr>
<td>protein</td>
<td>43.5K</td>
<td>205.6K</td>
<td>29</td>
<td>3</td>
</tr>
<tr>
<td>BlogCatalog</td>
<td>10.3K</td>
<td>668.0K</td>
<td>128</td>
<td>39</td>
</tr>
<tr>
<td>reddit_small</td>
<td>46.6K</td>
<td>1.4M</td>
<td>602</td>
<td>41</td>
</tr>
<tr>
<td>reddit_middle</td>
<td>233.0K</td>
<td>23.2M</td>
<td>602</td>
<td>41</td>
</tr>
<tr>
<td>reddit_full</td>
<td>2.2M</td>
<td>571.0M</td>
<td>300</td>
<td>50</td>
</tr>
<tr>
<td>enwiki</td>
<td>3.2M</td>
<td>222.1M</td>
<td>300</td>
<td>12</td>
</tr>
</tbody>
</table>

- Pubmed citation network
- Protein-protein interaction graphs
- BlogCatalog Social network
- Reddit online discussion forum
- Wikidata
Ngram and TF diverge with increasing sparsity
Ring-based streaming is effective

**Graphs**

- **GCN enwiki**
- **GG-NN enwiki**
- **CommNet enwiki**

- **Speed up** vs **GPU#**
- **Ring-based** vs **non-ring**

**Legend**
- *ring-based*
- *non-ring*
N-Gram scheduling strategy is effective

![Graph showing runtime comparison between different scheduling strategies for various graph neural network models.](image-url)
Outline

1. Motivation
2. NGRA Programming Abstraction
3. NGRA System
4. Parallel Processing with Multiple GPUs
5. Evaluation
6. Conclusion
Weaknesses

- No available source code
- No reason to use their tool when we can use PyTorch Geometric
- Ring-based streaming is just a simple heuristic
- Doesn’t scale to multi-host setting
- Evaluations aren’t very exciting - we already knew TF wasn’t great at this
Lessons Learned

- More exotic NN architectures can be handled by TF/PyTorch, provided we efficiently create and schedule their computation graphs.
- Reducing sparsity important for efficient GPU use.
- Vertex-centric programming model is becoming more and more prominent.