ACCELERATING GRAPH ALGORITHMS WITH RAPIDS

Joe Eaton, Ph.D.  Technical Lead for Graph Analytics
AGENDA

• Introduction - Why Graph Analytics?
• Graph Libraries - nvGraph and cuGraph
• Graph Algorithms - What’s New
• Conclusion - What’s Next
How do I get the software?

- https://github.com/rapidsai
- https://anaconda.org/rapidsai/
- https://pypi.org/project/cudf
- https://pypi.org/project/cuml

- https://hub.docker.com/r/rapidsai/rapidsai/
WHY GRAPH ANALYTICS

Cyber Security

1. Build a User-to-User Activity Graph
   • Property graph with temporal information

2. Compute user behavior changes over time
   • PageRank - changes in user’s importance
   • Jaccard Similarity - changes in relationship to others
   • Louvain - changes in social group, groups of groups
   • Triangle Counting - changes in group density

3. Look for anomalies
WHAT IS NEEDED

• Fast Graph Processing

• Use GPUs  (Shameless Marketing)
32GB V100 DGX-1
Now with 256GB of GPU Memory

1 PFLOPS | 8x Tesla V100 32GB | 300 GB/s NVLink Hybrid Cube Mesh
2x Xeon | 8 TB RAID 0 | Quad IB 100Gbps, Dual 10GbE | 3U – 3200W
DGX-1 HYBRID-CUBE MESH
DGX-2

16x Tesla V100 32GB
12x NVSwitch

NVLink Plane Card
8x EDR IB/100 GigE
2x Xeon Platinum
1.5TB System Memory
30TB NVME SSDs
PCIe Switch Complex

2 PFLOPS | 512GB HBM2 | 16 TB/sec Memory Bandwidth | 10 kW / 160 kg
DGX-2 INTERCONNECT

16 Tesla V100 32GB Connected by NVSwitch | On-chip Memory Fabric Semantic Extended Across All GPUs

Every GPU-to-GPU at 300 GB/sec
NVGRAPH IN RAPIDS

Keep What you have Invested in Graph Analytics

More! GPU Optimized Algorithms

Reduced cost & Increased performance

Integration with RAPIDS data IO, preparation and ML methods

Performance Constantly Improving
GRAPH ANALYTIC FRAMEWORKS

For GPU Benchmarks

• Gunrock from UC Davis
• Hornet from Georgia Tech (also HornetsNest)
• nvGraph from NVIDIA
PAGERANK

• Ideal application: influence in social networks
• Each iteration involves computing:
  \[ y = Ax \]
  \[ x = y / \text{norm}(y) \]
• Merge-path load balancing for graphs
• Power iteration for largest eigenpair by default
• Implicit Google matrix to preserve sparsity
• Advanced eigensolvers for ill-conditioning
PAGERANK PIPELINE BENCHMARK
Graph Analytics Benchmark

Proposed by MIT LL.

Apply supercomputing benchmarking methods to create scalable benchmark for big data workloads.

Four different phases that focus on data ingest and analytic processing, details on next slide....

Reference code for serial implementations available on GitHub. https://github.com/NVIDIA/PRBench
High Performance Exact Triangle Counting on GPUs
Mauro Bisson and Massimiliano Fatica

Useful for:
- **Community Strength**
- Graph statistics for summary
- Graph categorization/labeling

Figure 1. Example of triangle counting via multiplication of the two halves of an adjacency matrix. The sum is restricted to only the grey elements of the original $L$ matrix.
TRAVERSAL/BFS

Common Usage Examples:

Path-finding algorithms:
- Navigation
- Modeling
- Communications Network

Breadth first search
building block
fundamental graph primitive

Graph 500 Benchmark
BFS PRIMITIVE
Load balancing

Frontier: 4 8 9 5

Corresponding vertices degree: 2 3 0 1

Exclusive sum: 0 2 5 5 6

k = max (k’ such as exclusivesum[k] <= thread_idx)
For this thread:
source = frontier[k]
Edge_index = row_ptr[source] + thread_idx - exclusivesum[k]
BOTTOM UP

Motivation

• Sometimes it’s better for children to look for parents (bottom-up)
CLUSTERING ALGORITHMS

- **Spectral**
  Build a matrix, solve an eigenvalue problem, use eigenvectors for clustering

- **Hierarchical / Agglomerative**
  Build a hierarchy (fine to coarse), partition coarse, propagate results back to fine level

- **Local refinements**
  Switch one node at a time
SPECTRAL EDGE CUT MINIMIZATION

80% hit rate

Balanced cut minimization

Ground truth
SPECTRAL MODULARITY MAXIMIZATION

84% hit rate

Spectral Modularity maximization

Ground truth

A. Fender, N. Emad, S. Petiton, M. Naumov. 2017. “Parallel Modularity Clustering.” ICCS
HEIRARCHICAL LOUVAIN CLUSTERS

Check the size of each cluster
If size > threshold: recluster

Dict = {'0': initial clusters,
        '1': reclustering on data from '0',
        '2': reclustering on data from '1' ...... }
LOUVAIN SINGLE RUN

Speed up comparing to Networkx

speed up

preferentialAtta...
caidaRouterLevel
coAuthorsDBLP
dblp-2010
citationCiteseer
coPapersDBLP
coPapersCitese...
as-Skitter
# 32GB V100

Single and Dual GPU on Commodity Workstation

<table>
<thead>
<tr>
<th>RMAT</th>
<th>Nodes</th>
<th>Edges</th>
<th>Single</th>
<th>Dual</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1,048,576</td>
<td>16,777,216</td>
<td>0.019</td>
<td>0.020</td>
</tr>
<tr>
<td>21</td>
<td>2,097,152</td>
<td>33,554,432</td>
<td>0.047</td>
<td>0.035</td>
</tr>
<tr>
<td>22</td>
<td>4,194,304</td>
<td>67,108,864</td>
<td>0.114</td>
<td>0.066</td>
</tr>
<tr>
<td>23</td>
<td>8,388,608</td>
<td>134,217,728</td>
<td>0.302</td>
<td>0.162</td>
</tr>
<tr>
<td>24</td>
<td>16,777,216</td>
<td>268,435,456</td>
<td>0.771</td>
<td>0.353</td>
</tr>
<tr>
<td>25</td>
<td>33,554,432</td>
<td>536,870,912</td>
<td>1.747</td>
<td>0.821</td>
</tr>
<tr>
<td>26</td>
<td>67,108,864</td>
<td>1,073,741,824</td>
<td>1.880</td>
<td></td>
</tr>
</tbody>
</table>

Scale 26 on a single GPU can be achieved by using Unified Virtual Memory. Runtime was 3.945 seconds. Larger sizes exceed host memory of 64GB.
# DATASETS

Mix of social network and RMAT

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nodes</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>soc-twitter-2010</td>
<td>21,297,772</td>
<td>530,051,618</td>
</tr>
<tr>
<td>Twitter.mtx</td>
<td>41,652,230</td>
<td>1,468,365,182</td>
</tr>
<tr>
<td>RMAT - Scale 26</td>
<td>67,108,864</td>
<td>1,073,741,824</td>
</tr>
<tr>
<td>RMAT - Scale 27</td>
<td>134,217,728</td>
<td>2,122,307,214</td>
</tr>
<tr>
<td>RMAT - Scale 28</td>
<td>268,435,456</td>
<td>4,294,967,296</td>
</tr>
</tbody>
</table>
FRAMEWORK COMPARISON
PageRank on DGX-1, Single GPU

Framework Comparison for PageRank
soc-tweet data

- Gunrock
- Hornet
- nvGraph
PAGERANK ON DGX-1
Using Gunrock, Multi-GPU
BFS ON DGX-1
Using Gunrock, Multi-GPU

BFS Runtimes

Seconds

soc-twitter
0.10 0.06 0.04 0.04

twitter
0.49 0.26 0.17 0.12

RMAT 26
0.40 0.32 0.15 0.19

1 GPU 2 GPUs 4 GPUs 8 GPUs
DGX-2

PageRank Runtimes

Number of GPUS

Seconds

soc-twitter | RMAT 27 | RMAT 28

1 | 6.686 | 1.703 | 17.189
2 | 3.445 | 0.945 | 8.850
4 | 0.621 | 0.203 | 4.554
8 | 0.428 | 0.106 | 2.515
10 | 1.934 | 0.363 | 0.886
16 | 0.886 | 0.363 |
DGX-1 VS. DGX-2

PageRank Twitter Dataset Runtime

Seconds

Number of GPUs

16.729
15.826
9.299
8.760
5.583
5.158
4.596
3.152
2.647

DGX-1
DGX-2
**RMAT SCALING, STAGE 4 PRBENCH PIPELINE**

Near Constant Time Weak Scaling is Real Due to NVLINK

<table>
<thead>
<tr>
<th>GPU Count</th>
<th>Max RMAT scale</th>
<th>Comp time (sec)</th>
<th>Gedges/sec</th>
<th>MFLOPS</th>
<th>NVLINK Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
<td>1.4052</td>
<td>7.6</td>
<td>15282.90</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>26</td>
<td>1.3914</td>
<td>15.4</td>
<td>30867.37</td>
<td>1.4</td>
</tr>
<tr>
<td>4</td>
<td>27</td>
<td>1.3891</td>
<td>30.9</td>
<td>61838.78</td>
<td>2.8</td>
</tr>
<tr>
<td>8</td>
<td>28</td>
<td>1.4103</td>
<td>60.9</td>
<td>121815.46</td>
<td>4.1</td>
</tr>
<tr>
<td>16</td>
<td>29</td>
<td>1.4689</td>
<td>117.0</td>
<td>233917.04</td>
<td>8.1</td>
</tr>
</tbody>
</table>
WHAT’S NEXT?
Ease of Use, Multi-GPU, new algorithms
HORNET

- Designed for sparse and irregular data - great for powerlaw datasets

- Essentially a multi-tier memory manager
  - Works with different block sizes -- always powers of two (ensures good memory utilization)
  - Supports memory reclamation
  - Superfast!

- Hornet in RAPIDS: Will be part of cuGraph.
  - Streaming data analytics and GraphBLAS good use cases.
  - Data base operations such as join size estimation.
  - String dictionary lookups, fast text indexing.
**HORNET**

Performance - Edge Insertion

- Results on the NVIDIA P100 GPU
- Supports over 150M updates per second
  - Checking for duplicates
  - Data movement (when newer block needed)
  - Memory reclamation
- Similar results for deletions
Generality
  • Supports many algorithms

Programmability
  • Easy to add new methods

Scalability
  • Multi-GPU support

Performance
  • Competitive with other GPU frameworks

https://gunrock.github.io
Programming Model

A generic graph algorithm:

A group of V or E
Do something
Resulting group of V or E

Loop until convergence

A group of V or E
Do something
Another resulting group of V or E

Data-centric abstraction
- Operations are defined on a group of vertices or edges & a frontier
=> Operations = manipulations of frontiers

Bulk-synchronous programming
- Operations are done one by one, in order
- Within a single operation, computing on multiple elements can be done in parallel, without order

Yangzihao Wang, Yuechao Pan, Andrew Davidson, Yuduo Wu, Carl Yang, Leyuan Wang, Muhammad Osama, Chenshan Yuan, Weitang Liu, Andy T. Riffel, and John D. Owens.
CONCLUSIONS
We Can Do Real Graphs on GPUs!

- The benefits of full NVLink connectivity between GPUs is evident with any analytic that needs to share data between GPUs.
- DGX-2 is able to handle graphs scaling into the billions of edges.
- Frameworks need to be updated to support more than 8 GPUs, some have hardcoded limits due to DGX-1.
- More to come! We will be building ease-of-use features with high priority, we can already share data with cuML and cuDF.