# 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

# RAPIDS

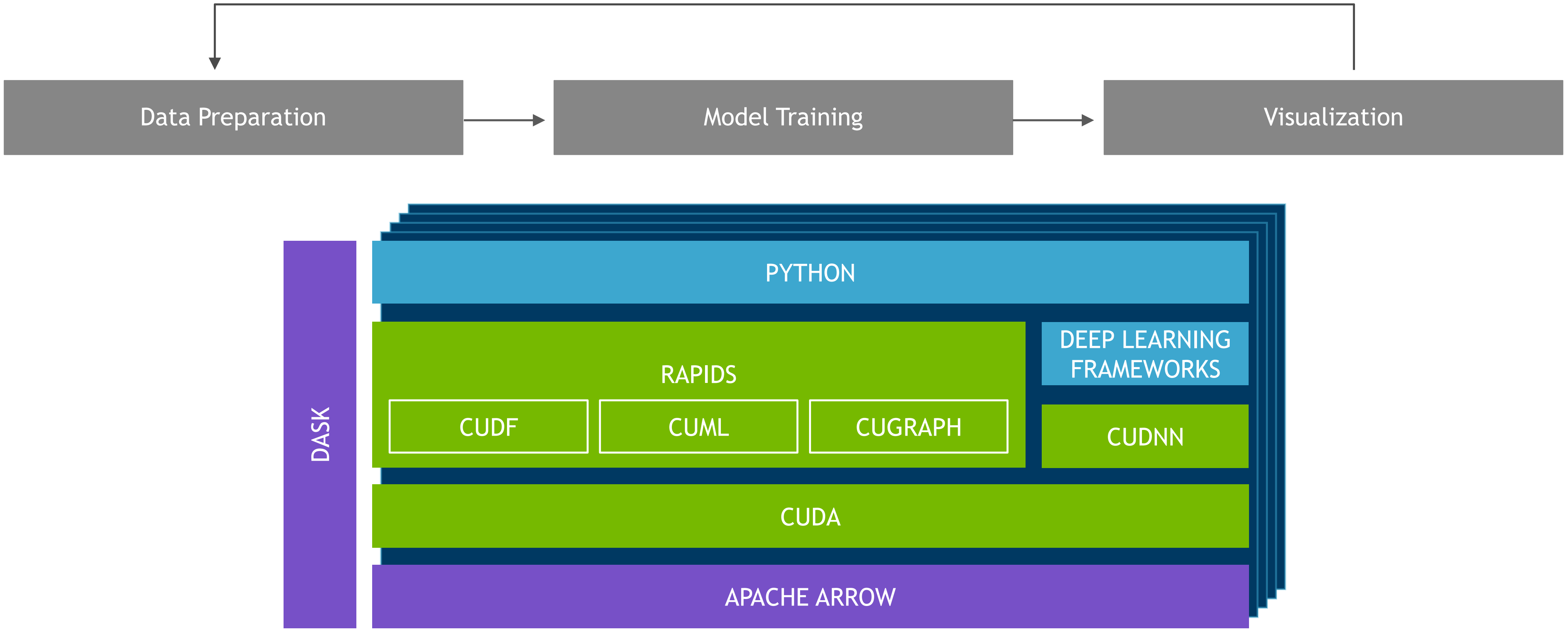
## How do I get the software?

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

- https://ngc.nvidia.com/registry/nvidia-rapidsai-rapidsai
- https://hub.docker.com/r/rapidsai/rapidsai/

# RAPIDS — OPEN GPU DATA SCIENCE

## Software Stack



# WHY GRAPH ANALYTICS

## Cyber Security

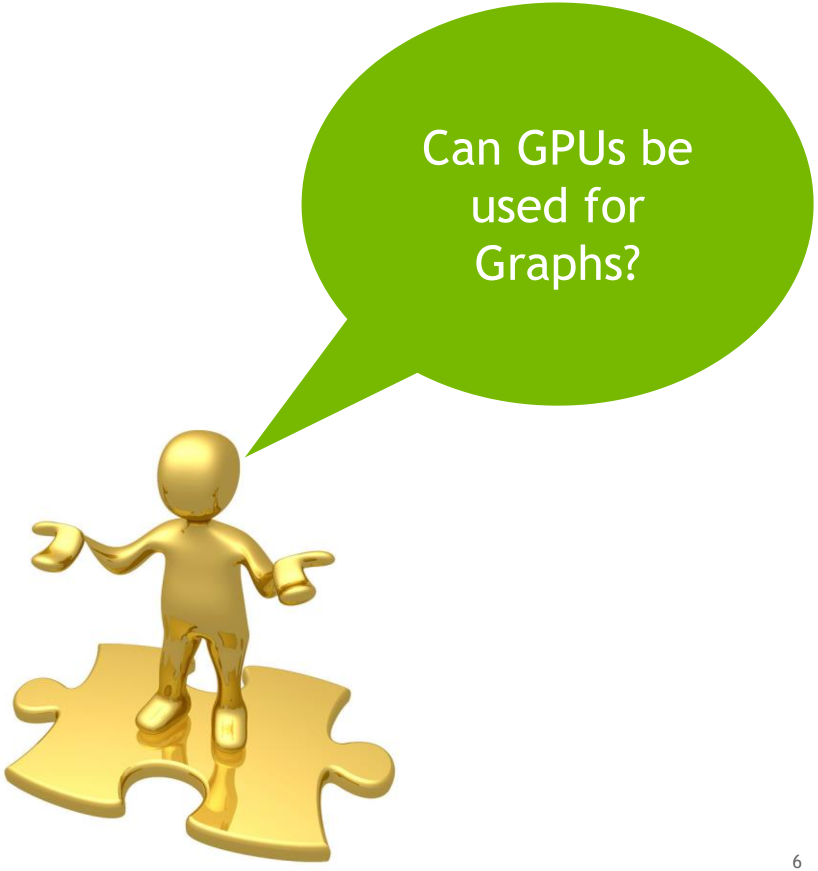
1. Build a User-to-User Activity Graph
   - Property graph with temporal information

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

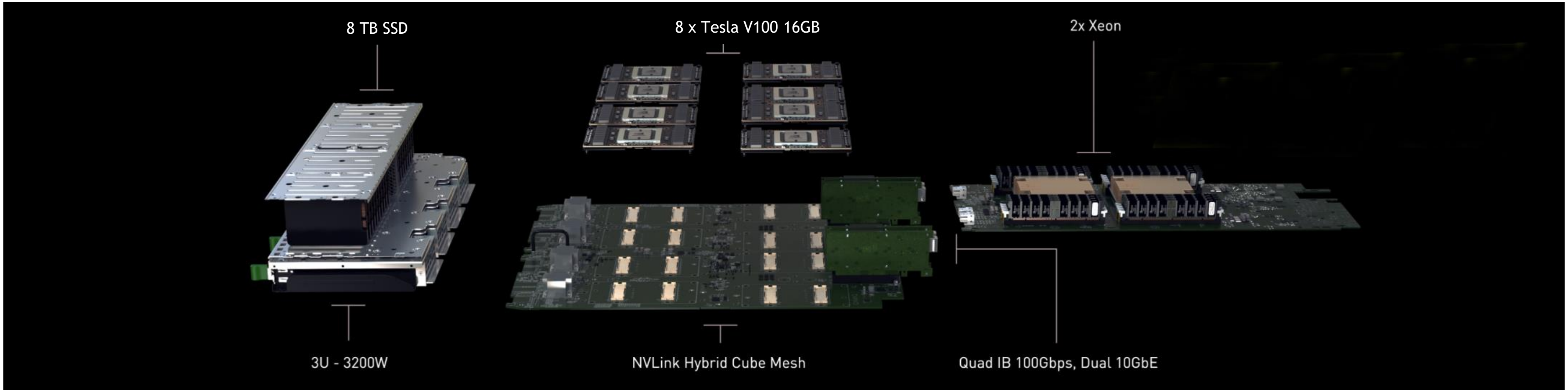
# WHAT IS NEEDED

- Fast Graph Processing
- Use GPUs (Shameless Marketing)



# 32GB V100 DGX-1

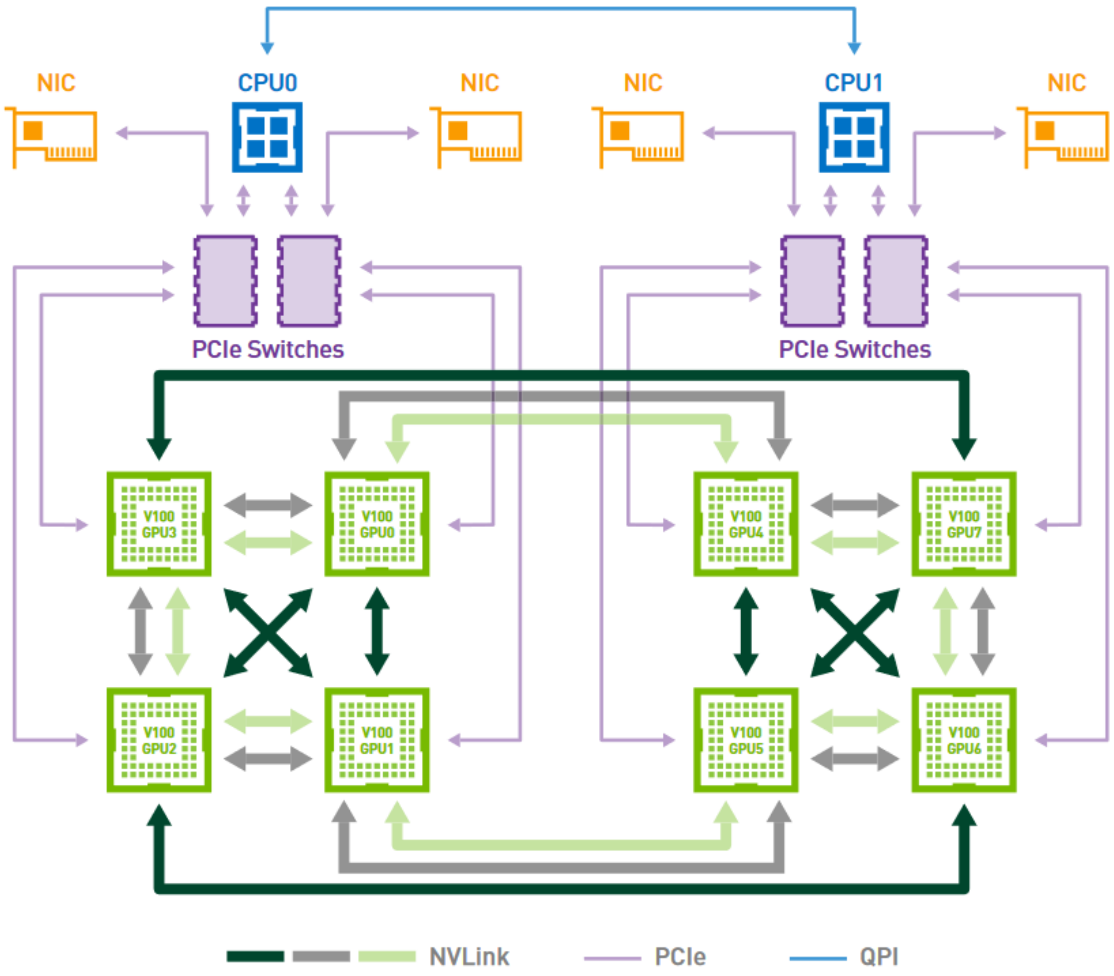
## Now with 256GB of GPU Memory




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

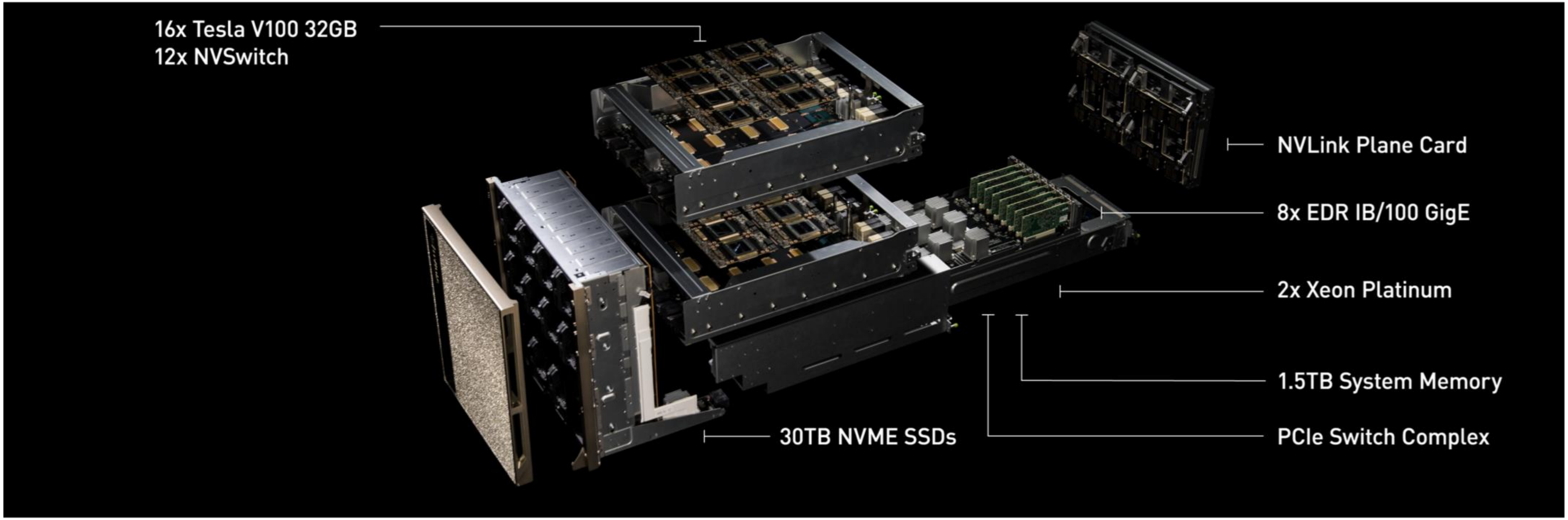
# DGX-1 HYBRID-CUBE MESH





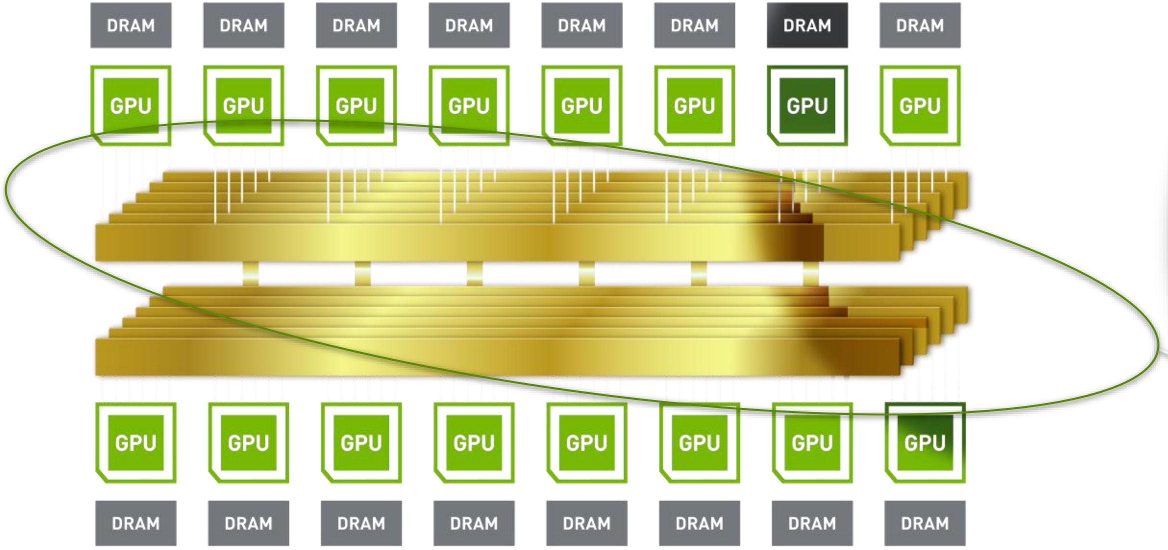
NVIDIA

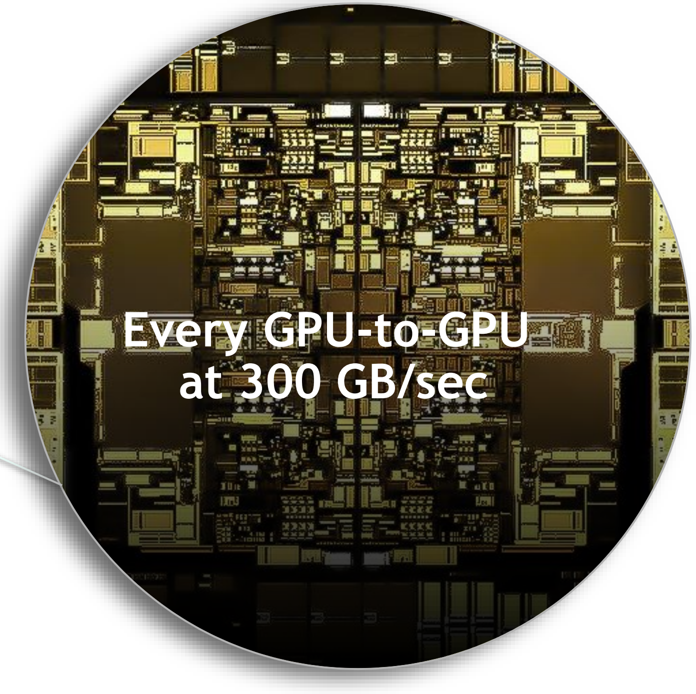
# DGX-2




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

# DGX-2 INTERCONNECT



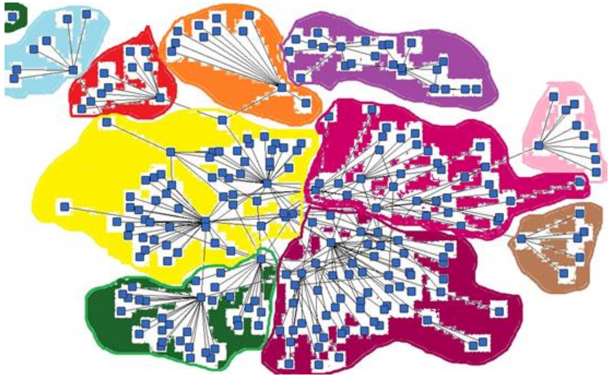



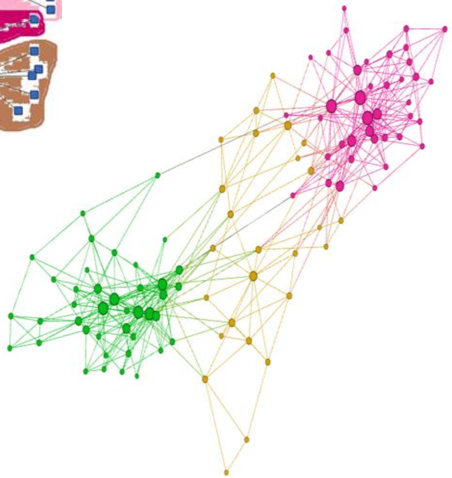

16 Tesla V100 32GB Connected by NVSwitch | On-chip Memory Fabric Semantic Extended Across All GPUs

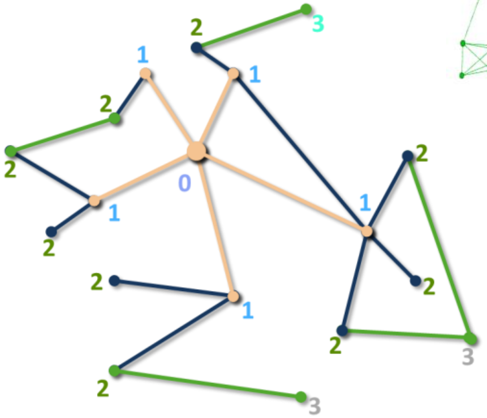
# NVGRAPH IN RAPIDS

## 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

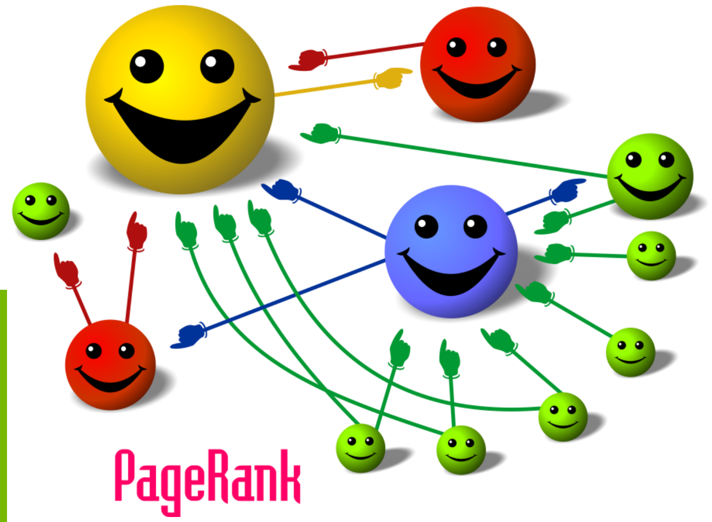
# PAGERANK

- Ideal application: influence in social networks

- Each iteration involves computing:

$$y = A\,x$$
$$x = y/norm(y)$$

- 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

# TRIANGLE COUNTING

*High Performance Exact Triangle Counting on GPUs*
Mauro Bisson and Massimiliano Fatica

Useful for:

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

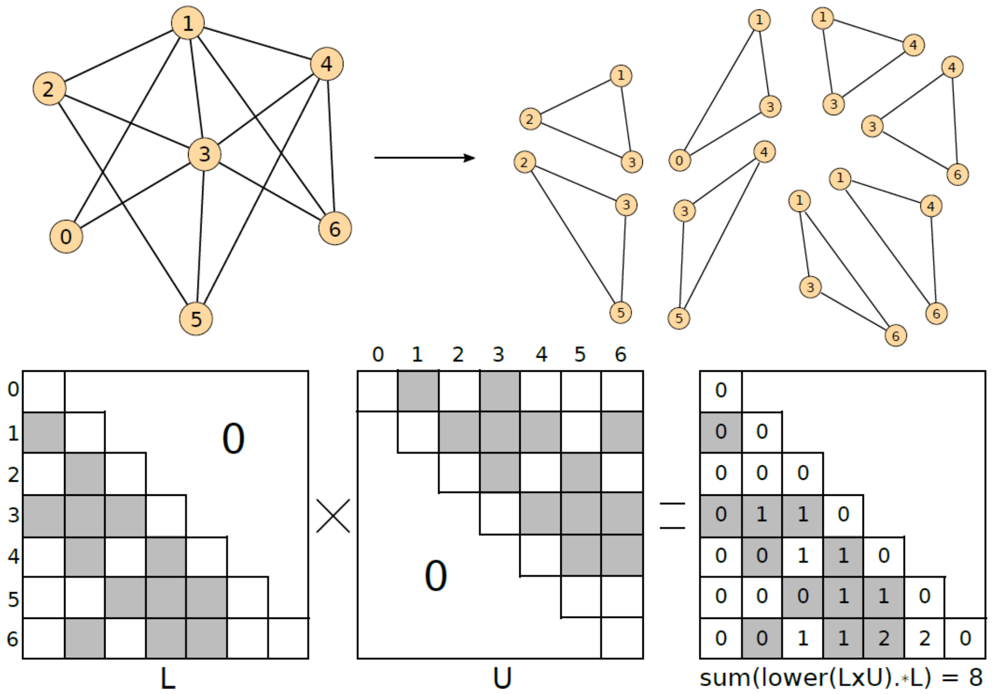



Figure 1. Example of triangle counting via multiplication of the two halves of an adjacency matrix. The sum is restricted to only the grey elements of the original $L$ matrix.

# TRAVERSAL/BFS

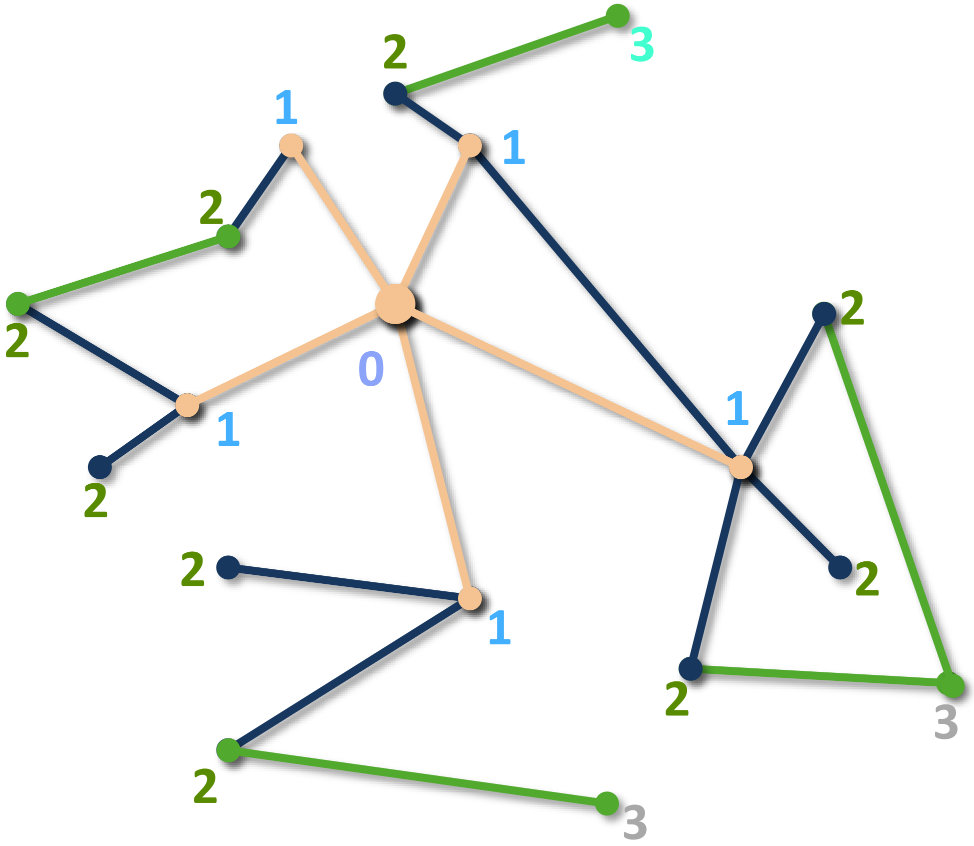
Common Usage Examples:

Path-finding algorithms:

- Navigation
- Modeling
- Communications Network

Breadth first search
building block
fundamental graph primitive

Graph 500 Benchmark



# BFS PRIMITIVE

## Load balancing

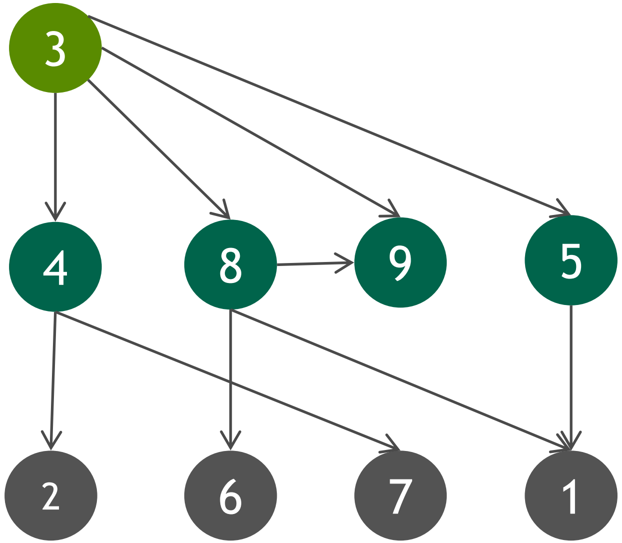
Frontier : 4 8 9 5

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]

Binary search

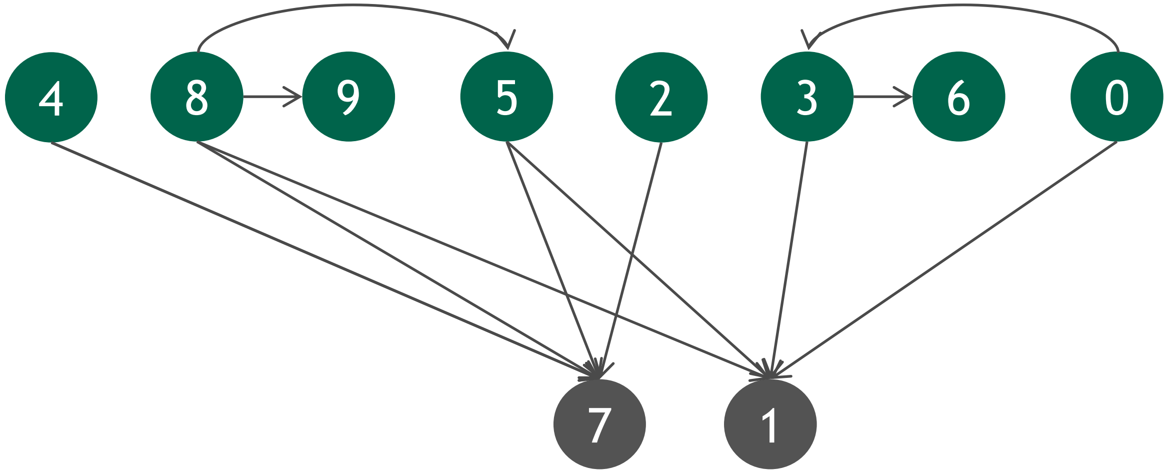
# BOTTOM UP

## Motivation

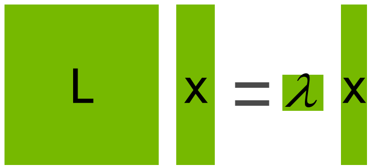
- Sometimes it’s better for children to look for parents (bottom-up)

Frontier depth=3

Frontier depth=4



# CLUSTERING ALGORITHMS

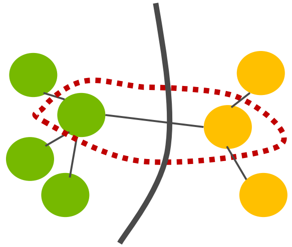



- Spectral

  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

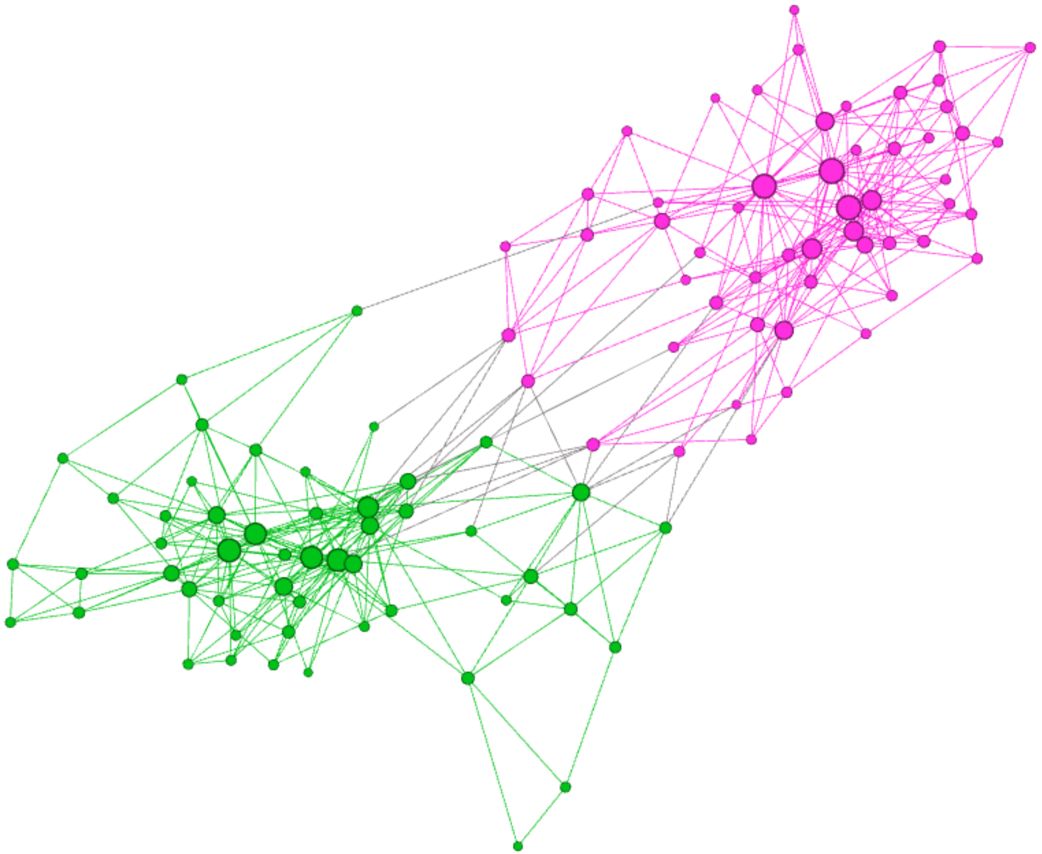


- Local refinements

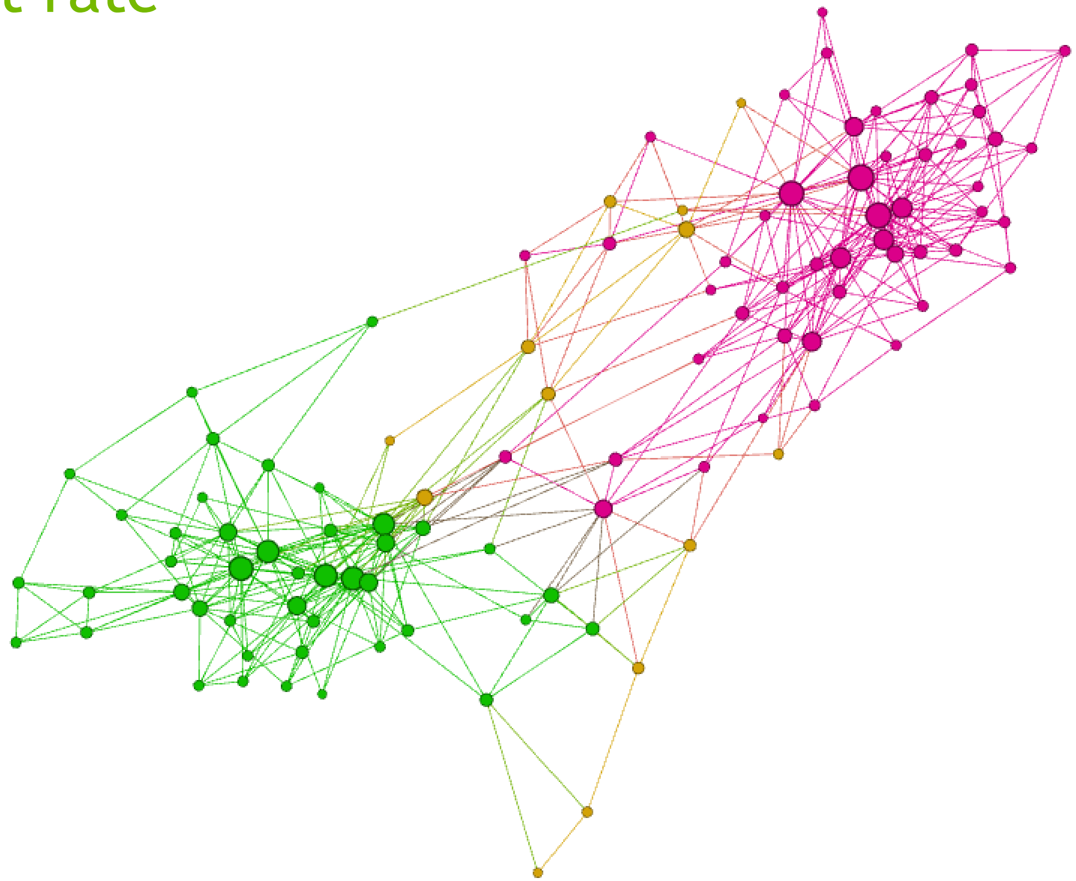
  Switch one node at a time

# SPECTRAL EDGE CUT MINIMIZATION

80% hit rate



Balanced cut minimization

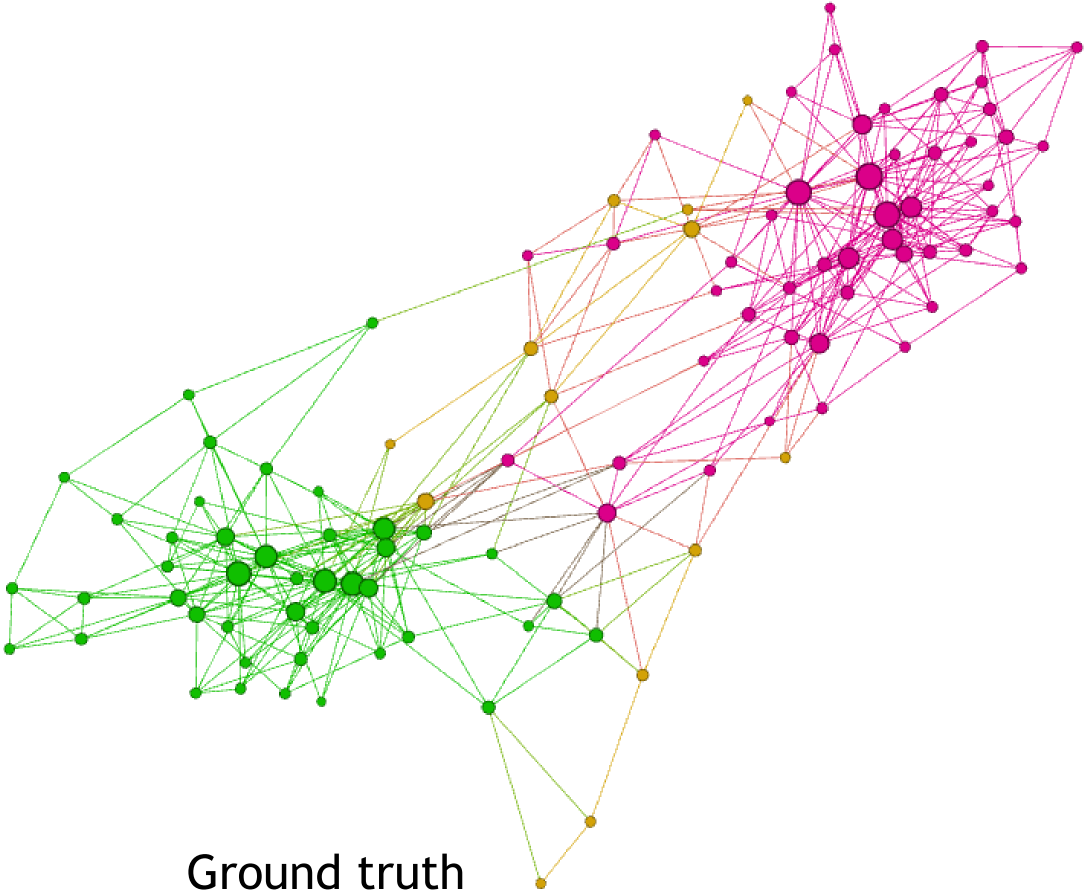


Ground truth

# SPECTRAL MODULARITY MAXIMIZATION

## 84% hit rate

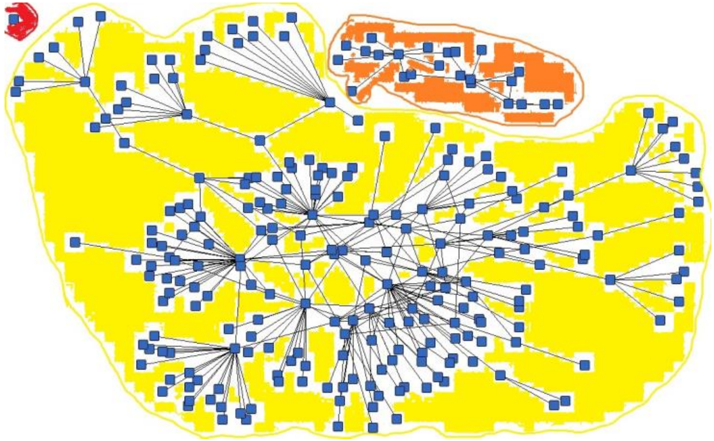
Spectral Modularity maximization



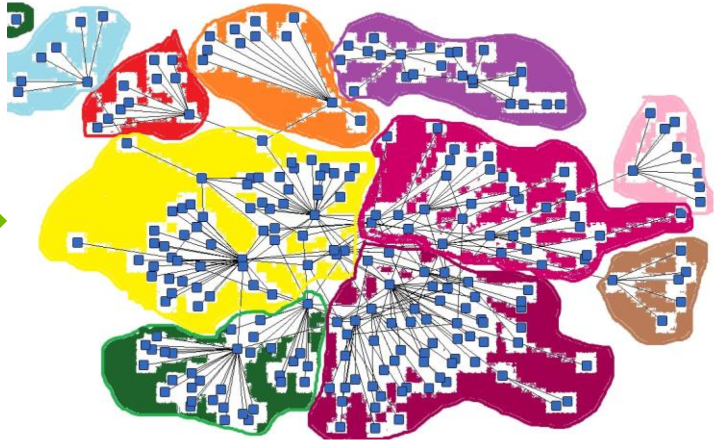
Ground truth

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

# HEIRARCHICAL LOUVAIN CLUSTERS

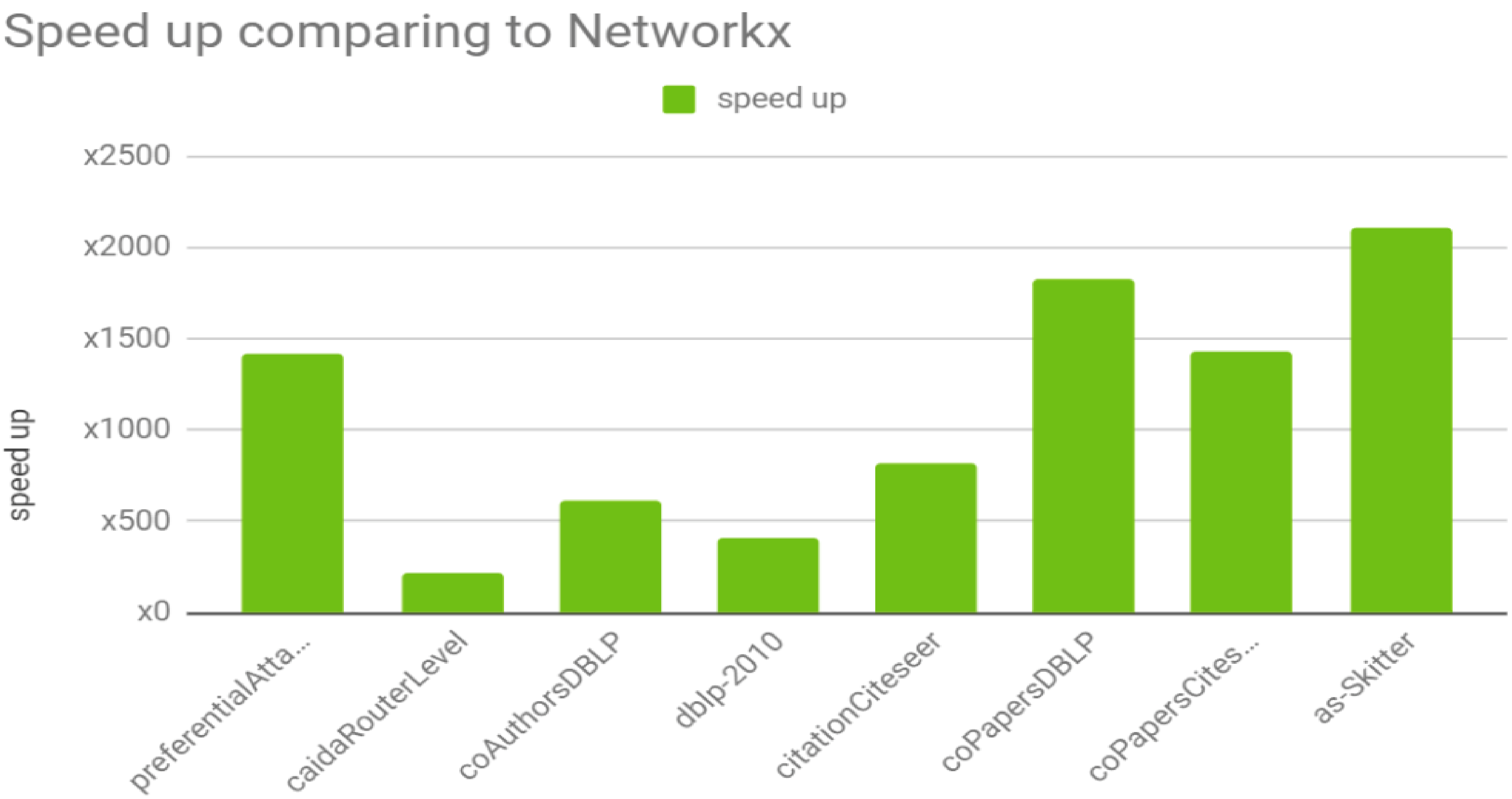


Check the size of each cluster
If size> threshold : recluster



Dict = {‘0’ : initial clusters ,
‘1’ : reclustering on data from ‘0’ ,
‘2’ : reclustering on data from ‘1’ …… }

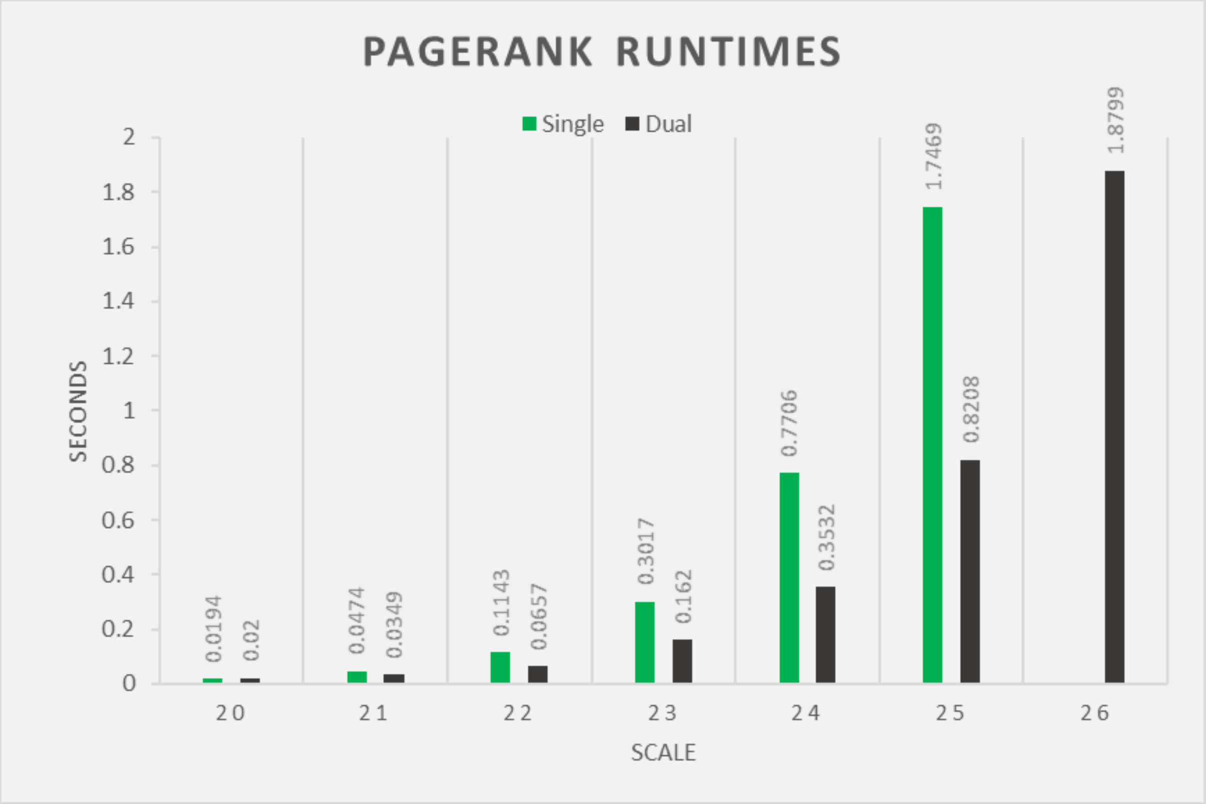
# LOUVAIN SINGLE RUN



# 32GB V100

## Single and Dual GPU on Commodity Workstation

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




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

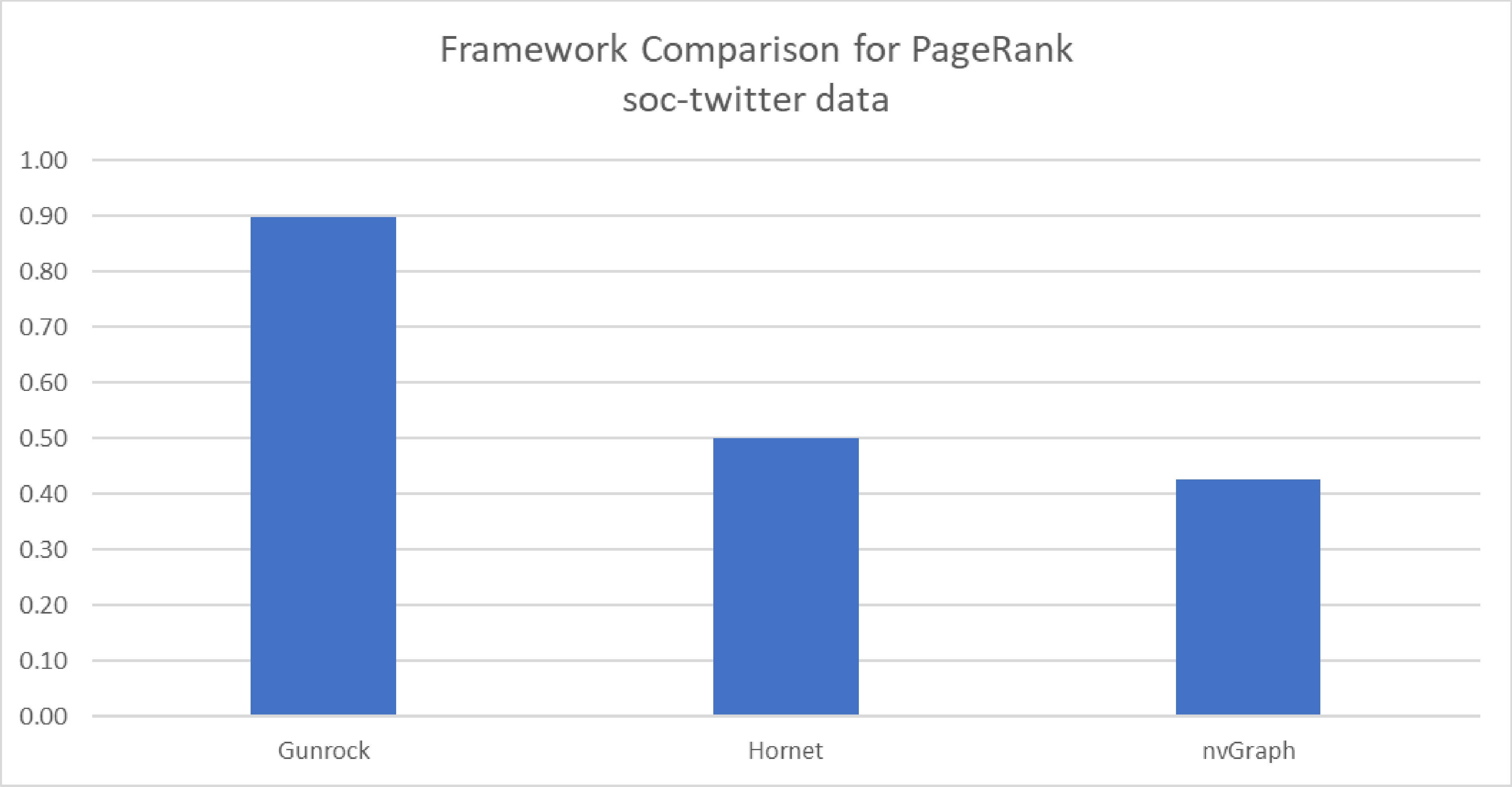
# DATASETS

## Mix of social network and RMAT

| Dataset | Nodes | Edges |
|---|---:|---:|
| soc-twitter-2010 | 21,297,772 | 530,051,618 |
| Twitter.mtx | 41,652,230 | 1,468,365,182 |
| RMAT – Scale 26 | 67,108,864 | 1,073,741,824 |
| RMAT – Scale 27 | 134,217,728 | 2,122,307,214 |
| RMAT - Scale 28 | 268,435,456 | 4,294,967,296 |

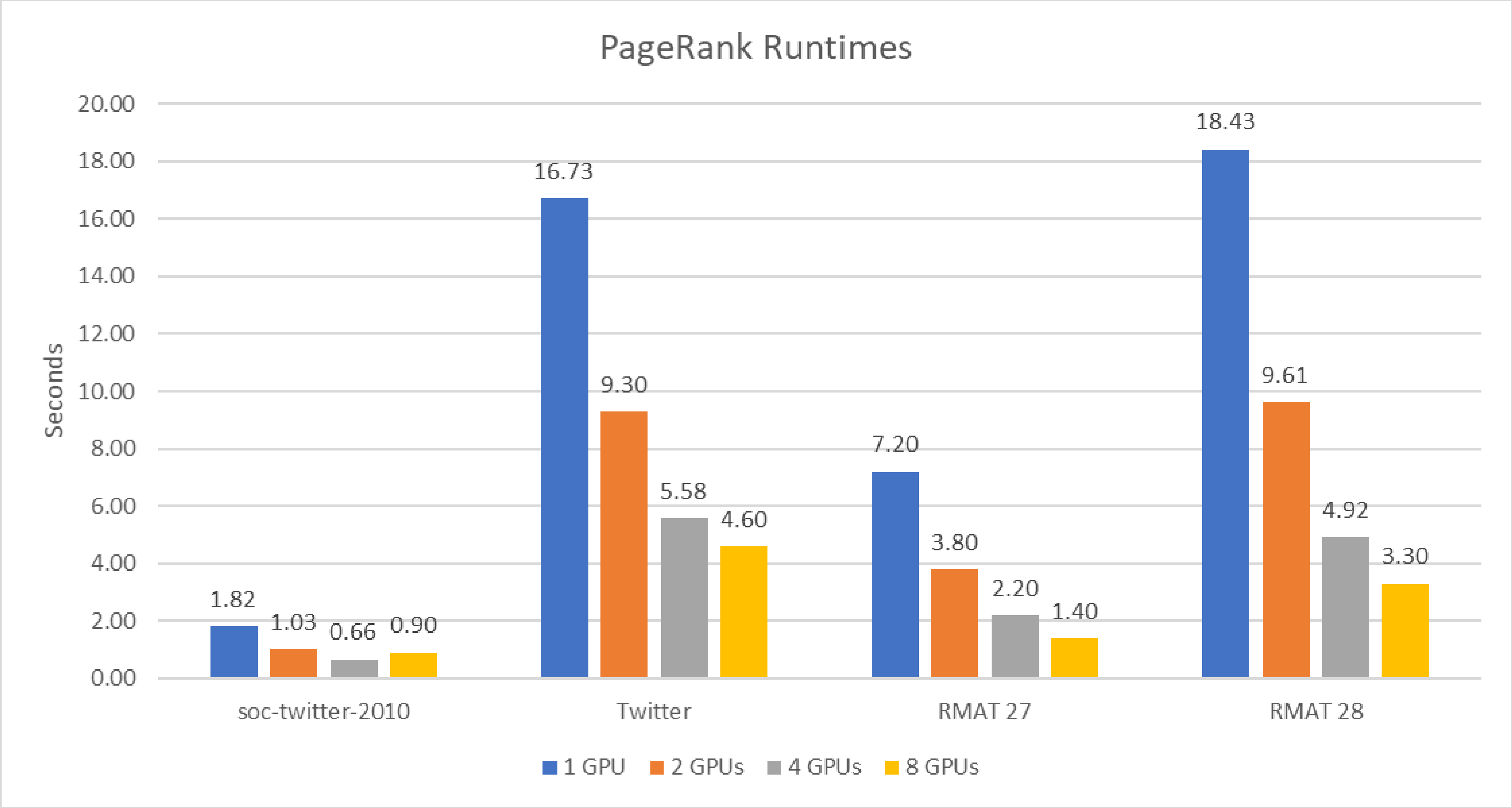
# FRAMEWORK COMPARISON

## PageRank on DGX-1, Single GPU



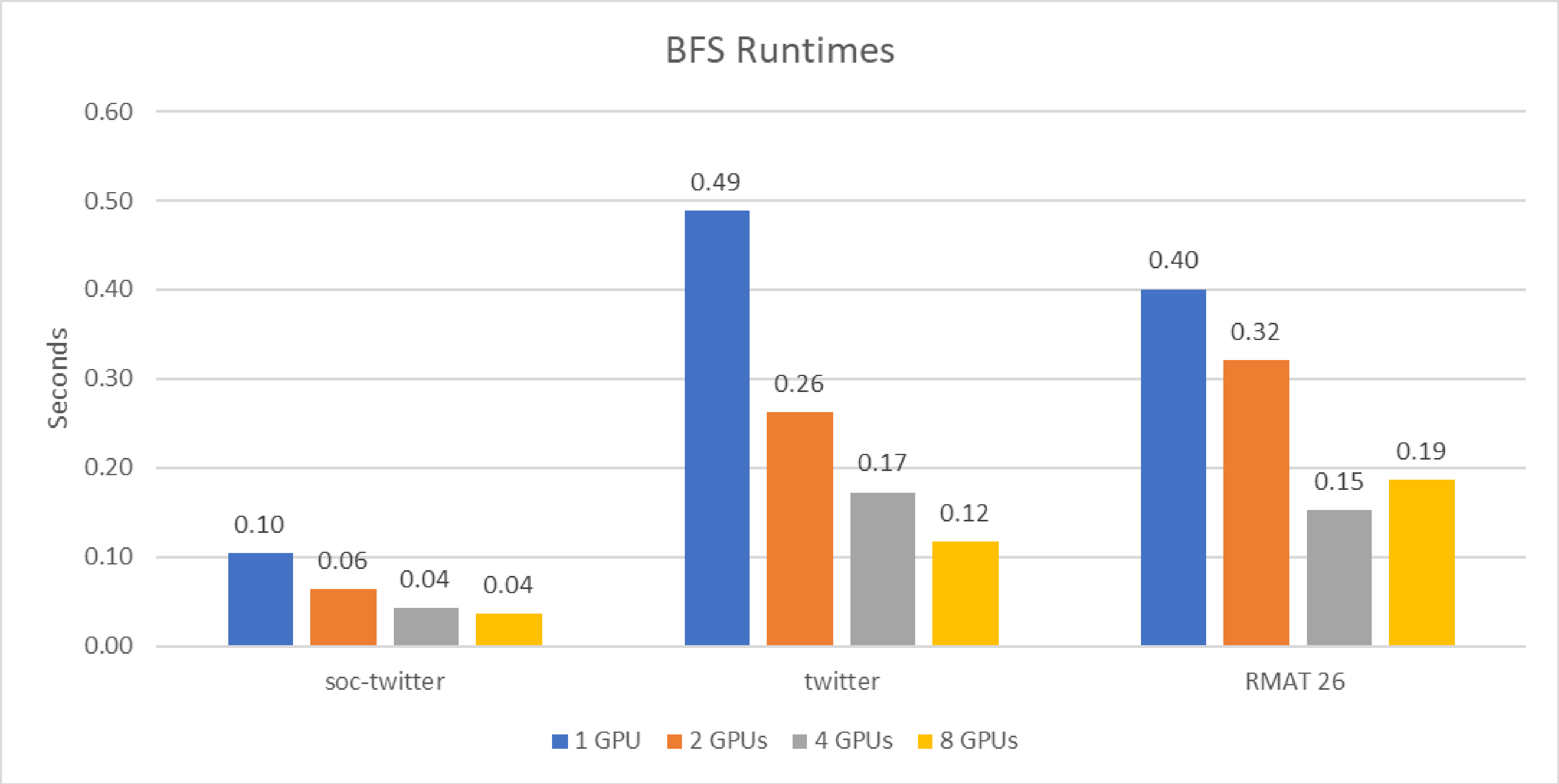
# PAGERANK ON DGX-1

## Using Gunrock, Multi-GPU



# BFS ON DGX-1

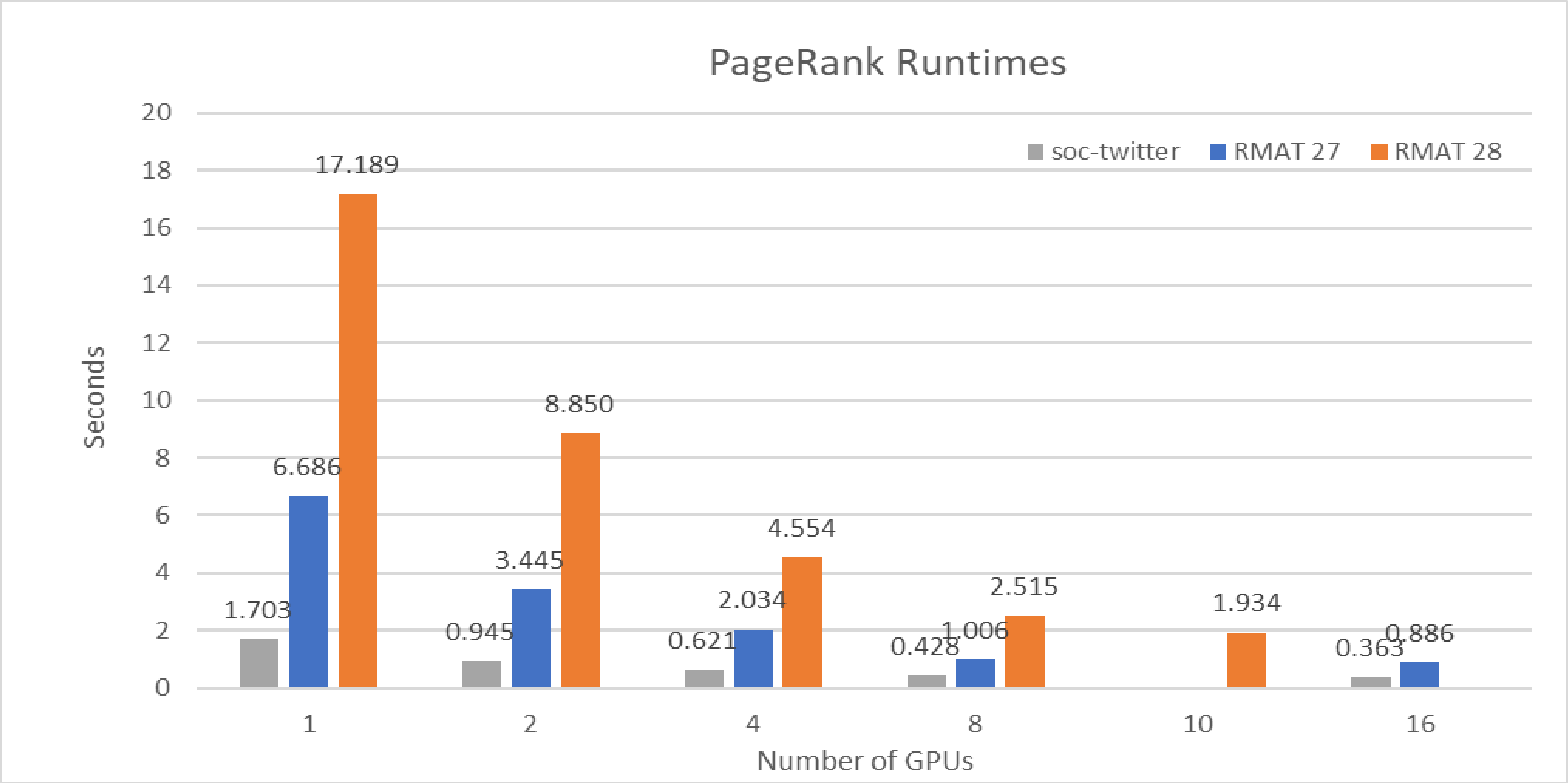
## Using Gunrock, Multi-GPU



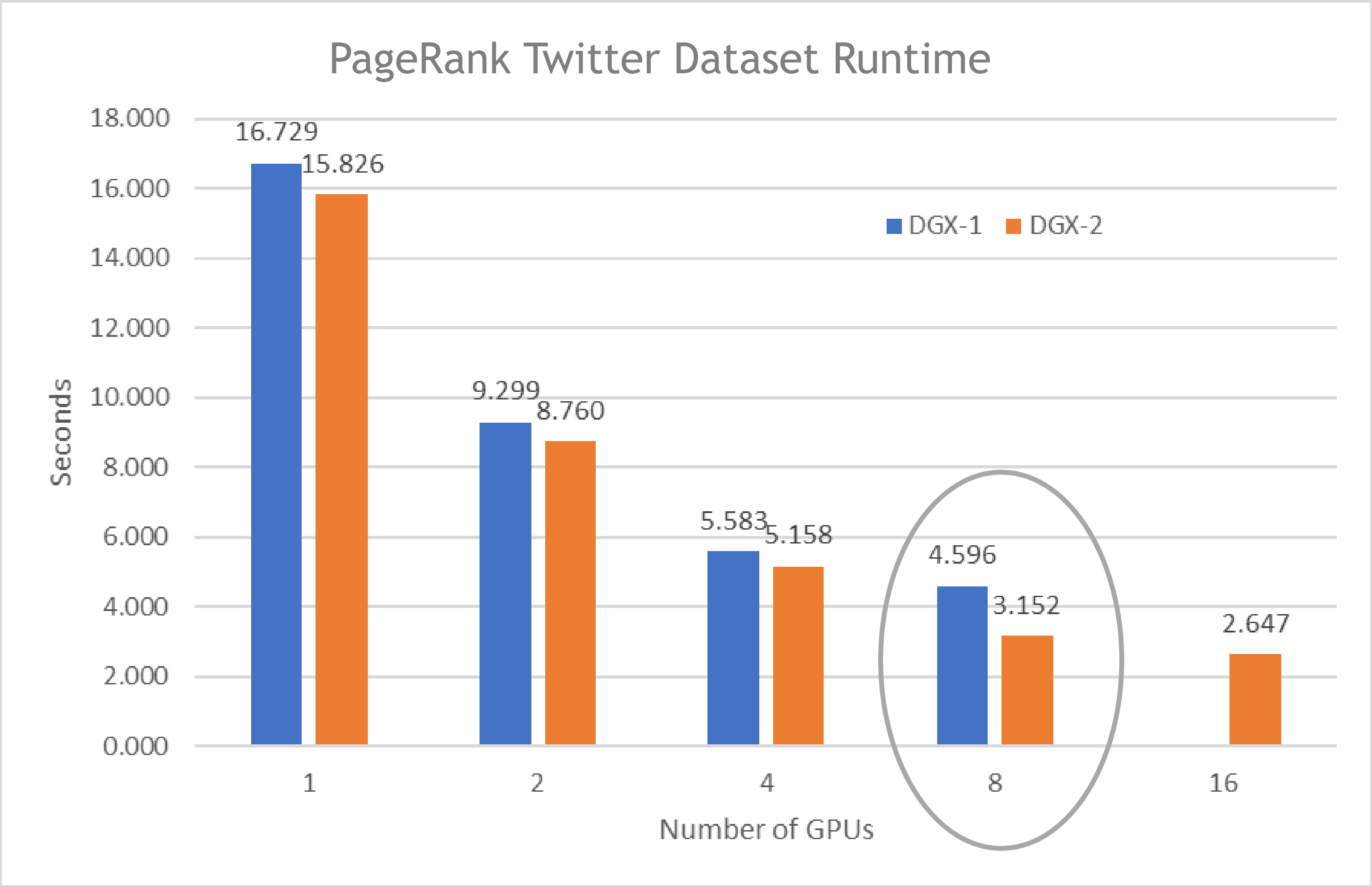


NVIDIA

# DGX-2


PageRank Runtimes
Seconds
20
18
16
14
12
10
8
6
4
2
0
soc-twitter
RMAT 27
RMAT 28
17.189
6.686
1.703
8.850
3.445
0.945
4.554
2.034
0.621
2.515
1.006
0.428
1.934
0.886
0.363
1
2
4
8
10
16
Number of GPUs

# DGX-1 VS. DGX-2



PageRank Twitter Dataset Runtime
Seconds
18.000
16.000
14.000
12.000
10.000
8.000
6.000
4.000
2.000
0.000
16.729
15.826
9.299
8.760
5.583
5.158
4.596
3.152
2.647
DGX-1
DGX-2
1
2
4
8
16
Number of GPUs

# RMAT SCALING, STAGE 4 PRBENCH PIPELINE

## Near Constant Time Weak Scaling is Real Due to NVLINK

| GPU Count | Max RMAT scale | Comp time (sec) | Gedges/sec | MFLOPS | NVLINK Speedup |
|---|---|---|---|---|---|
| 1 | 25 | **1.4052** | 7.6 | 15282.90 | 1.0 |
| 2 | 26 | **1.3914** | 15.4 | 30867.37 | 1.4 |
| 4 | 27 | **1.3891** | 30.9 | 61838.78 | 2.8 |
| 8 | 28 | **1.4103** | 60.9 | 121815.46 | 4.1 |
| 16 | 29 | **1.4689** | 117.0 | 233917.04 | 8.1 |

# WHAT'S NEXT?

Ease of Use, Multi-GPU, new algorithms

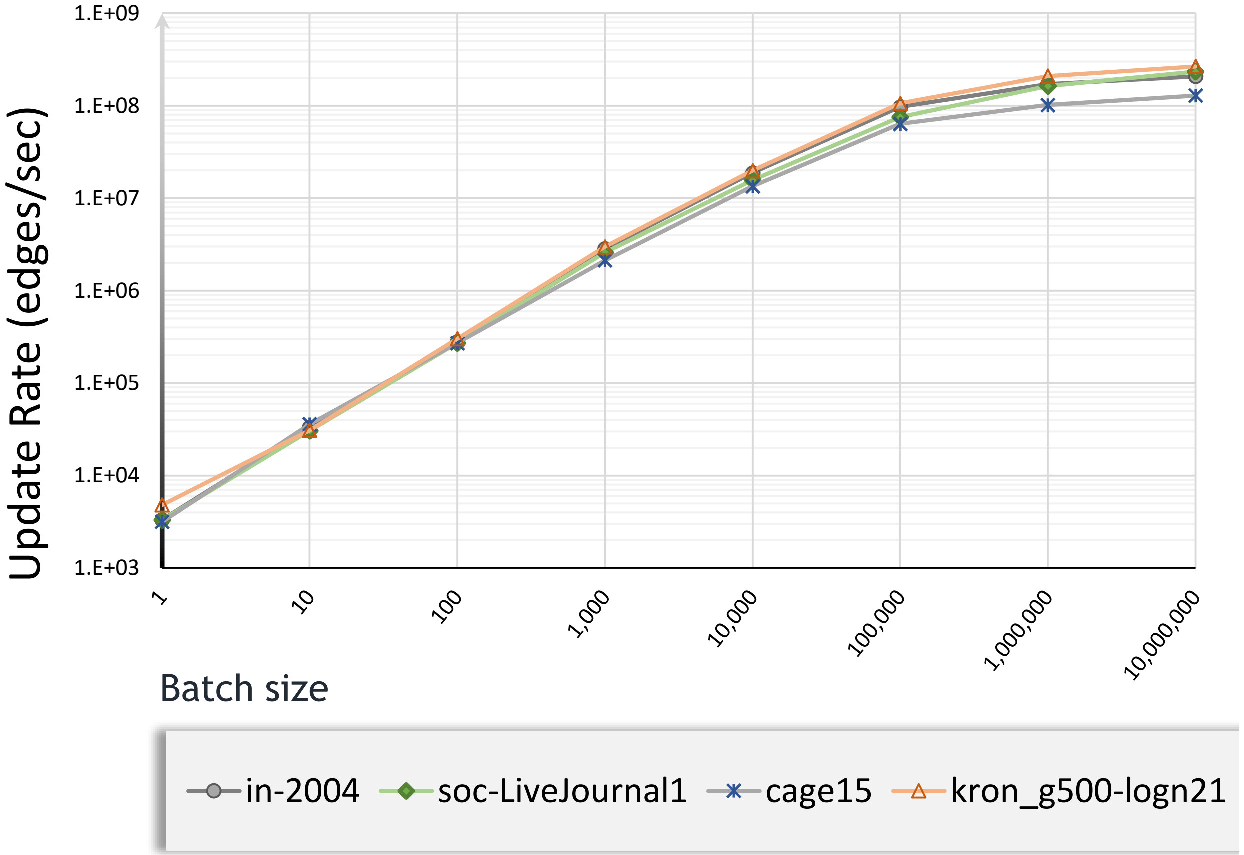
# HORNET

- 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



GUNROCK

GUNROCK
GPU GRAPH ANALYTICS FRAMEWORK AT UC DAVIS

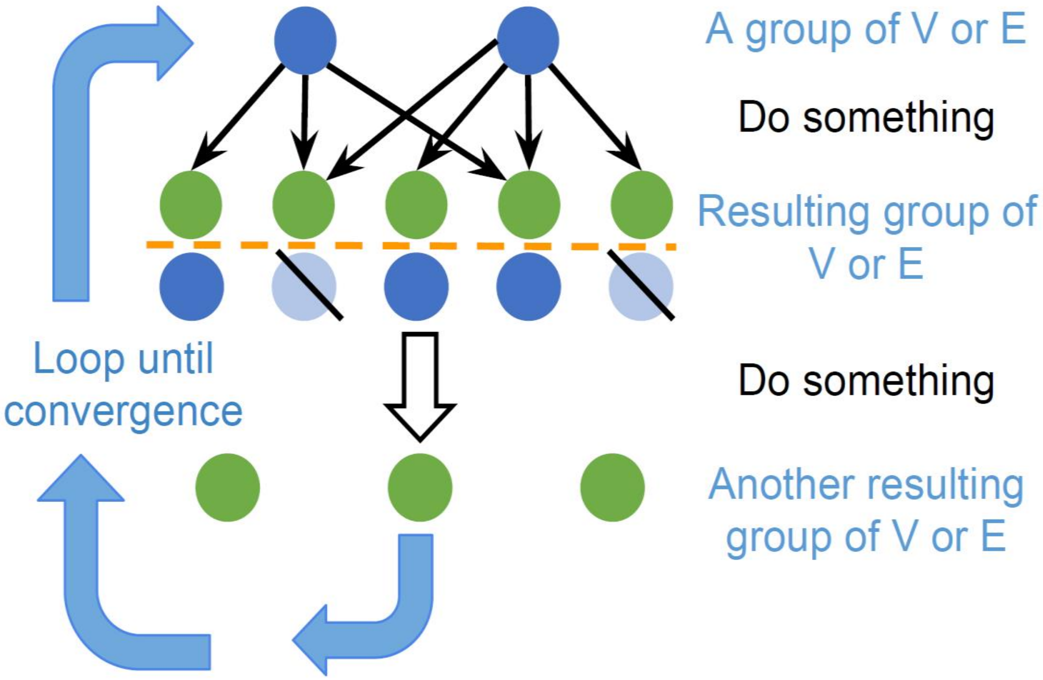
UC DAVIS
UNIVERSITY OF CALIFORNIA

https://gunrock.github.io

- Generality
  - Supports many algorithms
- Programmability
  - Easy to add new methods
- Scalability
  - Multi-GPU support
- Performance
  - Competitive with other GPU frameworks

# Programming Model

A generic graph algorithm:




**Data-centric abstraction**

- Operations are defined on
  a group of vertices or edges $\stackrel{\text{def}}{=}$ 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.

RAPIDS

https://rapids.ai

NVIDIA®