Pacific Northwest NATIONAL LABORATORY

WASHINGTON STATE UNIVERSITY

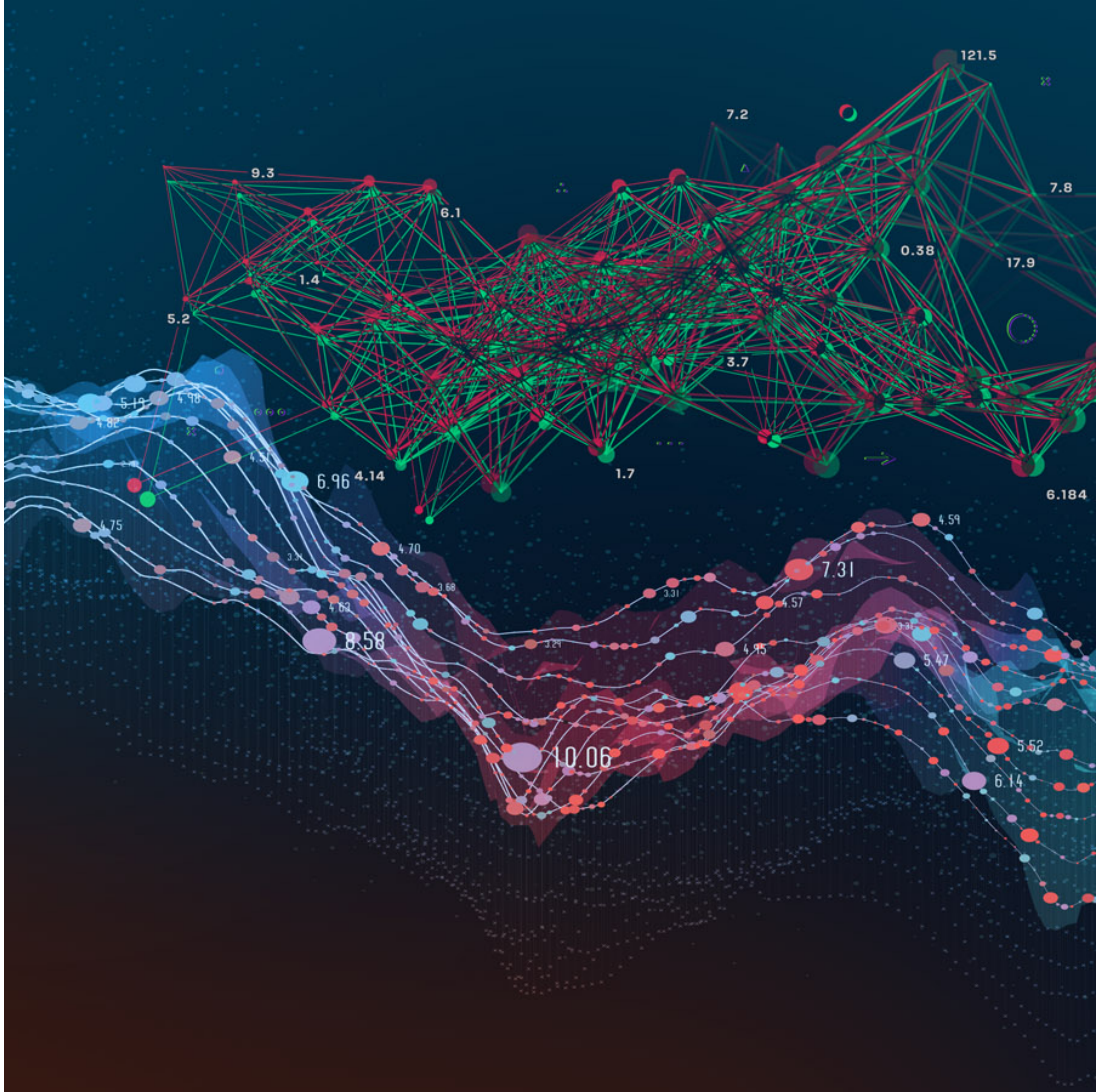
cuRipples: Influence Maximization on Multi-GPU Systems

ICS 2020

Marco Minutoli
Maurizio Drocco
Mahantesh Halappanavar
Antonino Tumeo
Ananth Kalyanaraman

U.S. DEPARTMENT OF ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy



The Team

Marco Minutoli
(PNNL)

Maurizio Drocco
(IBM)

Mahantesh Halappanavar
(PNNL)

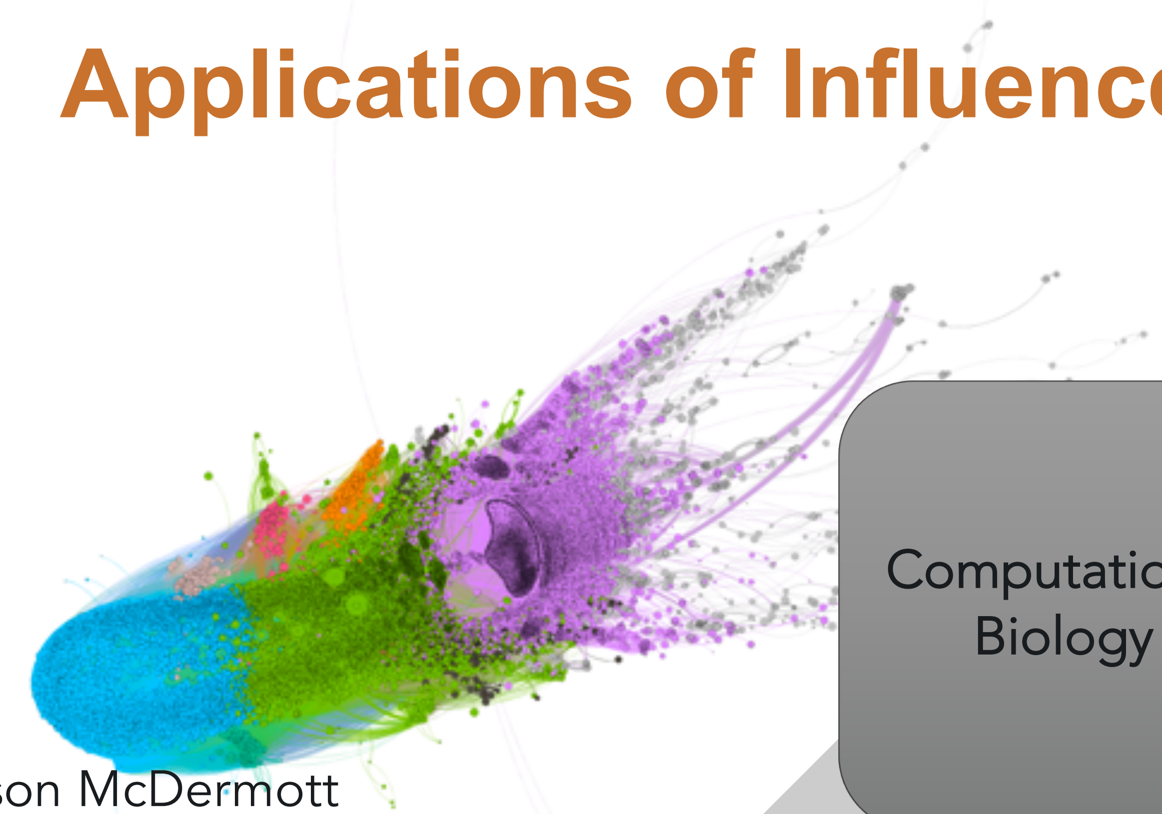
Ananth Kalyanaraman
(WSU)

Antonino Tumeo
(PNNL)

The Influence Maximization Problem

- The objective of the *Influence Maximization Problem (Inf-Max)* is to identify a small set of individuals in a social network, which when activated, will very likely result in the activation of the maximum number of vertices
  - Problem statement coming from the social sciences
    - ✓ Pedro M. Domingos and Matthew Richardson. “Mining the network value of customers”. In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA, August 26-29, 2001*. ACM, 2001, pp. 57–66
  - How does (word-of-mouth) **information propagate**?
  - Who are the **key individuals** that optimize the information diffusion?

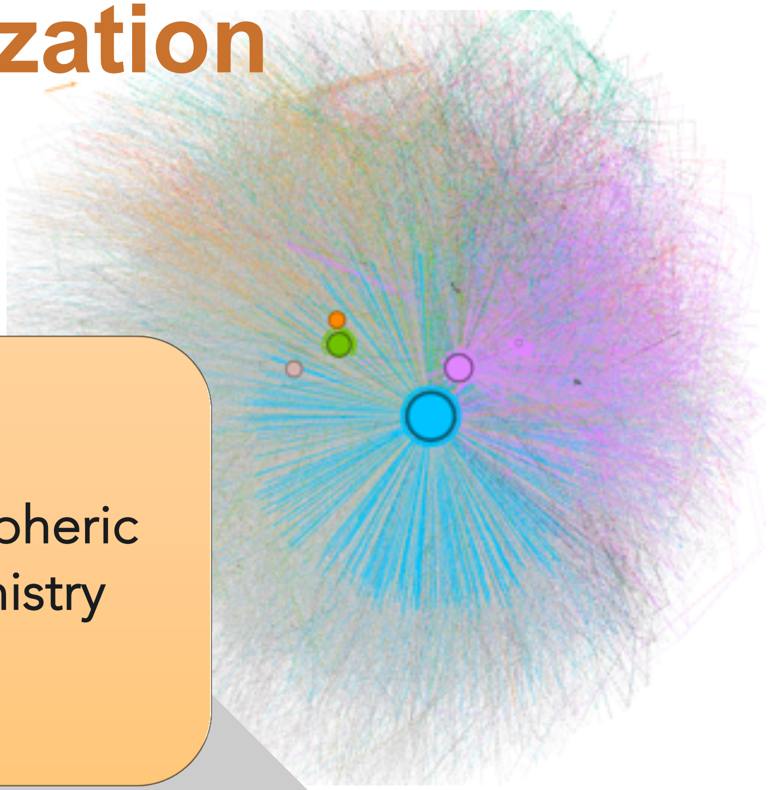
Applications of Influence Maximization



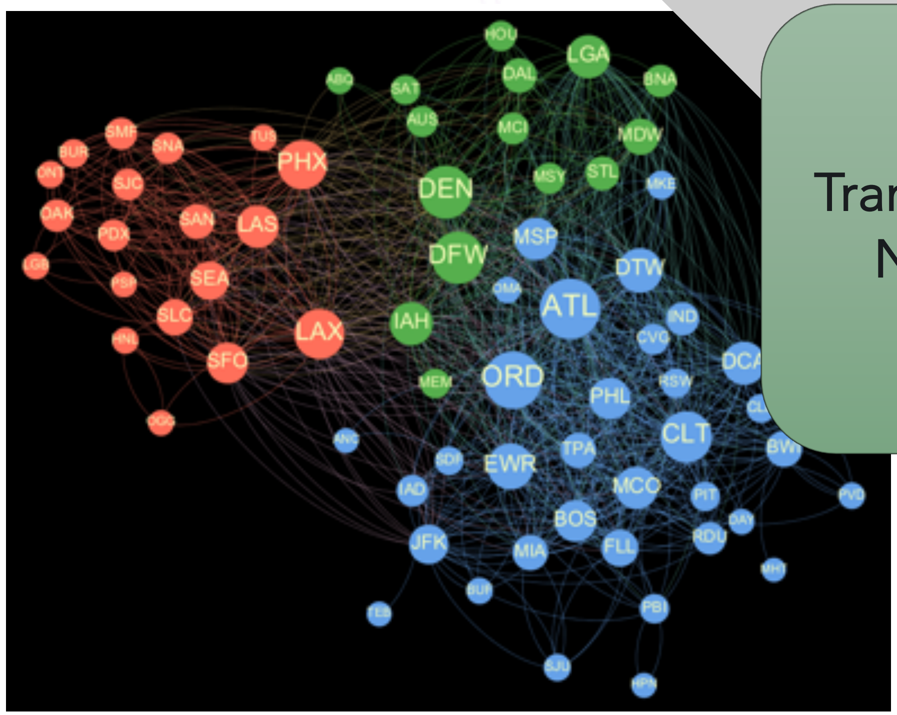
Jason McDermott

Computational Biology

Atmospheric Chemistry

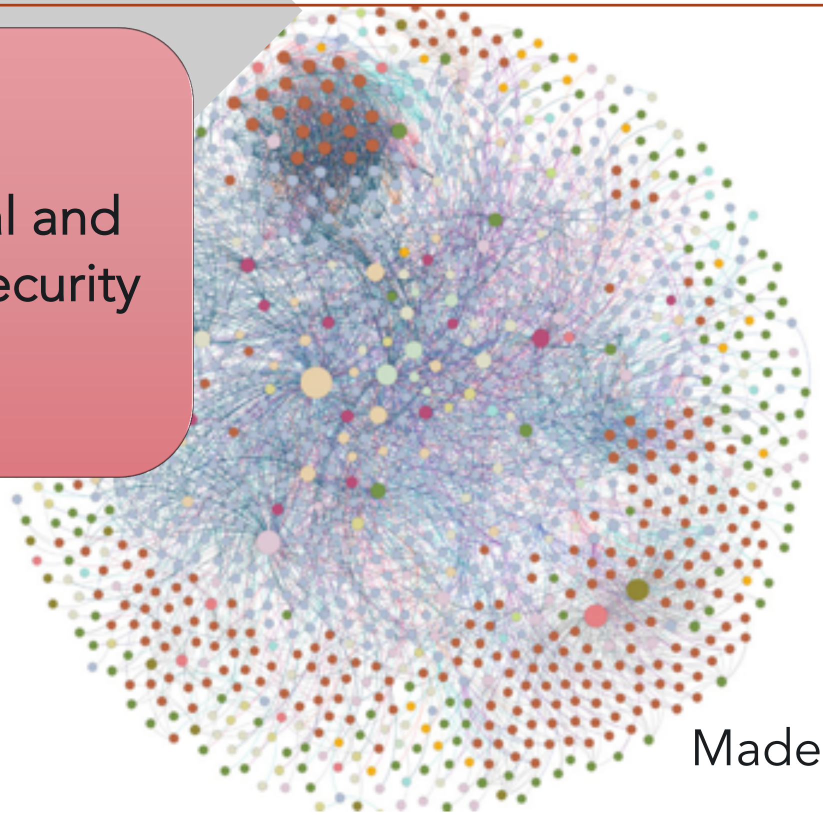


Sam Silva



Transportation Networks

National and Cyber Security



Arun Sathanur

Madelyn Dunning

Influence Maximization: Problem Definition

- Given: A graph *G(V, E)*, a diffusion model (how a vertex gets activated based on the state of its neighbors), and a budget *k*, the influence maximization problem is stated as follows:
- Find a set of *k* vertices called the seed set *S*, that when initially activated result in maximal activations on the network amongst all possible sets of *k* vertices
- Two diffusion models studied in our work:
  - Linear Threshold: A vertex can get activated if a fraction of neighboring vertices that are active is greater than a threshold λ_v
  - Independent Cascade: One shot chance for an activated vertex to activate its neighbor

The Greedy Hill Climbing Algorithm

- Uses the sub-modularity property of the Influence Function
- Approximation Factor: (1-1/e) - **ε**

1. Generate a set of **n** random samples **SG**
   - Different instantiations of **G** are computed based on the edge probabilities

1

2. Repeat until **k** most influential nodes are chosen:
   1. Compute the influence of all remaining nodes across different samples w.r.t. the current seed set **S**
   2. Pick the best influential node, and add to **S**

2

Kempe, David, Jon Kleinberg, and Éva Tardos. "Maximizing the spread of influence through a social network." *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2003.

The IMM algorithm

"Who is influencing me?" instead of "Who am I influencing?"

- Decide the number of experiments
- Generating the reverse reachability information with graph explorations
- Greedy seed selection

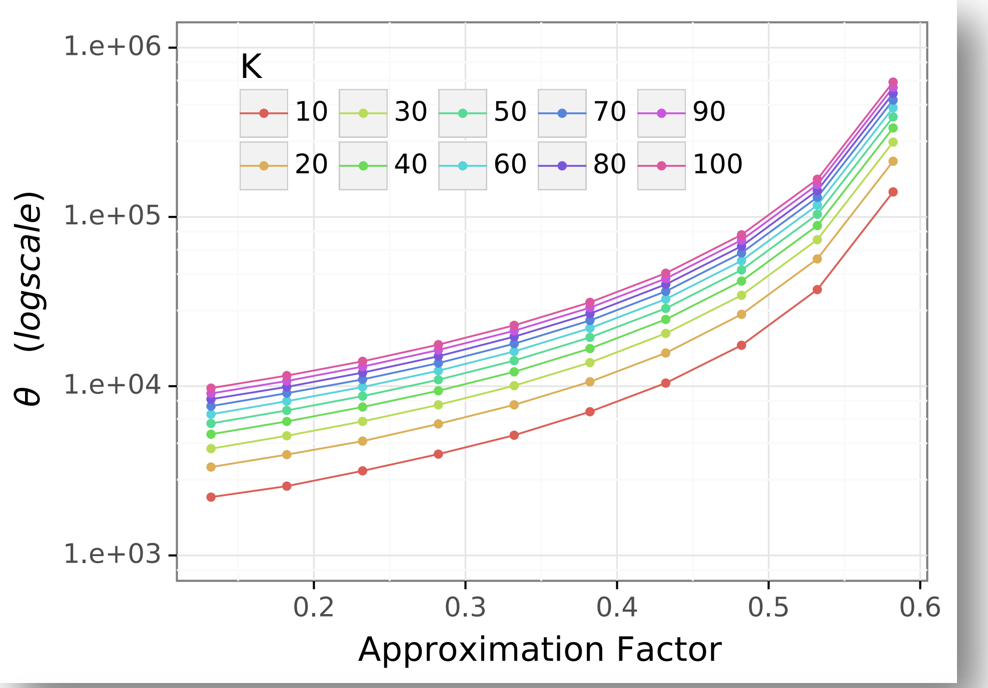
- EstimateTheta builds on Sample and SeedSelect

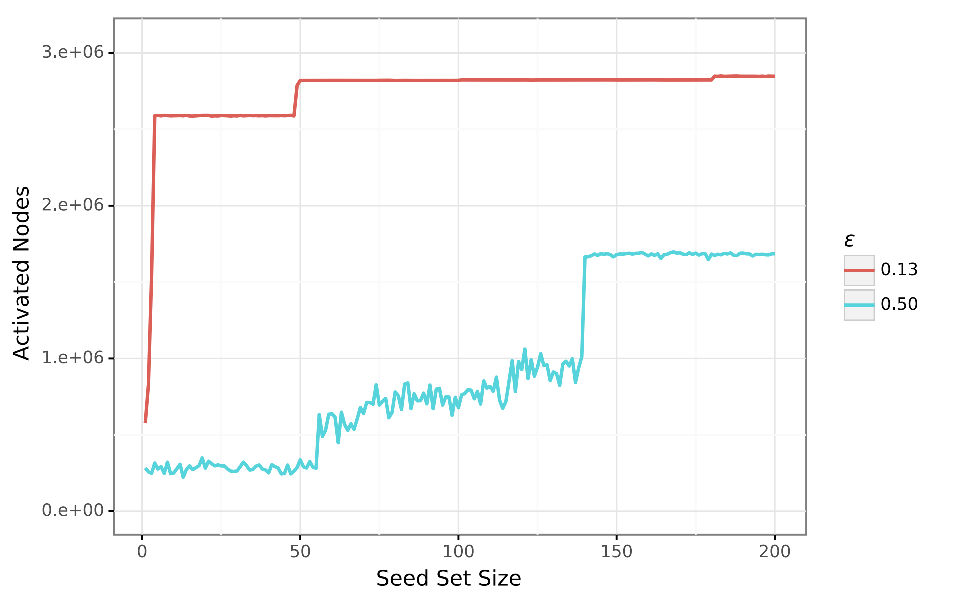
Input : G, k, ϵ
Output: S
begin
    $\langle \mathbb{R}, \theta \rangle \leftarrow$ `EstimateTheta`(G, k, ϵ)
    $\mathbb{R} \leftarrow$ `Sample`(G, $\theta - |\mathbb{R}|$, $\mathbb{R}$)
    S $\leftarrow$ `SelectSeeds`(G, k, $\mathbb{R}$)
    return S
end

Tang, Youze, Yanchen Shi, and Xiaokui Xiao. "Influence maximization in near-linear time: A martingale approach." *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 2015.

Influence Maximization: Challenges

- Approximation Factor: (1 - 1/e) - **ε**
- Algorithms are computationally expensive (high order polynomial for the greedy hill climbing approach of Kempe et al.)
- Alternative methods (random reverse reachable paths) are memory and compute intensive (HPC)
- Nonlinear growth in work relative to approximation factor (**ε**) and number of seeds required (*k*)

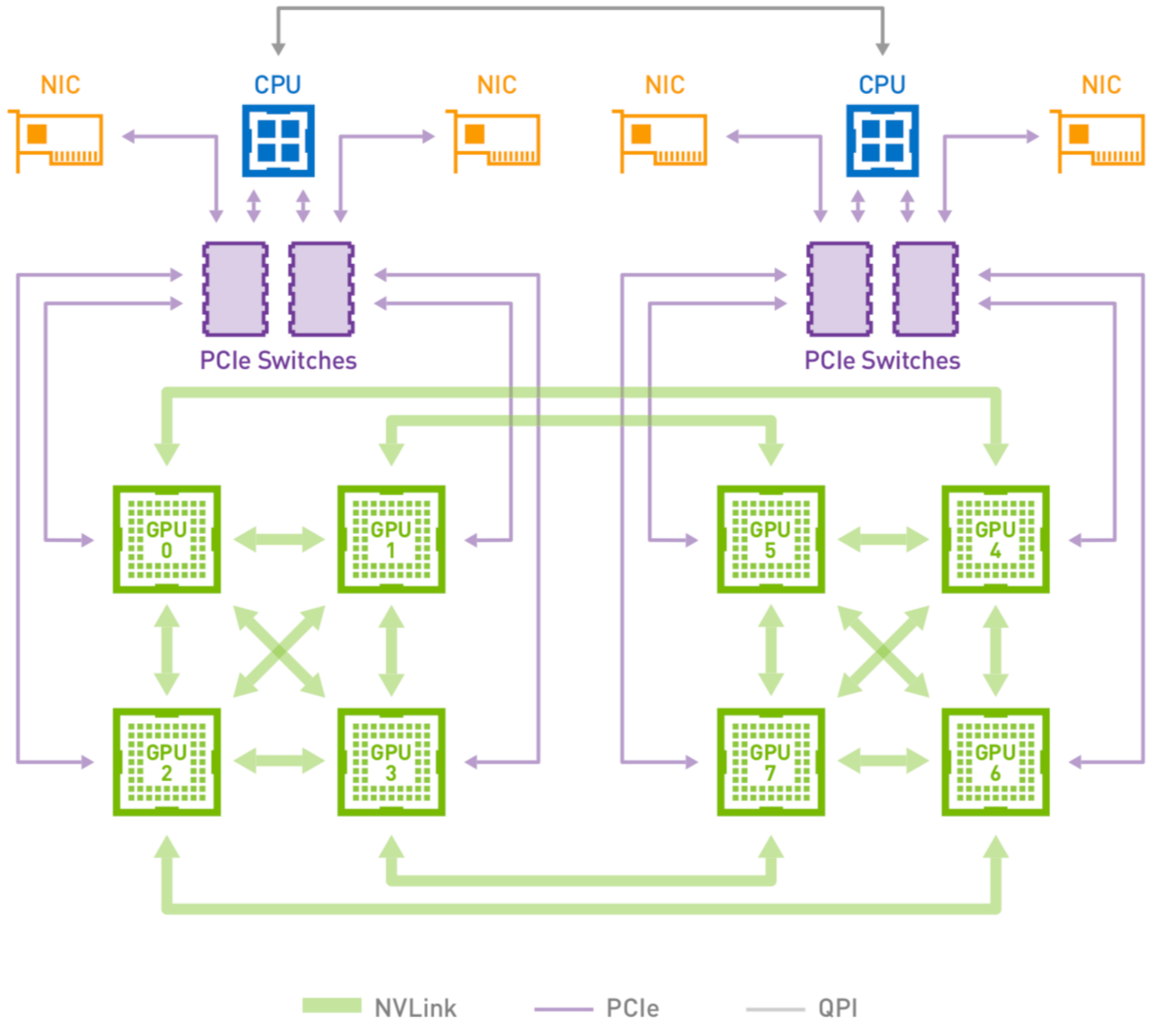




Architectures

DGX-1 Volta

- 8 x V100 GPUs, NVLINK2
  - 5120 CUDA Cores/640 Tensor Cores/4096-bit memory bus/16 GB HBM2
- GPUs not fully interconnected (some at 2 hops)
  - Not all GPUs with peer-to-peer atomic memory operations
- V100 has 6 peer-to-peer links, some connections are faster

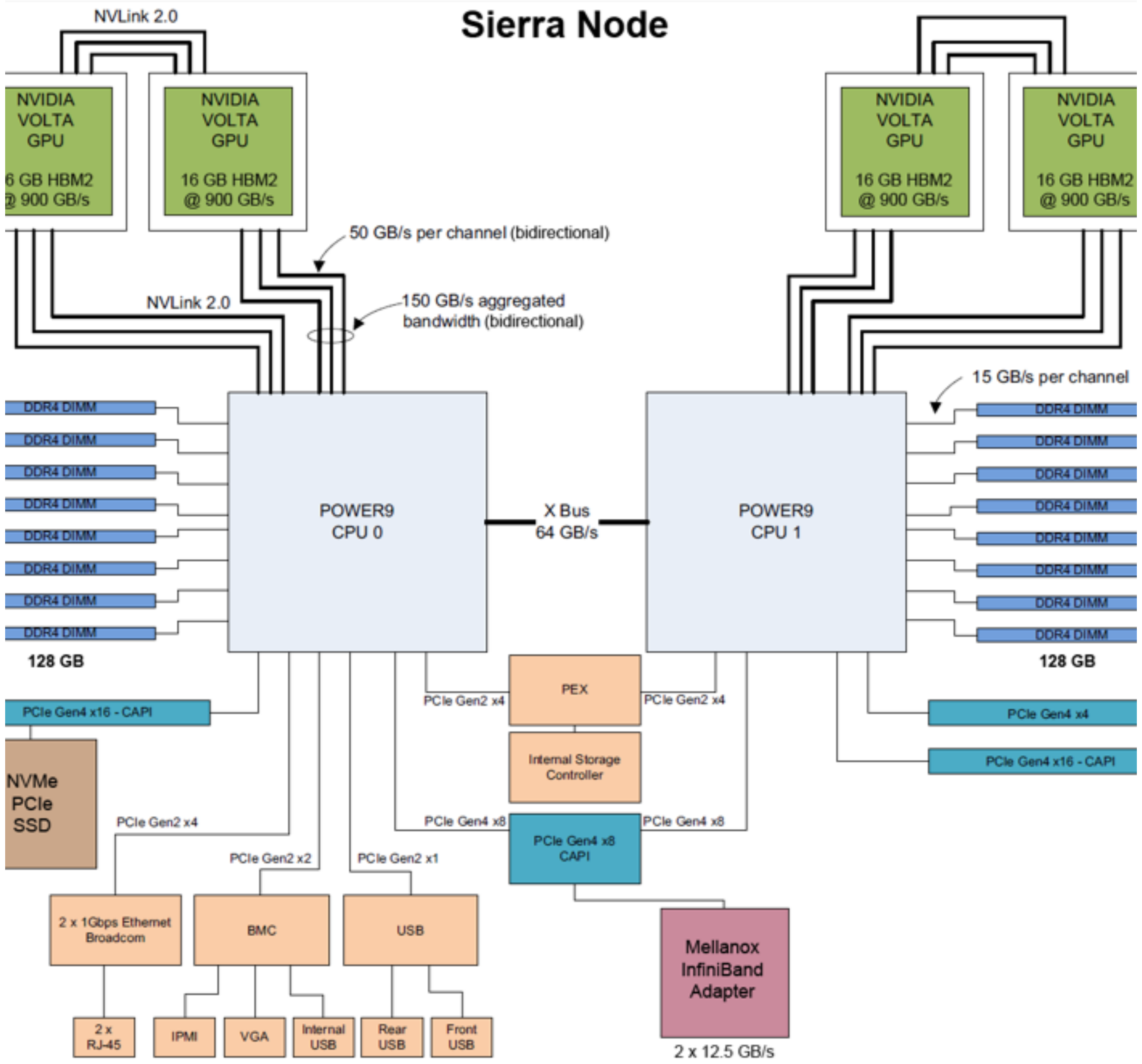


From the DGX-1 System Architecture Manual

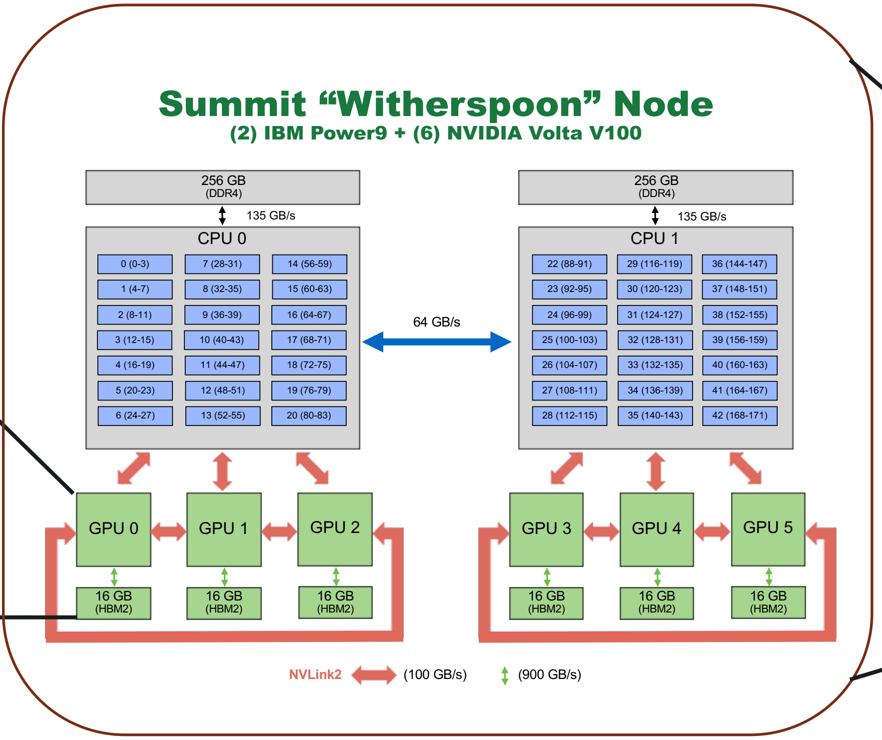
Pacific Northwest
NATIONAL LABORATORY

Newell@PNNL

- 2 Power9 CPUs with a total of 128 logical cores per system
- 4 NVIDIA V100 GPUs with NVLINK2 (16GB per GPU)
    - 3 links (GPU-GPU, CPU-GPU)
- X-Bus allows atomic memory operations for GPUs connected to different sockets
- 1TB of system memory per node
- EDR Infiniband internal network



Summit: Unprecedented Parallelism



Single GPU
(2048 x 80 threads)

Single Node
(6 GPUs)

Distributed Multi-GPU Cluster
(4608 nodes)

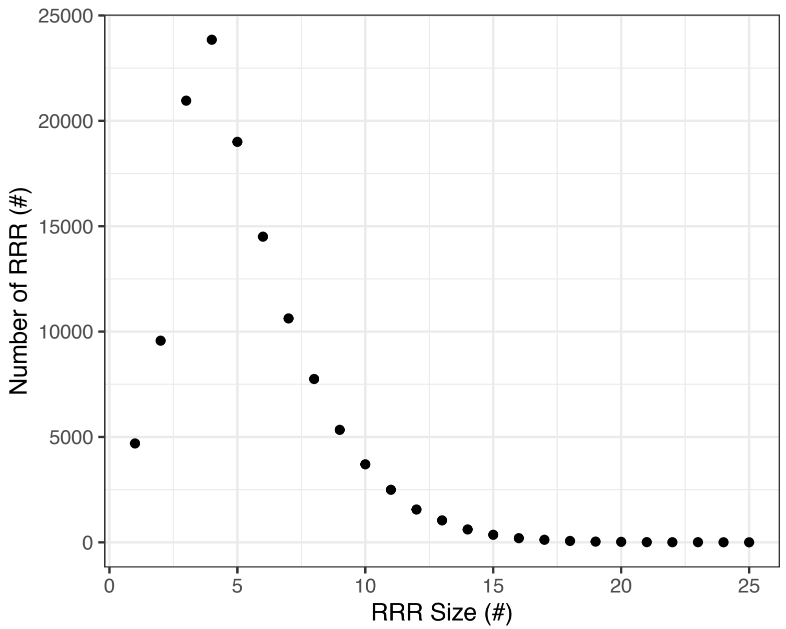
$$\frac{\mathbf{2048}\ \text{Threads}}{\text{SM}} \times \frac{\mathbf{80}\ \text{SMs}}{\text{GPU}} \times \frac{\mathbf{6}\ \text{GPUs}}{\text{Node}} \times \mathbf{4608}\ \text{Nodes} = \mathbf{4.5\ Billion}\ \text{GPU Threads}$$

cuRipples: The CPU-GPU Dispatching Engine

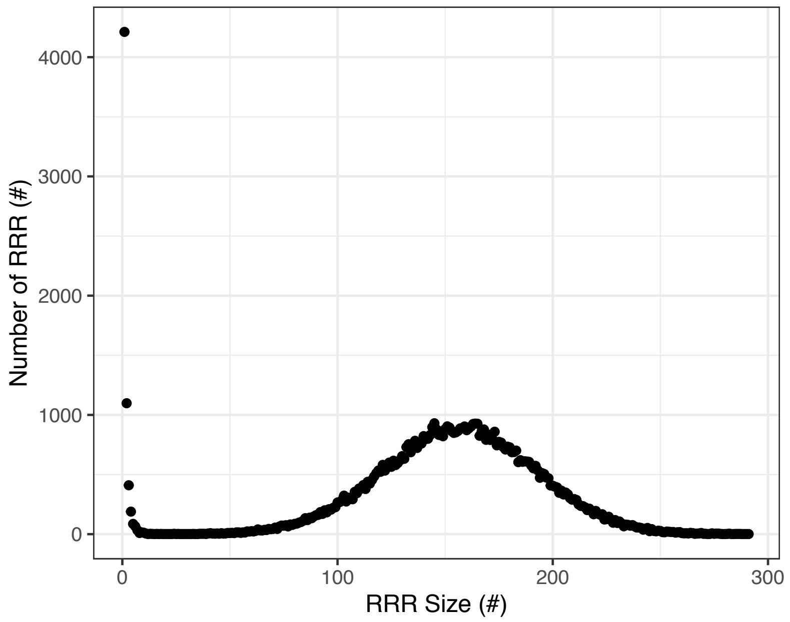
- The engine instantiates a thread pool
  - Usually 1 thread per core on the CPUs of system
- Each GPU has a dedicated CPU thread offloading work with the possibility to over-subscribe
  - More than 1-thread pushing work to the same device (Hyper-Q)
- The engine builds a representation of the topology of GPUs
  - To structure reductions between GPUs
  - Topology built query the CUDA runtime
- CPU and GPU workers steal from the same “task queue”

Sampling

- Two different strategies for IC and LT models
- For the LT model
  - Each GPU thread performs a randomized BFS, but is limited to visit 8 vertices at most
  - When the limit is exceeded the tasks is invalidated and replayed on the CPU
- For the IC model
  - Parallel BFS derived from the nvgraph.
- Each worker has Parallel Random Number Generator
  - Sequences split with the leap-frog scheme
  - GPU threads do round-robin among them



LT model



IC model

Seed Selection

- Greedily select the covering the greatest number of RRR sets
  - Build a histogram of the vertices occurrences for those not yet selected as seeds

- The histogram can be updated or rebuilt from scratch
  - Partition the RRR in covered and uncovered
  - Rebuild works on the uncovered
  - Update works on the newly covered.

- CPU and GPU histograms are then reduced
  - GPUs use a local histogram later reduced using a tree reduction

Experimental Results: Strong scaling on Summit

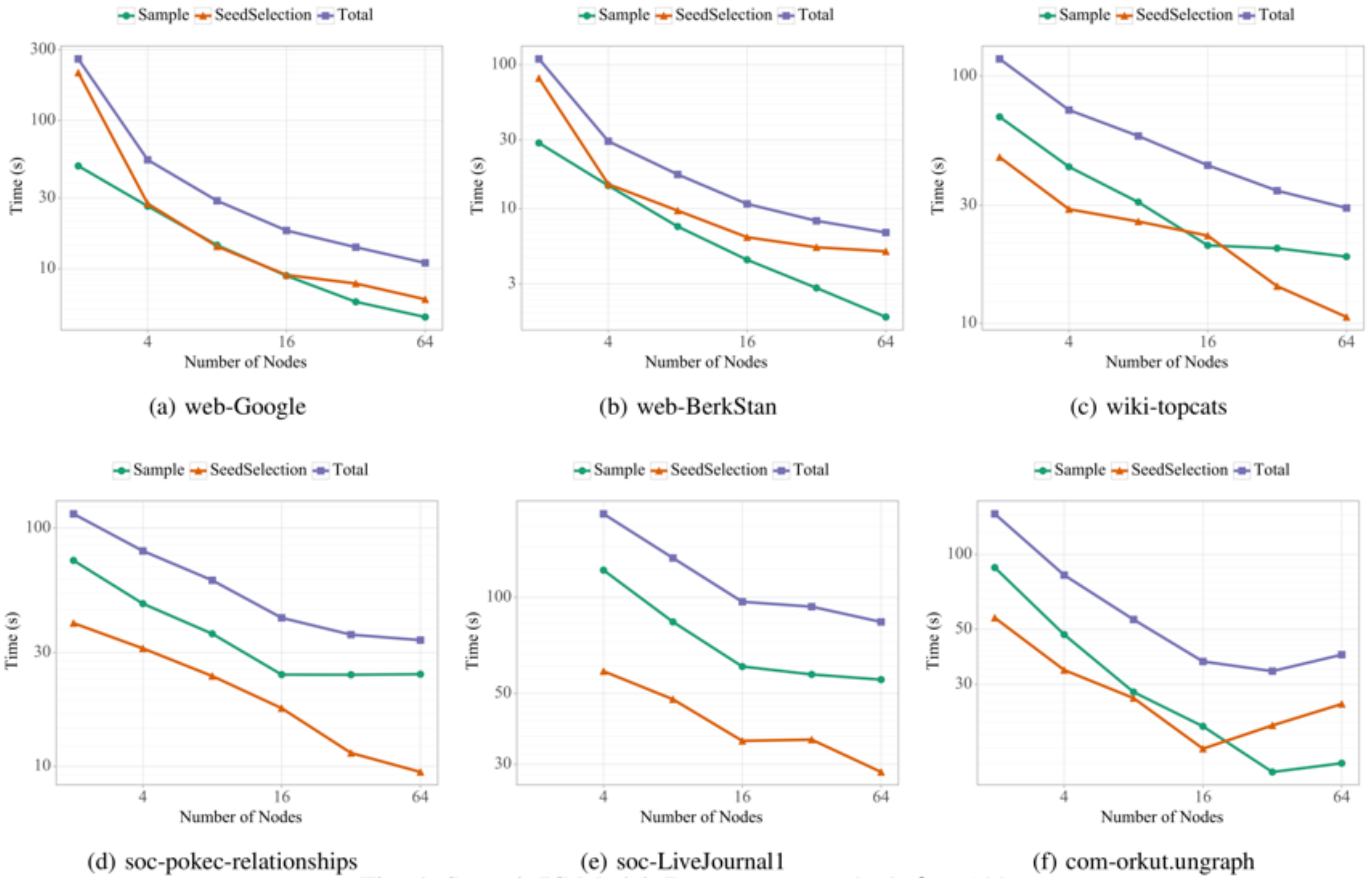


Fig. 4: Summit IC Model. Parameters: $\epsilon = 0.13$, $k = 100$.

Experimental Results on DGX-1 with V100

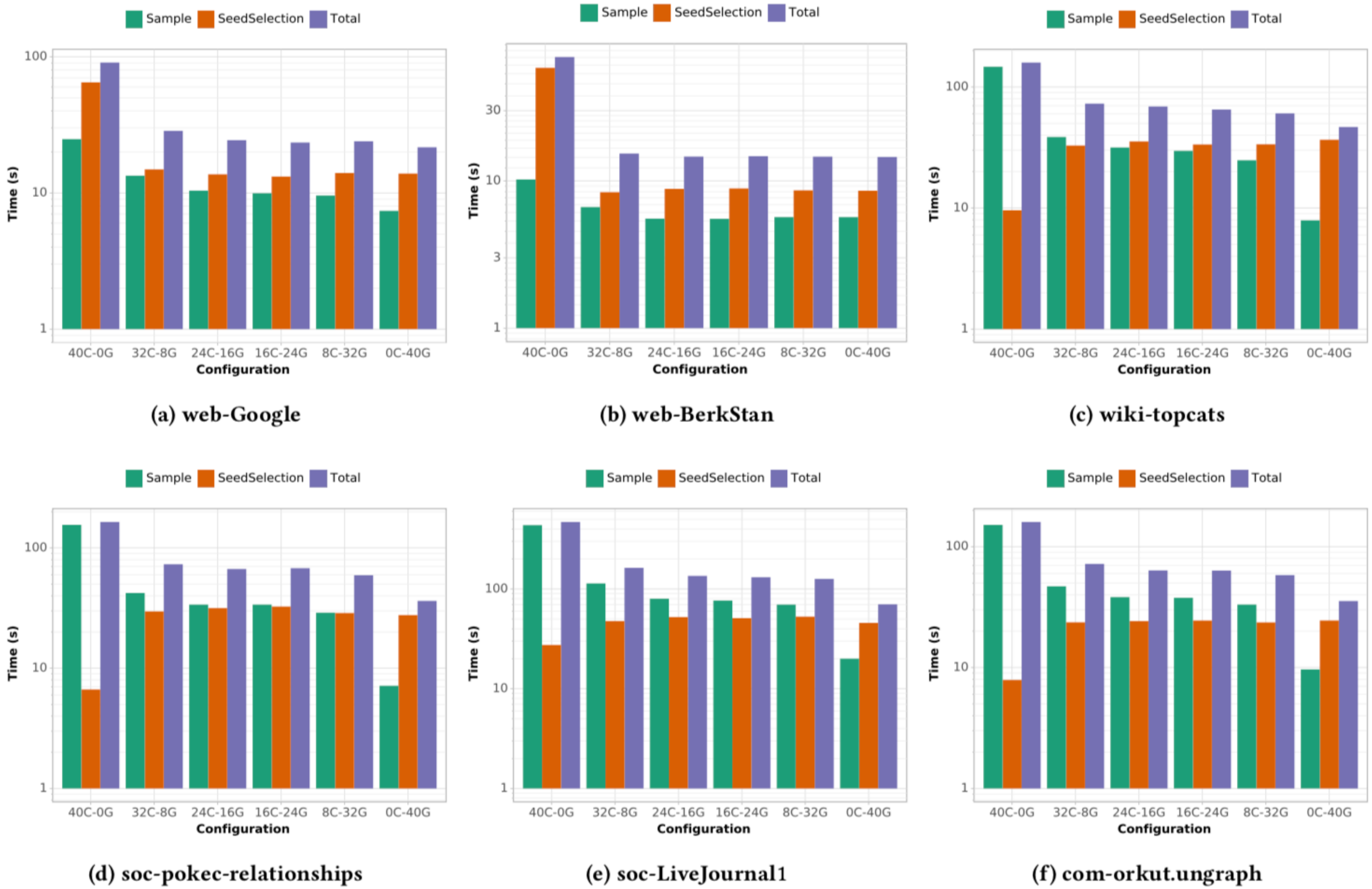
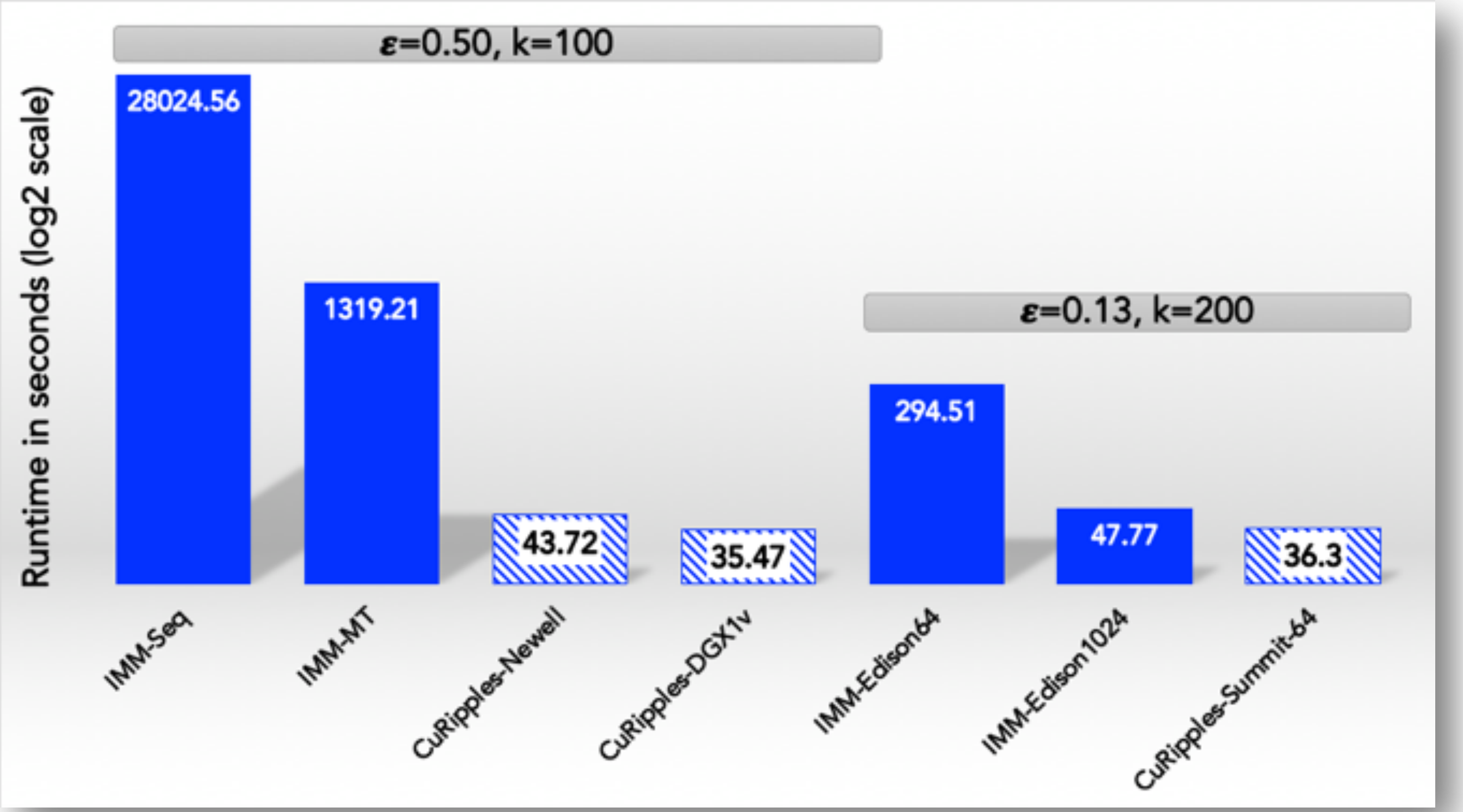


Figure 2: DGX-1v IC Model ($\epsilon = 0.5$, $k = 100$). The configuration reports the number of CPU workers(C) and GPU workers(G)

Summary of our contributions

- Scalable implementations (shared and distributed memory systems)
  https://github.com/pnnl/ripples



CuRipples achieves a speedup of 790x over a state-of-the-art serial implementation, while also significantly improving the quality. The input network is com-Orkut.

TABLE II: Comparative evaluation of `cuRipples` relative to previous implementations of IMM—both serial (IMM_{seq}) [2] and parallel ($\text{IMM}_{\text{opt/mt/edison}}$) [3]. Abbreviations used: No. Cores (C), GPUs (G), Nodes (N).

System	Time (s)	Speedup	Scale
com-Orkut (ϵ=0.5,k=100)			
$\textbf{IMM}_{\text{seq}}$	28024.56	1.00×	1C
$\textbf{IMM}_{\text{opt}}$	9027.50	3.10×	1C
$\textbf{IMM}_{\text{mt}}$	1319.21	21.24×	20C (1N)
$\textbf{CuRipples}_{\text{dgx-1v}}$	35.47	**790.09**×	80C+8G (1N)
$\textbf{CuRipples}_{\text{newell}}$	43.72	641.00×	128C+4G (1N)
com-Orkut (ϵ=0.13,k=200)			
$\textbf{IMM}_{\text{edison}}$	294.51	95.16×	3,072C (64N)
$\textbf{IMM}_{\text{edison}}$	47.77	586.61×	49,152C (1024N)
$\textbf{CuRipples}_{\text{summit}}$	36.30	**772.03**×	2,688C+384G (64N)
soc-LiveJournal1 (ϵ=0.5,k=100)			
$\textbf{IMM}_{\text{seq}}$	16434.81	1.00×	1C
$\textbf{IMM}_{\text{opt}}$	3954.59	4.16×	1C
$\textbf{IMM}_{\text{mt}}$	1026.21	16.02×	20C
$\textbf{CuRipples}_{\text{dgx-1v}}$	70.23	234.01×	80C+8G (1N)
$\textbf{CuRipples}_{\text{newell}}$	65.26	251.84×	128C+4G (1N)
soc-LiveJournal1 (ϵ=0.13,k=200)			
$\textbf{IMM}_{\text{edison}}$	190.94	86.07×	3,072C (64N)
$\textbf{IMM}_{\text{edison}}$	55.12	298.16×	49,152C (1024N)
$\textbf{CuRipples}_{\text{summit}}$	106.43	154.42×	2,688C+384G (64N)

Influence Maximization References

- M Minutoli, M Halappanavar, A Kalyanaraman, A Sathanur, R Mcclure, J McDermott. “Fast and scalable implementations of influence maximization algorithms.” In Proceedings of the *IEEE Cluster conference* (CLUSTER'19), 12 pages, 2019.
- A Sathanur, M Halappanavar, Y Shi, and Y Sagduyu. "Exploring the Role of Intrinsic Nodal Activation on the Spread of Influence in Complex Networks." in *Lecture Notes in Social Networks* (LNSN). 2018.
- U Bhatia, S Chatterjee, A Ganguly, M Halappanavar, R Tipireddy, and R Brigantic. "Aviation Transportation, Cyber Threats, and Network-of-Networks: Conceptual Framing and Modeling Perspectives for Translating Theory to Practice." Accepted for publication in proceedings of the *IEEE International Symposium on Technologies for Homeland Security* (HST). October 23 - 24, 2018 Woburn, MA USA.
- A Sathanur, M Halappanavar, S Chaterjee, A Ganguly, and K Clark, “Identification of Critical Airports from the Perspective of Delay and Disruption Propagation in Air Travel Networks”, *IEEE Symposium on Technologies for Homeland Security*, Boston, Nov 2019
- A Sathanur, and M Halappanavar. "Influence Maximization on Complex Networks with Intrinsic Nodal Activation." Accepted for publication in proceedings of the *8th International Conference on Social Informatics* (SocInfo 2016). Bellevue, WA, USA. November 2016.
- Halappanavar M, A Sathanur, and A Nandi. "Accelerating the Mining of Influential Nodes in Complex Networks through Community Detection." In proceedings of the ACM *International Conference on Computing Frontiers*. May 16 - 18, 2016. Como, Italy.
- A Kalyanaraman, M Halappanavar, D Chavarría-Miranda, H Lu, K Duraisamy and P Pratim Pande. "Fast Uncovering of Graph Communities on a Chip: Toward Scalable Community Detection on Multicore and Manycore Platforms", *Foundations and Trends® in Electronic Design Automation*: Vol. 10: No. 3, pp 145-247.

Pacific Northwest NATIONAL LABORATORY

- **Resources:**
  - Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory (Contract No. DE-AC05-00OR22725)
  - the National Energy Research Scientific Computing Center (Contract DE-AC02-05CH11231).
- **Funding:**
  - U.S. DOE ExaGraph project
  - NSF awards CCF 1815467 and OAC1910213

Thank you

Marco Minutoli
marco.minutoli@pnnl.gov

https://github.com/pnnl/ripples