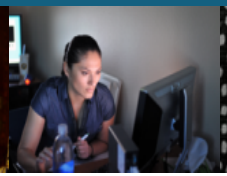
SAND2022-6020C

Sandia National Laboratories

# Exagraph Tutorial, ECP 2022



## ExaGraph: Partitioning & Coloring

Click to edit master subtitle style

Erik Boman , Sandia

Contributions from: Siva Rajamanickam, Seher Acer, Ian Bogle, [illegible]



Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

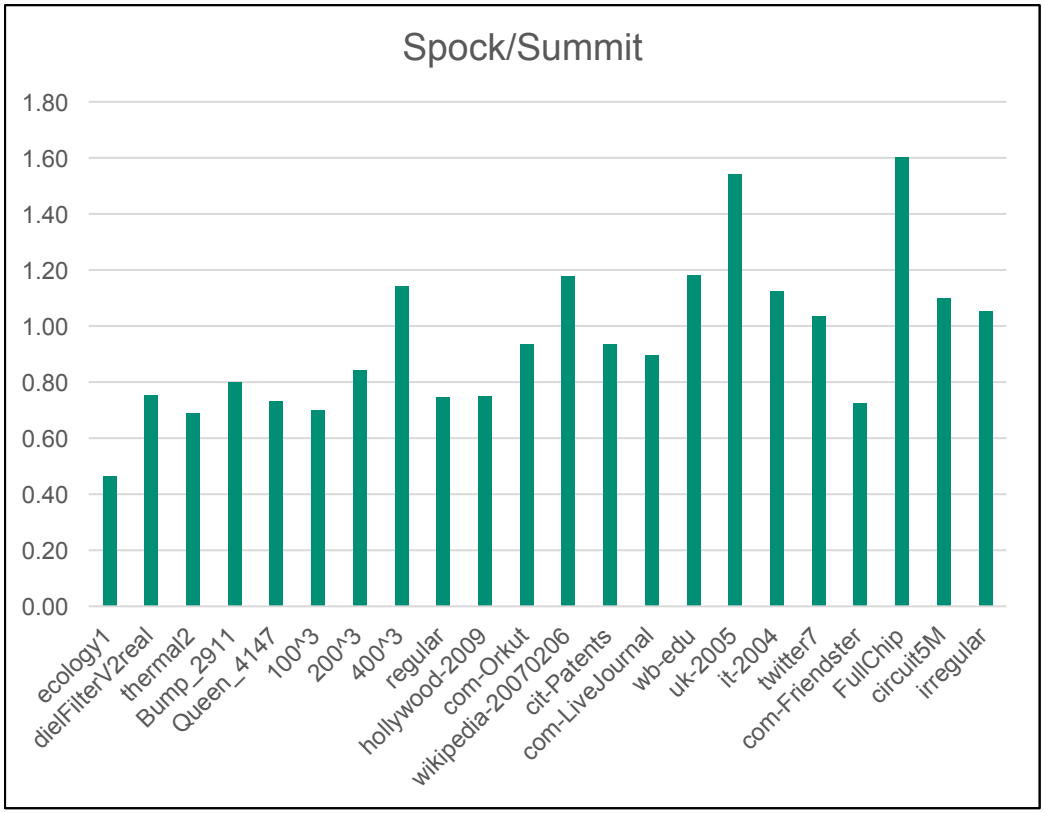
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

# Partitioning: Background/Motivation

- Graph partitioning successful approach to load balancing
- We are revisiting graph partitioning problem, because:
  - DoE/ECP supercomputers will use different accelerators
    - AMD, Intel, NVIDIA GPUs
  - No accelerator-enabled graph partitioning tool exists
  - We provide a new partitioner Sphynx to fill this gap
    - Distributed-memory parallel, accelerator-enabled, and portable
- We are pursuing two strategies:
  - Sphynx: Spectral methods use linear-algebra kernels, which are more amenable to parallelization on accelerators
  - Multilevel graph partitioning on GPU: hard!

# Partitioning: Status

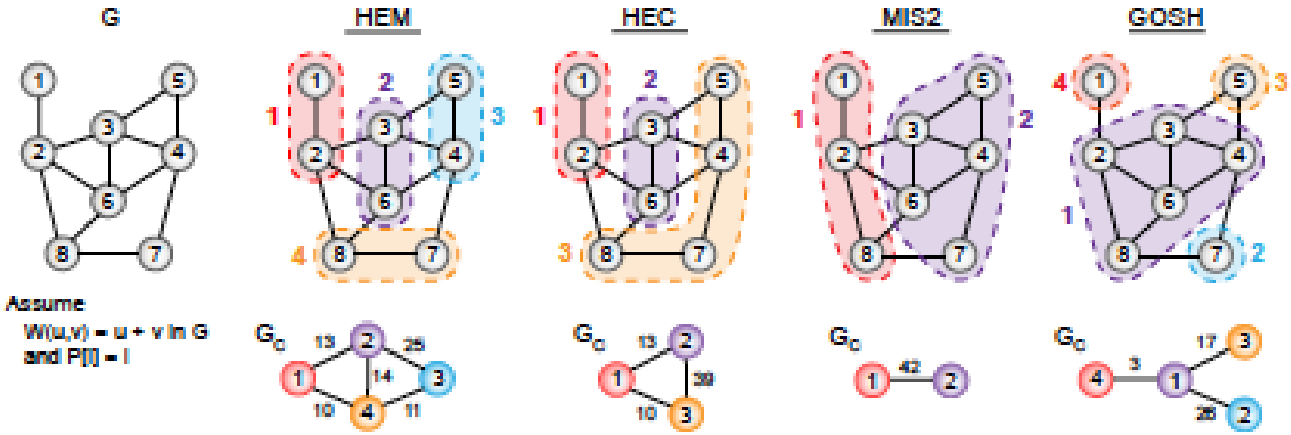
- Sphynx has been released in Trilinos/Zoltan2
- First multi-GPU graph partitioner!
  - Works on both Nvidia and AMD GPU



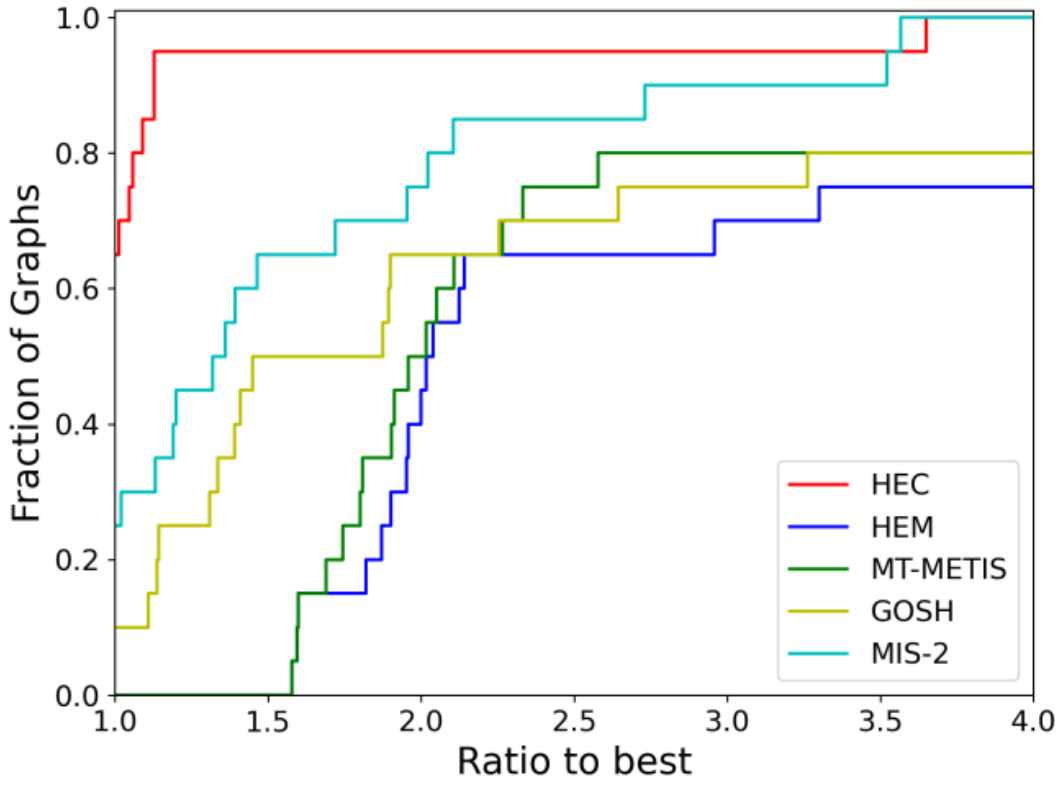

Sphynx partitioning time on Spock relative to Summit (same # GPU).

# Graph coarsening on GPU

- Collaboration with Kamesh Madduri and Mike Gilbert (Penn State)
- Developed and compared several GPU-based graph coarsening methods
- Used Kokkos for performance portability (CPU, GPU)
- Coarsening can be used for graph partitioning, clustering, and potentially algebraic multigrid
- The code is available in KokkosKernels
- Future plan includes a GPU based multilevel graph partitioner
  - GPU based refinement in progress

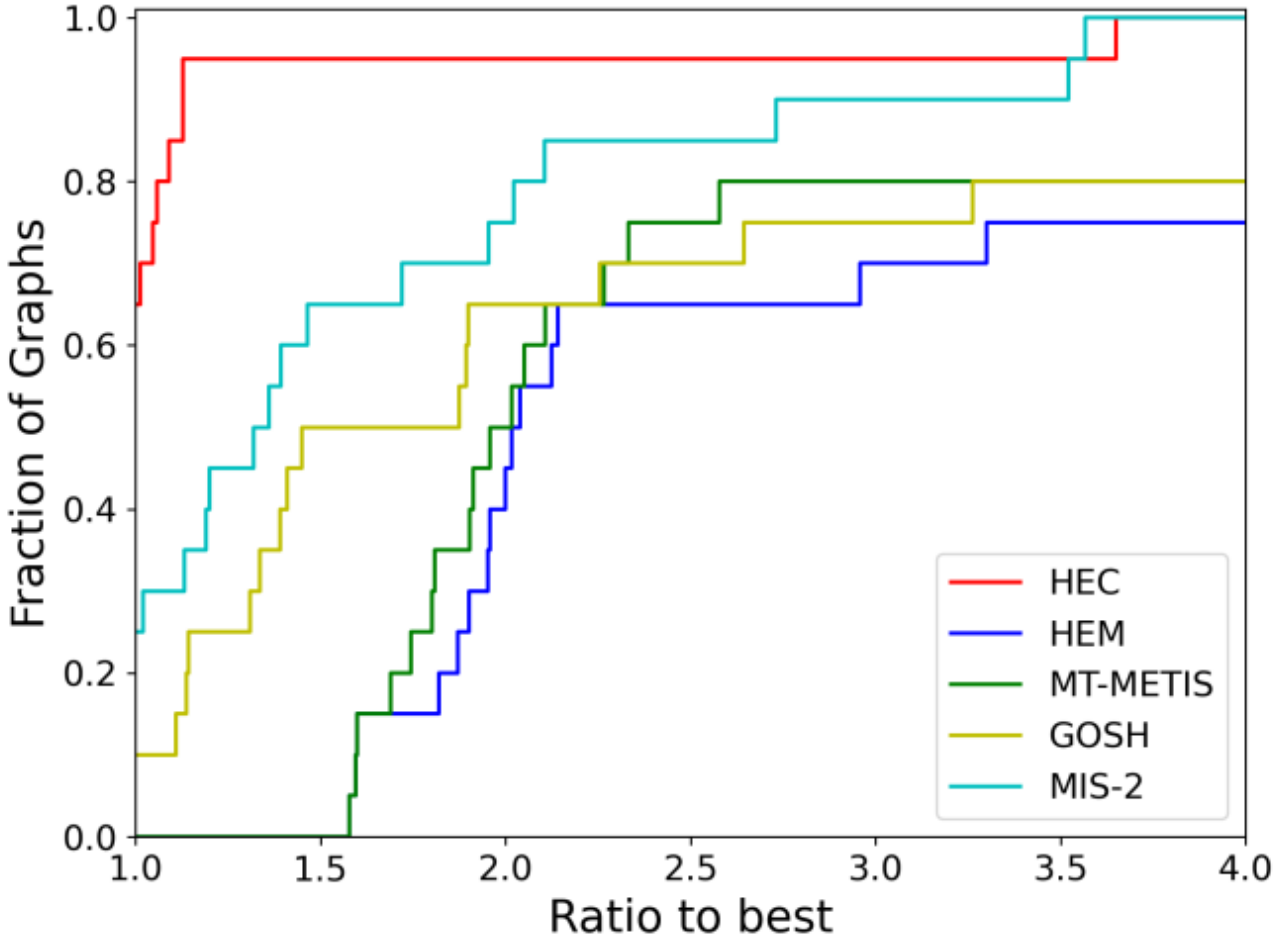


# Graph coarsening - time



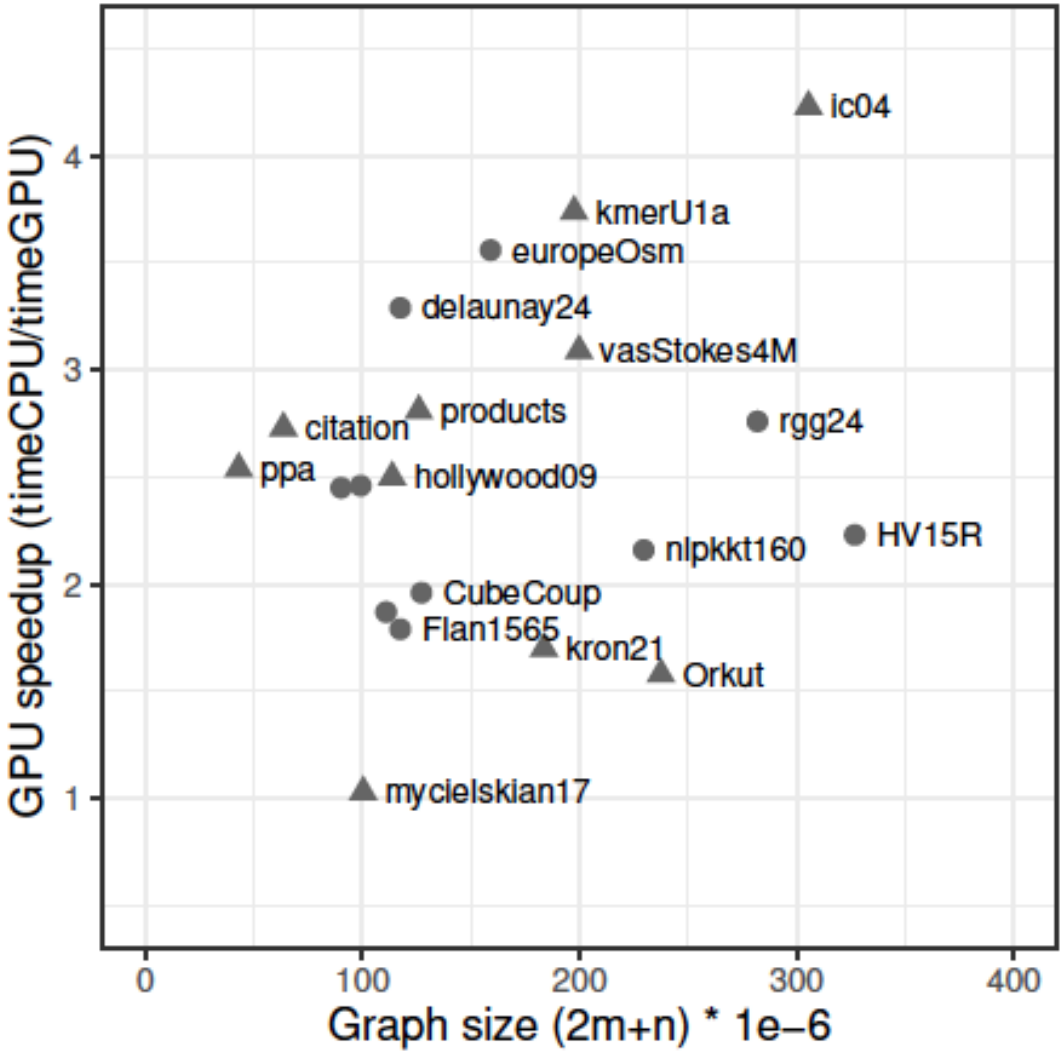
# Graph coarsening - quality

Quality is measured by cut-size in spectral bisection



# HEC coarsening – CPU vs GPU time

HEC was the best method in our tests. Compare one GPU vs 32 CPU cores.

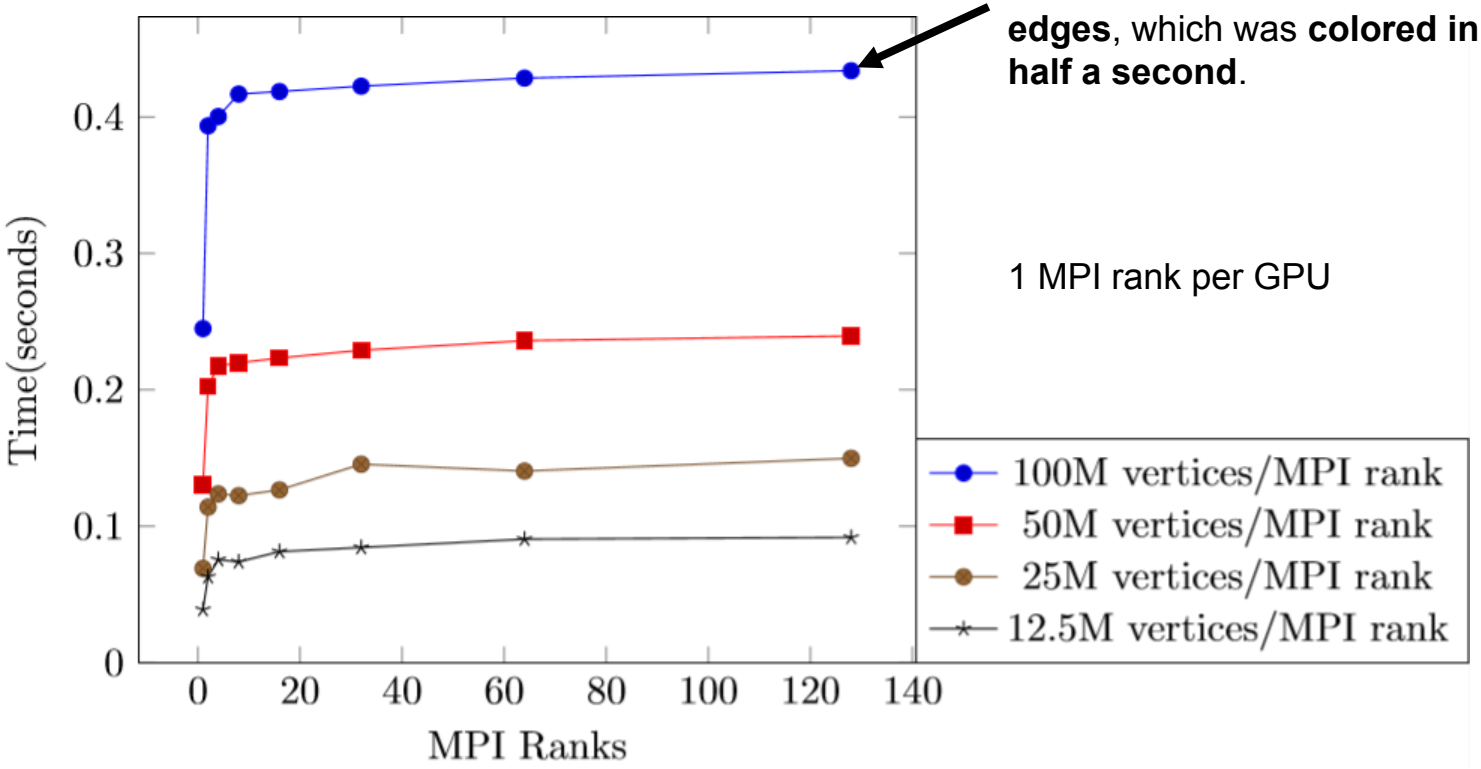


# Graph Coloring on multi-GPU

- Collaboration with RPI (Bogle, Slota)
- Software for graph coloring:
  - Zoltan: Parallel distributed-memory dist-1 and dist-2 coloring
  - KokkosKernels: Parallel shared-memory dist-1 and dist-2 coloring, tuned for GPU
  - ColPack (Purdue): Many models and algorithms, but limited to single node (CPU)
  - NEW: Hybrid MPI+Kokkos coloring for distributed multi-GPU systems
- We reuse algorithms & code
  - Speculative coloring (Gebremedhin & Manne 2000), as used in Zoltan
  - We use KokkosKernels coloring on the node for optimal GPU performance
  - Performance portable to many architectures (all ECP platforms)
  - Will be delivered through Trilinos/Zoltan2 (soon)
- Applications
  - ATDM/SNL Empire: Coloring for Jacobians
  - ATDM/SNL Scalable solvers: Coarsening for MueLu

# Coloring: Weak Scaling on multi-GPU

3D mesh, run on Nvidia GPU cluster at RPI




The largest input we ran has **12.8 Billion vertices** and **76.7 Billion edges**, which was **colored in under half a second**.

1 MPI rank per GPU

# ECP Collaboration: SPARC

- SPARC is a SNL National Security application
  - Computational bottleneck is nonlinear solves
    - Sequence of linear solves, same pattern
  - Want to use adjoints and AD for sensitivity studies
  - Need graph coloring of Jacobian matrix
    - This is distance-2 coloring of the bipartite graph
  - Developed user-friendly Tpetra::CrsColorer interface
    - Thanks to Eric Phipps!
    - Currently only CPU version, but multi-GPU support in progress
- SPARC team is testing it on up to 288 cores/processors.

# Thank you!

Papers:

- Gilbert et al., “Performance Portable Graph Coarsening for Multilevel Graph Analysis”, Proc. Of IPDPS’21.
- Bogle et al., “Parallel graph coloring for multi-GPU systems”, Proc. Of IAAA, 2020.
- Acer, Boman, Glusa, Rajamanickam, “Sphynx: a Parallel Multi-GPU Partitioner”, Parallel Computing, 2021.

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.


EXASCALE COMPUTING PROJECT

# Backup slides

# Highlights: Sphynx partitioner

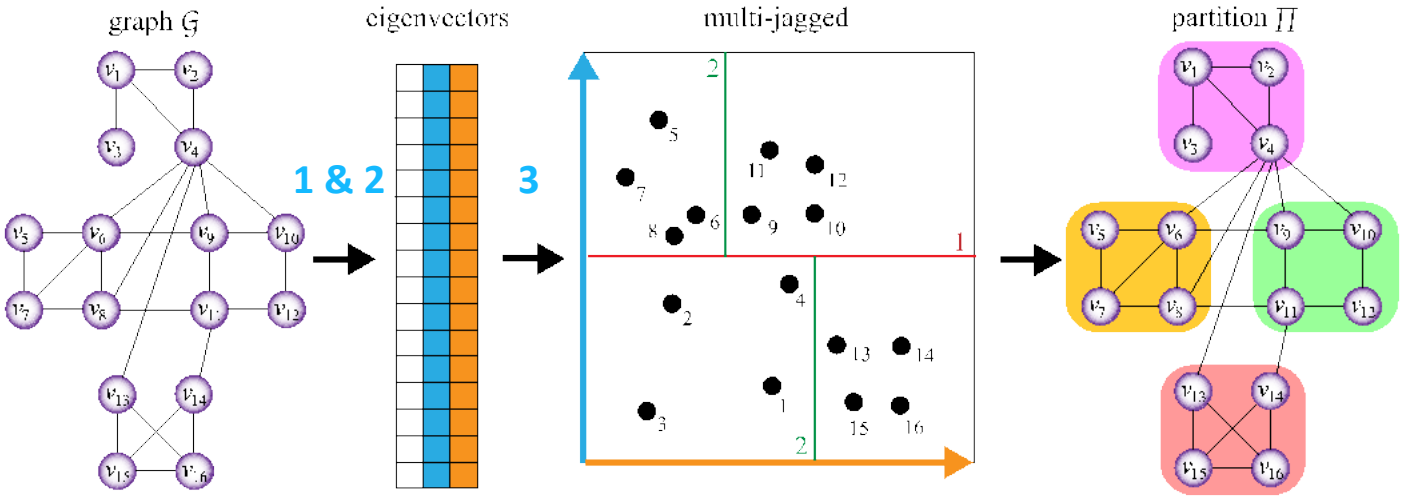
- Sphynx: Spectral Partitioning for HYbrid aNd aXelarator-based systems
- Sphynx uses several Trilinos packages using Kokkos for performance portability
- Sphynx is the first multi-GPU partitioner for distributed-memory systems
- Compared to ParMETIS, Sphynx is faster on irregular graphs and obtains a close cutsize on regular graphs

# Spectral partitioning

- Use graph Laplacian for embedding into low-dimensional space
- Adjacency matrix $A = (a)_{ij} = \begin{cases} 1 & \text{if } e_{i,j} \in E \\ 0 & \text{otherwise} \end{cases}$
- Degree matrix $D = (d)_{ij} = \begin{cases} \deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$
- Form a Laplacian matrix:
  - Combinatorial Laplacian $L_C = D - A$
  - Normalized Laplacian $L_N = I - D^{-1/2} A D^{-1/2}$
- The 2nd lowest eigenvector (Fiedler vector) approximately minimizes the edge cut
  - Spectral recursive bisection was introduced by Pothen, Simon, Liou (SIMAX, 1990)
- Sphynx computes $(\log K + 1)$ eigenvectors on the Laplacian, all at once
  - No recursive bisection, only one eigensolve

# Trilinos for spectral partitioning

1. Create Laplacian $L$ for $G$ – Tpetra CrsMatrix, Kokkos parallel_for
2. Compute $(\log K + 1)$ eigenvectors of $L$ using LOBPCG [1] – Anasazi
   - First eigenvector: trivial, not used
   - Remaining vectors: coordinates to embed $G$ into $\log K$-dimensional space
3. Compute a $K$-way partition on coordinates using multi-jagged [2] – Zoltan2




[1] A. V. Knyazev,"Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method," SIAM Journal on Scientific Computing, vol. 23, no. 2, pp. 517–541, 2001.

[2] M. Deveci, S. Rajamanickam, K. D. Devine, and U. V. Catalyurek, "Multi-jagged: A scalable parallel spatial partitioning algorithm," IEEE Transactions on Parallel and Distributed Systems, vol. 27, pp. 803–817, March 2016.

# Preconditioning

- Number of iterations in LOBPCG is a bottleneck
- LOBPCG allows using a preconditioner
  - Reduces #iterations and overall time
- Sphynx uses three preconditioners
  1. Jacobi: $M = diag(A)^{-1}$ (Ifpack2)
     - scaling each row by the inverse of the diagonal, easy to parallelize
  2. Polynomial: $M = p_k(A)$ (Belos)
     - SpMV to apply, highly parallel
     - based on GMRES polynomial
  3. (Algebraic) Multigrid: $A_{\ell+1} = RA_\ell P$ (MueLu)
     - multilevel, scalable solver but costlier setup

# Experiments

- The GPU focus: MPI+Cuda
- Performed on Summit
- General experiments performed on 24 GPUs
  - Desired number of parts = K = 24
- Strong scaling performed on 6, 24, 96 GPUs
- Weak scaling performed on 4, 32, 256 GPUs
- Each GPU is exclusively used by one MPI rank (default)
- Device allocations in the Unified Virtual Memory (default)
- Initial distribution of the test graphs: 1D block
  - This is the default distribution with Tpetra CrsMatrix
- Parameter sensitivity and comparison against ParMETIS
  - Performance metrics: cutsize and runtime

# Results

Eigenvalue Problem:

| Average results normalized w.r.t combinatorial | | | | | |
|---|---|---|---|---|---|
| | | generalized | | normalized | |
| | preconditioner | runtime | cutsize | runtime | cutsize |
| regular | Jacobi | 0.81 | 1.15 | 0.43 | 2.26 |
| | Polynomial | 0.73 | 1.21 | 0.54 | 2.45 |
| | MueLu | 0.99 | 1.12 | 0.95 | 2.20 |
| irregular | Jacobi | 0.75 | 0.83 | 0.26 | 1.36 |
| | Polynomial | 0.36 | 0.84 | 0.02 | 0.83 |
| | MueLu | 0.71 | 0.90 | 0.31 | 1.68 |

Default: combinatorial for regular graphs,

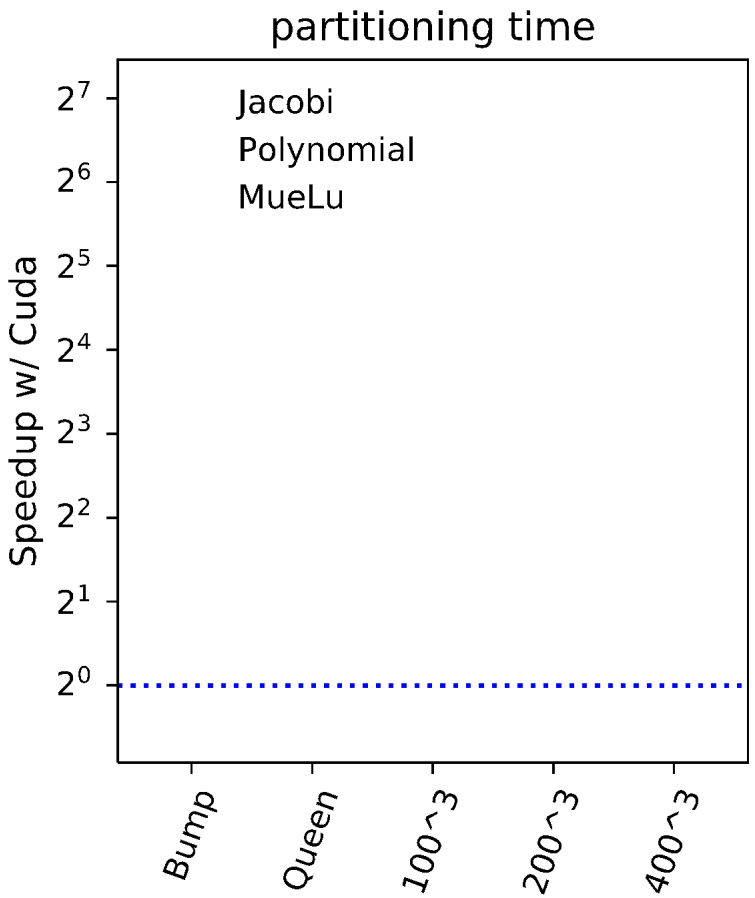
generalized for irregular graphs with Jacobi and MueLu, and

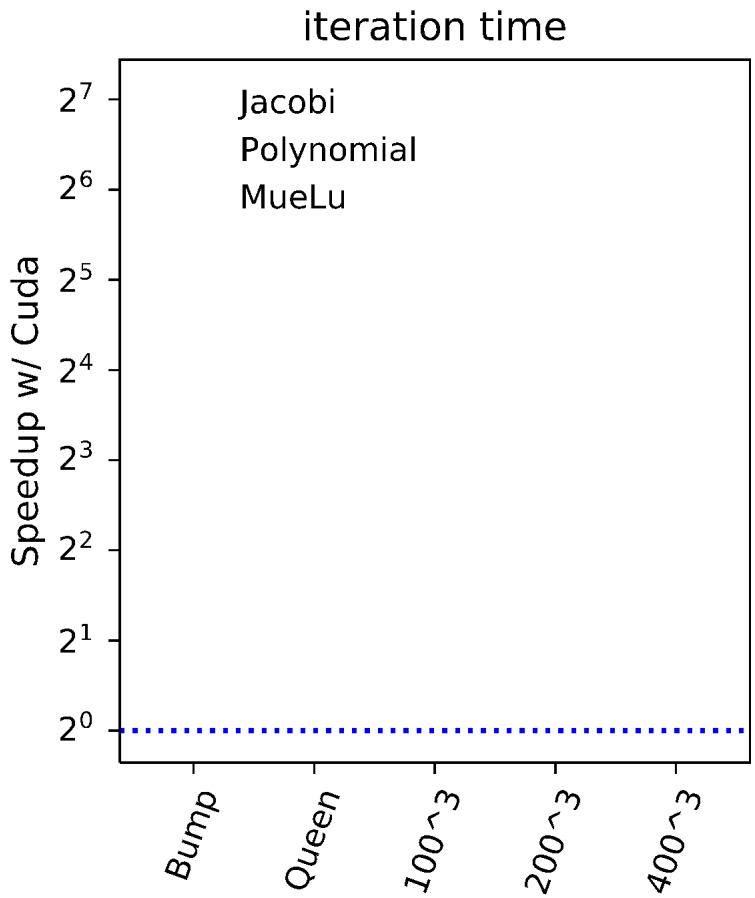
normalized for irregular graphs with Polynomial.

# Results

GPU vs CPU:

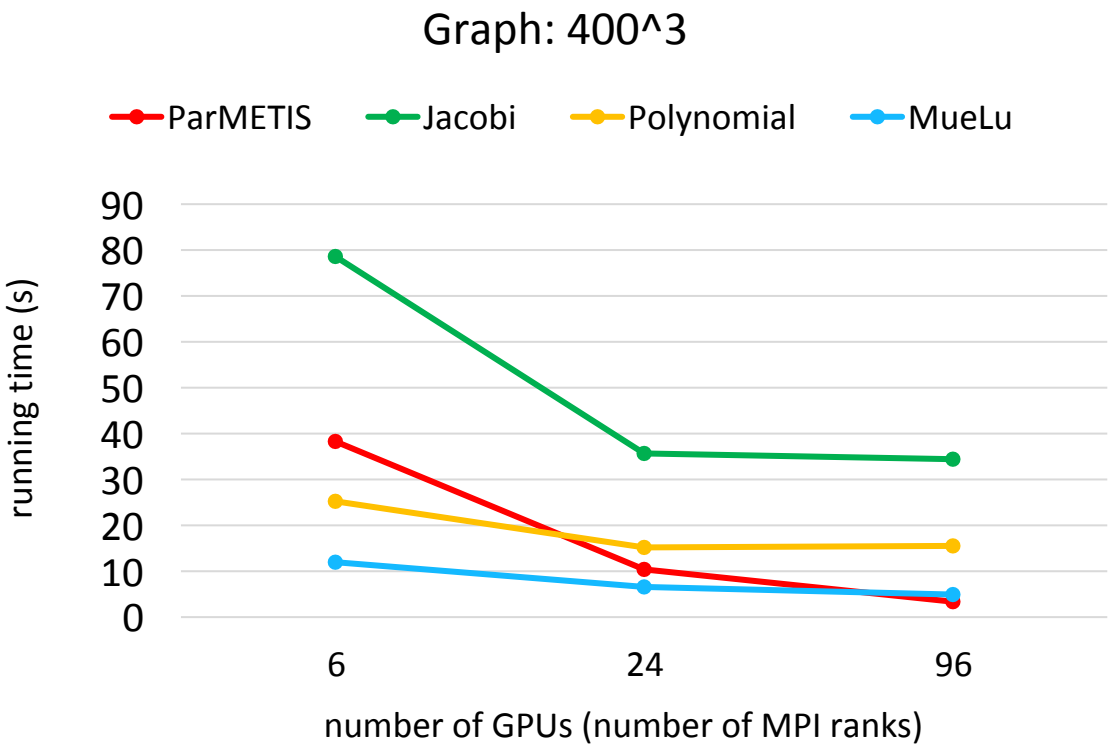
Typical application use case. 24 GPUs vs 24 MPI ranks on CPU.



# Results

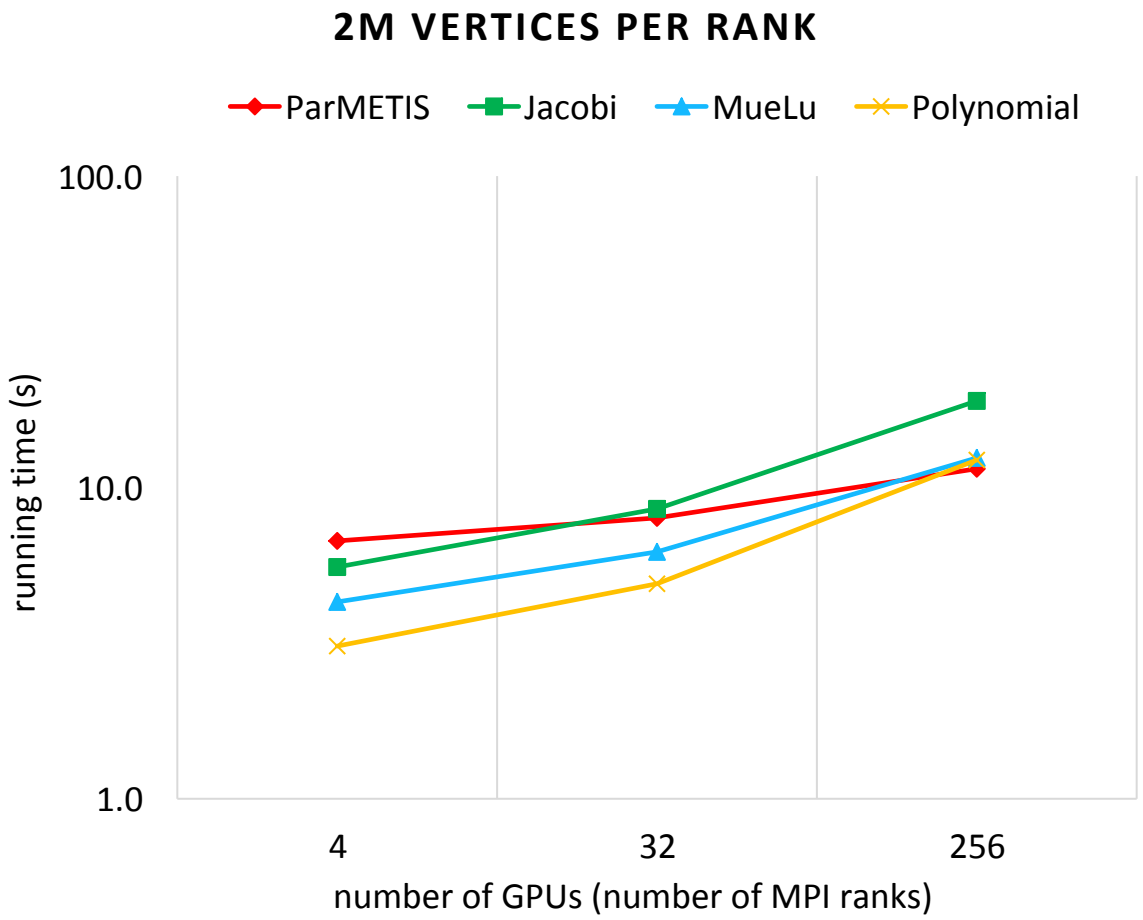
Strong Scaling:



# Results

Weak Scaling:

(tolerance = 1e-2 for all)



# Conclusions: Sphynx

- First multi-GPU partitioner on distributed-memory systems
  - Spectral method is based on linear algebra, so well suited for GPU
  - Many knobs to tune the performance (preconditioners, problem type,..)
- Built on top of other Trilinos packages, lots of code reuse
  - Performance portability to many GPU architectures via Kokkos
  - Any improvement in Kokkos, Anasazi, MueLu, …, will lead to an improvement in Sphynx
- Sphynx is a subpackage of Zoltan2 (in Trilinos):

  https://github.com/trilinos/Trilinos/tree/master/packages/zoltan2/sphynx