Effective Parallel Computation of Eigenpairs to Detect Anomalies in Very Large Graphs

Michael M. Wolf, **Benjamin A. Miller**

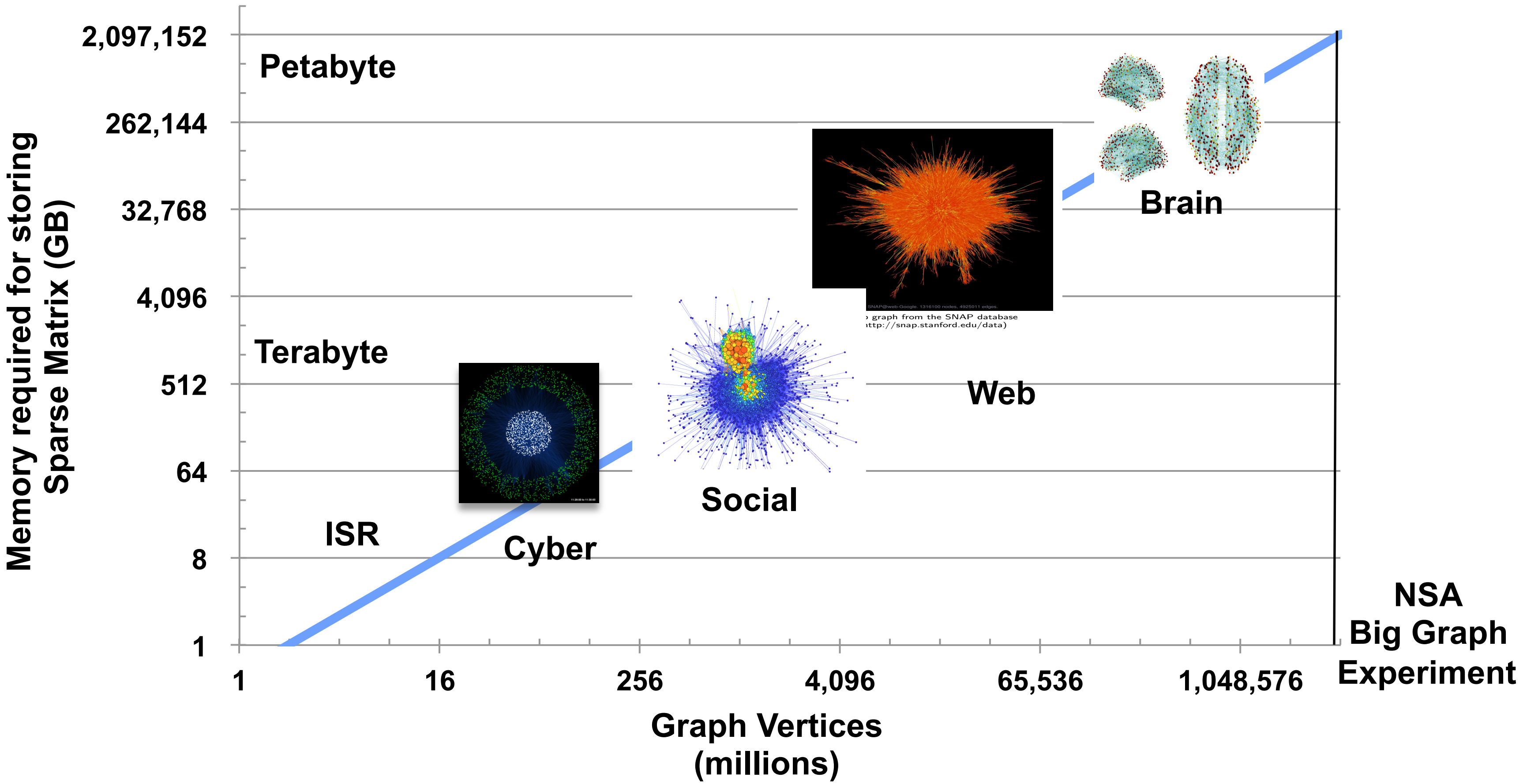
SIAM PP14

18 February 2014

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

This work is sponsored by the Intelligence Advanced Research Projects Activity (IARPA) under Air Force Contract FA8721-05-C-0002. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

Big Data Challenge

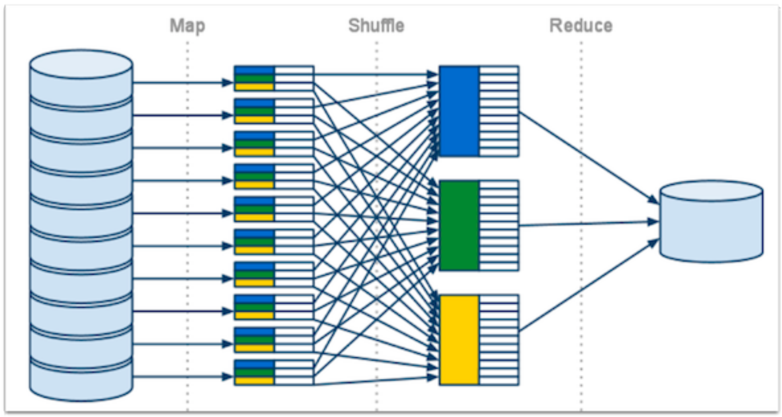


How do we address the data storage and compute challenges posed by the problem scales of interest to the DoD/IC community?

P. Burkhardt and C. Waring, “An NSA Big Graph experiment,” National Security Agency, Tech. Rep. NSA-RD-2013-056002v1, 2013.

Big Data and HPC

Current approach: Map/Reduce



- Map (<k1,v1>) → <k2,v2>
- Reduce (k2, {<k2,v2>}) → v3
- Each map-reduce step reads from and writes to disk

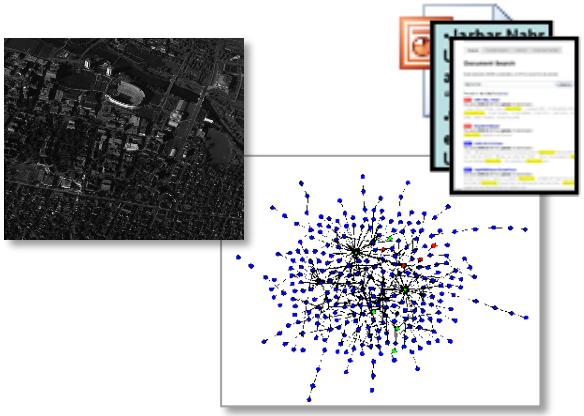
- Map/Reduce provides one way to deal with large problem sizes, but is too limited and too slow
  - Poorly suited for iterative sparse matrix and graph algorithms when fast runtime is essential
- Our approach uses High Performance Computing techniques to tackle big data
  - Leverage HPC sparse linear algebra packages (e.g., Trilinos)

Outline

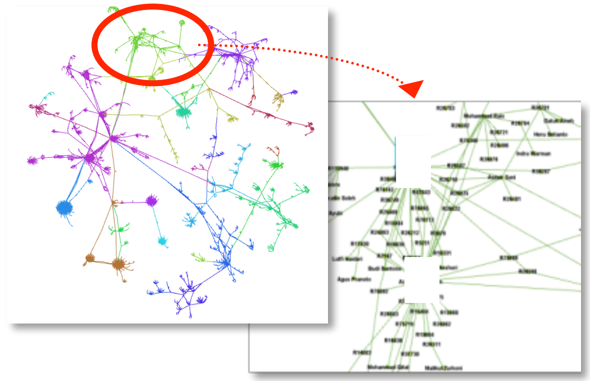
- Big Data and High Performance Computing
- Anomaly Detection in Graphs
- Signal Processing for Graphs (SPG)
- Improving Sparse Matrix-Vector Multiplication (SpMV) Performance
- Improving Performance of Moving Average Filter
- Related Ongoing and Future Work
- Summary

Example Applications of Graph Analytics

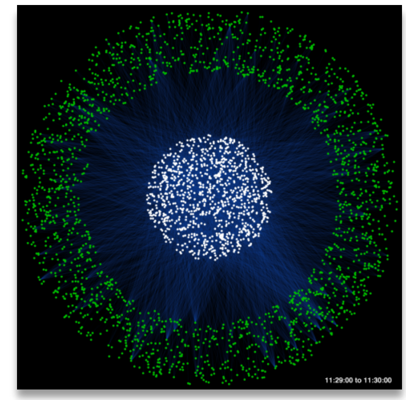
ISR	Social	Cyber



- Graphs represent entities and relationships detected through multiple sources
- 1,000s – 1,000,000s tracks and locations
- GOAL: Identify anomalous patterns of life



- Graphs represent relationships between individuals or documents
- 10,000s – 10,000,000s individual and interactions
- GOAL: Identify hidden social networks

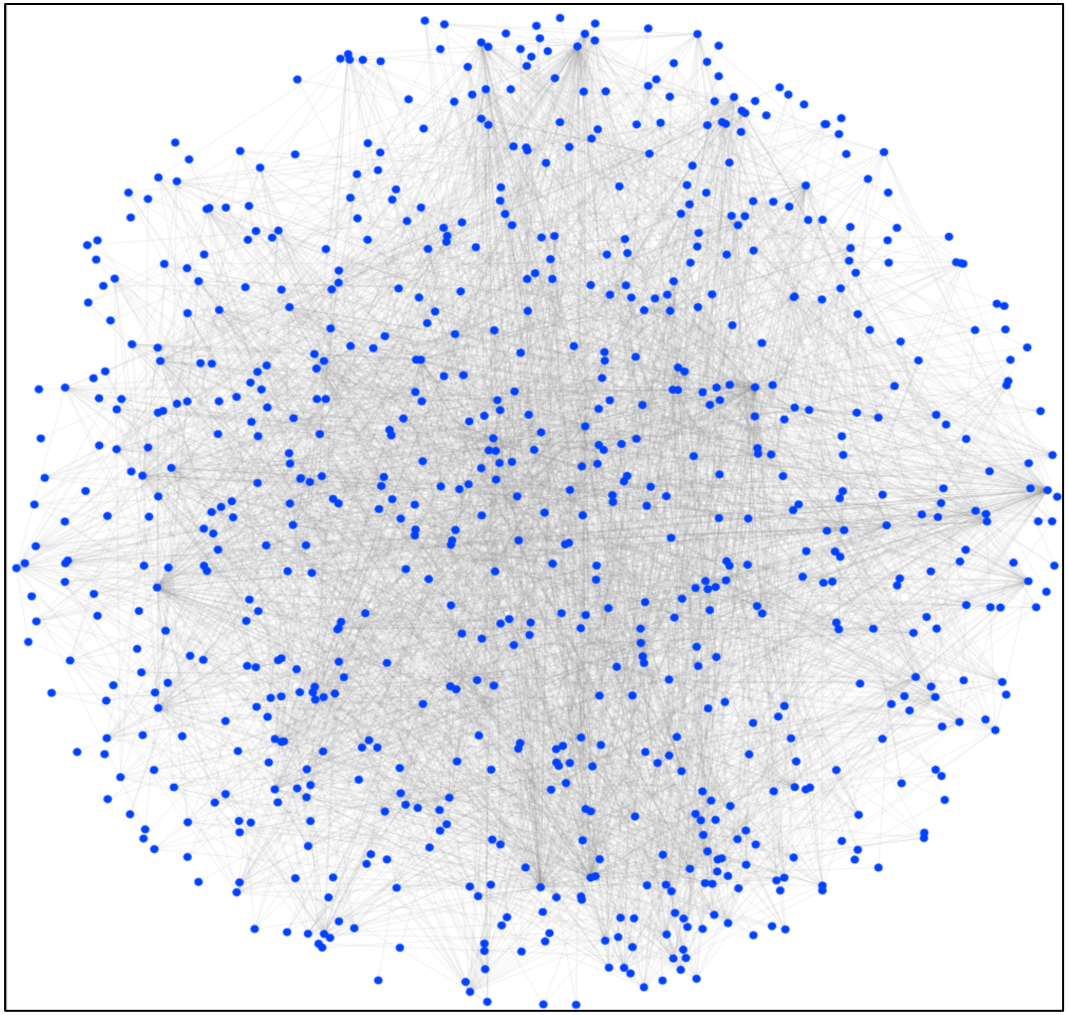


- Graphs represent communication patterns of computers on a network
- 1,000,000s – 1,000,000,000s network events
- GOAL: Detect cyber attacks or malicious software

Cross-Mission Challenge:
Detection of subtle patterns in massive multi-source noisy datasets

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

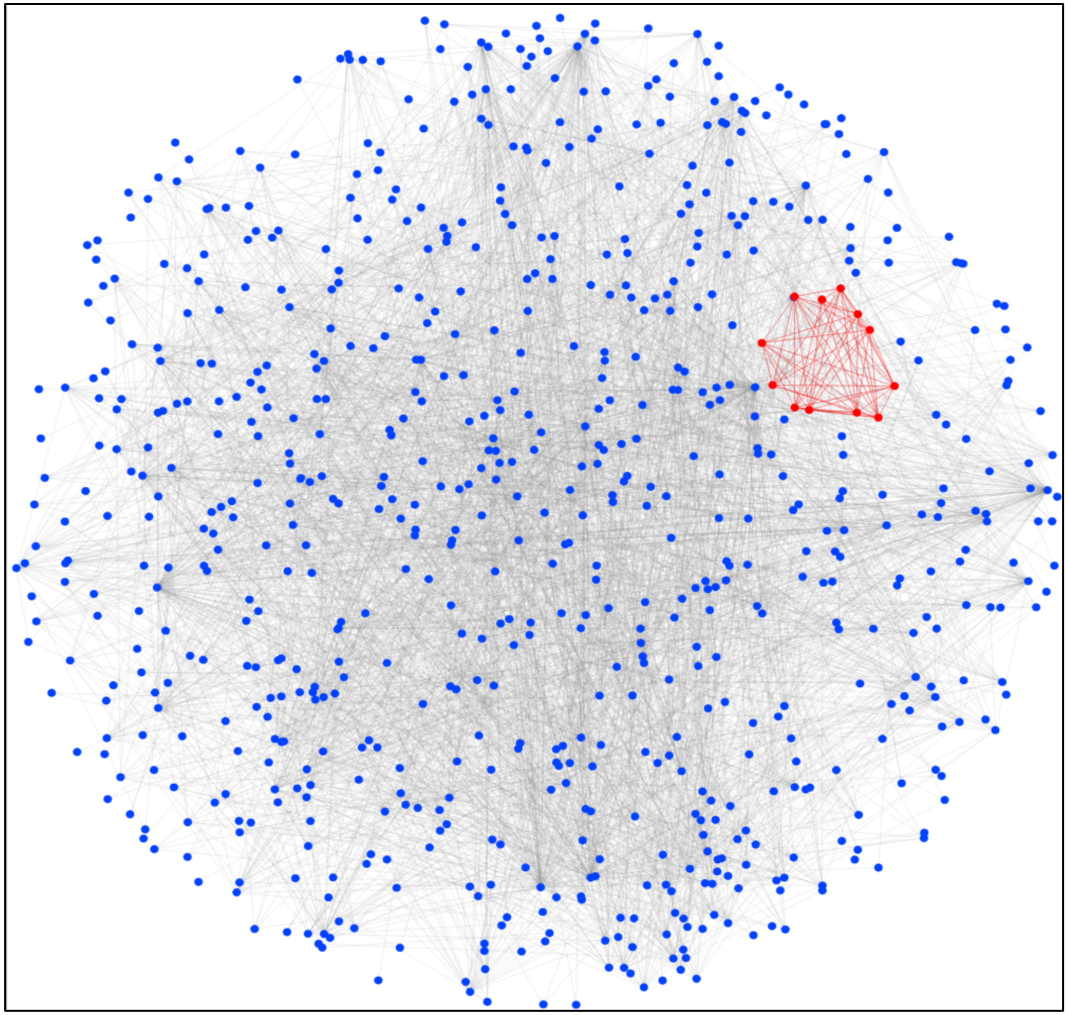
Example: Network Traffic Surrogate



Graph Statistics

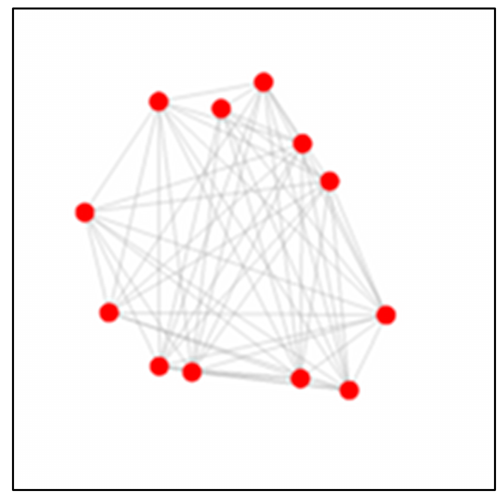
- R-Mat
- Parameters derived from network traffic data
- 1024 vertices

Big Data Challenge: Activity Signatures



Graph Statistics

- R-Mat
- Parameters derived from network traffic data
- 1024 vertices
- Anomaly: 12 vertices
- Anomaly: 1% of graph (often smaller)



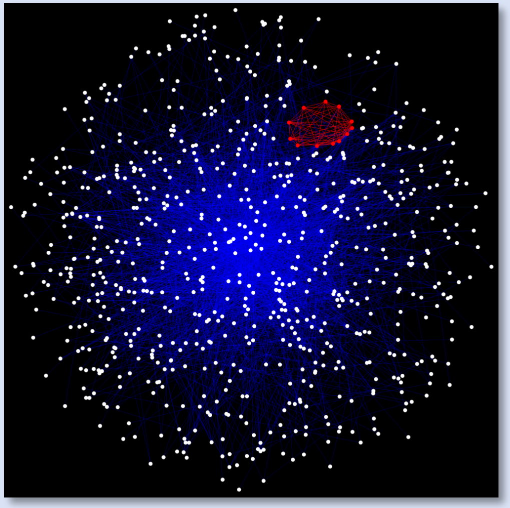
Challenge: Activity signature is typically a weak signal

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

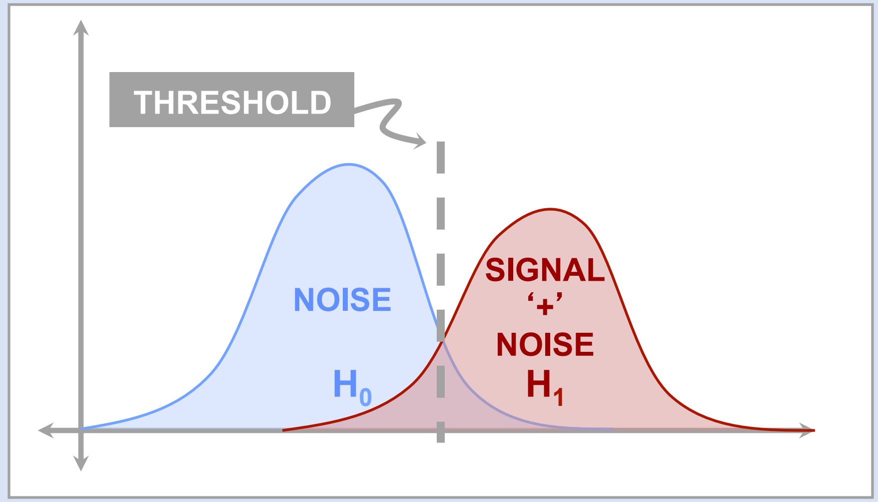
Outline

- Big Data and High Performance Computing
- Anomaly Detection in Graphs
- Signal Processing for Graphs (SPG)
- Improving Sparse Matrix-Vector Multiplication (SpMV) Performance
- Improving Performance of Moving Average Filter
- Related Ongoing and Future Work
- Summary

Statistical Detection Framework for Graphs



Graph Theory



Detection Theory

Develop fundamental graph signal processing concepts

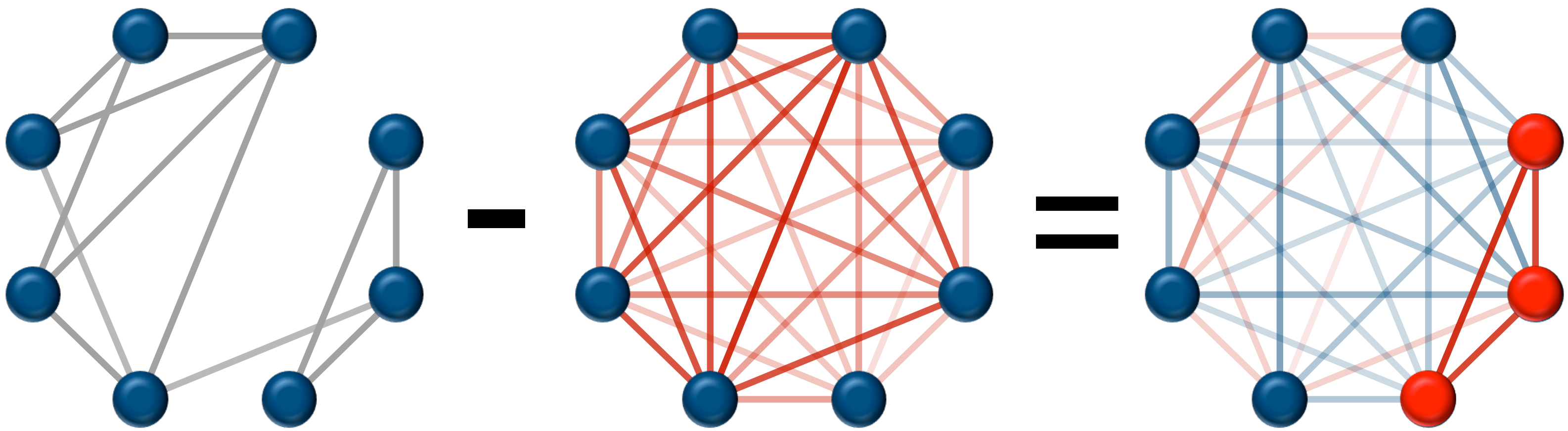
Demonstrate in simulation

Apply to real data

Residuals Example: Anomalous Subgraph

Observed Graph	Graph Model	Residual Graph
G_1	$E[G]$	$R[G_1]$

H_1



- Residual graph represents the difference between the observed and expected
- *Coordinated* vertices (subsets of vertices connected by edges with large edge weights) in residual graph will produce much stronger signal than uncoordinated vertices

Detection framework is designed to detect coordinated deviations from the expected topology

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SPG Processing Chain

DIMENSIONALITY REDUCTION

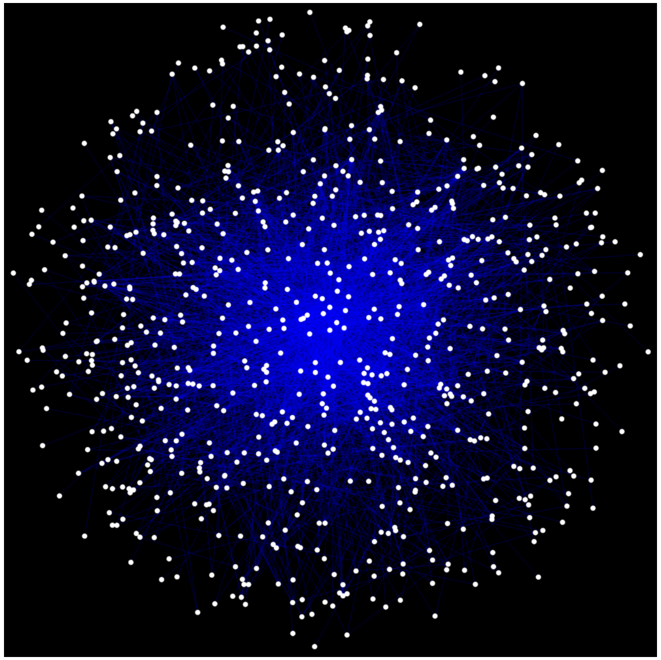
TEMPORAL INTEGRATION → **GRAPH MODEL CONSTRUCTION** → **RESIDUAL DECOMPOSITION** → **COMPONENT SELECTION** → **ANOMALY DETECTION** → **IDENTIFICATION**

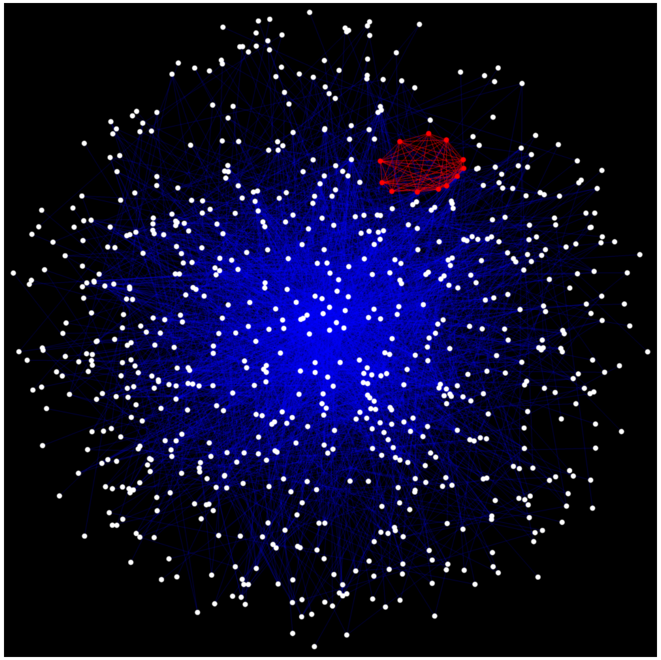
Input

- Graph
- No cue

Output

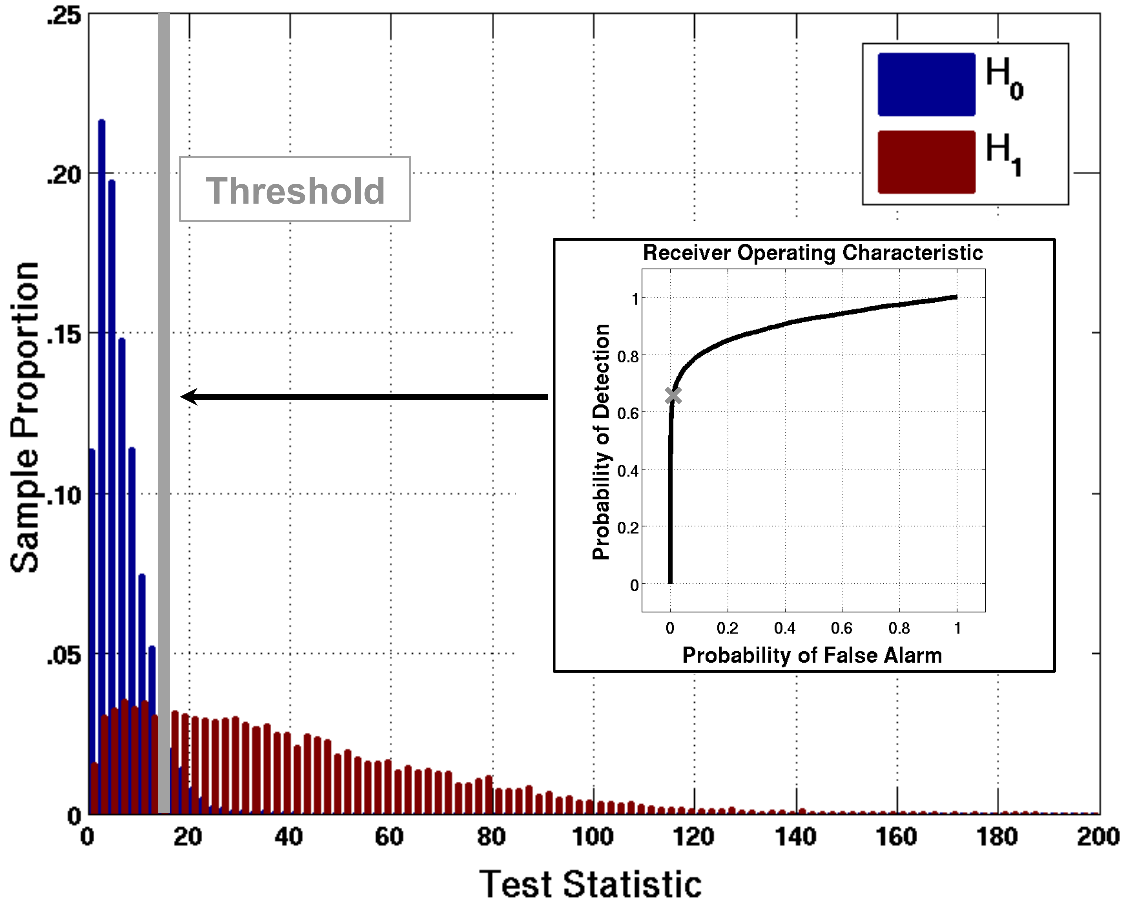
- Statistically anomalous subgraph(s)





Anomaly Detection: Setup Phase

Temporal Integration → Graph Model Construction → Residual Decomposition → Component Selection → **Anomaly Detection** → Identification



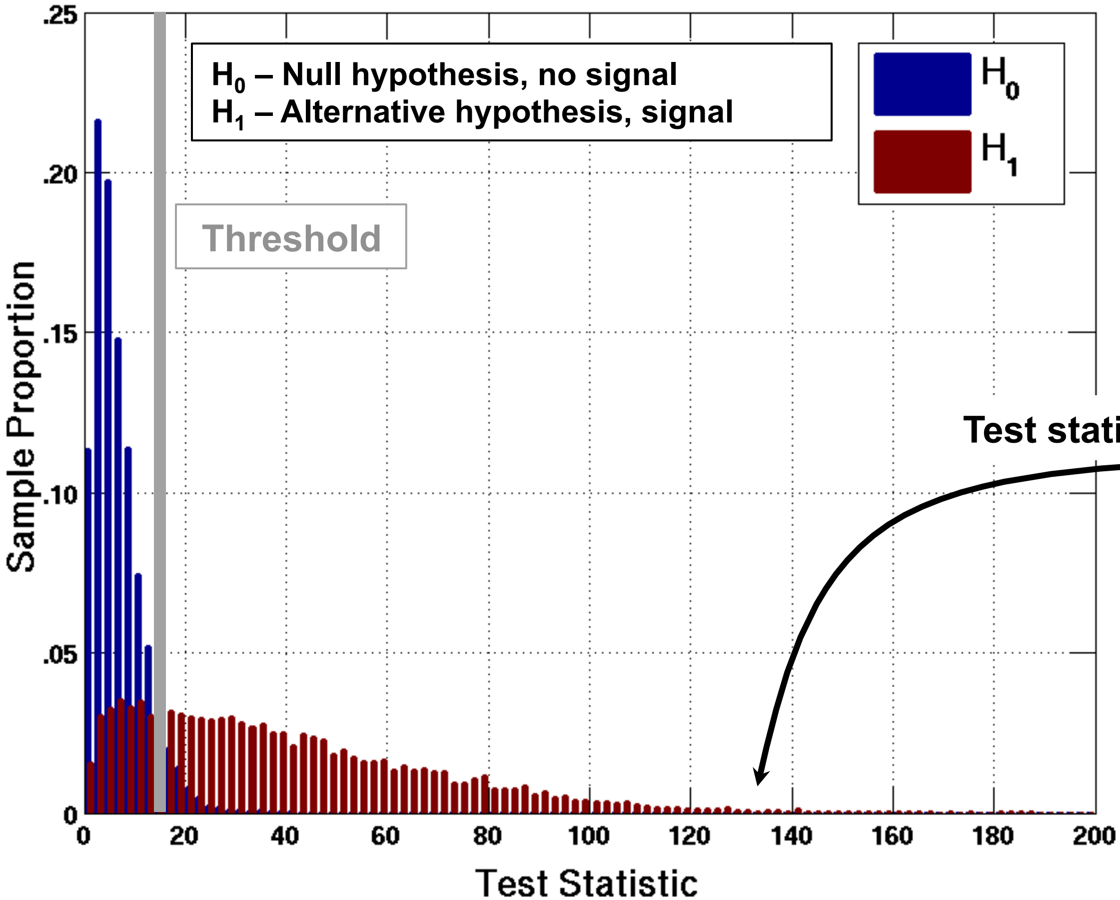
H_0 – Null hypothesis, no signal
H_1 – Alternative hypothesis, signal

Detection Setup

1. Monte-Carlo simulations to generate density functions
2. ROC-curve generated from density function

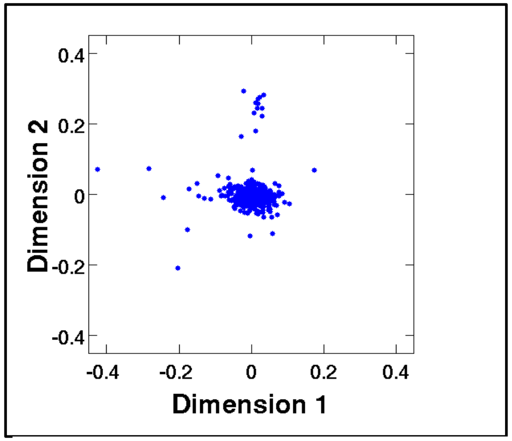
Anomaly Detection

TEMPORAL INTEGRATION → GRAPH MODEL CONSTRUCTION → RESIDUAL DECOMPOSITION → COMPONENT SELECTION → **ANOMALY DETECTION** → IDENTIFICATION



Test statistic

Test statistic calculated for observed graph:



$$\chi^2_{\max} = \max_{\theta} \chi^2 \left(\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} [u_1 \; u_2]^T \right)$$

Test statistic value significantly larger than test statistic value threshold corresponding to 1% false alarm rate

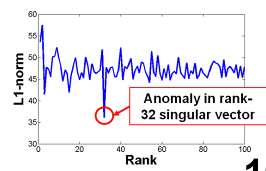
Detection Methods, Effectiveness, and Cost

Notional Comparison of Power and Effectiveness

Detection Power

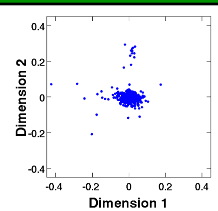
SPCA

Eigenvectors L_1 Norms



100s EVs

χ^2 in 2 Principal Components



2 EV

Spectral Norm

σ_1, λ_1

1 EV

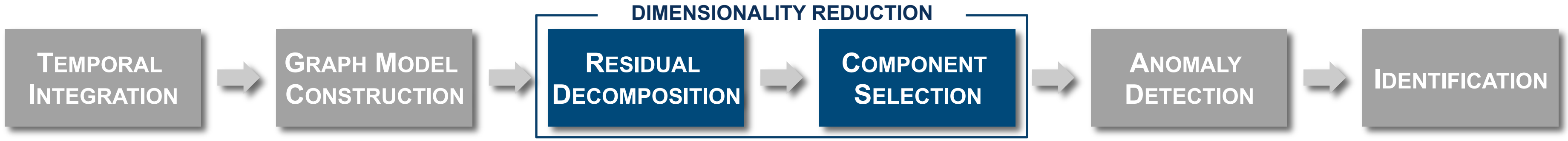
Computation Cost

- More powerful methods require more computation
- For detection of subtle anomalies, need to calculate 100s of eigenvectors fast

$O((|E|r+|V|r^2+r^3)h)^*$ to compute r eigenvectors

SPCA = Sparse Principal Component Analysis

Computational Focus: Dimensionality Reduction



- Dimensionality reduction dominates computation
- Eigen decomposition is key computational kernel
- Parallel implementation required for very large graph problems
  - Fit into memory
  - Minimize runtime

Need fast parallel eigensolvers

Eigenvalue Problems

$$B = (A - E[A])$$

Solve:

$$Bx_i = \lambda_i x_i, i = 1, \ldots, m$$

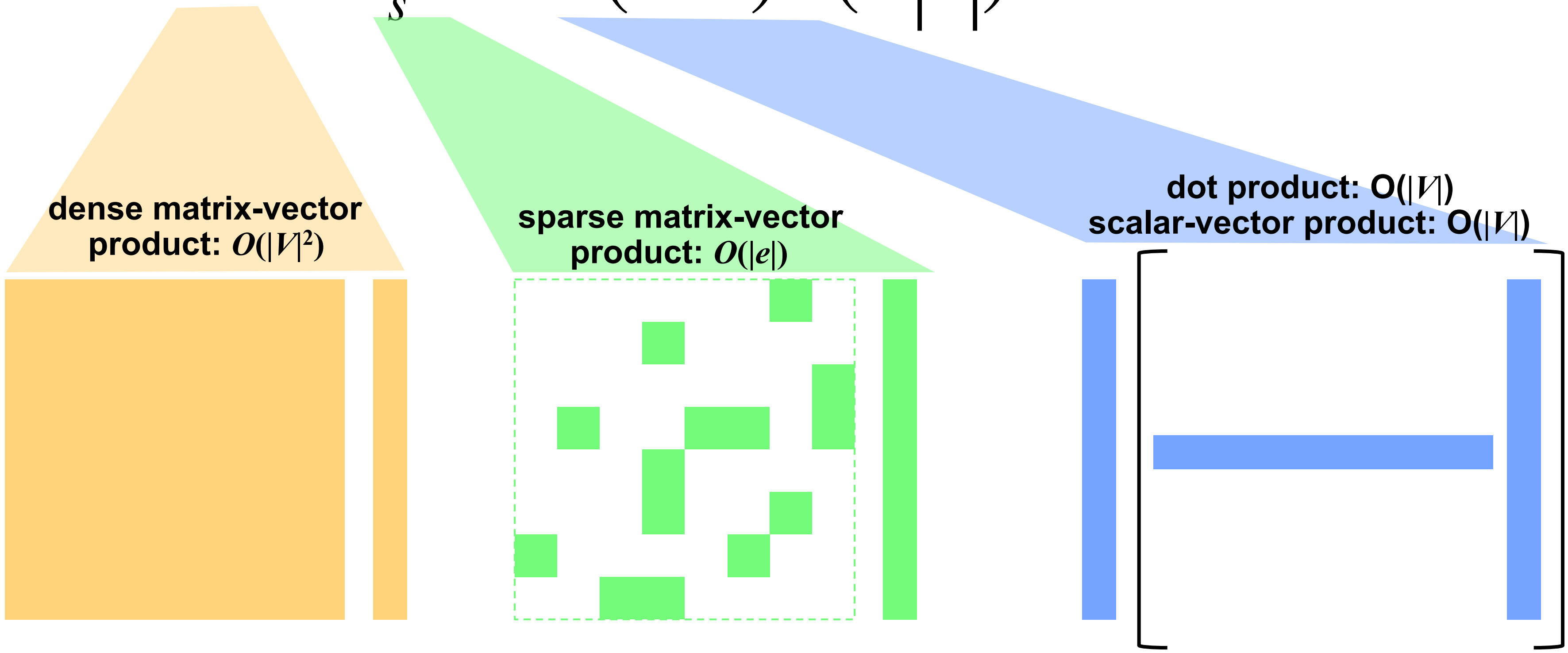
Modularity Matrix	Moving Average Filter
$E[A_s] = \dfrac{k\,k^T}{2\lvert e\rvert}$ $\lvert e\rvert$ – Number of edges in graph $G(A)$ k – degree vector k_i = degree(v_i), $v_i \in G(A)$	$E[A_s(t)] = \sum_{i=1}^{T} h_i A_s(t-i)$ $\vec{h} = \operatorname{argmin}_h \left\lVert A_s(t) - \sum_{i=1}^{T} h_i A_s(t-i) \right\rVert_F$

Modularity Matrix: Computation Breakdown

Matrix-vector multiplication is at the heart of eigensolver algorithms

Operator apply:

$$Bx = A_s x - k\,(k^T x)/(2|e|)$$



Bx **can be computed without storing** ***B*** **(modularity matrix)**

Moving Average Filter: Computational Breakdown

Matrix-vector multiplication is at the heart of eigensolver algorithms

Operator:

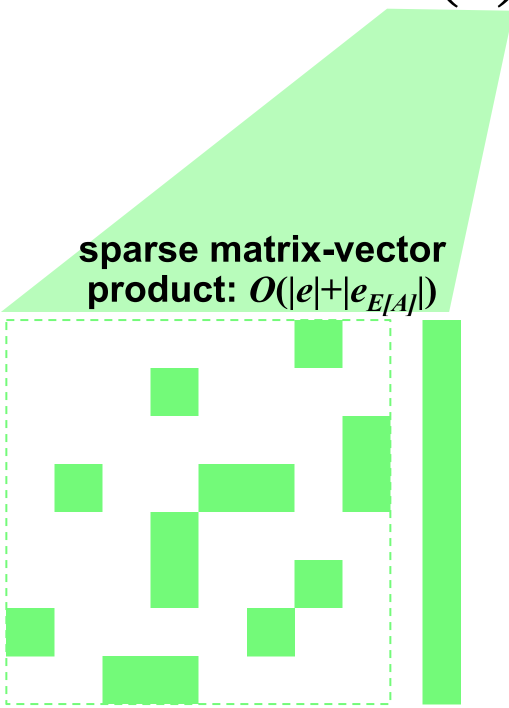
$$B(t) = A_s(t) - E[A_s(t)]$$

$$E[A_s(t)] = \sum_{i=1}^{T} h_i A_s(t-i)$$

Since *E[A(t)]* is sparse, *B(t)* will be sparse

Operator apply:

$$B(t)x$$



Key computational kernel is sparse matrix-dense vector multiplication

Parallel Implementation

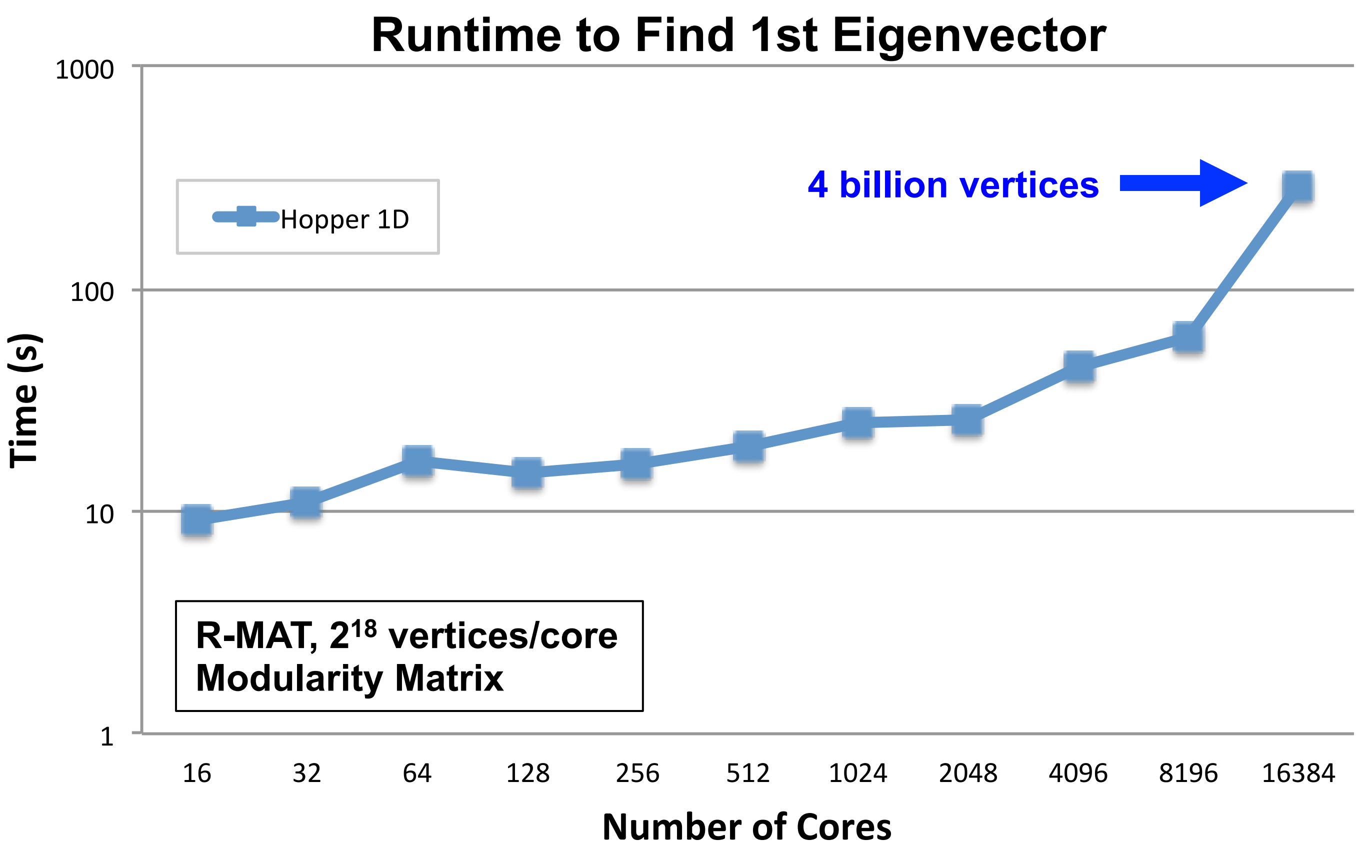
- Using Anasazi (Trilinos) Eigensolver
- 64 bit global ordinals
  - Necessary for graphs with 2^{31} vertices or more
- User defined operators
  - Modularity matrix
  - Moving average filter
  - Apply defined efficiently for particular operator
- Block Krylov-Schur method
  - Symmetric
  - Eigenvalues with largest real component
  - Blocksize=1

Numerical Experiments

- **Matrices**
  - **R-Mat (a=0.5, b=0.125, c=0.125, d=0.25)**
    - **Average nonzeros per row: 8**
    - **Number of rows: 2^{22} to 2^{32}**
- **Two systems**
  - **LLGrid (MIT LL)**
    - **274 compute nodes (8,768 cores)**
    - **Node: two 16-core AMD Opteron 6274 (2.2 GHz)**
    - **Network: 10 GB Ethernet**
  - **Hopper* (NERSC)**
    - **Cray XE6**
    - **6,384 nodes (153,216 cores)**
    - **Node: two 12-core AMD 'MagnyCours' (2.1 GHz)**
    - **Network: 3D torus (Cray Gemini)**
- **Initially: 1D random row distribution (good load balance)**

* This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Weak Scaling – Hopper*

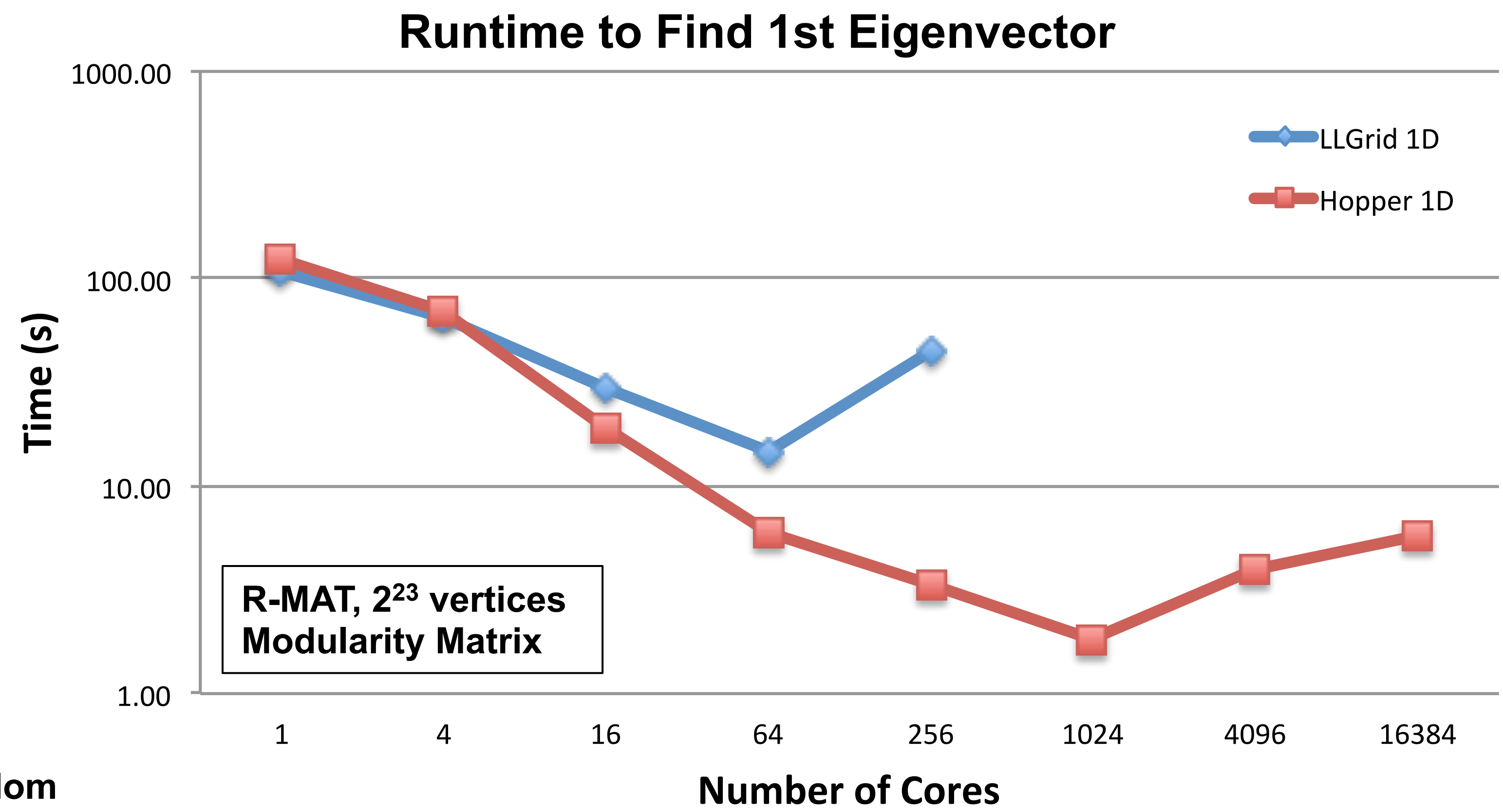


1D random partitioning

Solved system for up to 4 billion vertex graph

* This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Strong Scaling Results

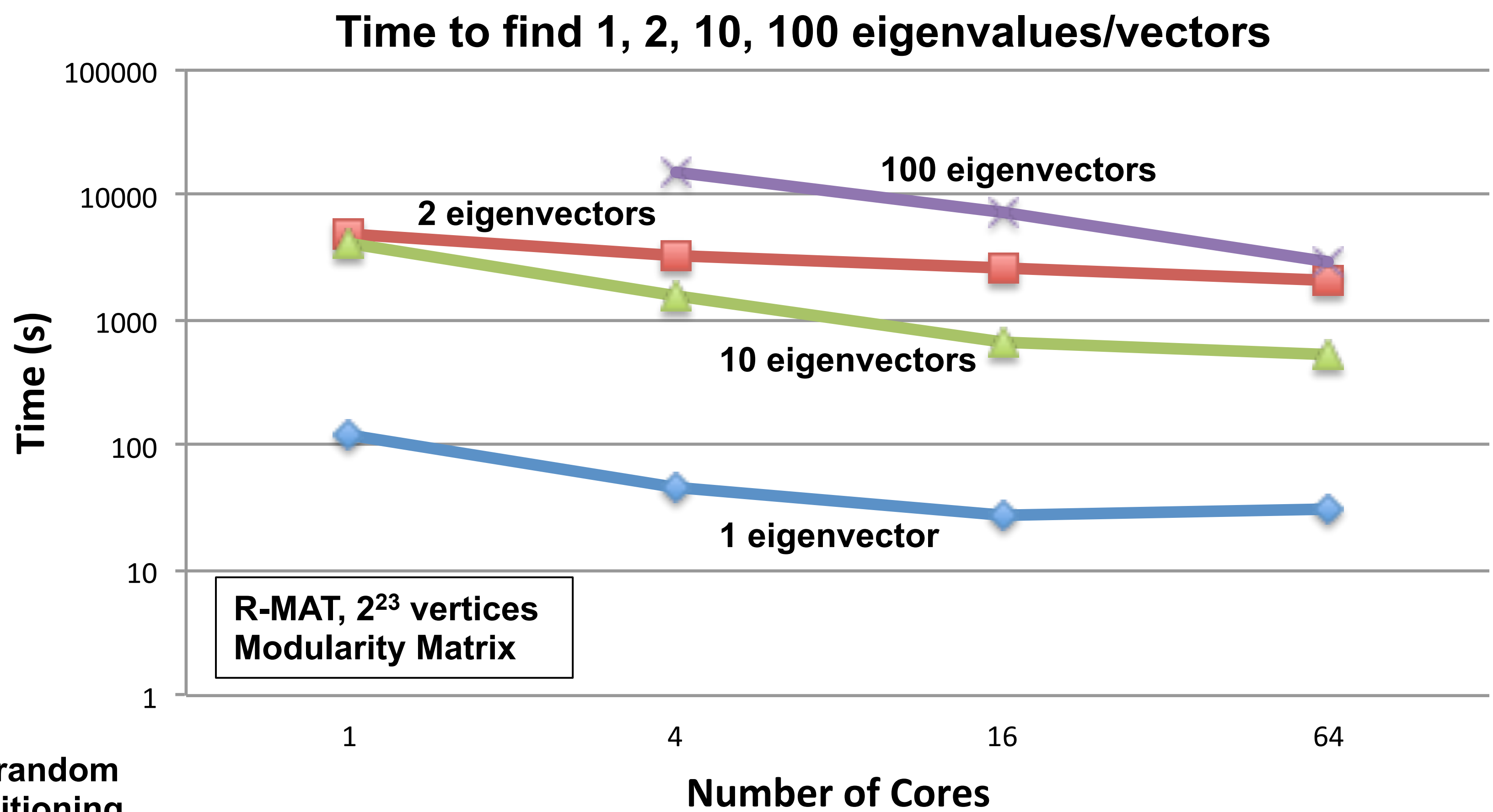


1D random partitioning

Scalability limited and runtime increases for large numbers of cores

* This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Finding Multiple Eigenvectors – LLGrid



1D random partitioning

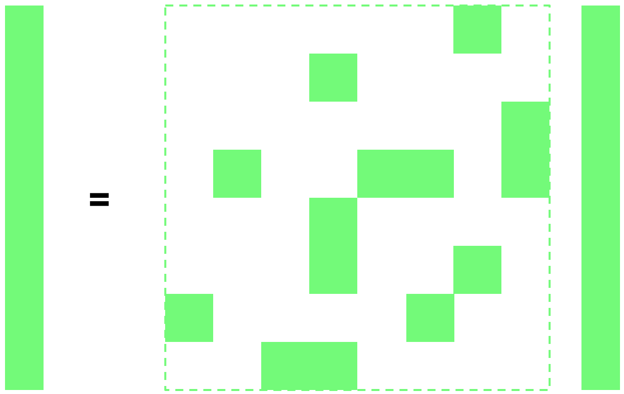
LLGrid system

Significant increase in runtime when finding additional eigenvectors

Outline

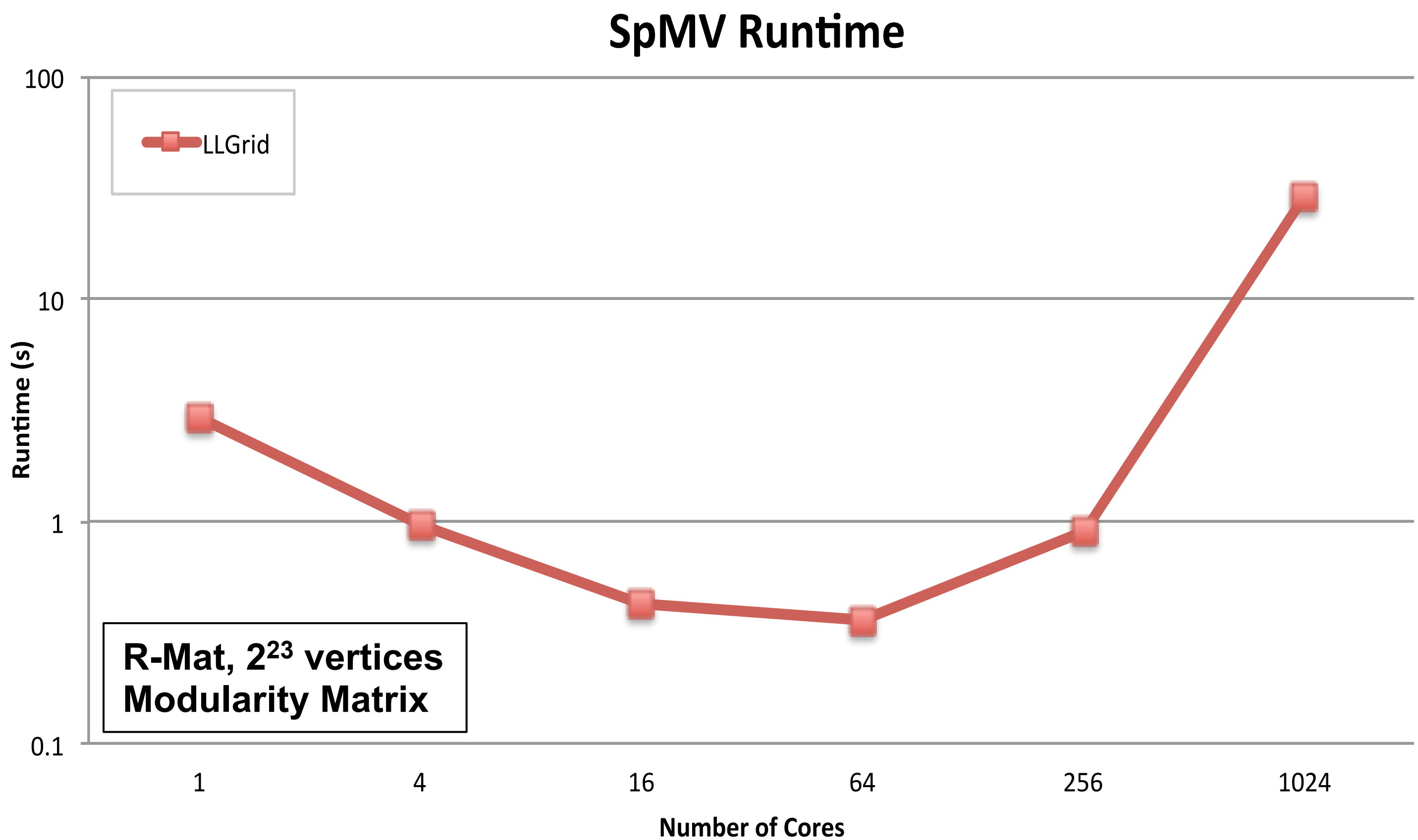
- **Big Data and High Performance Computing**
- **Anomaly Detection in Graphs**
- **Signal Processing for Graphs (SPG)**
- **Improving Sparse Matrix-Vector Multiplication (SpMV) Performance**
- **Improving Performance of Moving Average Filter**
- **Summary**

Sparse Matrix-Vector Multiplication



- **Sparse matrix-dense vector multiplication (SpMV) key computational kernel in eigensolver**
- **Performance of SpMV challenging for matrices resulting from power-law graphs**
  - **Load imbalance**
  - **Irregular communication**
  - **Little data locality**
- **Important to improve performance of SpMV**

SpMV Strong Scaling -- LLGrid



1D random partitioning

Scalability limited and runtime increases for large numbers of cores

Data Partitioning to Improve Parallel Sparse Matrix-Dense Vector Multiplication

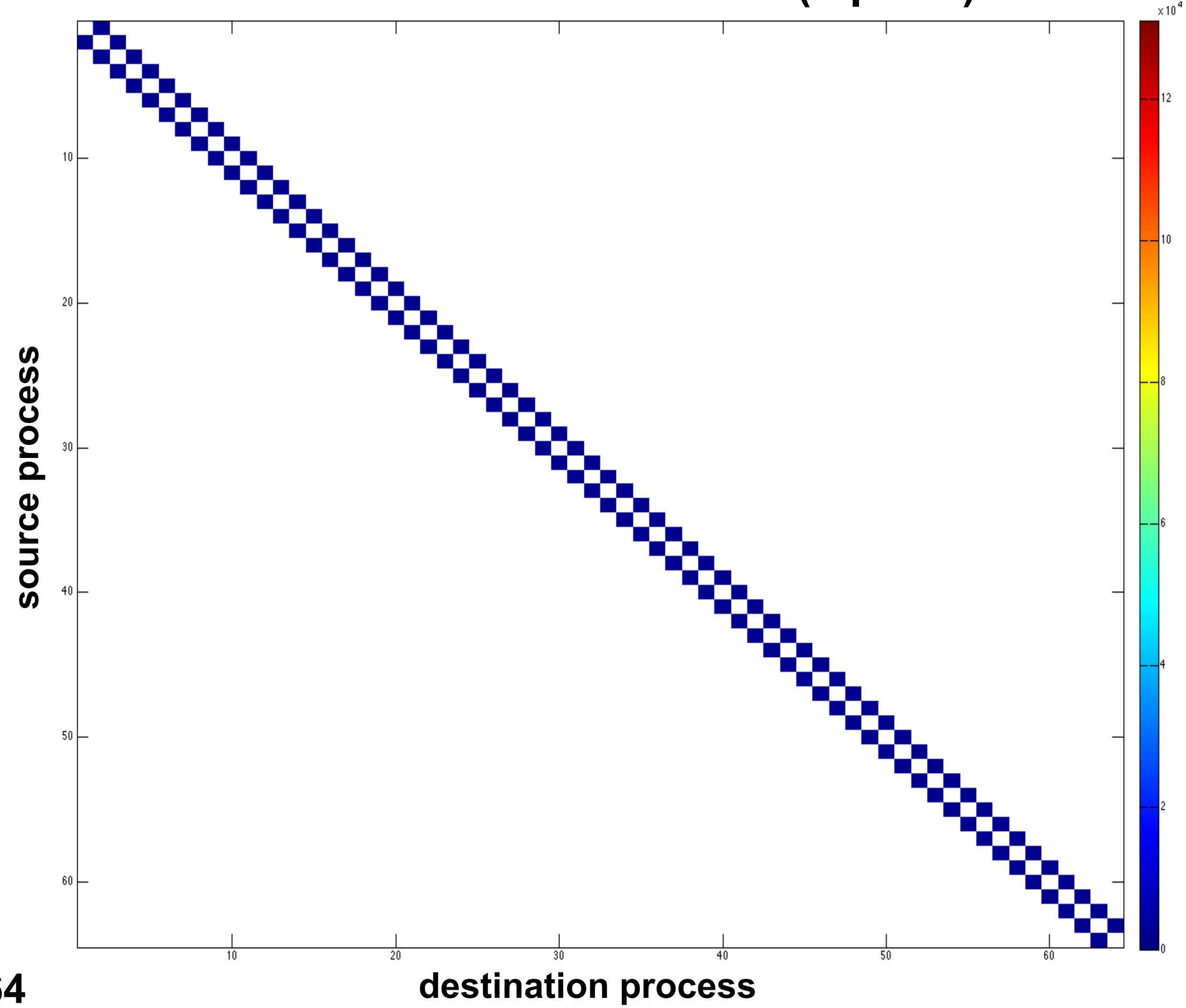
$$
\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix}
=
\begin{bmatrix}
1 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 1 & 9 & 0 & 5 & 0 & 0 & 0 \\
0 & 8 & 1 & 7 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 1 & 0 & 0 & 0 & 7 \\
0 & 0 & 0 & 0 & 1 & 8 & 0 & 0 \\
4 & 0 & 0 & 0 & 3 & 1 & 3 & 0 \\
0 & 0 & 0 & 6 & 0 & 9 & 1 & 4 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 1
\end{bmatrix}
\begin{bmatrix} 1 \\ 2 \\ 4 \\ 3 \\ 1 \\ 4 \\ 2 \\ 1 \end{bmatrix}
$$

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

- Partition matrix nonzeros
- Partition vectors

Communication Pattern: 1D Block Partitioning

2D Finite Difference Matrix (9 point)



P=64

Number of Rows: 2^{23}
Nonzeros/Row: 9

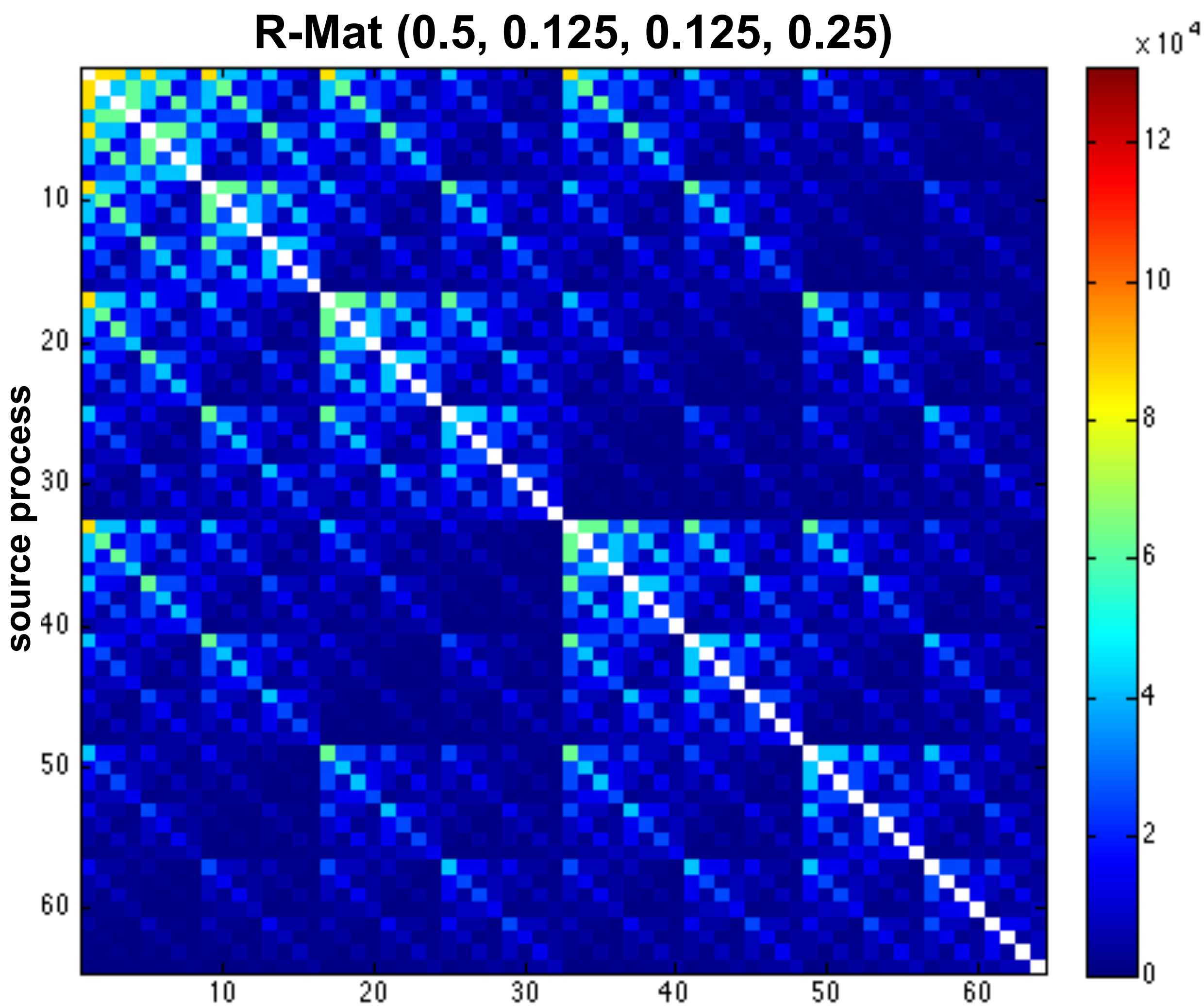
NNZ/process
min: 1.17E+06
max: 1.18E+06
avg: 1.18E+06
max/avg: 1.00

 # Messages (Phase 1)
total: 126
max: 2

Volume (Phase 1)
total: 2.58E+05
max: 4.10E+03

Nice properties:
Great load balance
Small number of messages
Low communication volume

Communication Pattern: 1D Block Partitioning



P=64

Number of Rows: 2^{23}
Nonzeros/Row: 8

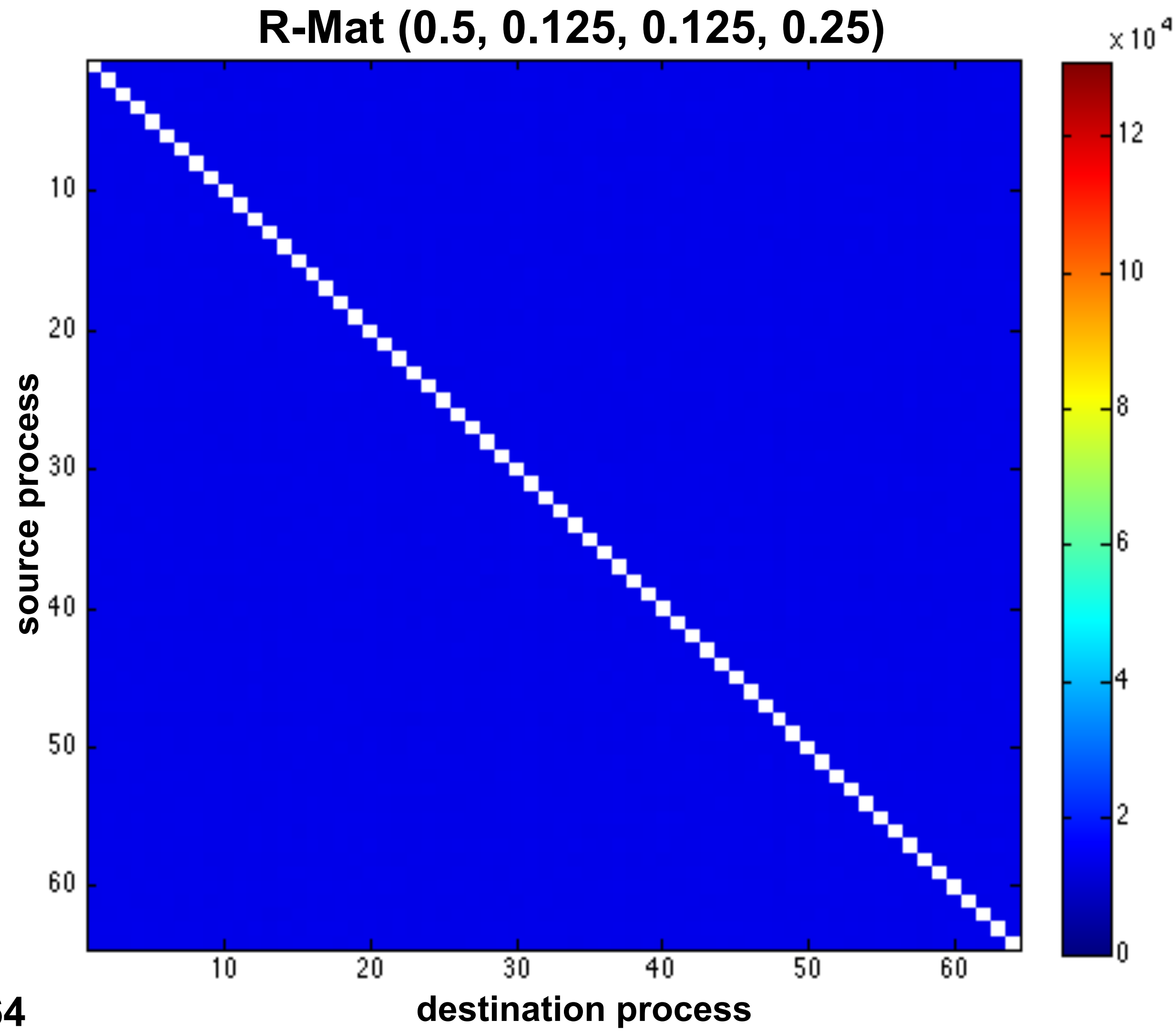
NNZ/process
min: 1.88E+05
max: 4.00E+06
avg: 1.06E+06
max/avg: 3.78

 # Messages (Phase 1)
total: 4032
max: 63

Volume (Phase 1)
total: 4.02E+07
max: 1.48E+06

Challenges:
Poor load balance
All-to-all communication

Communication Pattern: 1D Random Partitioning



P=64

Number of Rows: 2^{23}
Nonzeros/Row: 8

NNZ/process
min: 1.05E+06
max: 1.07E+06
avg: 1.06E+06
max/avg: 1.01

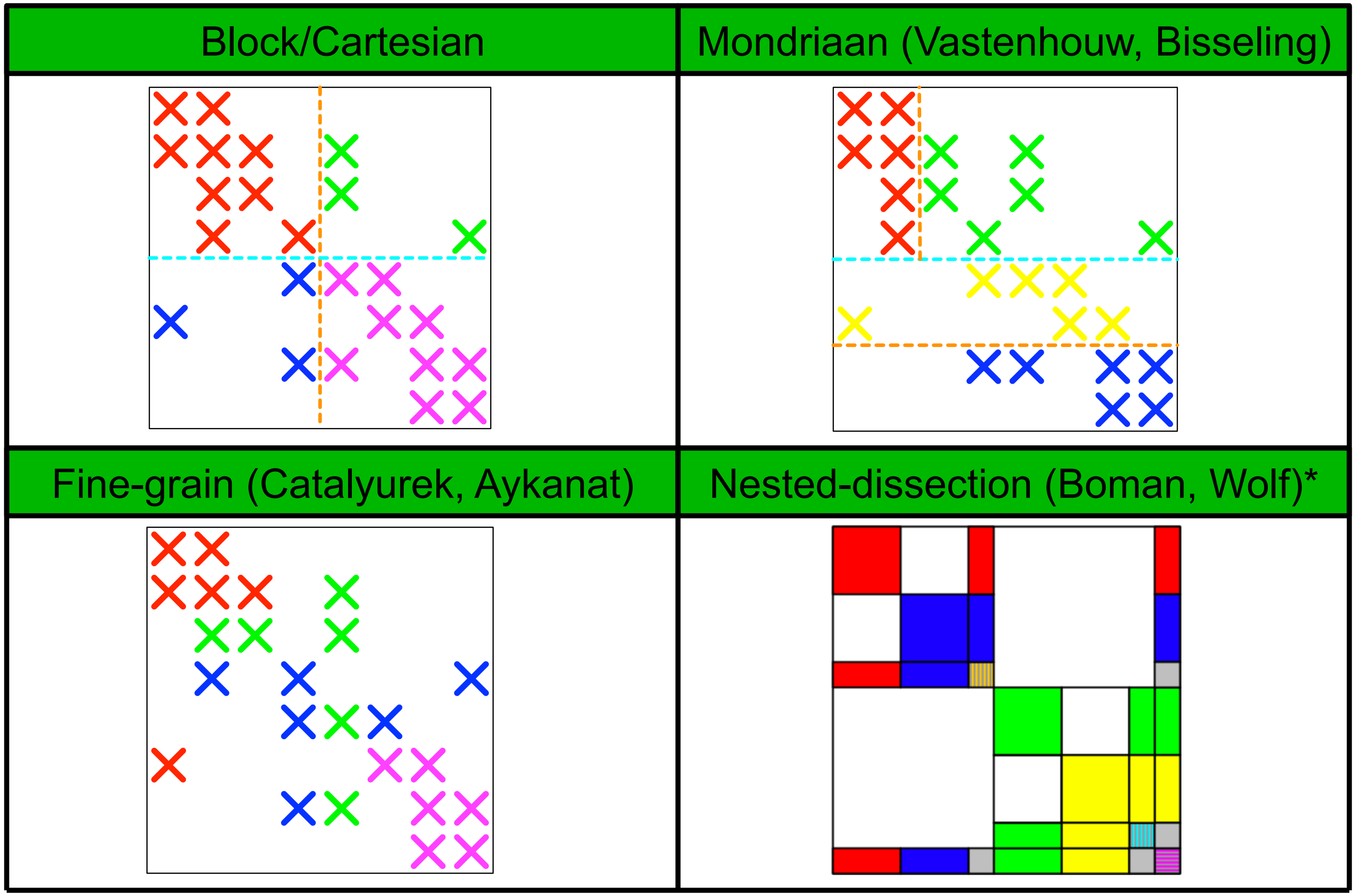
 # Messages (Phase 1)
total: 4032
max: 63

Volume (Phase 1)
total: 5.48E+07
max: 8.62E+05

Nice properties:
Great load balance

Challenges:
All-to-all communication

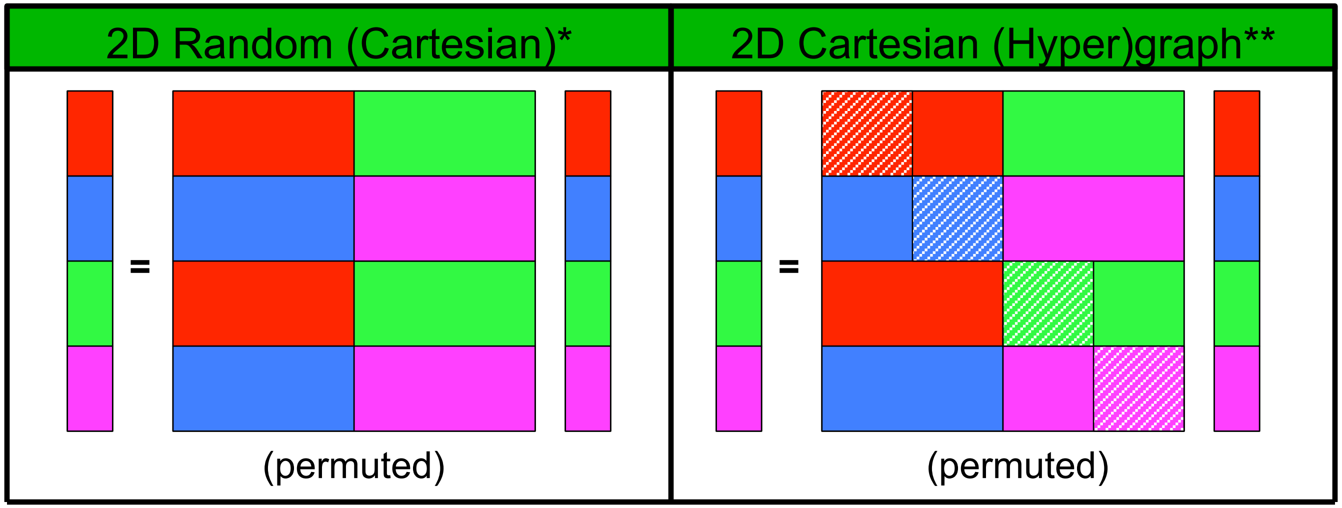
2D Partitioning



- More flexibility: no particular part for entire row or column
- More general sets of nonzeros assigned parts

*Boman and Wolf, "A Nested Dissection Partitioning Method for Parallel Sparse Matrix-Vector Multiplication," IEEE HPEC 2013.

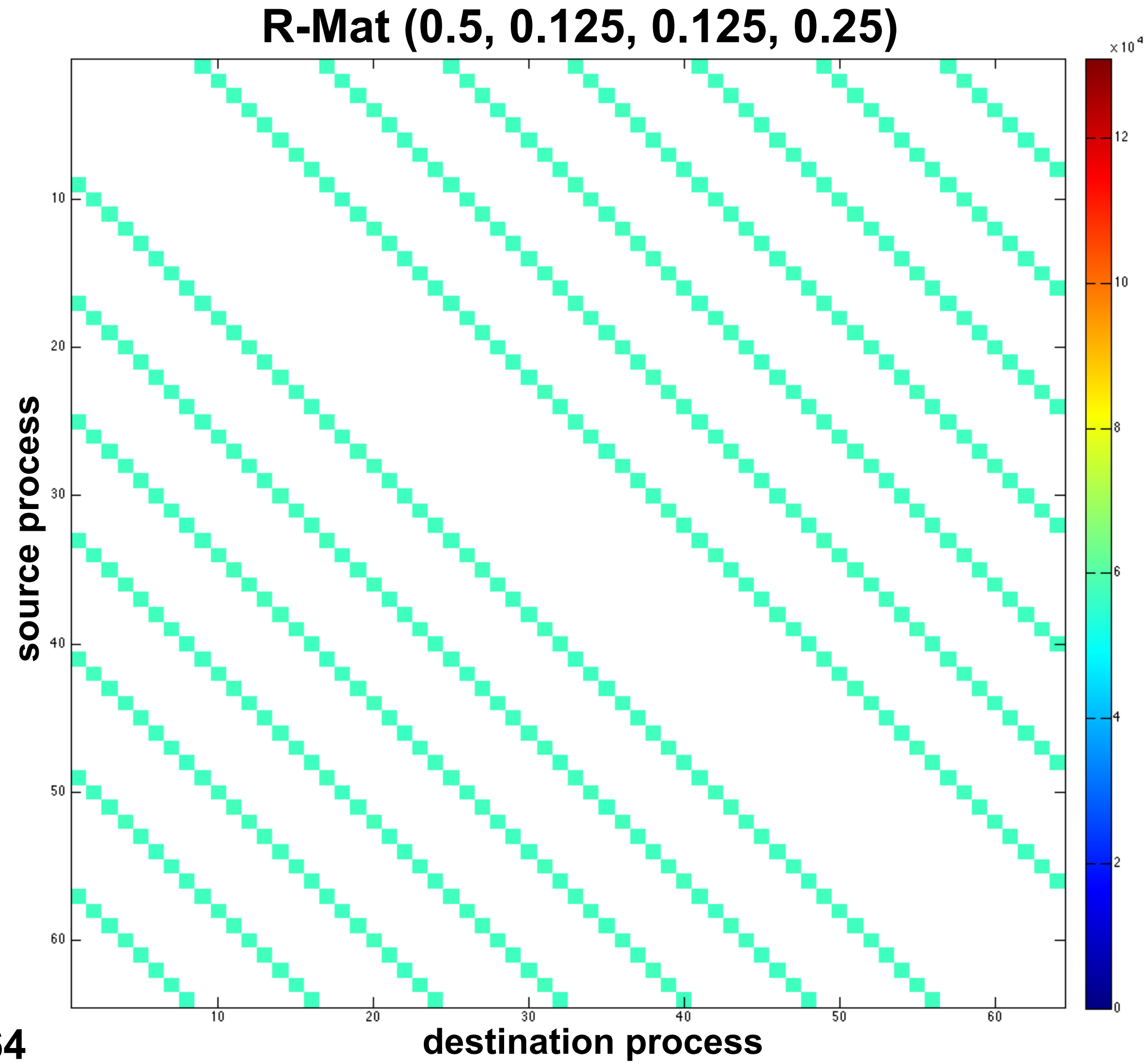
Bounding Number of Messages with 2D Partitioning



- Use flexibility of 2D partitioning to bound number of messages
  – Distribute nonzeros in permuted 2D Cartesian block manner
- 2D Random (Cartesian) – (Hendrickson, et al., Bisseling, Yoo)
  – Block Cartesian with rows/columns randomly distributed
  – Cyclic striping to minimize number of messages
- 2D Cartesian (Hyper)graph
  – Replace random partitioning with hyper(graph) partititioning to minimize communication volume

*Hendrickson, et al.; Bisseling; Yoo, et al.
**Boman, Devine, Rajamanickam, “Scalable Matrix Computations on Large Scale-Free Graphs Using 2D Partitioning, SC2013.

Communication Pattern: 2D Random Partitioning Cartesian Blocks (2DR)



P=64

Number of Rows: 2^{23}
Nonzeros/Row: 8

NNZ/process
min: 1.04E+06
max: 1.05E+06
avg: 1.05E+06
max/avg: 1.01

 # Messages (Phase 1)
total: 448
max: 7

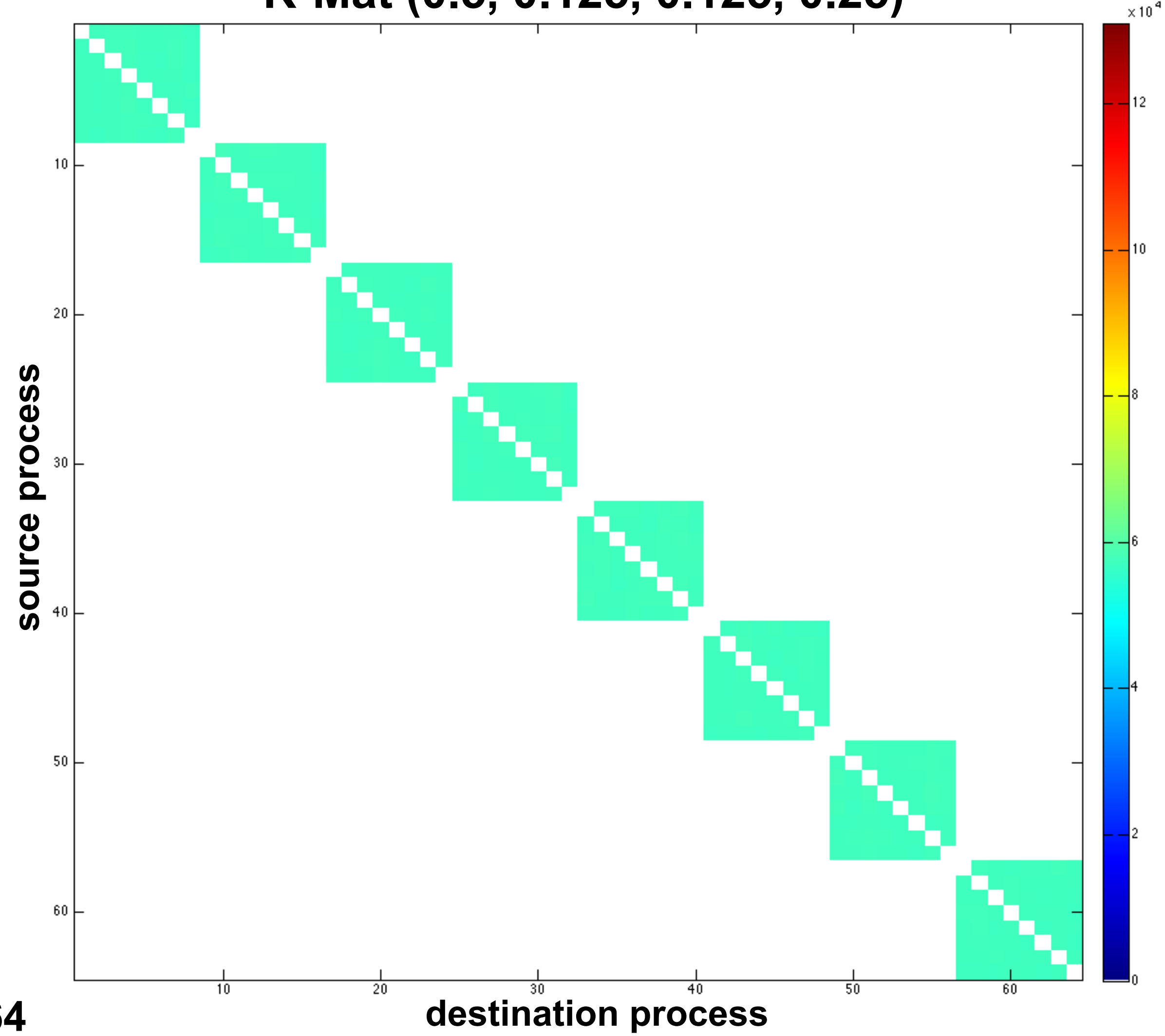
Volume (Phase 1)
total: 2.57E+07
max: 4.03E+05

Nice properties:
No all-to-all communication
Total volume lower than 1DR

1DR = 1D Random

Communication Pattern: 2D Random Partitioning Cartesian Blocks (2DR)

R-Mat (0.5, 0.125, 0.125, 0.25)



P=64

Number of Rows: 2^{23}
Nonzeros/Row: 8

NNZ/process
min: 1.04E+06
max: 1.05E+06
avg: 1.05E+06
max/avg: 1.01

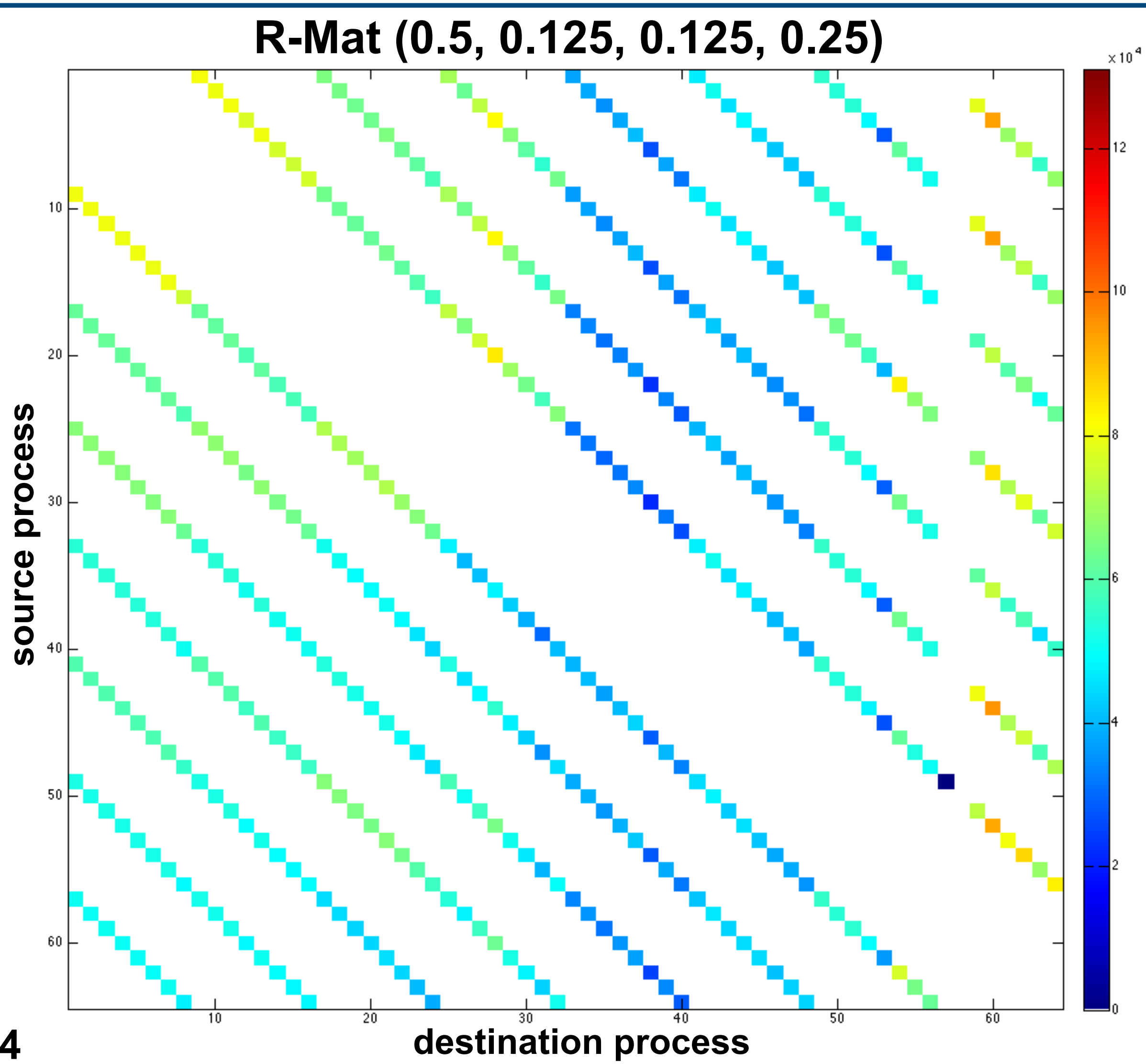
 # Messages (Phase 2)
total: 448
max: 7

Volume (Phase 2)
total: 2.57E+07
max: 4.03E+05

Nice properties:
No all-to-all communication
Total volume lower than 1DR

1DR = 1D Random

Communication Pattern: 2D Cartesian Hypergraph Partitioning



P=64

Number of Rows: 2^{23}
Nonzeros/Row: 8

NNZ/process
min: 5.88E+05
max: 1.29E+06
avg: 1.05E+06
max/avg: 1.23

 # Messages (Phase 1)
total: 448
max: 7

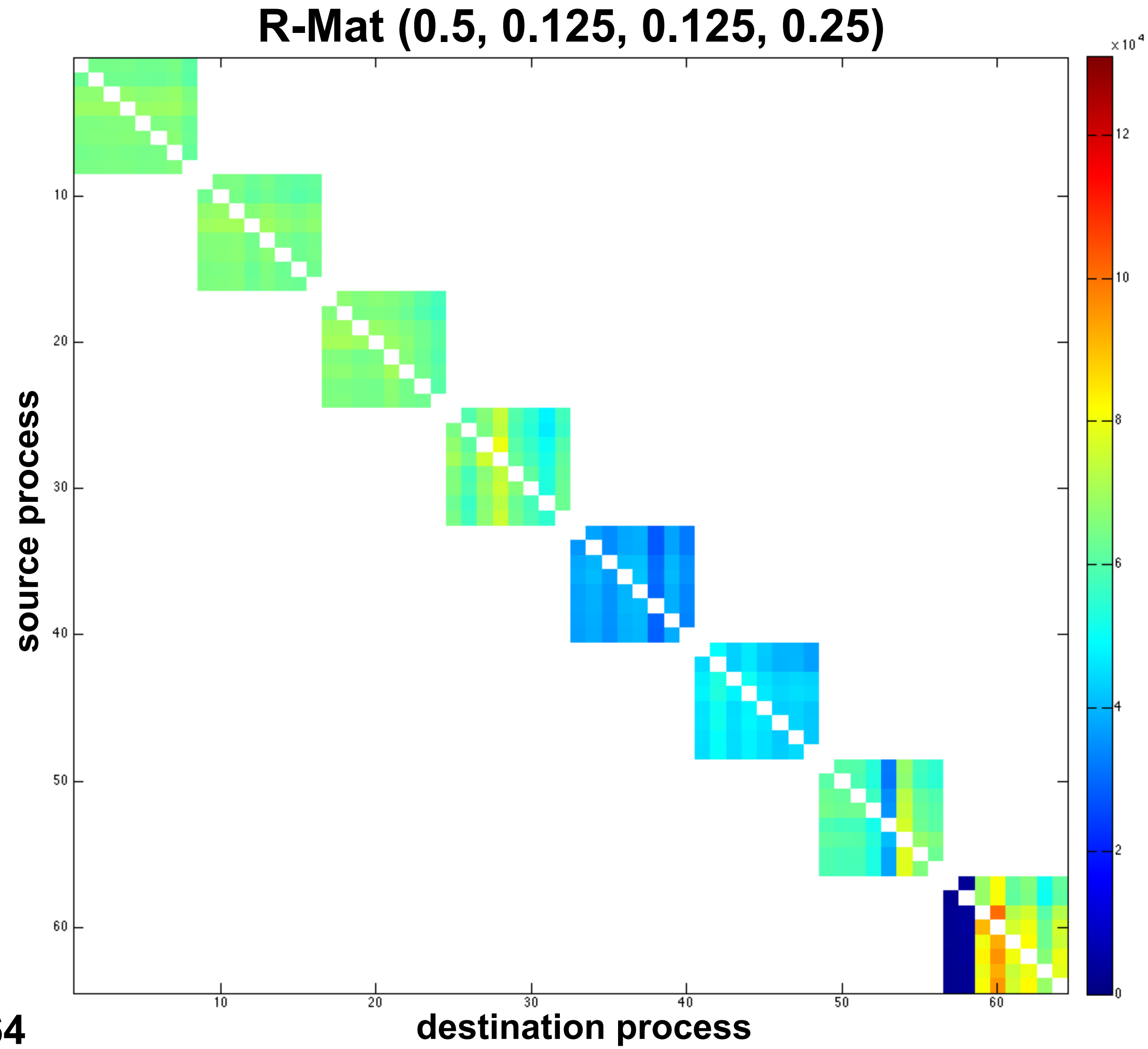
Volume (Phase 1)
total: 2.33E+07
max: 4.52E+05

Nice properties:
No all-to-all communication
Total volume lower than 2DR

Challenges:
Imbalance worse than 2DR

2DR = 2D Random Cartesian

Communication Pattern: 2D Cartesian Hypergraph Partitioning



P=64

Number of Rows: 2^{23}
Nonzeros/Row: 8

NNZ/process
min: 5.88E+05
max: 1.29E+06
avg: 1.05E+06
max/avg: 1.23

 # Messages (Phase 2)
total: 448
max: 7

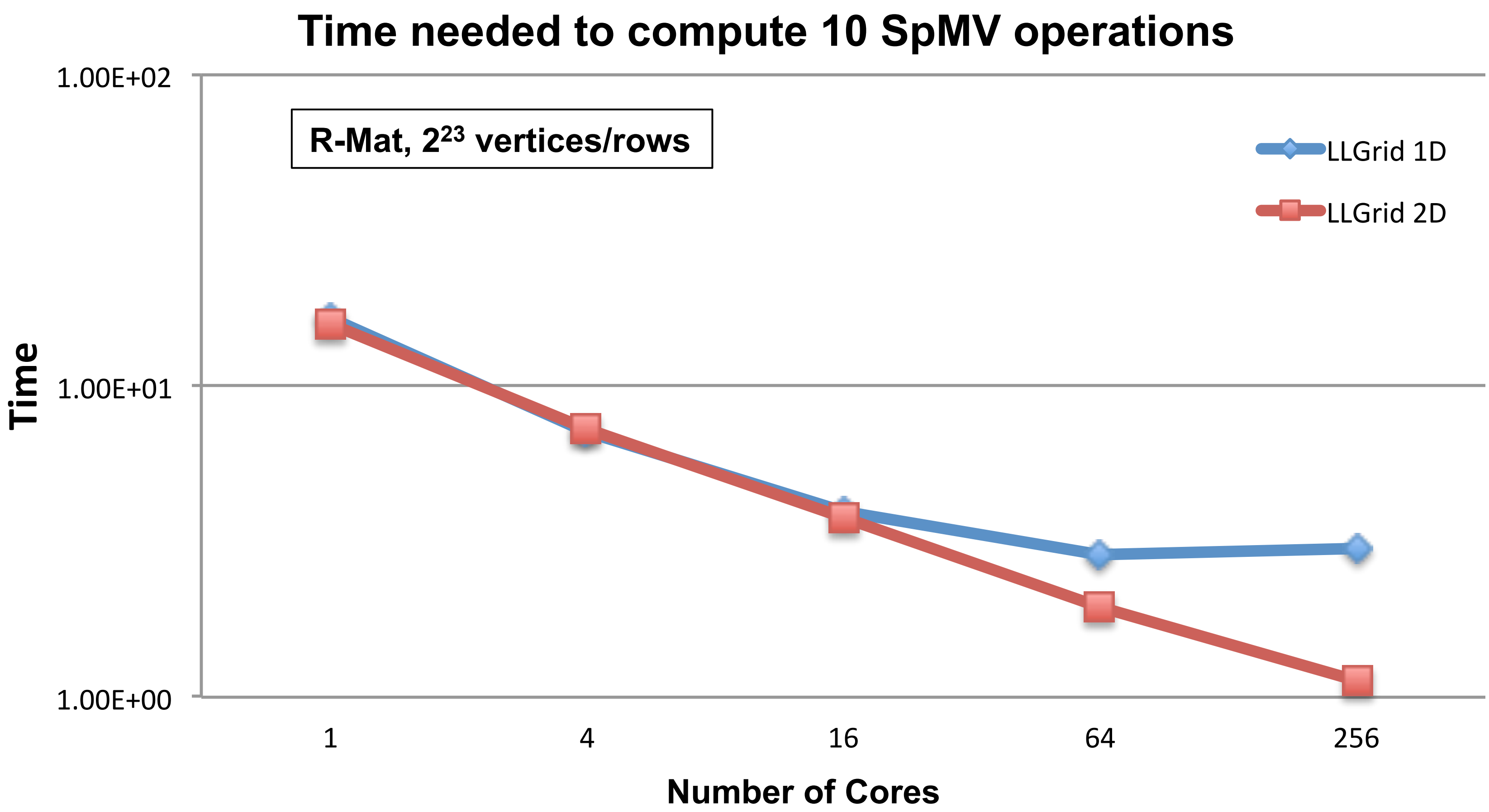
Volume (Phase 2)
total: 2.54E+07
max: 4.80E+05

Nice properties:
No all-to-all communication
Total volume lower than 2DR

Challenges:
Imbalance worse than 2DR

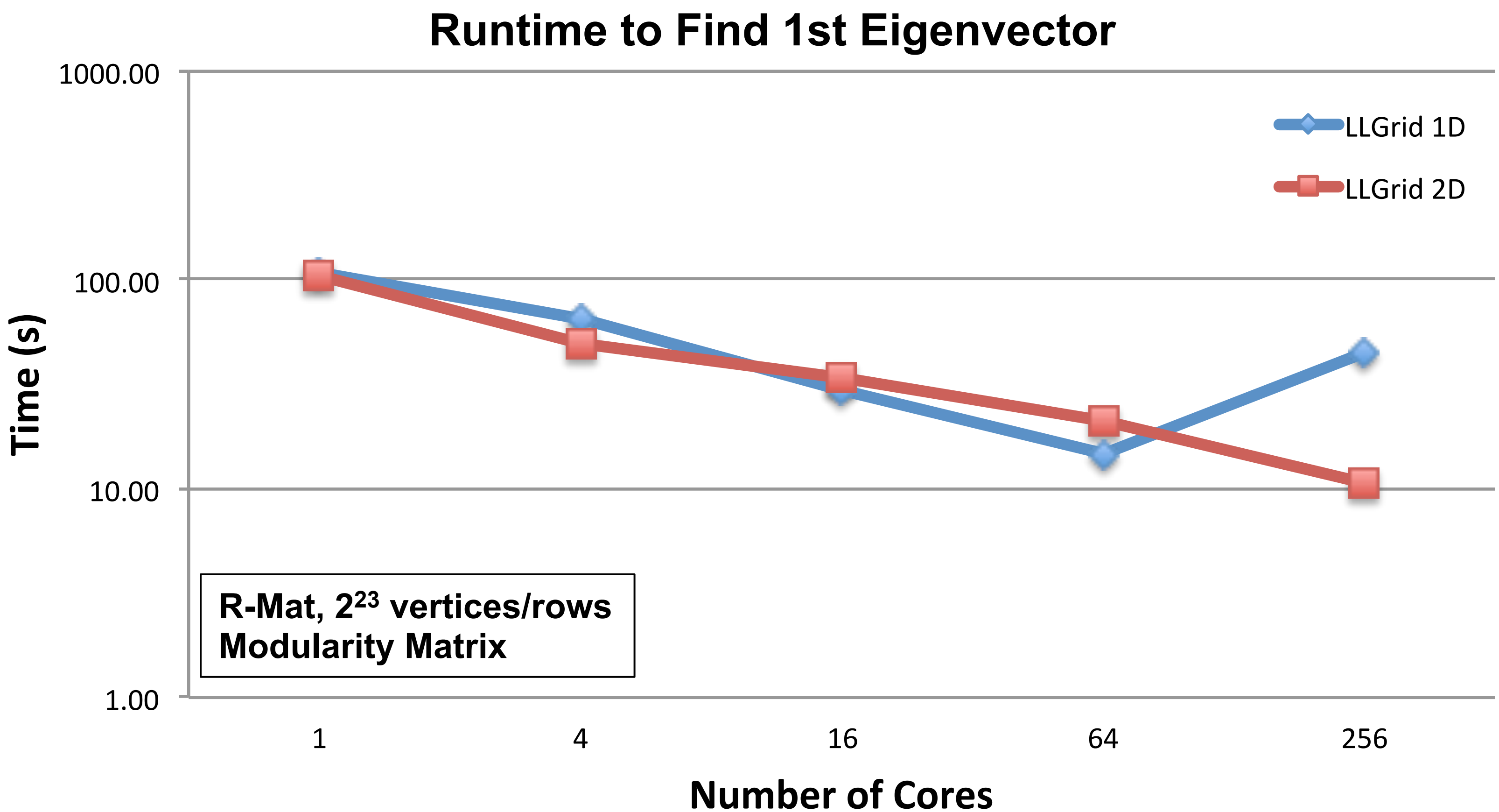
2DR = 2D Random Cartesian

Improved Results: SpMV – LLGrid



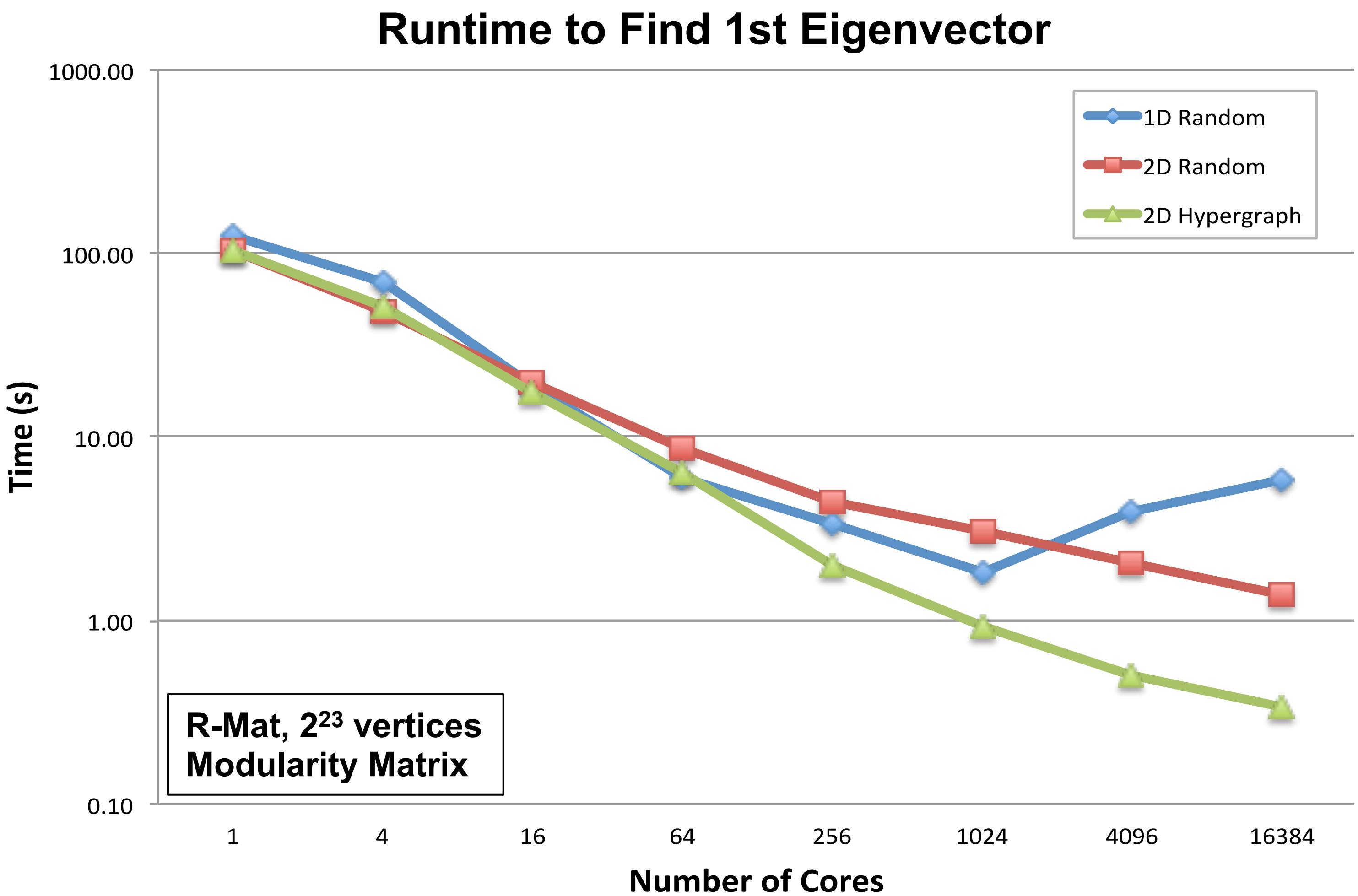
Simple 2D method shows improved scalability

Improved Results – LLGrid



Simple 2D method shows improved scalability

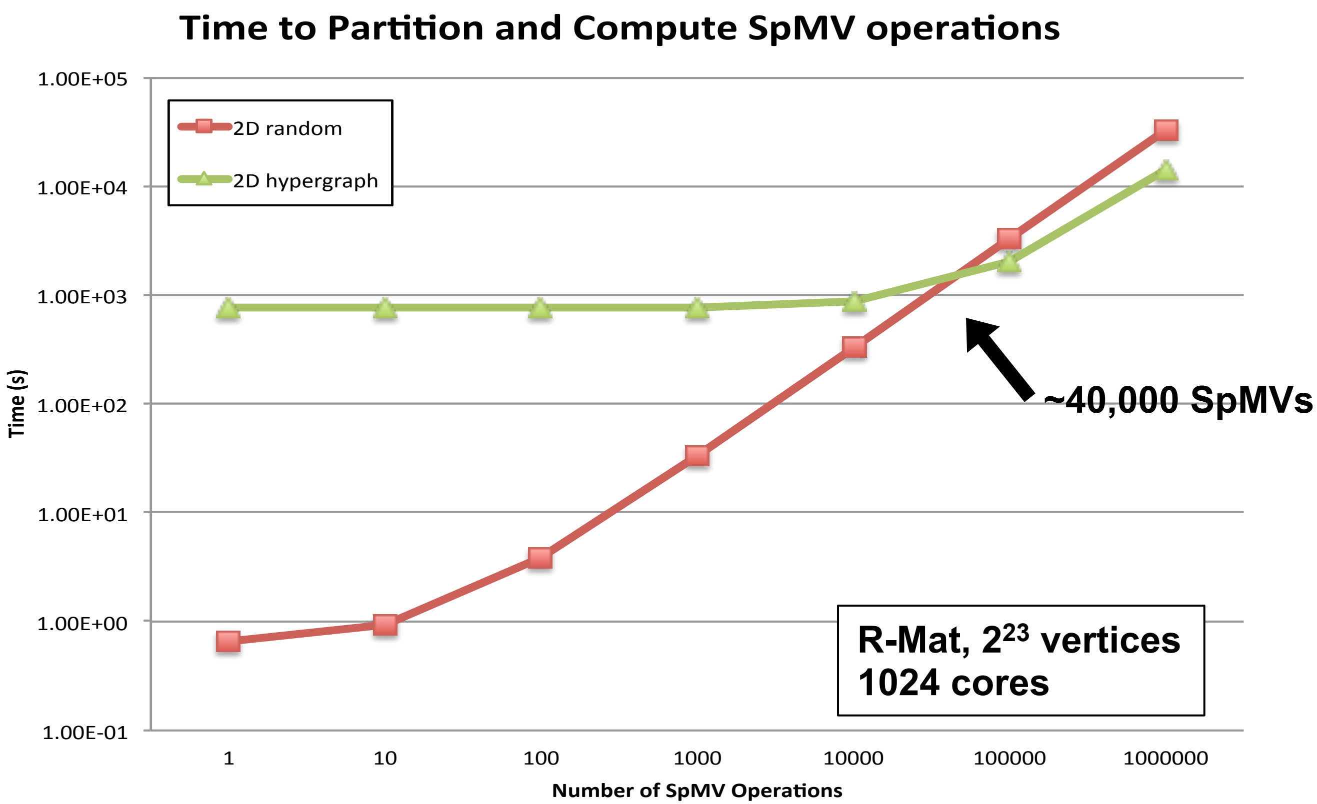
Improved Results – NERSC Hopper*



2D methods show improved scalability

* This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Challenge with Hypergraph/Graph Partitioning



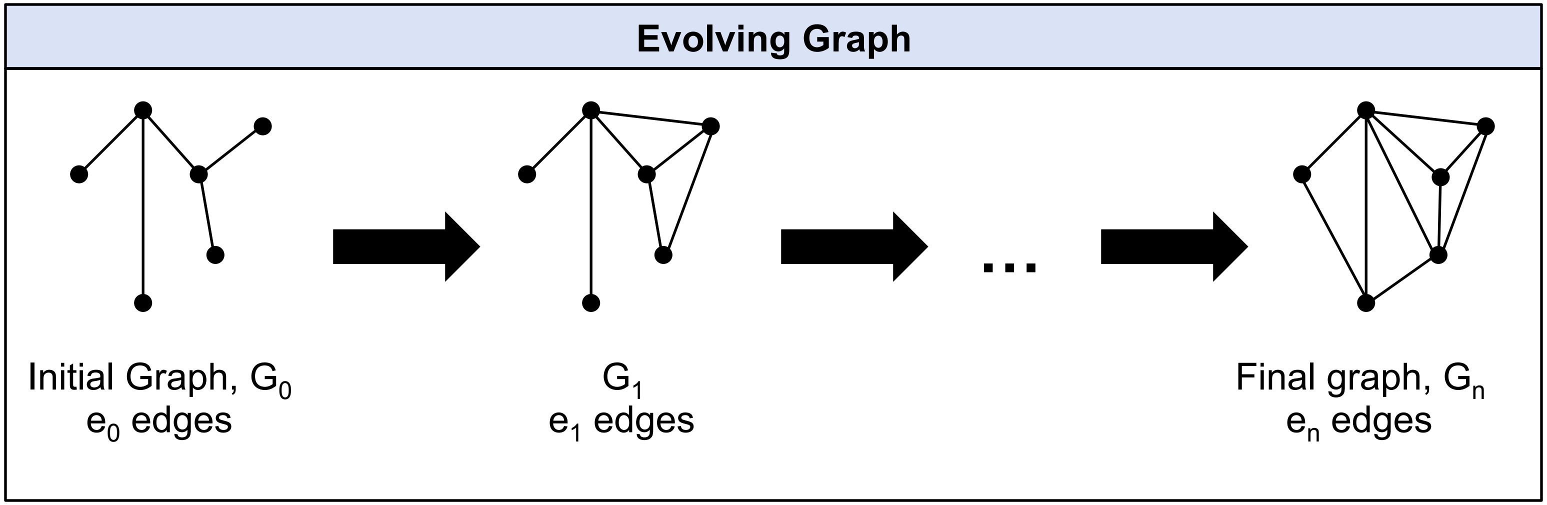
NERSC Hopper*

- High partitioning cost of graph/hypergraph methods must be amortized by computing many SpMV operations
- Detection** requires at most 1000s of SpMV operations
- Expensive partitions need to be effective for multiple graphs

**L1 norm method: computing 100 eigenvectors

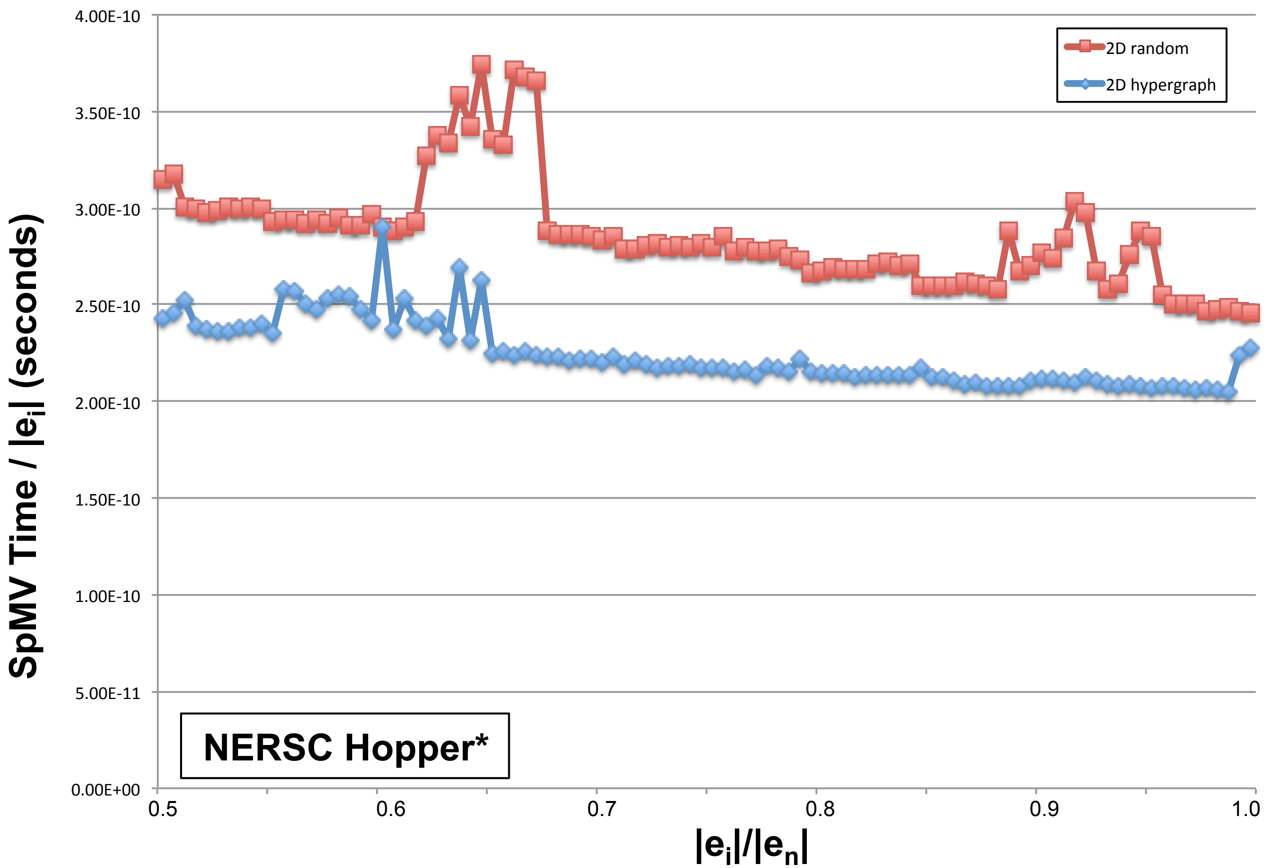
* This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Experiment Partitioning for Dynamic Graphs



- Key question: How long will a partition be effective?
- Initial experiment
  - Evolving R-Mat matrices: fixed number of rows, R-Mat parameters (a,b,c,d)
  - Start with a given number of nonzeros ($|e_0|$)
  - Iteratively add nonzeros until new number of nonzeros is reached ($|e_n|$)

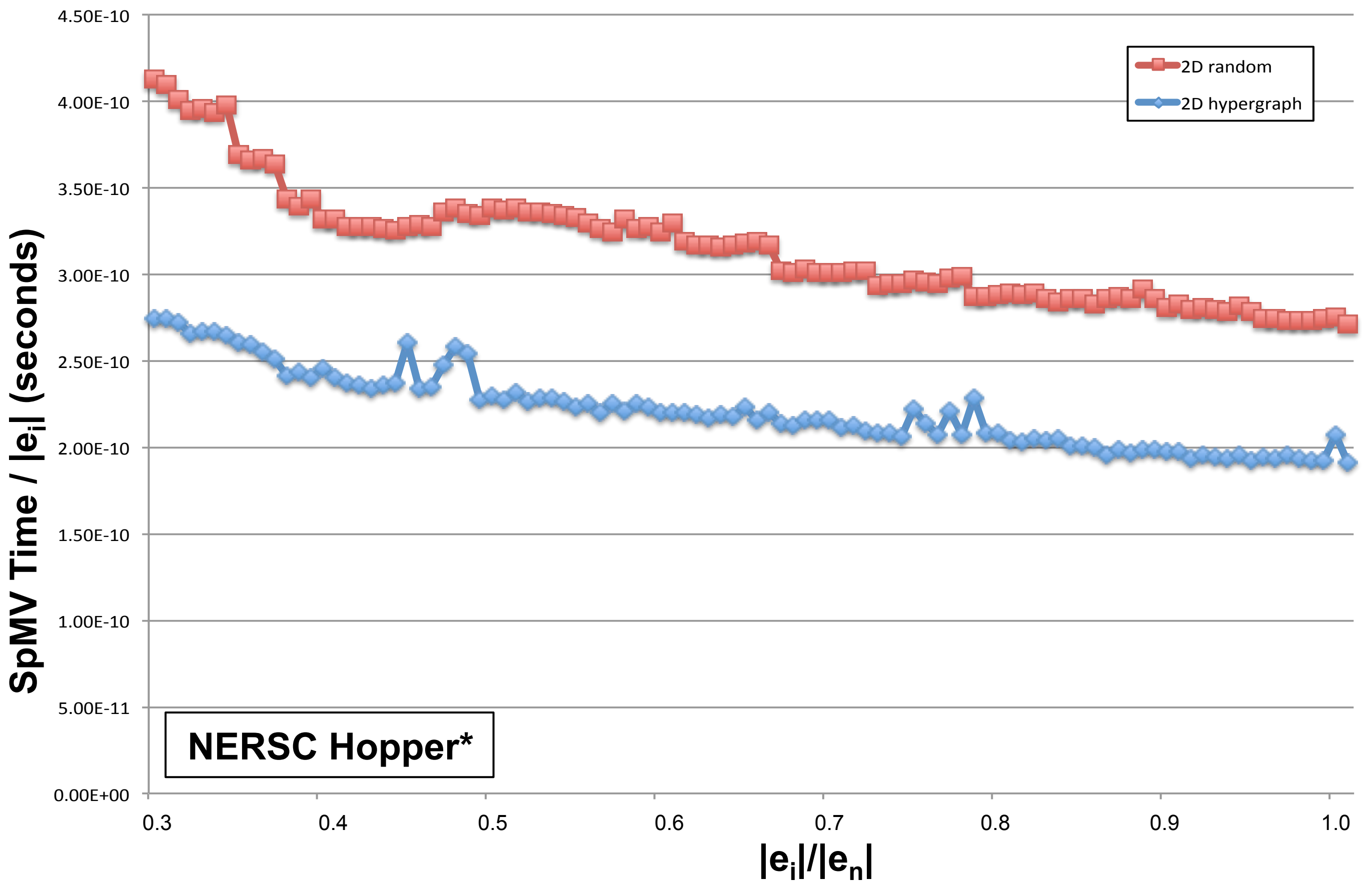
Results: Partitioning for Dynamic Graphs



- $|e_0| = 0.5\ |e_n|$
- **2D hypergraph surprisingly effective as edges are added to graph**

* This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Results: Partitioning for Dynamic Graphs



- $|e_0| = 0.3\ |e_n|$
- **2D hypergraph surprisingly effective as edges are added to graph**

* This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Outline

- Big Data and High Performance Computing
- Anomaly Detection in Graphs
- Signal Processing for Graphs (SPG)
- Improving Sparse Matrix-Vector Multiplication (SpMV) Performance
- Improving Performance of Moving Average Filter
- Summary

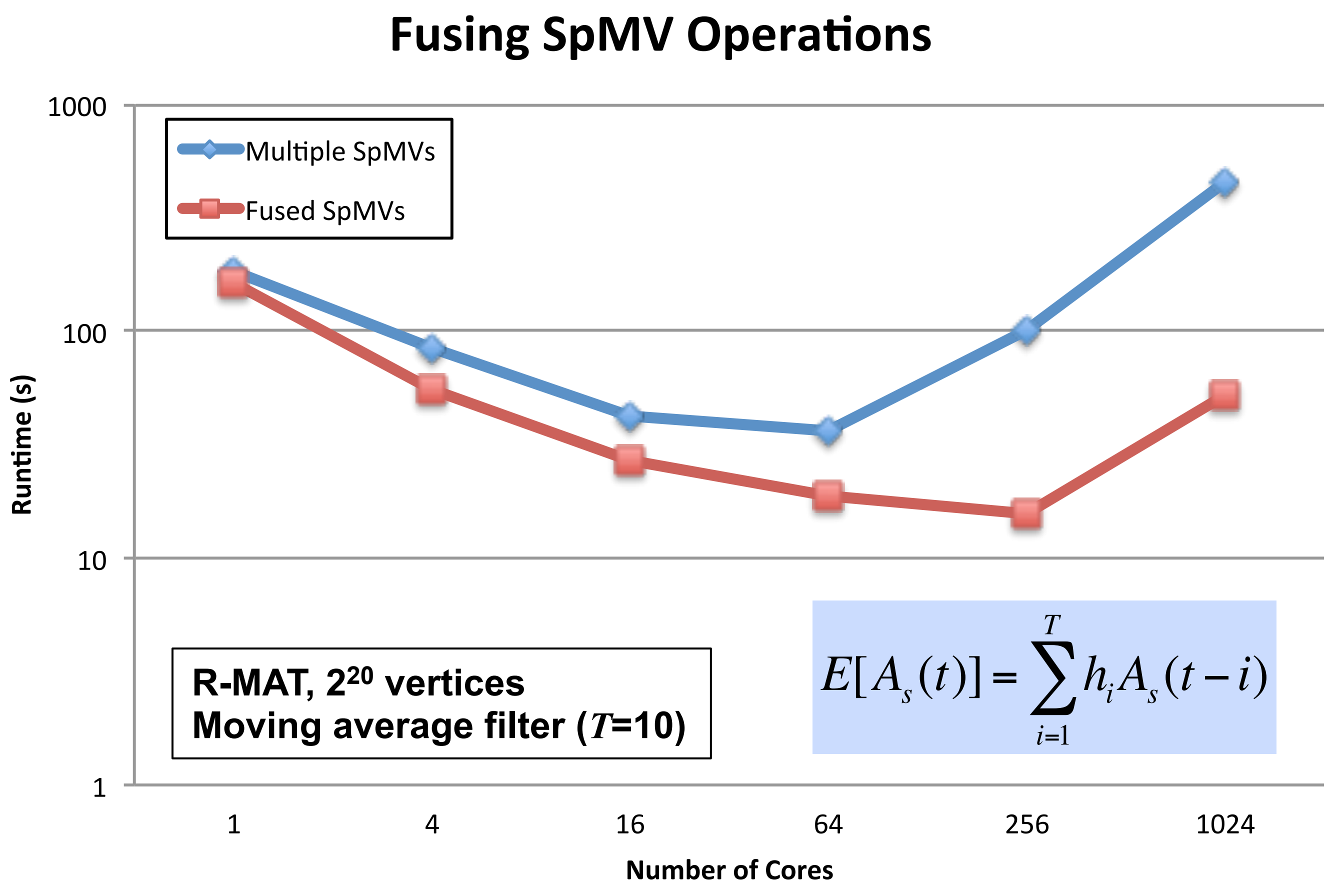
Moving Average Filter

Moving Average Filter
$$E[A_s(t)] = \sum_{i=1}^{T} h_i A_s(t-i)$$

$$B = (A - E[A])$$

- Option 1: explicitly form expected graph model matrix each time step
  - Pro: Less computation (when nonzeros collide) than T SpMV ops
  - Pro: Less communication than T *SpMV ops*
  - Con: Very expensive (have to add and subtract matrices to form)
- Option 2: don't explicitly form graph model matrix
  - Pro: Avoid expensive matrix formation
  - Con: Requires T SpMV ops (more communication, possibly more computation)
- Idea to improve option 2: fuse multiple SpMV operations
  - Perform communication once

Fused SpMV Operations



Fusing SpMV operations can effectively reduce runtime

Outline

- Big Data and High Performance Computing
- Anomaly Detection in Graphs
- Signal Processing for Graphs (SPG)
- Improving Sparse Matrix-Vector Multiplication (SpMV) Performance
- Improving Performance of Moving Average Filter
- Summary

Summary

- Outlined HPC approach to processing big data
  – Signal processing for graphs
  – Statistical framework for anomaly detection in graphs
- Key component is eigensolver for dimensionality reduction
- Solving eigensystems resulting from power law graphs challenging
  – Load imbalance
  – Poor data locality
- SpMV key computational kernel
  – 1D data partitioning limits performance due to all-to-all communication
  – 2D data partitioning can be used to improve scalability
- Dynamic graphs pose new computational challenges
  – New computational kernels may be necessary (e.g., fused sparse matrix-dense vector operations)

Acknowledgements

- Nicholas Arcolano (MITLL)
- Michelle Beard (MITLL)
- Nadya Bliss (ASU)
- Jeremy Kepner (MITLL)
- Dan Kimball (MITLL)
- Lisie Michel (MITLL)
- Sanjeev Mohindra (MITLL)
- Eddie Rutledge (MITLL)
- Scott Sawyer (MITLL)
- Matt Schmidt (MITLL)