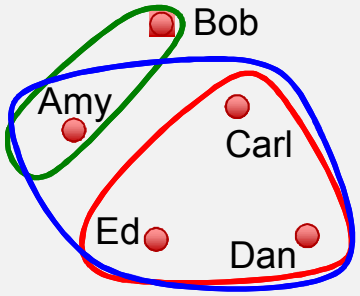
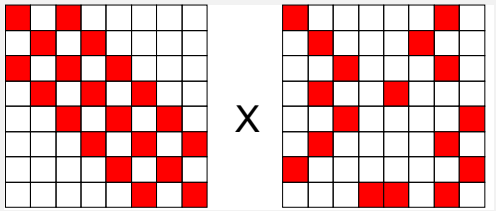
SAND2018-1750C





TriData: High Performance Linear Algebra-Based Data Analytics

Michael Wolf, Danny Dunlavy, Rich Lehoucq, Jon Berry, Daniel Bourgeois

CCR
Center for Computing Research

SIAM PP18

2018-03-08

U.S. DEPARTMENT OF ENERGY
NNSA National Nuclear Security Administration

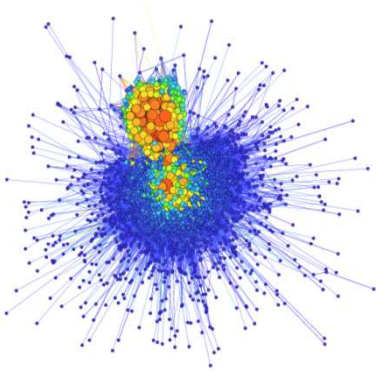
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

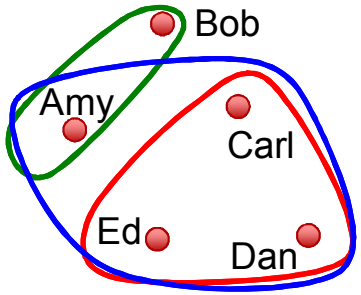
TriData Overview

Sandia National Laboratories

- TriData – Trilinos for Large-Scale Data Analysis
  - Leverages Trilinos Framework (Sandia National Labs)
    - High performance linear algebra, traditional focus on CSE
    - High performing eigensolvers, linear solvers
    - Scales to billions of matrix rows/vertices
- Vision: Sparse Linear Algebra-Based Data Analysis
  - Apply sparse linear algebra techniques to data analysis
  - Target: very large data problems
  - Target: distributed memory and single node HPC architectures
- Additionally
  - Vehicle for improving how Trilinos can be leveraged for data analytics (e.g., submatrix extraction, preconditioning, load-balancing)
  - Support GraphBLAS-like linear algebra analysis techniques
- Focus: Graph and Hypergraph Analysis

TriData Capabilities



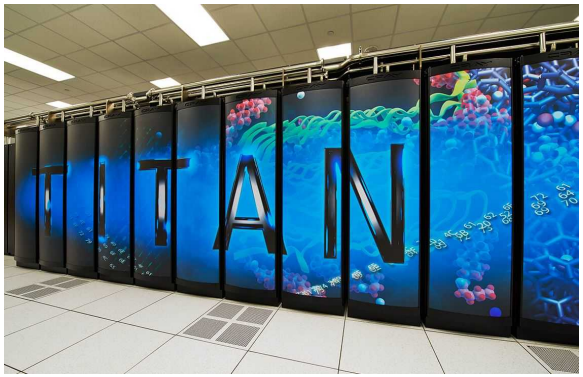


- Eigen solver based capabilities
  - Spectral Clustering, Vertex/Edge eigencentrality (graphs, hypergraphs)
  - Supports several eigensolvers (through Trilinos): LOBPCG, TraceMin-Davidson, Riemannian Trust Region, Block Krylov-Schur
- Linear solver based capability
  - Mean hitting time analysis on graphs
  - Support for different linear solvers (typically use CG) and preconditioners
- Other
  - K-means++, metrics (conductance, modularity, jaccard index)
  - Random graph and hypergraph models, hypothesis testing techniques/infrastructure for evaluation of clustering software

TriData Approach

Sandia National Laboratories

TriData



Distributed Memory (DM)

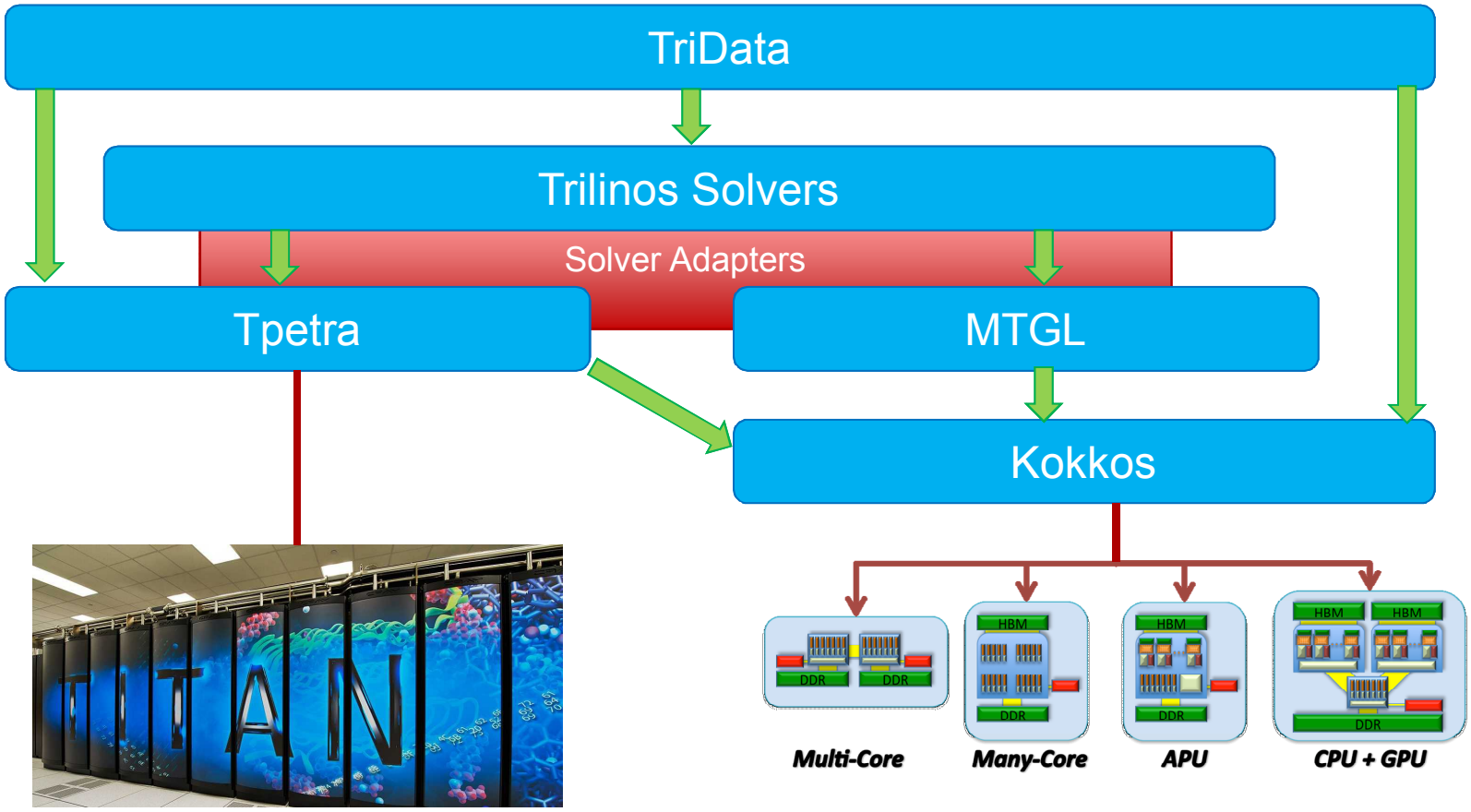
- Clusters, supercomputers
- Tpetra (MPI, DM data structures)
- Kokkos (node level parallelism)

Workstation

- CPUs, **GPUs**, KNLs, …
- Kokkos
- MTGL

Goal: Write algorithms once, run on both types of architectures

TriData Software Stack



Distributed memory computations

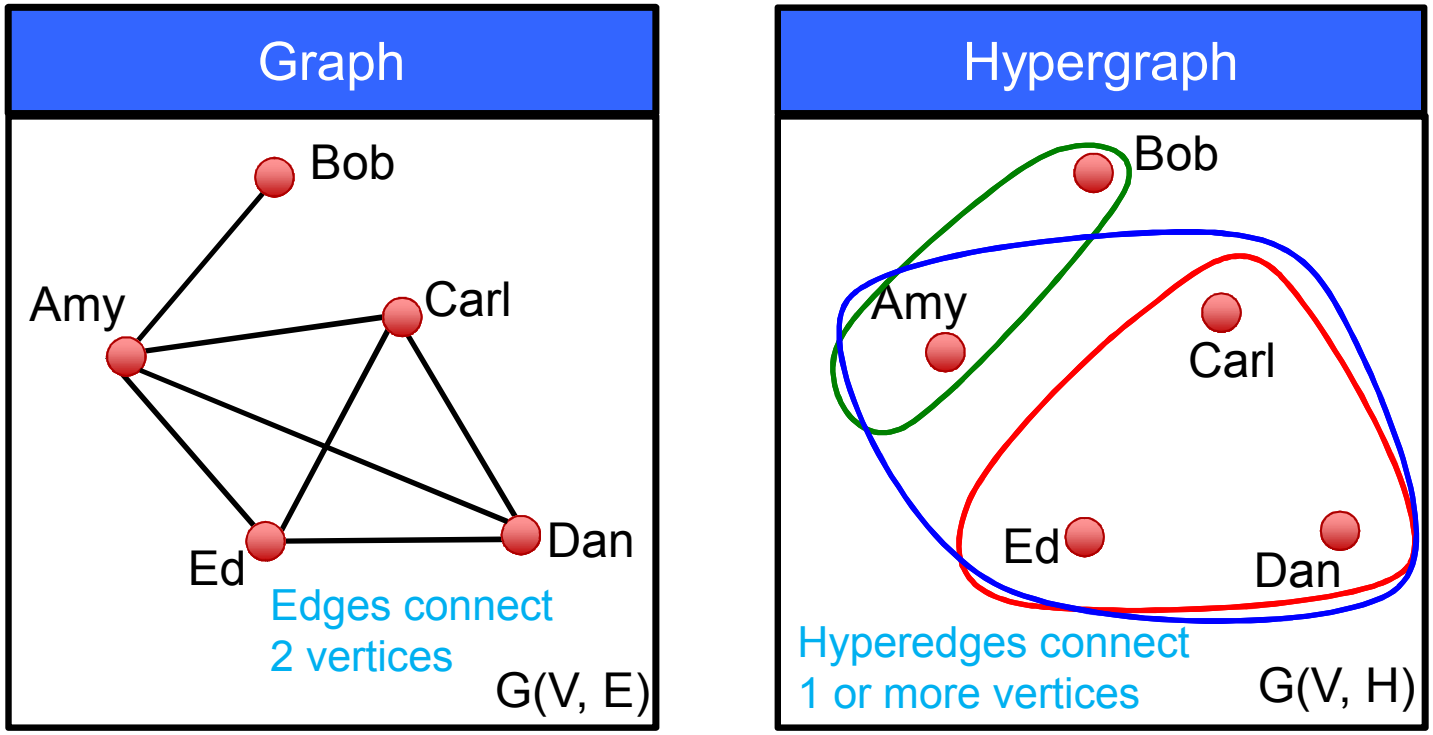
Portable on-node performance

Flexible solver adapters enable solution for both architectures

Outline

- TriData Overview
- Focus 1: Hypergraphs and Incidence Matrices
- Focus 2: 2D Partitioning
- Focus 3: Interoperability
- Future TriData Focus: KokkosKernels
- Summary

Hypergraphs



- Generalization of graph
  - Hyperedges represent multiway relationships between vertices
  - Hyperedge – set of 1 or more vertices
  - Key feature: hyperedges can connect more than 2 vertices

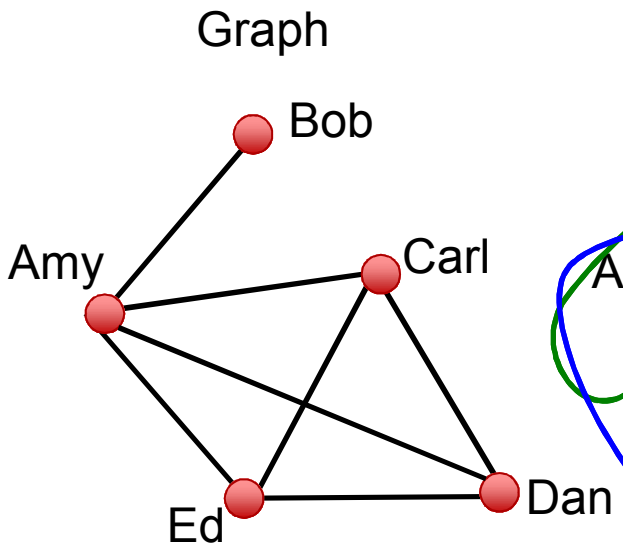
Why Hypergraphs?

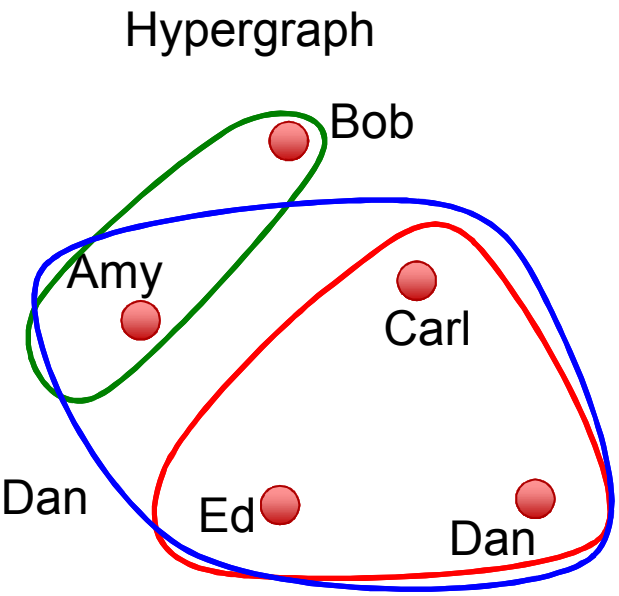
Sandia National Laboratories

Emails

Users	1	2	3
Amy	x		x
Bob	x		
Carl		x	x
Dan		x	x
Ed		x	x

Relational Data



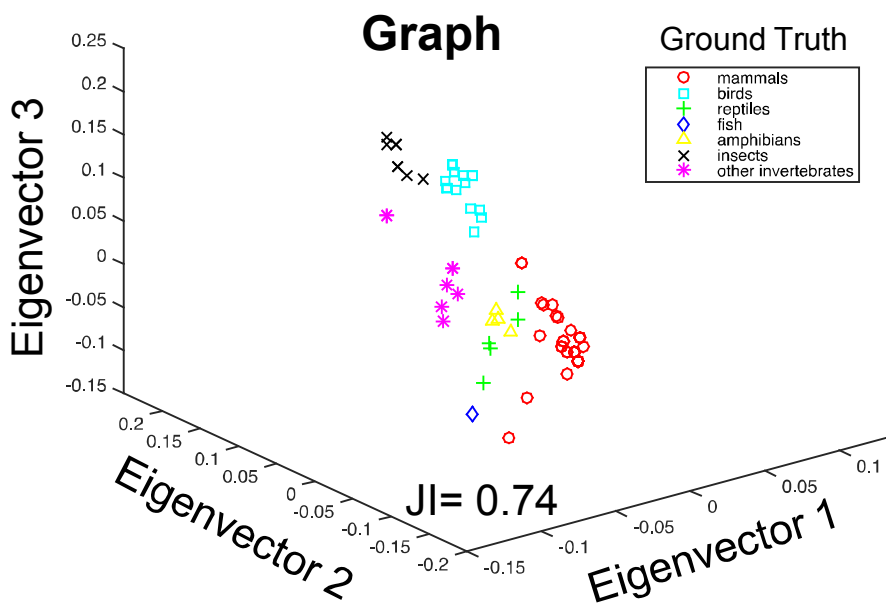


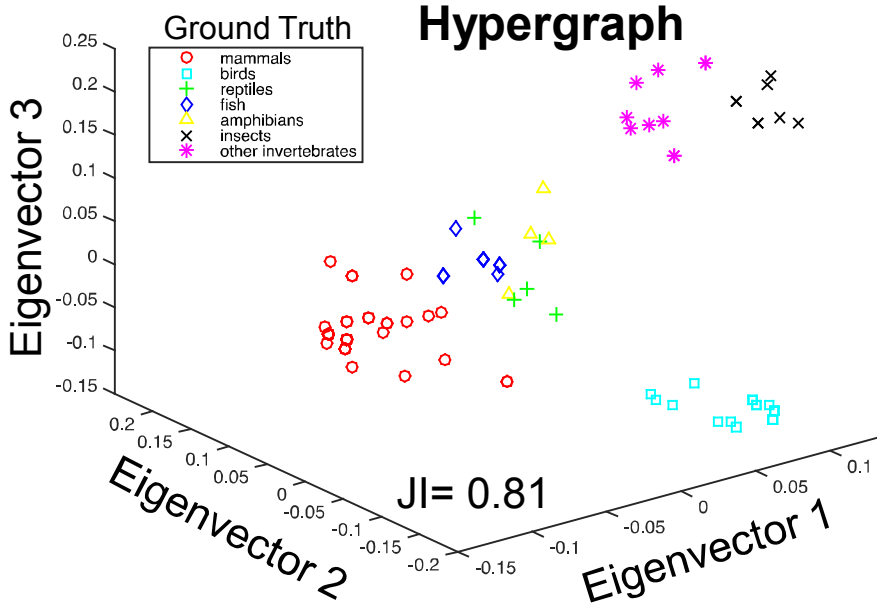
- Convenient representation of relational data
  - E.g., Each email represented by hyperedge (a subset of users)
- Multiway relationships can be represented nonambiguously
- Computation and storage advantages

Why Hypergraphs?

Sandia National Laboratories

Spectral Clustering (Normalized Laplacians)





- Improved Modeling
  - Clustering (above UCI ML Repository Zoo Data Set*)
    - Clustering vs. ground truth (7 clusters): Graph JI=0.74, Hypergraph JI= 0.81
    - Clustering vs. ground truth (3 clusters): Graph JI=0.87, Hypergraph JI= 1.00
  - Eigencentrality

* https://archive.ics.uci.edu/ml/

JI=Jaccard Index

Incidence Matrices

1		1
1		
	1	1
	1	1
	1	1

Hypergraph incidence matrix

1				1	1	1			
1									
	1	1		1			1	1	
	1		1		1		1		1
		1	1			1		1	1

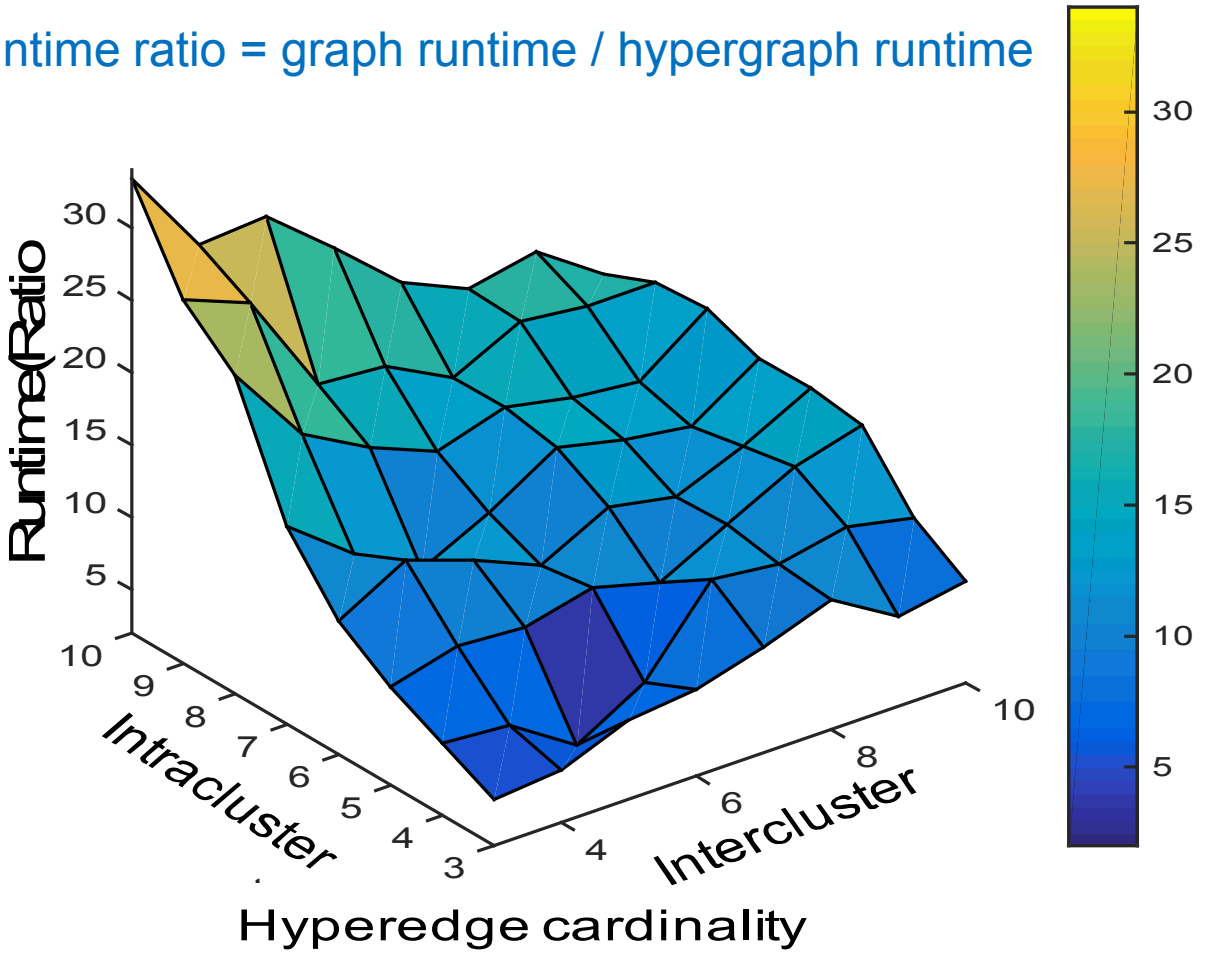
Graph Incidence matrix

- Compute with **hypergraph incidence matrices** when possible
  - Relational data is often stored as hypergraph incidence matrix*
  - Avoids costly SpGEMM operation for building adjacency matrices
  - Dynamic data: easier to update incidence matrices than adjacency matrices
  - Trilinos solver operators make this easy
- Hypergraphs require significantly **less storage space** and **fewer operations** than graphs generated using clique expansion

*Kepner, Jeremy, et al. "Dynamic distributed dimensional data model (D4M) database and computation system." *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012.

Runtimes – Graph vs. Hypergraph

Runtime ratio = graph runtime / hypergraph runtime

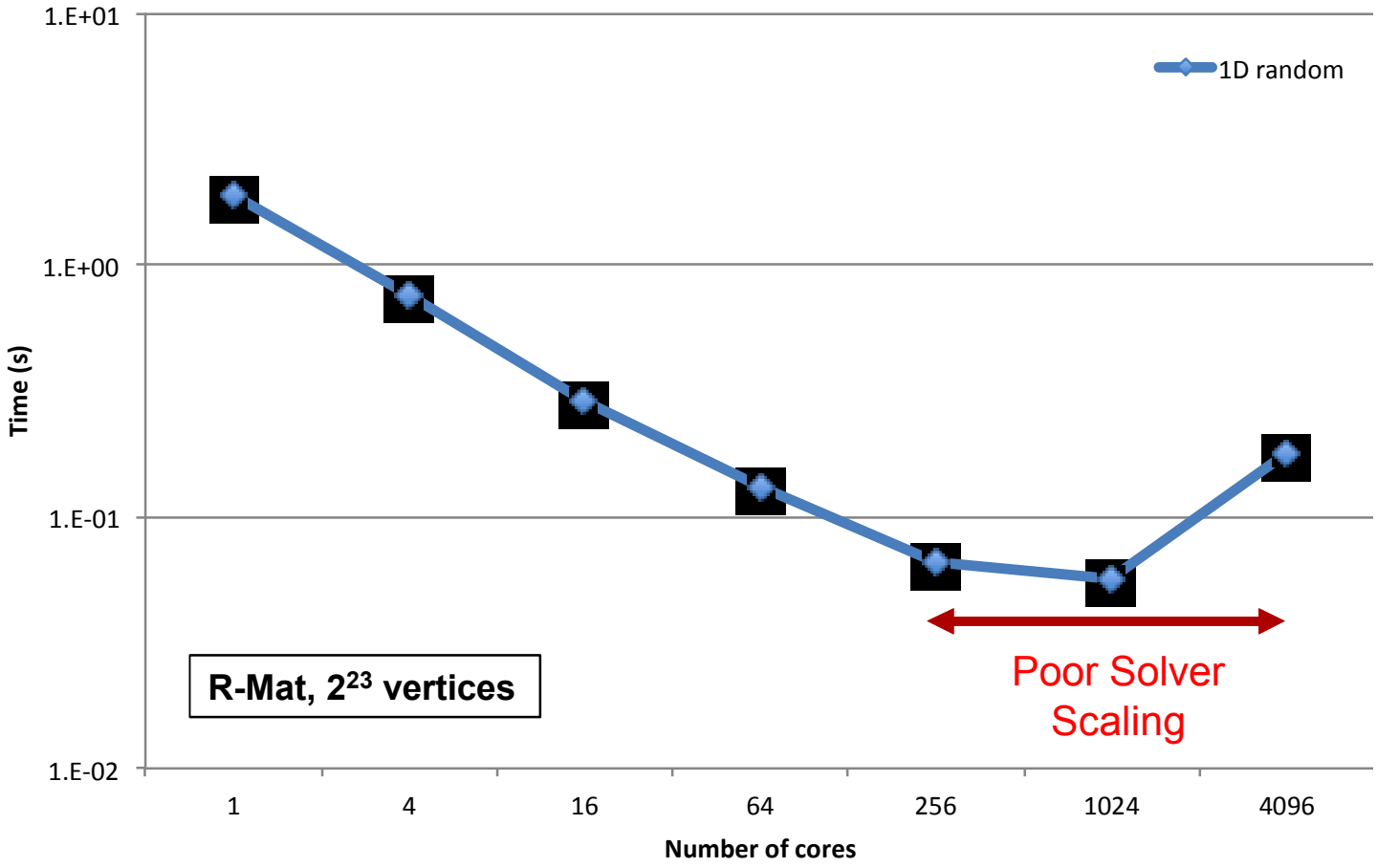


Hypergraph models (with incidence matrices) up to 30x faster than graph models

Outline

- TriData Overview
- Focus 1: Hypergraphs and Incidence Matrices
- Focus 2: 2D Partitioning
- Focus 3: Interoperability
- Future TriData Focus: KokkosKernels
- Summary

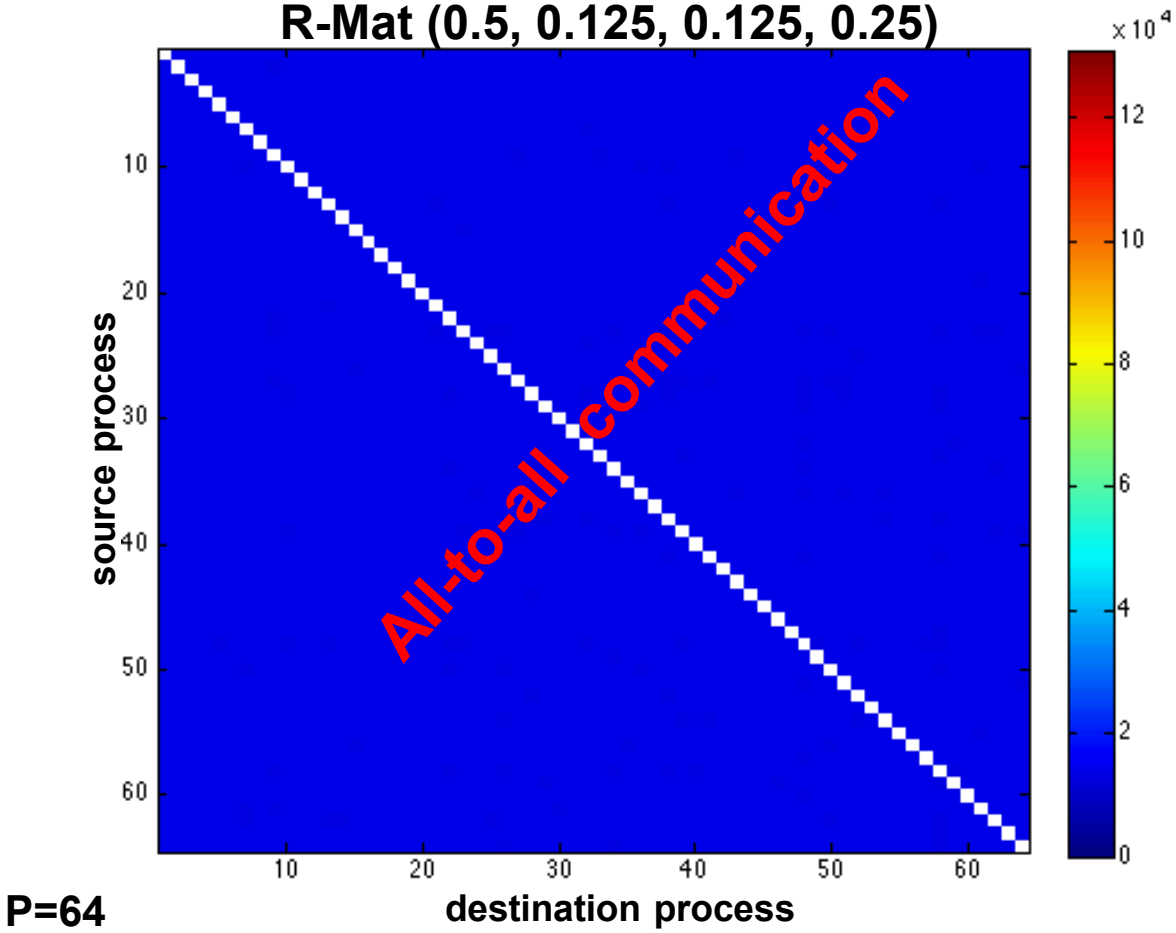
2D Partitioning: Motivation



Solver scalability severely limited when using one-dimensional distributions (data partitioned by row or vertex) for social network data

M. Wolf and B. Miller, "Sparse Matrix Partitioning for Parallel Eigenanalysis of Large Static and Dynamic Graphs," in *IEEE HPEC*, 2014.

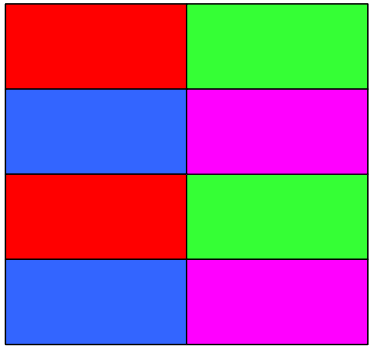
1D Partitioning: Communication Pattern



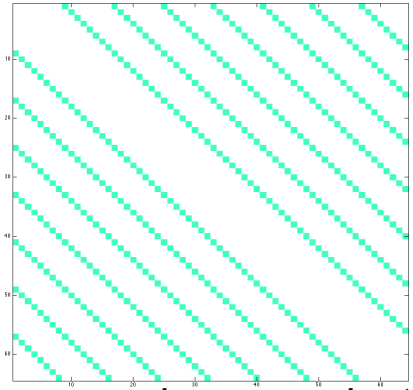
Latency due to all-to-all communication kills parallel performance for large numbers of processors

M. Wolf and B. Miller, “Sparse Matrix Partitioning for Parallel Eigenanalysis of Large Static and Dynamic Graphs," in *IEEE HPEC*, 2014.

2D Partitioning



2D Block Cartesian



Improved communication

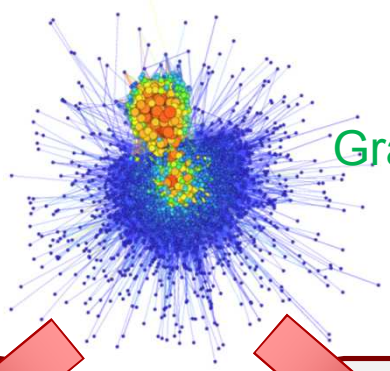
- 2D partitioning essential for high performance linear algebra-based data analytics
  - More flexibility: no particular part for entire row, general sets of nonzeros
  - Use flexibility of 2D partitioning to bound number of messages
- 2D Block Cartesian*
  - Random (Yoo, et al. SC'11)
  - Hypergraph (Boman, et al. SC'13; Wolf, Miller IEEE HPEC 2014)
  - Leads to further scaling
- **New Zoltan2 Support**: 2D block Cartesian partitioning

*Previous work by Hendrickson, et al.; Bisseling, et al.

Outline

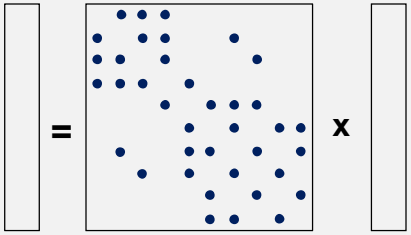
- TriData Overview
- Focus 1: Hypergraphs and Incidence Matrices
- Focus 2: 2D Partitioning
- Focus 3: Interoperability
- Future TriData Focus: KokkosKernels
- Summary

High Performance Graph Analytics



Graph

Linear Algebra



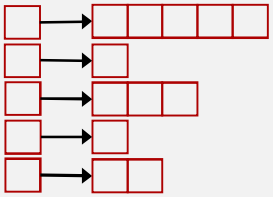
Clustering, eigenvector centrality, commute time, mean hitting time

TriData (SNL), GraphBLAS

Device Interface

System Architecture 1

Graph Traversal



BFS, connected components, page rank

MTGL (SNL), BGL (PNNL)

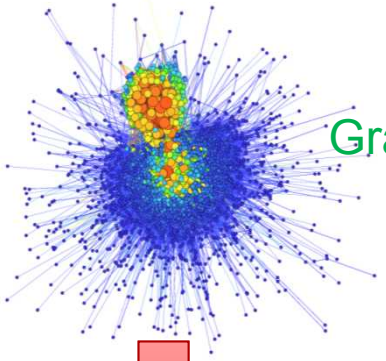
Device Interface

System Architecture 2

Two different approaches to graph analytics

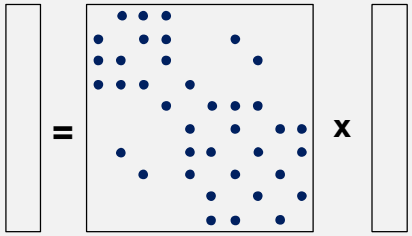
BFS = Breadth First Search; SNL=Sandia National Labs; PNNL (Pacific Northwest National Labs)

Interoperability: TriData + MTGL Graph Analysis



Graph

Linear Algebra



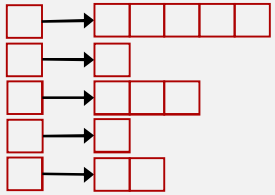
Clustering, eigenvector centrality, commute time, mean hitting time

TriData (SNL)

Kokkos

Graph data structure

Graph Traversal



BFS, connected components, page rank

MTGL (SNL)

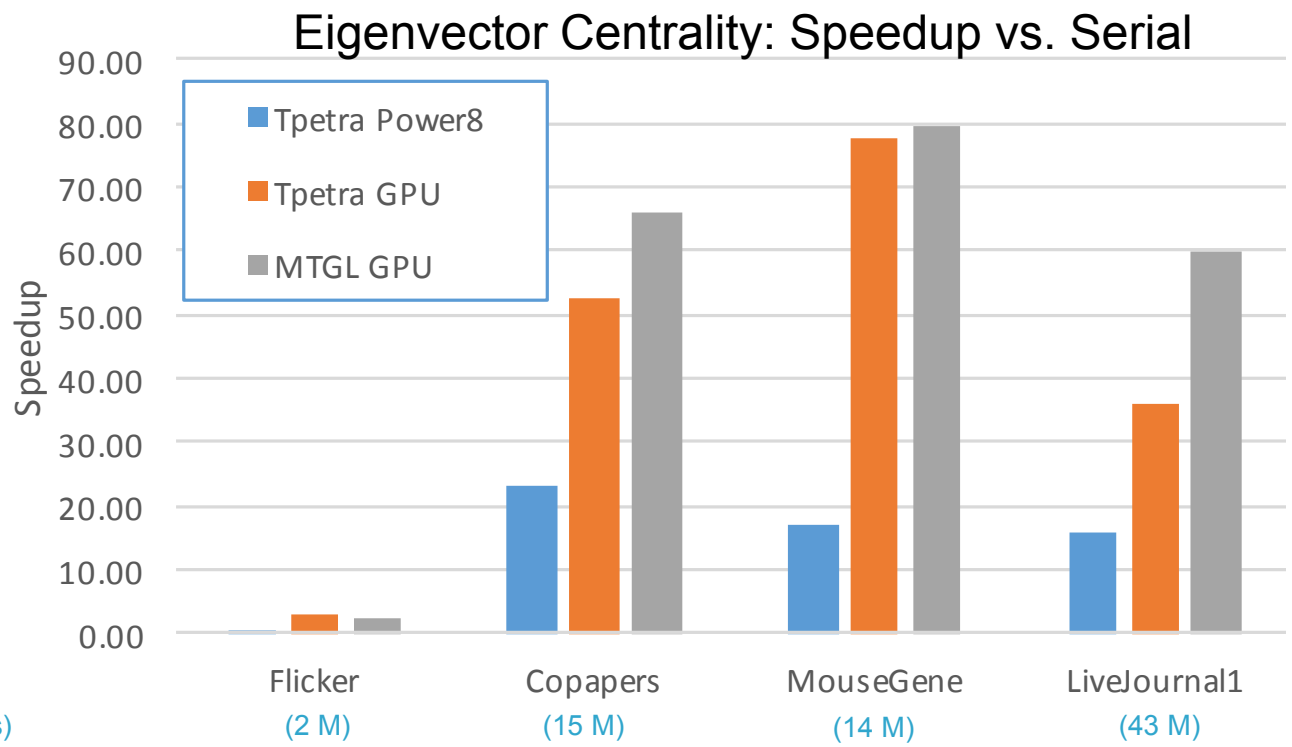
Kokkos

GPUs, Multicore CPUs, Intel KNL, …

TriData MTGL adapter allows us to run linear-algebra based algorithms directly on MTGL graphs

BFS = Breadth First Search

TriData Centrality Results: Tpetra and MTGL



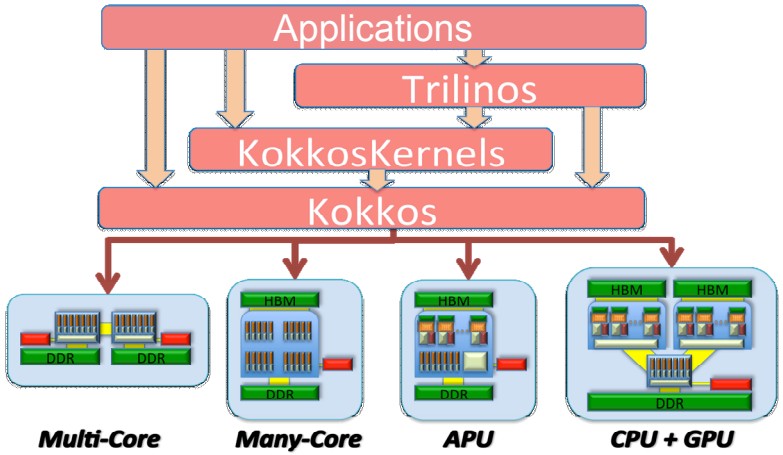
- CPU: 20 core IBM Power 8, 3.42 GHz (serial, multithreaded)
- GPU: NVIDIA Pascal P100 (Tpetra, Kokkos/MTGL)

- **GPU computation is up to 80x speedup over host serial**
- **TriData/MTGL in general better than TriData/Tpetra**

Outline

- TriData Overview
- Focus 1: Hypergraphs and Incidence Matrices
- Focus 2: 2D Partitioning
- Focus 3: Interoperability
- Future TriData Focus: KokkosKernels
- Summary

Future Focus: KokkosKernels for Node-Level Performance



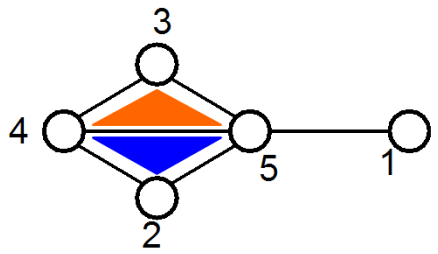
- Kokkos
  - Tools for performance portable node-level parallelism
  - Manages data access patterns, execution spaces, memory spaces
  - Performance portability not trivial for sparse matrix and graph algorithms
- KokkosKernels
  - Layer of performance-portable kernels for high performance
  - Sparse/Graph: SpMV, SpGEMM, triangle enumeration
- **Related Talk – Deveci, Friday 3:20pm, Room 52-303**
  - "Performance Portable Sparse Matrix Matrix Multiplication with Applications in Scientific Computing and Graph Analytics"

KokkosKernels for performance-portable sparse/graph kernels

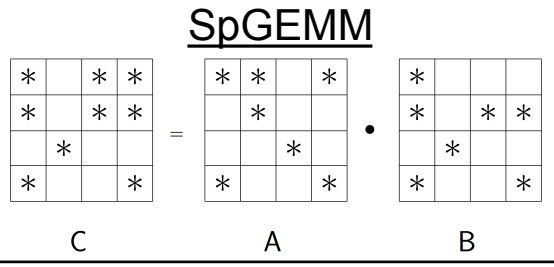
KokkosKernels and IEEE/DARPA Graph Challenge

KKTri

Linear Algebra Based Triangle Counting



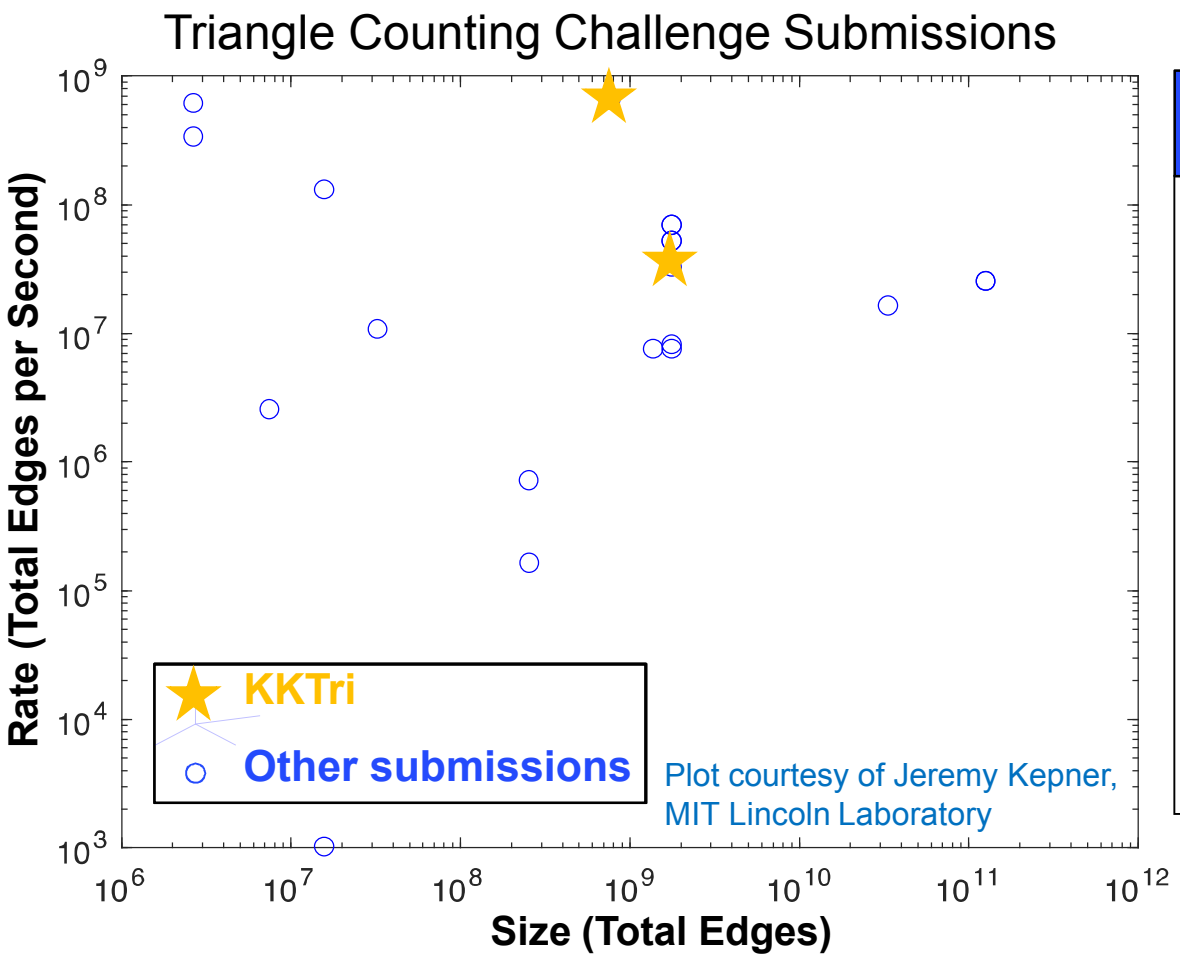
KKMEM: KokkosKernels Matrix-Matrix Multiply



- 2017 IEEE/DARPA Graph Challenge Submission
  - Wolf, Deveci, Berry, Hammond, Rajamanickam: "Fast Linear Algebra-Based Triangle Counting with KokkosKernels."
  - **Triangle Counting Champion** (focus: single node)
  - Counted 34.8B triangles in 1.2B edge graph in 43 secs (Twitter2010)

Vision: Build software on top of highly optimized KokkosKernels kernels (e.g., KKTri) to impact applications

Graph Challenge: Lessons Learned

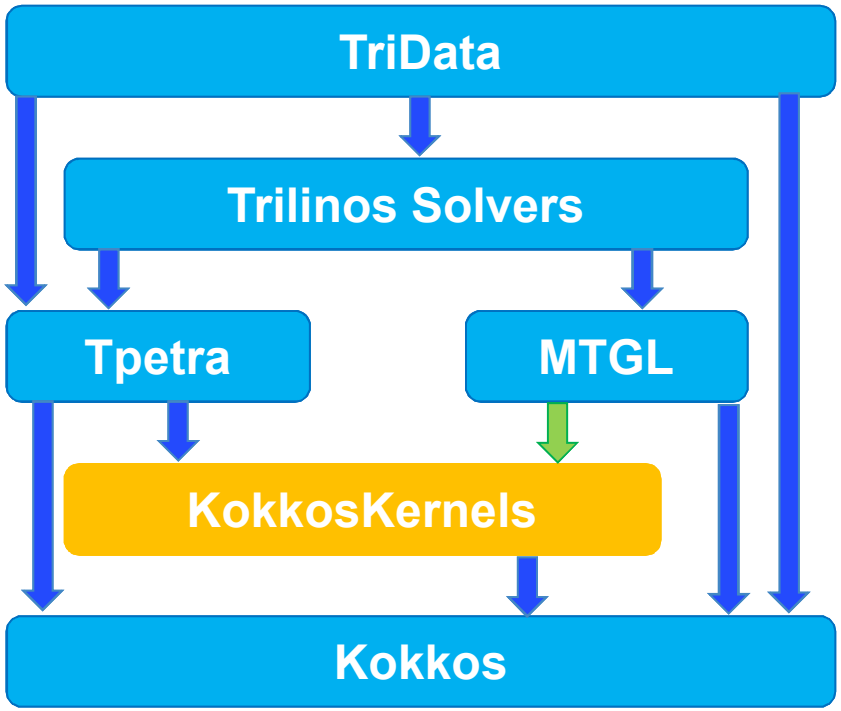


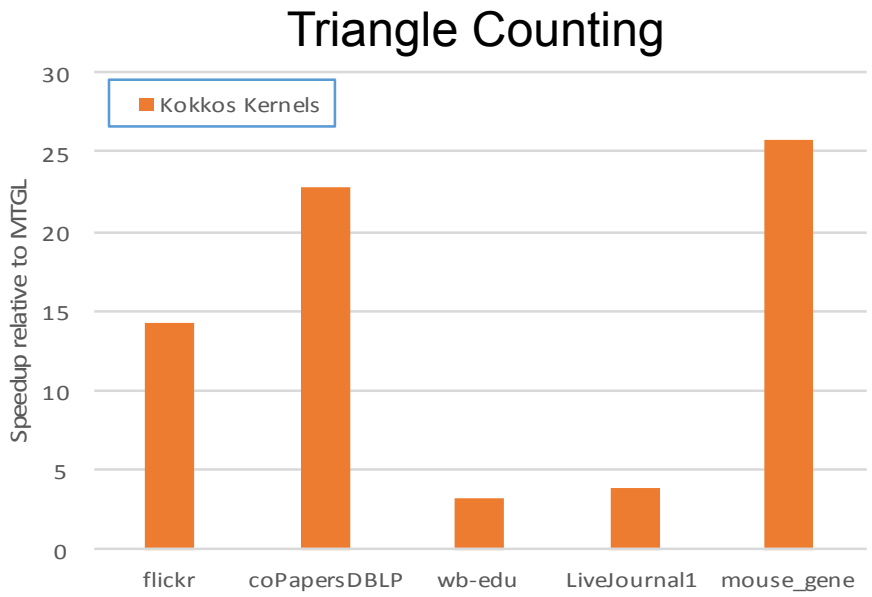
Lessons Learned

- Linear algebra approach can be competitive
- Avoiding unnecessary computation essential
- Data compression often helps performance
- Visitor pattern can add more flexibility to linear algebra approach

Linear algebra-based KKTri as good as or better than other state-of-the-art methods

Future TriData Software Stack





Leverage highly optimized KokkosKernels kernels for high performance data analysis

Summary

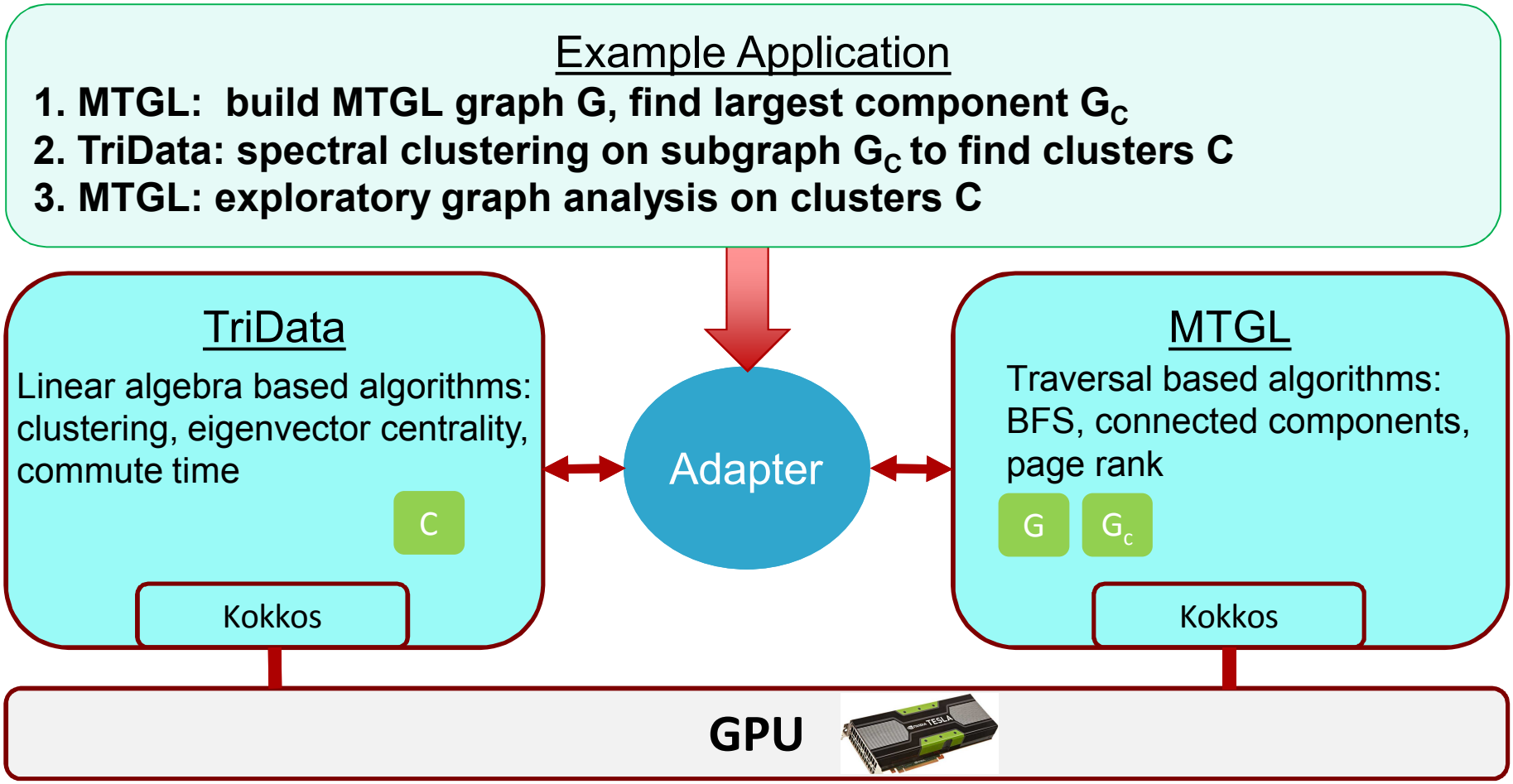
- Overview of TriData
  - Trilinos for Large-Scale Data Analytics
  - Targeting both workstations and supercomputers
- Several key foci of TriData that enable high performance
  - Use of hypergraphs, incidence matrices beneficial computationally
  - 2D partitioning necessary for scalable solvers (on network problems)
  - Flexible adapters support interoperability with MTGL
- Related Work
  - Wolf, Klinvex, Dunlavy. "Advantages to modeling relational data using hypergraphs versus graphs," *Proc of IEEE HPEC*, 2016.
  - Wolf, Deveci, Berry, Hammond, Rajamanickam. "Fast Linear Algebra-Based Triangle Counting with KokkosKernels," *Proc of IEEE HPEC*, 2017.
  - **Triangle Counting: https://github.com/Mantevo/miniTri**
  - **KokkosKernels: https://github.com/kokkos/kokkos-kernels**

Extra

Sandia National Laboratories

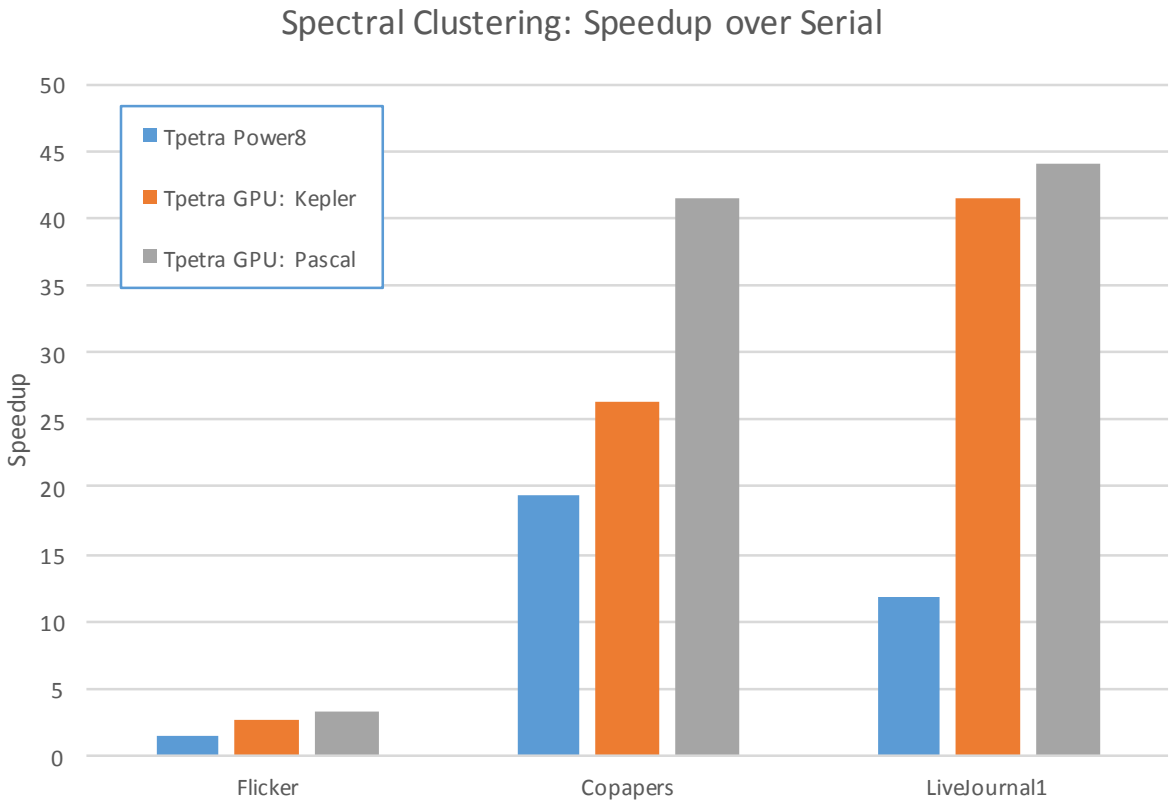
Example: TriData + MTGL GPU Graph Analysis

Sandia National Laboratories



- **TriData and MTGL uses Kokkos for performant GPU execution**
- **TriData use of MTGL graph directly (via adapter) avoids data copy**

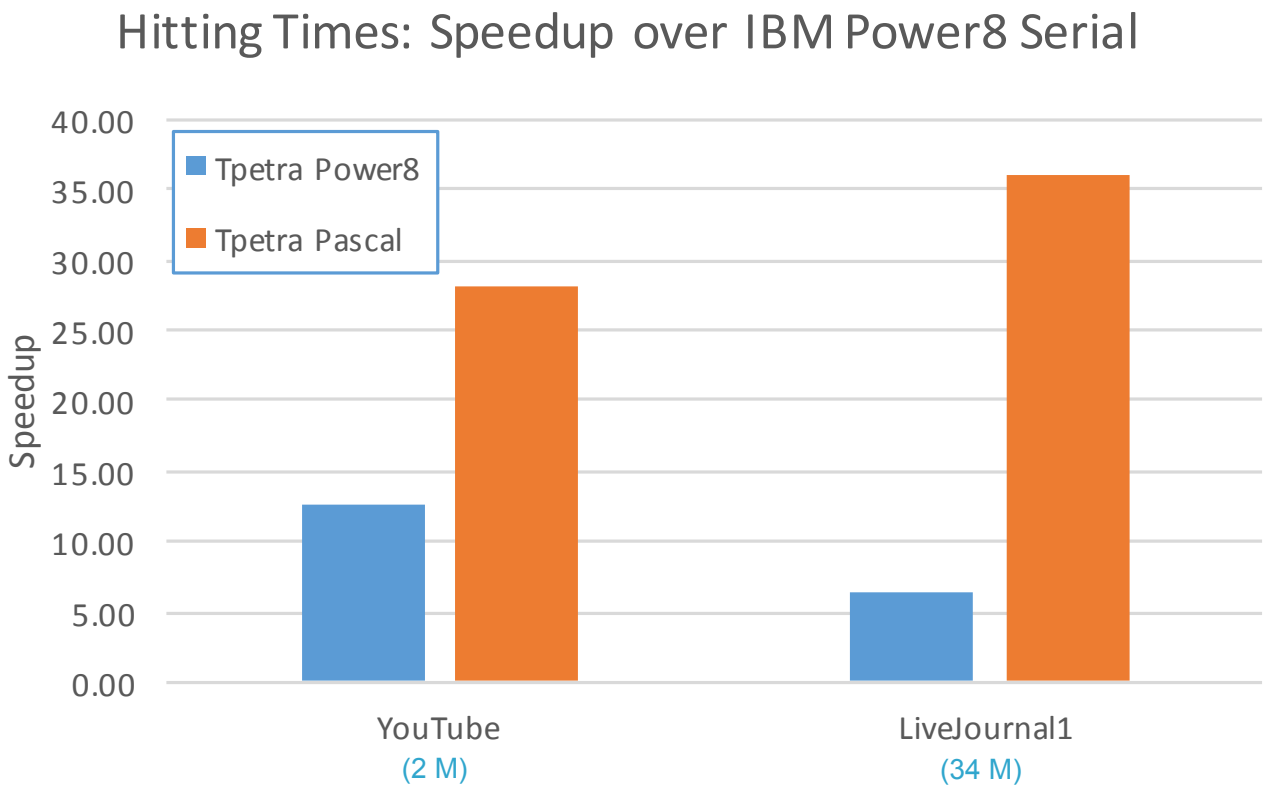
TriData Spectral Clustering Results



- CPU: 20 core IBM Power 8, 3.42 GHz (serial, multithreaded)
- GPU: NVIDIA Pascal P100 (Tpetra, Kokkos/MTGL)

- **GPU computation up to 45x speedup over host serial**

Mean Hitting Time Results

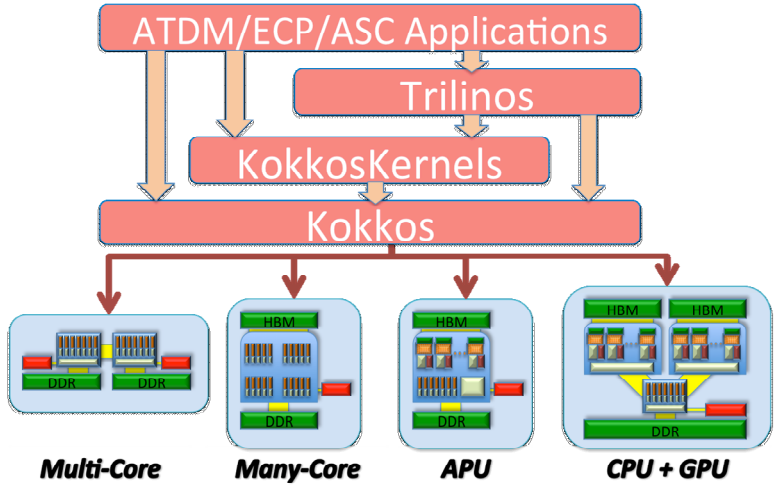


- Distributed memory TriData/Tpetra on GPU (not MTGL)

GPU computation is up to 35x speedup over host serial

Performance Portability

- Kokkos:
  - Layered collection of template C++ libraries
  - Manages data access patterns
  - Execution spaces, Memory spaces
- Kokkos provides tools for portability
  - Performance portability does not come for free.
  - Not trivial for sparse matrix and graph algorithms



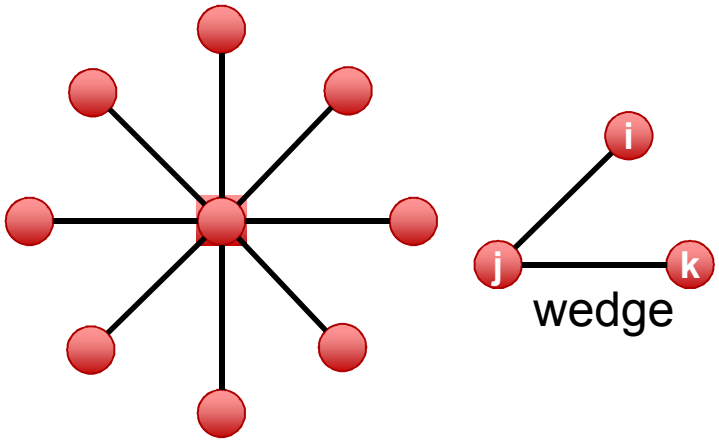
- KokkosKernels:
  - Layer of performance-portable kernels
  - SpGEMM

KKTri leverages Kokkos and KokkosKernels for performance-portable linear algebra-based triangle counting

GC 1: Vertex Ordering and Triangle Counting

Sandia National Laboratories

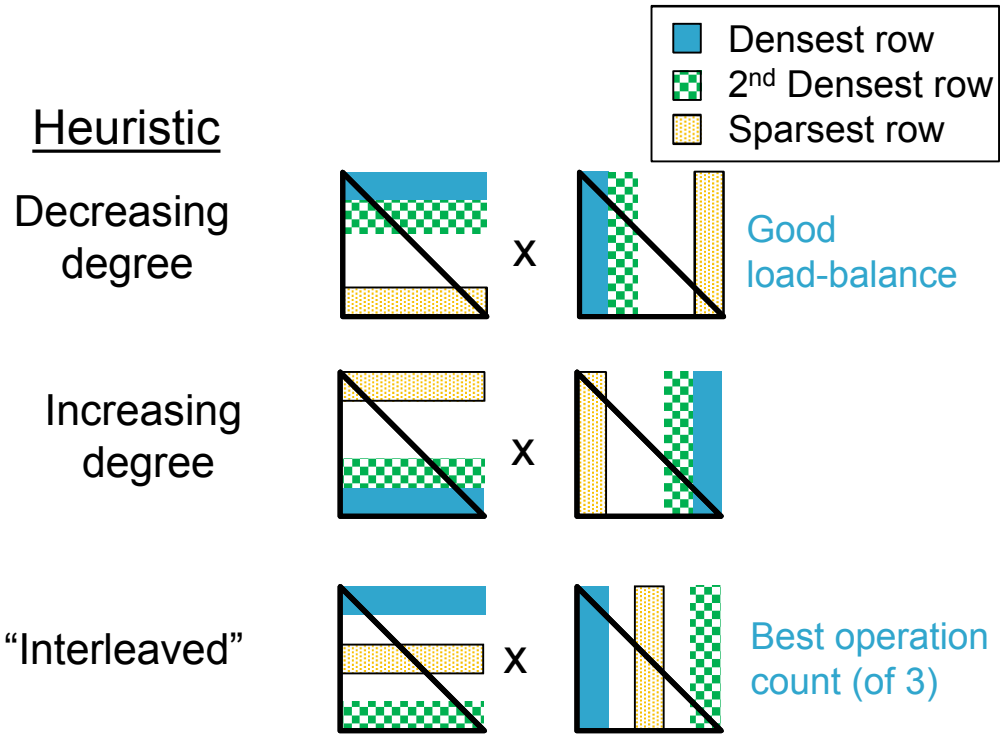
Vertex Ordering Matters



Wedges with d(i) < d(j) > d(k) : 56
Wedges with d(i) > d(j) < d(k) : 0

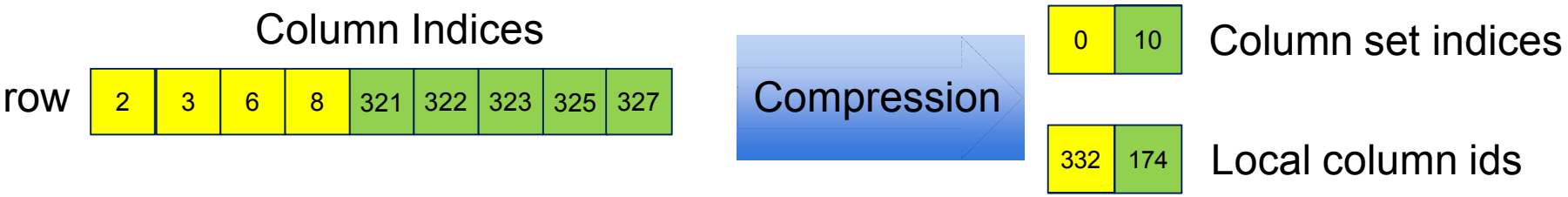
- **Ordering: essential first step for triangle counting**
- **Impacts # operations (# wedges visited)**

KKTri Ordering Challenging



Avoiding computation essential for efficient triangle counting!

GC 2: Matrix Compression



- Compression used to improve performance
  - Encodes columns using fewer integers
  - Reduces number of operations and memory required in symbolic phase
  - Allows ”vectorized” bitwise union/intersection of different rows
- Effectiveness of compression varies greatly with data
  - Large random graphs compress poorly (R-Mat <1% compression storage)
  - However, still helpful for many random graphs (e.g., power-law) – effective for dense rows (improves load balance, operation count)

Compression consistently improves triangle counting performance

GC 3: Visitor Pattern

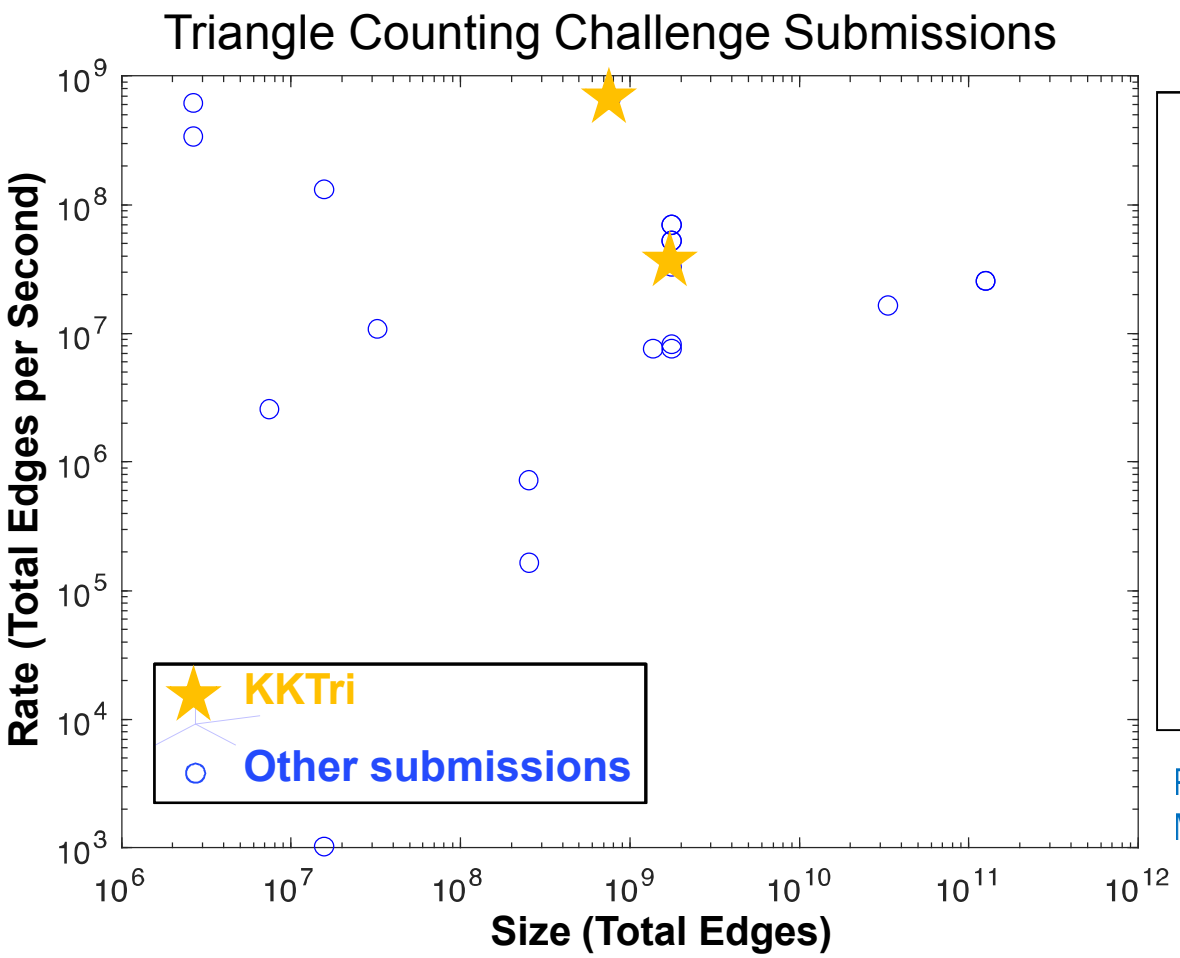
Sandia National Laboratories

- KKMEM based triangle counting supports visitor pattern
  - Concept fundamental to BGL and MTGL
- Functor passed to triangle identification function, which allows method to be run once triangle is found
  - For triangle counting: triangleCount++;
  - Flexibility allows for more complex analysis of triangles, miniTri

Visitor pattern support provides additional flexibility to analysts

BGL = Boost Graph Library, MTGL = MultiThreaded Graph Library

Graph Challenge Results



- 2017 Graph Challenge
  - Sponsors: DARPA, Amazon, IEEE HPEC, MIT Lincoln Lab
  - 3 problems, 5 champions
  - **KKTri**: **Champion** status for triangle counting
- Top rates per submission
  - Largest graph and rate
  - Highest rate and size

Plot courtesy of Jeremy Kepner, MIT Lincoln Laboratory

Linear algebra-based KKTri achieves higher peak rate than almost all other triangle counting submissions