SAND2016-6439C

Task Parallel Approach to the Linear Algebra-Based Implementation of miniTri

Michael Wolf

Center for Computing Research

SIAM Annual Meeting

11 July, 2016

Sandia National Laboratories

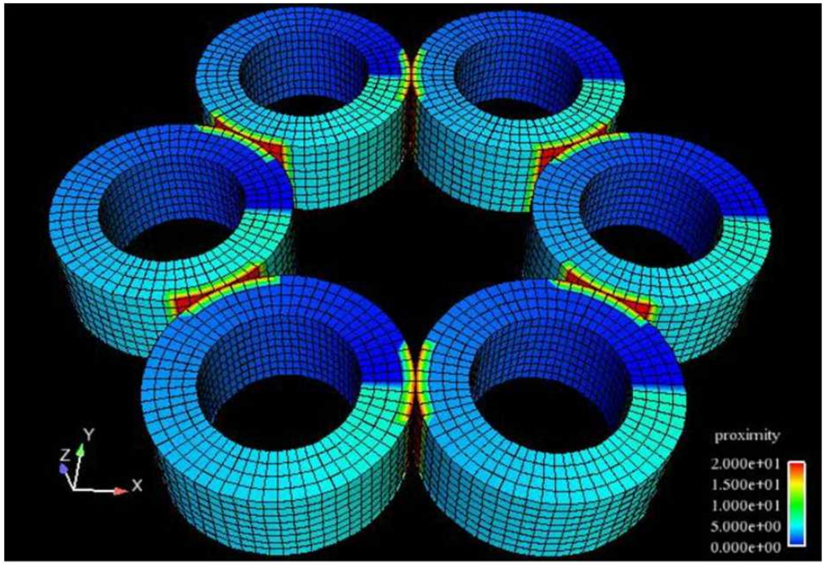
Exceptional service in the national interest

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2015-7696C

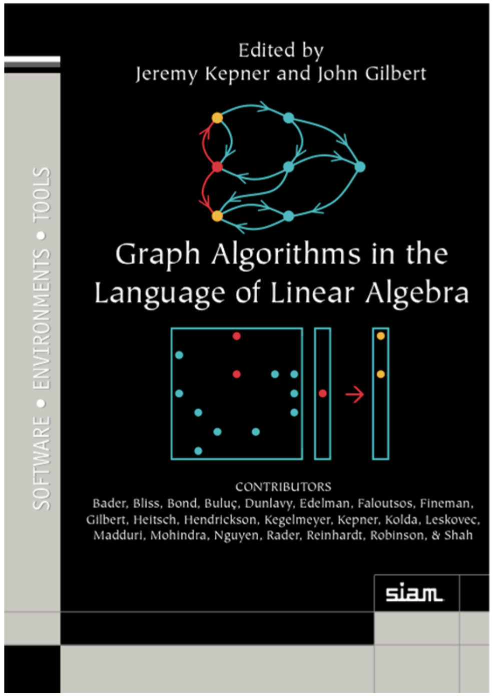
Miniapps



phdMesh (Mantevo)

- Small, self contained applications
- Proxy for important characteristics of full applications
- Important co-design tool: performance analysis
  - Target 1: existing applications on new and future architectures
  - Target 2: new applications on existing architectures
  - Strong partnership with industry
- Mantevo MiniApp Suite (Sandia): mantevo.org

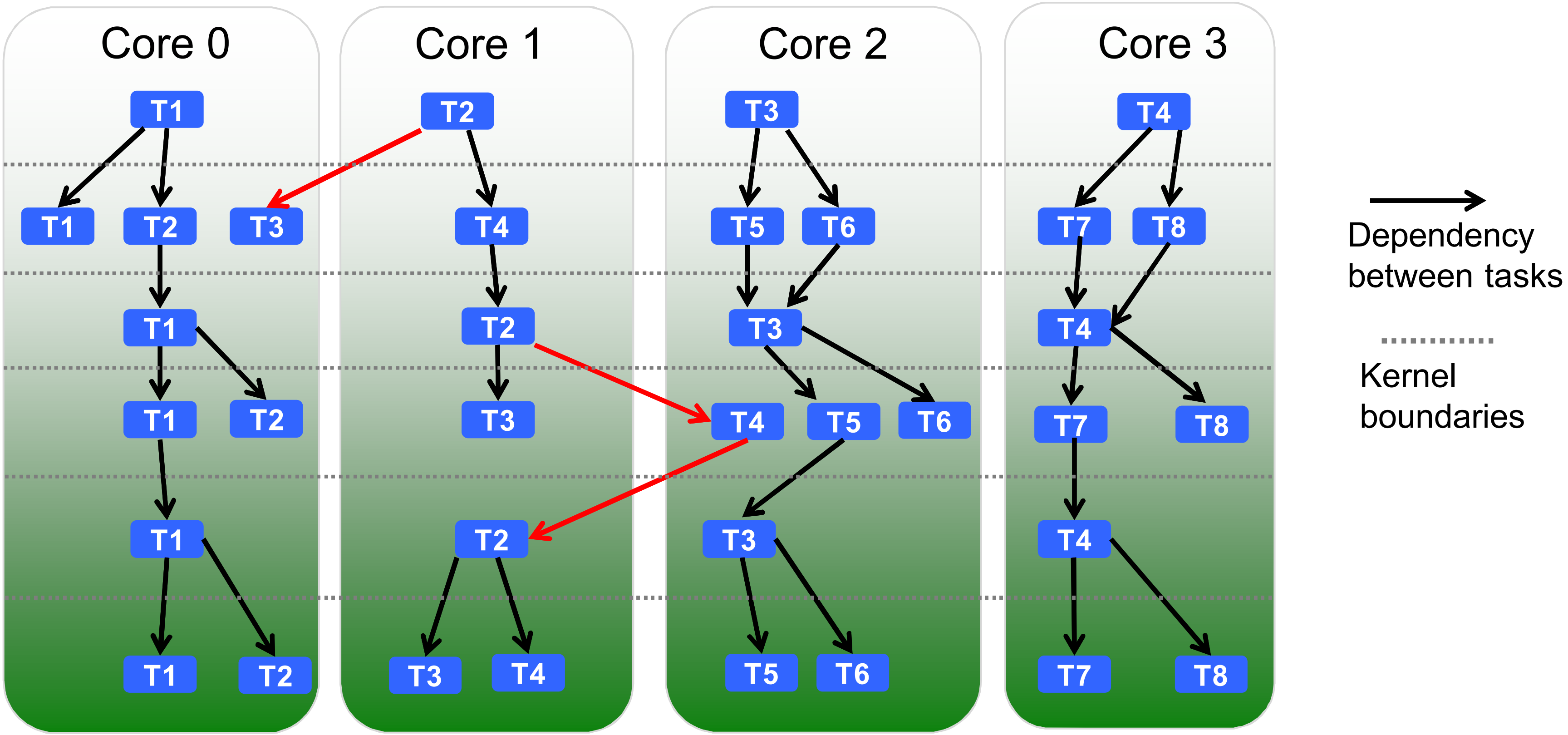
Graph BLAS



Eds. Kepner, Gilbert

- Effort to standardize building blocks for graph algorithms in language of linear algebra
  - Overloaded linear algebra kernels express most graph computations
  - Matrix-graph duality – highly impactful in CS&E (partitioning, solvers,…)
  - Promising for data sciences but many challenges

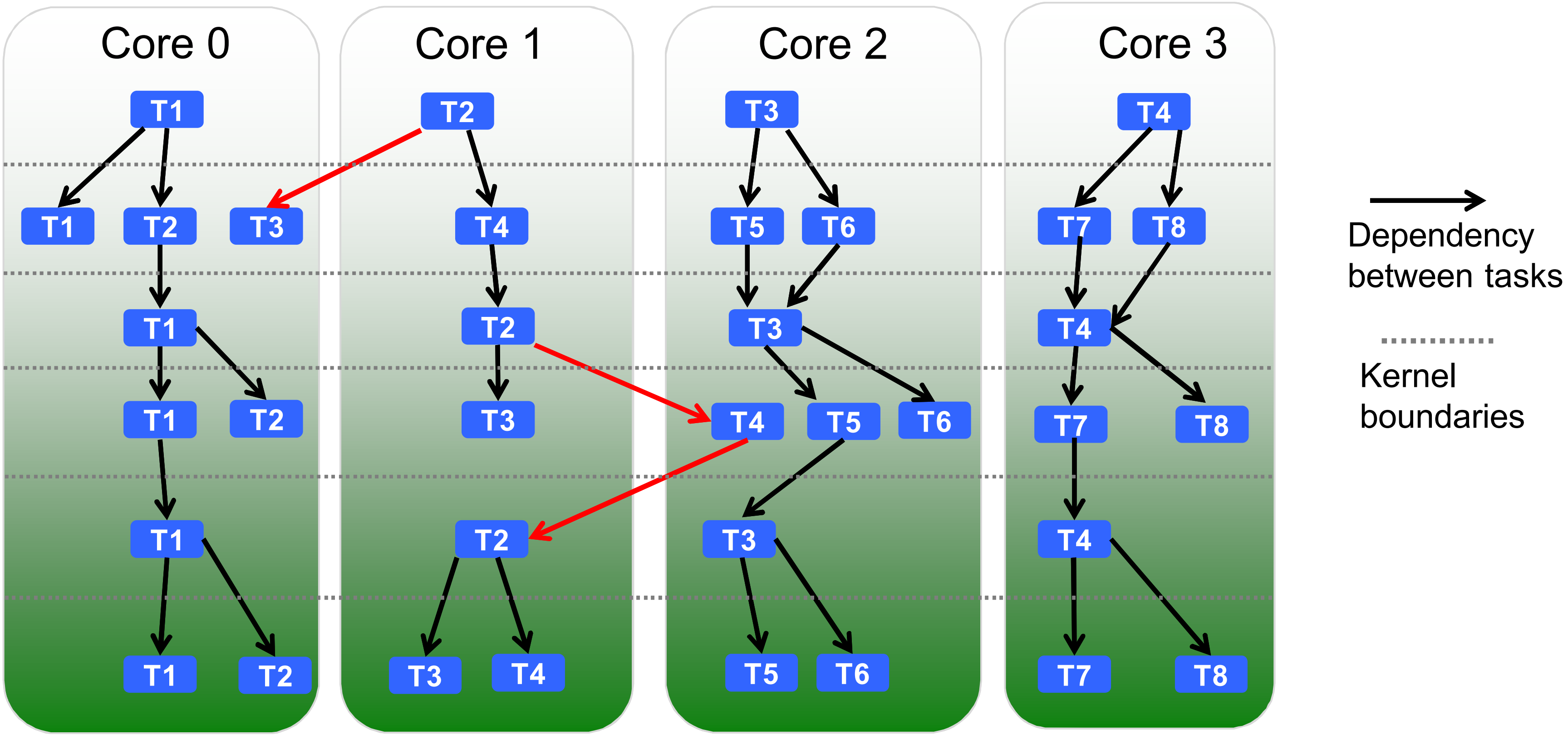
Task Parallelism



- Dataflow of application expressed through tasks/dependencies (Avoid explicit barriers between kernels)
- Overdecomposition of problem into tasks (# tasks > #cores)
- Tasks scheduled and moved to appropriate compute resources

Task Parallelism

Sandia National Laboratories

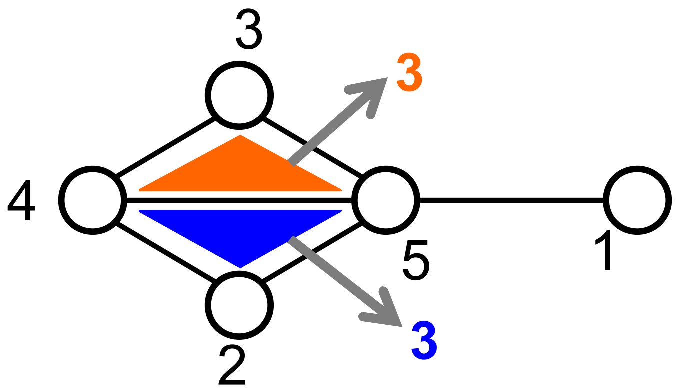


- Natural for data analytics (model intrinsically data-centric)
- Several task parallel models/libraries: **HPX**, **Kokkos/Qthreads**, Uintah, Legion, OCR, …

Outline

- Background
- ➡ miniTri
- Linear Algebra-Based miniTri (miniTriLA)
- Task Parallel Approach to miniTriLA
  - HPX
  - Kokkos/Qthreads
- Memory-Constrained Task Parallelism
- Summary

miniTri: Data Analytics Miniapp



k:

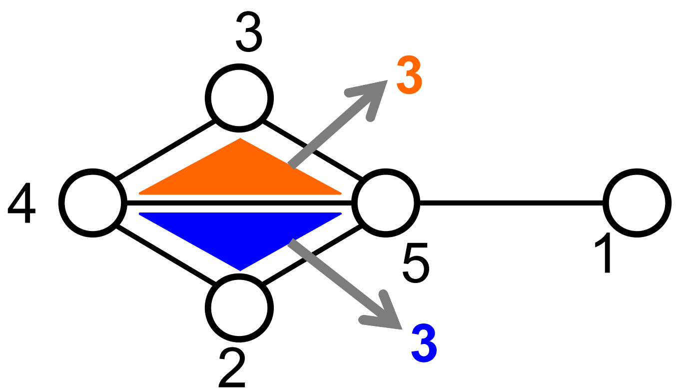
$$\arg\max_k \{(\min_{v \in t} t_v \geq \binom{k-1}{2}) \cap (\min_{e \in t} t_e \geq k-2)\}$$

Max clique size of given triangle

- Proxy for triangle based data analytics (Mantevo)
- Uses **triangle enumeration** + vertex/edge properties
- Key uses: **dense subgraph detection**, characterizing graphs, improving community detection, generating graphs
- Related applications in **cyber security**, **intelligence**, **functional biology**

miniTri is more application relevant than standard data analytics benchmarks such as Graph 500

miniTri: Overview



k:

$$\arg\max_k \{(\min_{v \in t} t_v \geq \binom{k-1}{2}) \cap (\min_{e \in t} t_e \geq k-2)\}$$

- miniTri Steps:
  - For each triangle, calculate triangle degrees for vertices and edges
  - For each triangle, calculate integer **k** given triangle degree info
- Developed 20+ variants
  - Different methods: buckets data structure, set intersection, linear algebra
  - Different programming models: OpenMP, MPI, HPX, Kokkos/Qthreads
- **Focus of talk:** linear algebra-based miniTri, task parallel models

Challenge: Can miniTri be implemented efficiently using Graph BLAS-like building blocks?

Outline

- Background
- miniTri
- ➡ Linear Algebra-Based miniTri (miniTriLA)
- Task Parallel Approach to miniTriLA
  - HPX
  - Kokkos/Qthreads
- Memory-Constrained Task Parallelism
- Summary

Linear Algebra Based miniTri (miniTriLA)

Sandia National Laboratories

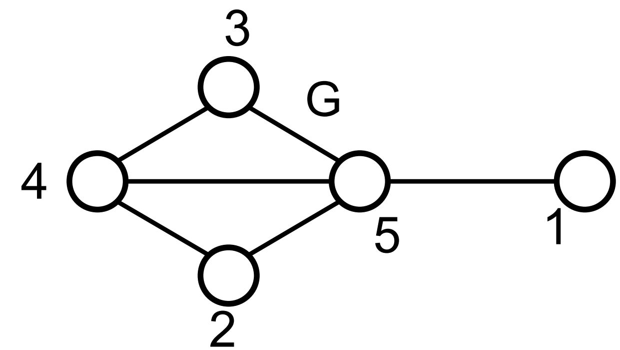
<u>miniTri</u>

$C = A * B$

$t_v = C * 1$

$t_e = C^T * 1$

$kcount(C, t_v, t_e)$



$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Adjacency matrix of G

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Incidence matrix of G

- Developed Graph BLAS-like formulation of miniTri
- Important stressor of Graph BLAS

A = adjacency matrix for graph, B = incidence matrix for graph,
1 = vector of ones

miniTriLA: Triangle Enumeration

<u>miniTri</u>

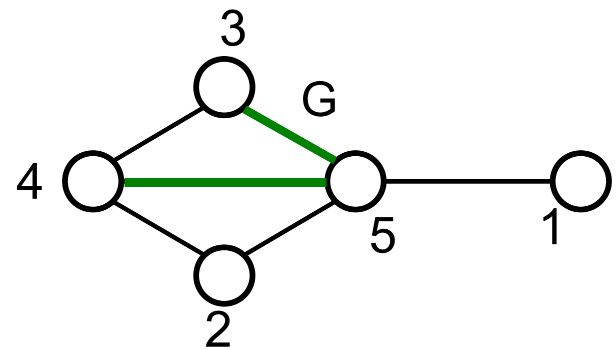
$C = A * B$

$t_v = C * 1$

$t_e = C^T * 1$

$kcount(C, t_v, t_e)$

Enumerates each triangle 3 times (once: C=L*B, where L is lower triangle part of A)



$$C = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{2,4,5\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{3,4,5\} \\ \emptyset & \{4,2,5\} & \{4,3,5\} & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{5,3,4\} & \{5,2,4\} & \emptyset \end{pmatrix}$$

Adjacency matrix of G — Incidence matrix of G — Matrix of Triangles

- Wedges implicitly stored in rows of A
  - Adj matrix: wedges = pairwise combinations of nonzero column ids + row #
  - E.g., row 5 wedges: {1,5,2}, {1,5,3}, {1,5,4}, {2,5,3}, {2,5,4}, {3,5,4}

A = adjacency matrix for graph, B = incidence matrix for graph, 1 = vector of ones

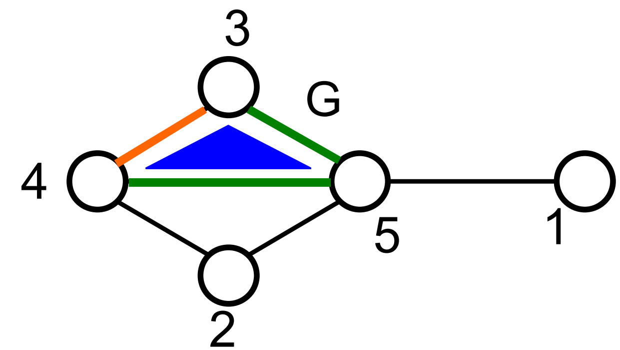
SPGEMM = Sparse Matrix Multiplication

miniTriLA: Triangle Enumeration

miniTri

```
C = A * B
t_v = C * 1
t_e = C^T * 1
kcount(C, t_v, t_e)
```

Enumerates each triangle 3 times (once: C=L*B, where L is lower triangle part of A)



$$C = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{2,4,5\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{3,4,5\} \\ \emptyset & \{4,2,5\} & \{4,3,5\} & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{5,3,4\} & \{5,2,4\} & \emptyset \end{pmatrix}$$

Adjacency matrix of G — Incidence matrix of G — Matrix of Triangles

- Columns of B tell us whether wedge is "closed" to form triangle
- Overloaded SpGEMM yields triangle enumeration:
  - If $A_{i,x}=A_{i,y}=1$ and $B_{x,j}=B_{y,j}=1$ (or equivalently $A_{i,*}B_{*,j}=2$), $C_{i,j}$ = triangle {i,x,y}
  - Else, no triangle

A = adjacency matrix for graph, B = incidence matrix for graph, 1 = vector of ones

SPGEMM = Sparse Matrix Multiplication

miniTriLA: Triangle Degree Calculation

Sandia National Laboratories

<u>miniTri</u>

C = A * B
$t_v = C * 1$
$t_e = C^T * 1$
kcount(C, t_v, t_e)

$$C=\begin{pmatrix} \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{2,4,5\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{3,4,5\} \\ \emptyset & \{4,2,5\} & \{4,3,5\} & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{5,3,4\} & \{5,2,4\} & \emptyset \end{pmatrix}$$

$$t_v = \begin{pmatrix} \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{2,4,5\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{3,4,5\} \\ \emptyset & \{4,2,5\} & \{4,3,5\} & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{5,3,4\} & \{5,2,4\} & \emptyset \end{pmatrix} * \mathbf{1} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 2 \\ 2 \end{pmatrix}$$

Triangles with v_4

$$t_e = \begin{pmatrix} \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{2,4,5\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{3,4,5\} \\ \emptyset & \{4,2,5\} & \{4,3,5\} & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{5,3,4\} & \{5,2,4\} & \emptyset \end{pmatrix}^T * \mathbf{1} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 2 \end{pmatrix}$$

Triangles with $e_{4,5}$

Overloaded SpMV to count triangles in rows

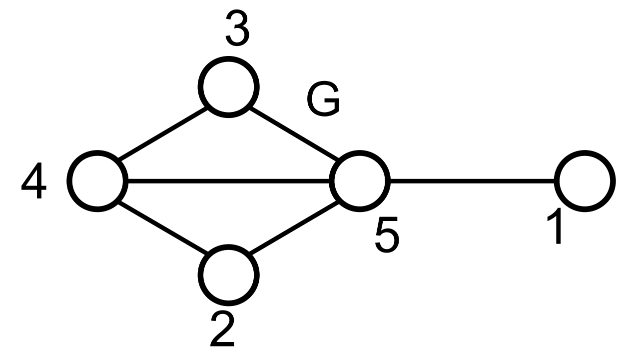
- Triangle degree calculation
  - Each triangle represented once in C for each of its edges and vertices
  - Triangle vertex and edge degrees are number of nz in rows and columns

A = adjacency matrix for graph, B = incidence matrix for graph,
1 = vector of ones

miniTriLA: Kcounts

<u>miniTri</u>

```
C = A * B
t_e = C * 1
t_v = C^T * 1
kcount(C, t_e, t_v)
```



k:

$$\arg\max_k \{(\min_{v \in t} t_v \geq \binom{k-1}{2}) \cap (\min_{e \in t} t_e \geq k-2)\}$$

K	1	2	3
triangle count	0	0	2

- Compute **k** for each triangle (summarize in table)
- **k**-count table gives us upper bound on largest clique in graph
  - Largest *c* such that Comb(*c*,3) triangles have *k*-counts at least *c*

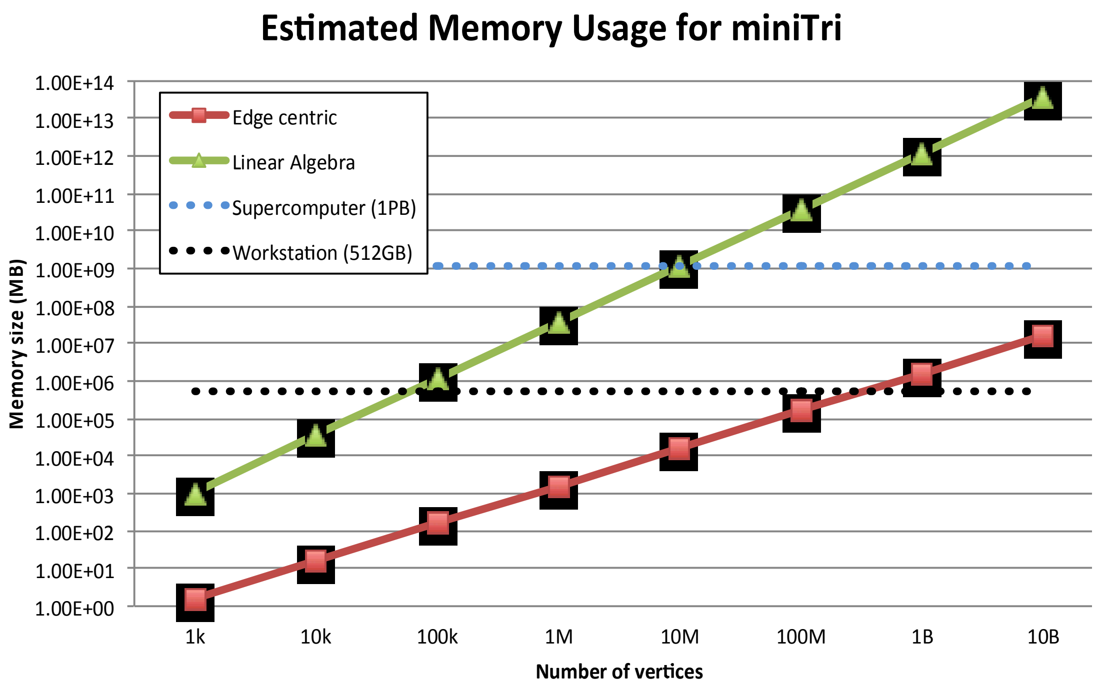
A = adjacency matrix for graph, B = incidence matrix for graph,
1 = vector of ones

miniTriLA: Challenges

Sandia National Laboratories

miniTri

```
C = A * B
t_e = C * 1
t_v = C^T * 1
kcount(C, t_e, t_v)
```



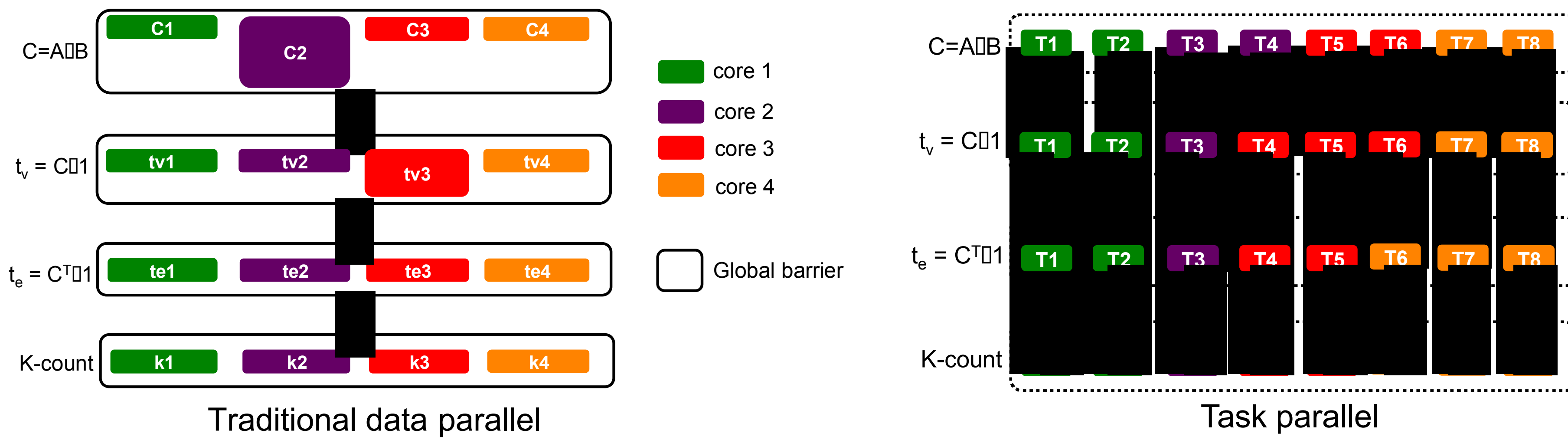
- Challenge 1: Computation difficult to load-balance
- Challenge 2: Typical Graph BLAS approach forms C, which means storing all triangles in graph
  - Worst case: $O(|E|^{3/2})$ triangles in graph, typical: 100-1000 triangles/edge
  - Severely limits size of graph

A = adjacency matrix for graph, B = incidence matrix for graph,
1 = vector of ones

Outline

- Background
- miniTri
- Linear Algebra-Based miniTri (miniTriLA)
- ➡ Task Parallel Approach to miniTriLA
  - HPX
  - Kokkos/Qthreads
- Memory-Constrained Task Parallelism
- Summary

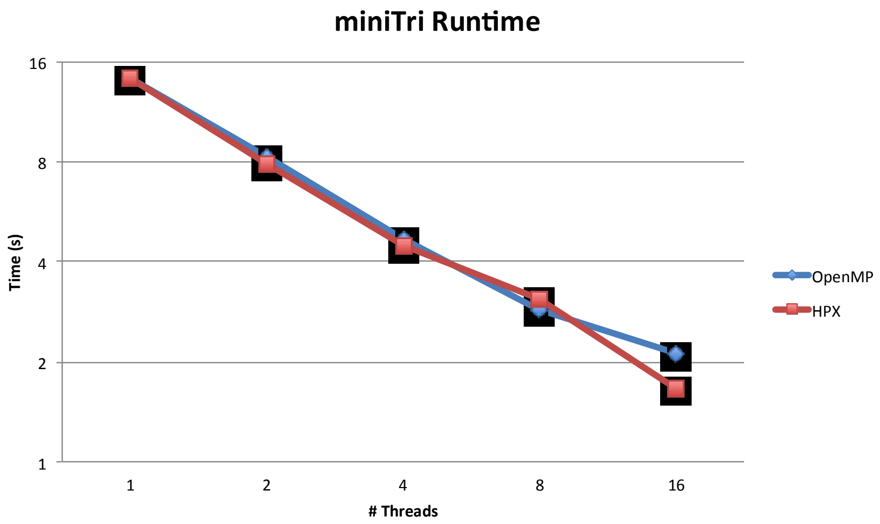
Task Parallel Approach

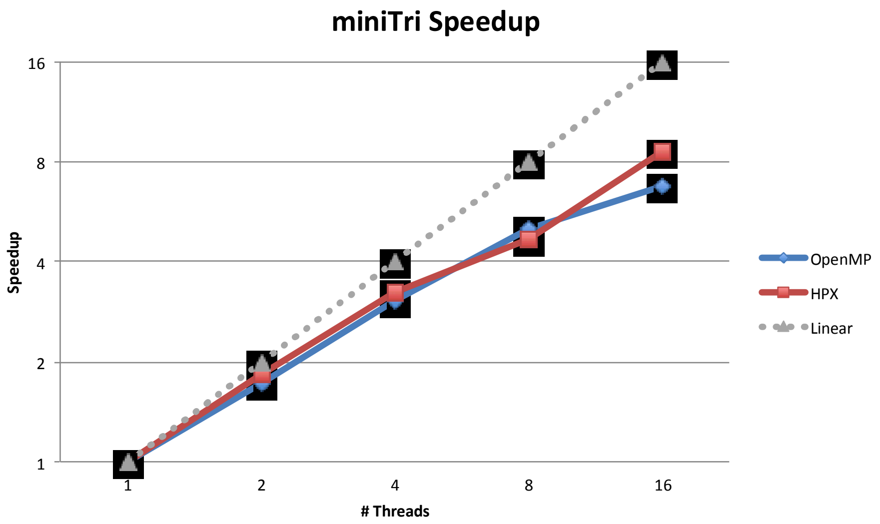


Illustrative Example of GraphBLAS Approaches

- Each block of linear algebra operations assigned to a task
  - Block of rows, 2D block of elements
- Global barriers between kernels removed
  - Replaced by dependencies between tasks
  - Tasks in subsequent kernels can start/finish before first kernel finishes

HPX-3 Implementation

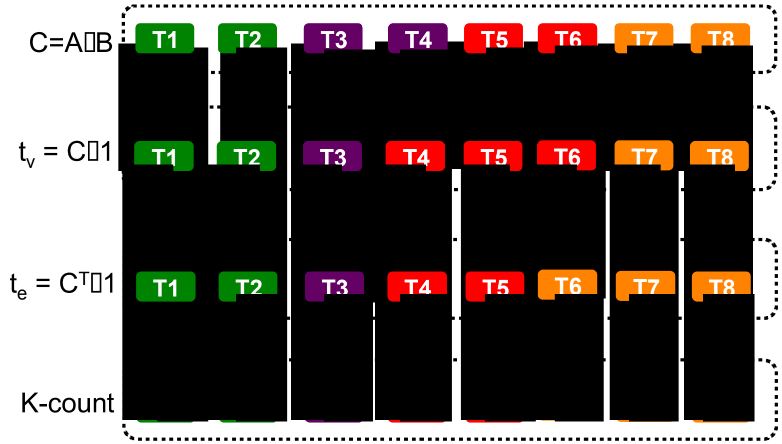




- HPX-3 (LSU)
  - General purpose C++ runtime system
  - Supports both on-node and inter-node tasks parallelism
  - Active global address space (AGAS)
  - Lightweight control objects instead of global barriers

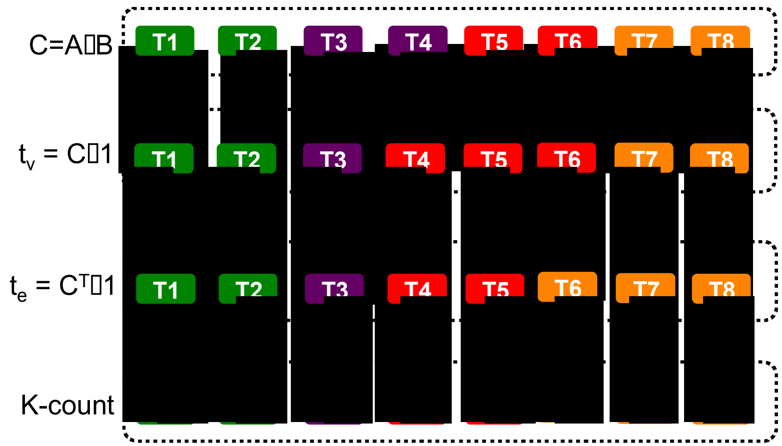
Preliminary results show similar performance of task parallel and data parallel approaches

Kokkos/Qthreads Implementation



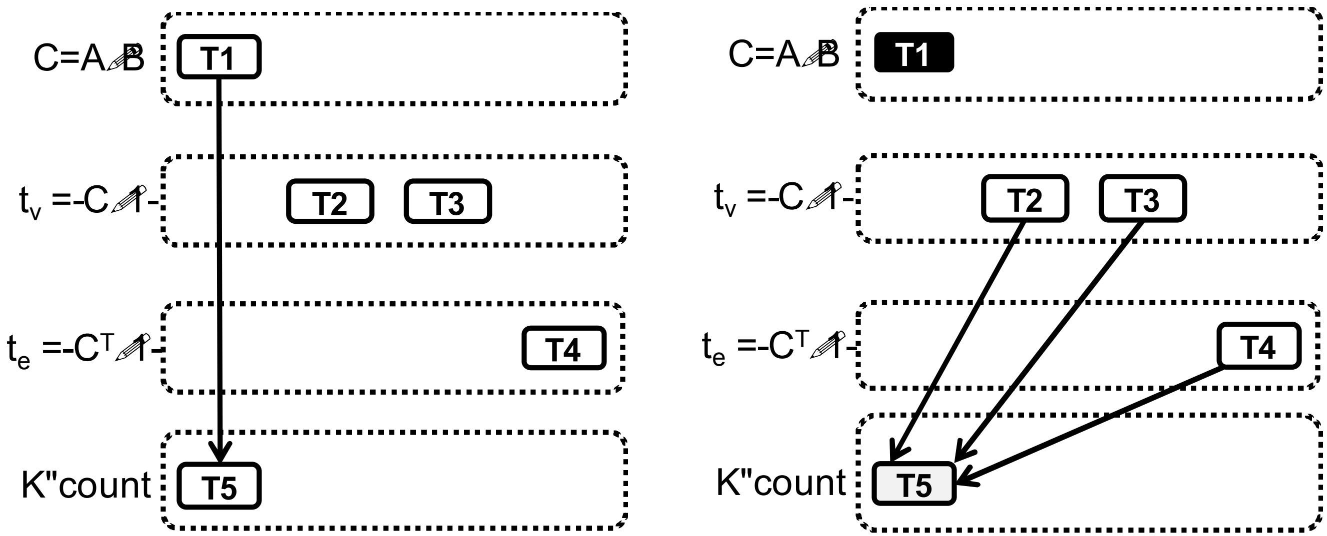
- Kokkos
  - Performance portable programming model and C++ library implementation for intra-node (shared memory) parallelism
  - Supports diverse manycore architectures (GPUs, CPUs, Intel MIC, …)
  - **policy** manages how tasks are scheduled using task-DAG pattern
- Qthreads
  - C-based multithreading library inspired by MTA/XMT architectures
  - Runtime system to execute Kokkos API's task-DAG pattern

Kokkos Task Parallel API



- Task Creation
  - f = policy.create(func);
  - f – future; func – functor that creates/executes tasks
- Setting Dependencies
  - policy.add_dependence(f1, f2);
  - f1, f2 – futures such that task corresponding to f1 depends on task corresponding to f2
- Launch Tasks
  - policy.spawn(f); // f is future

Dynamic Task Dependencies of miniTri



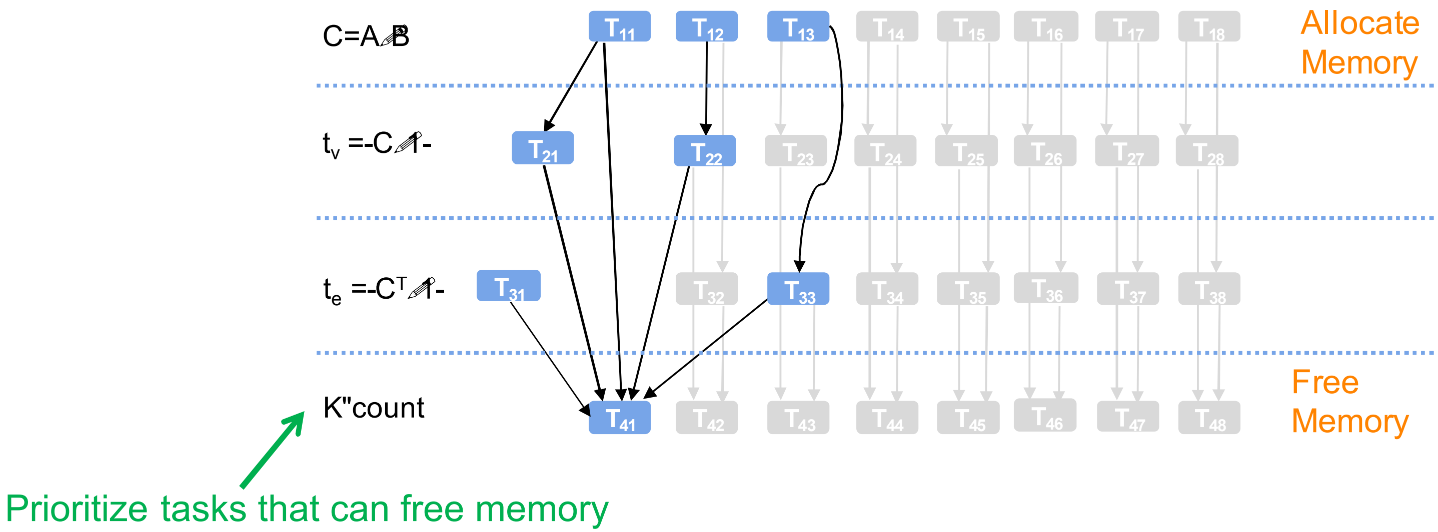
T5 initially spawned | T5 respawned once T1 complete

- Complication with asynchronous task parallel miniTri
  - kcount task (e.g., T5) dependencies not known until after corresponding triangle enumeration task (e.g., T1) is complete
- Kokkos provides respawn functionality
  - Relaunches task
  - Necessary for portability to GPUs (tasks can't yield to other tasks)
  - miniTri exploits this feature to handle dynamic dependencies

Outline

- Background
- miniTri
- Linear Algebra-Based miniTri (miniTriLA)
- Task Parallel Approach to miniTriLA
  - HPX
  - Kokkos/Qthreads
- ➡ Memory-Constrained Task Parallelism
- Summary

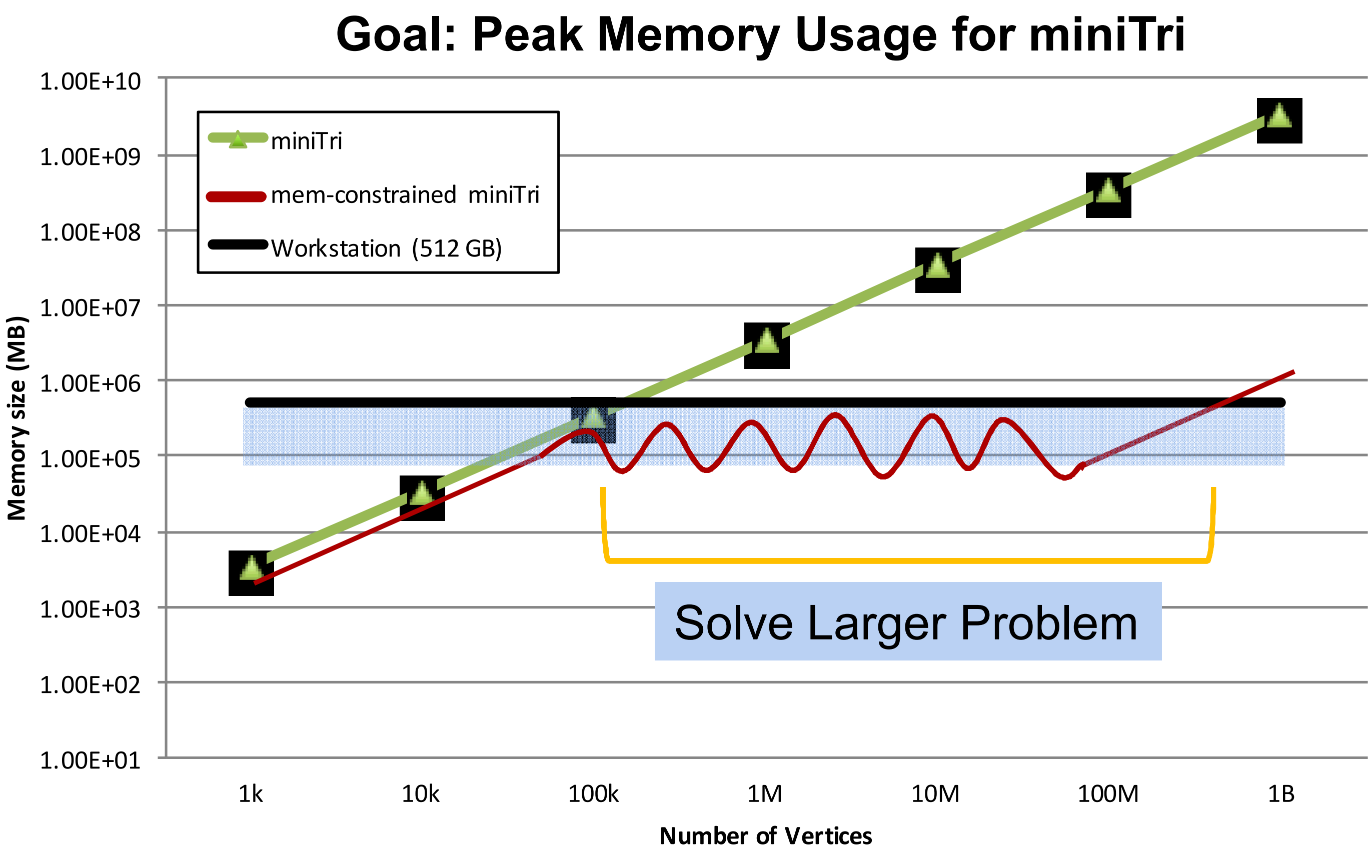
Addressing Challenge #2 with Memory-Constrained Task Parallelism



- Key insight: Tasks can be scheduled asynchronously to free temporary memory early
  - Prioritize k-count tasks to free blocks of triangles from memory
  - **Need runtime system to support advanced resource management/priorities (on-going effort: Kokkos/Qthreads and HPX)**

Asynchrony can be exploited to reduce peak memory of application

Memory-Constrained Task Parallelism



Task parallelism with memory-constrained scheduling allows solution of larger problems

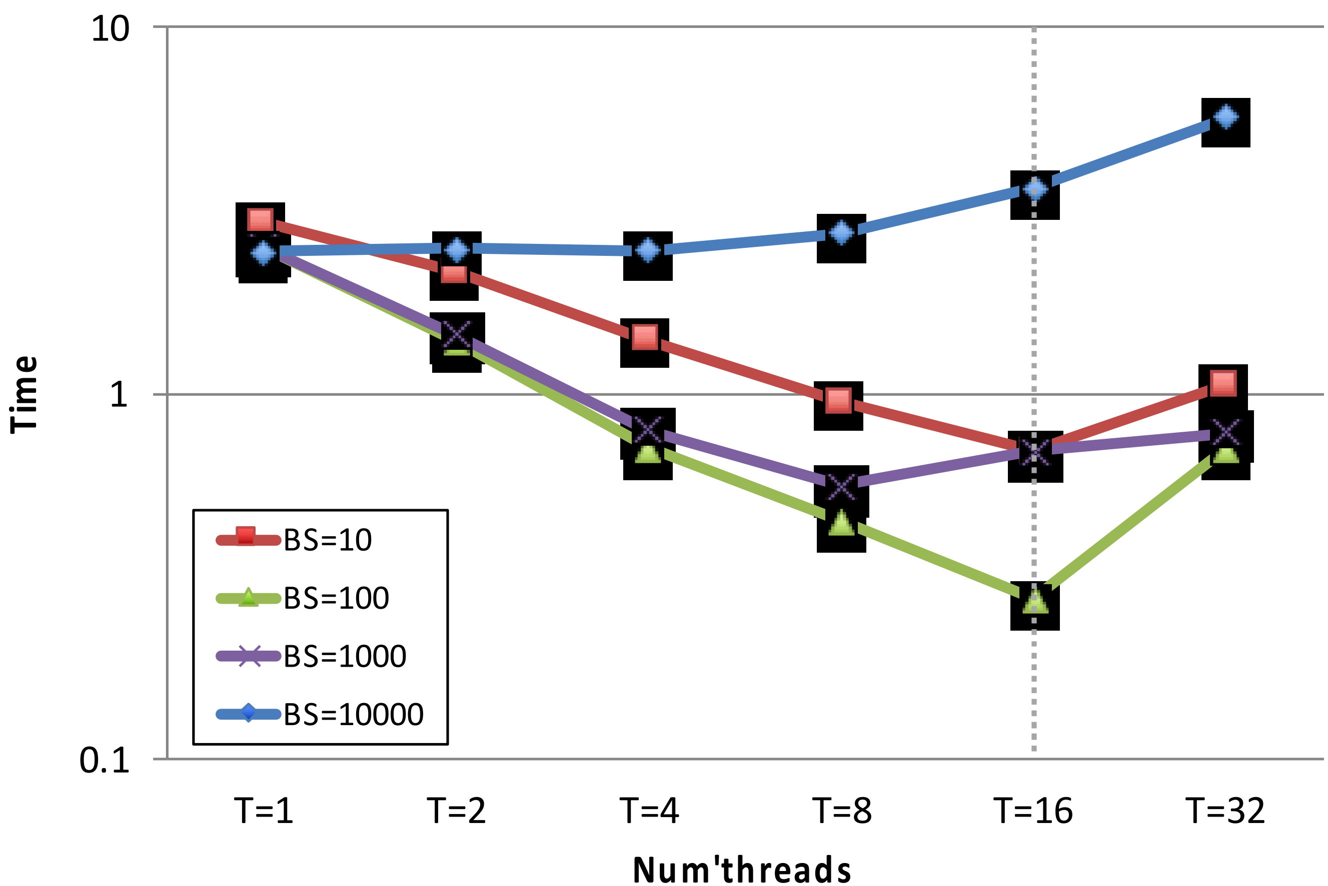
Kokkos/Qthreads miniTri Experiments

Graph	|V|	|E|	|T|	|T|/|E|
Oregon-1	11174	23409	19894	0.85
email-Enron*	36692	183831	727044	3.95
ca-AstroPh*	18772	198110	1352469	6.87
com-Youtube	1157827	2987624	3056386	1.02

- 16 core workstation with 64 GB of memory (dual processor Intel Xeon E5-2630 v3 @ 2.40 GHz)
- SNAP datasets (http://snap.stanford.edu)
  - Oregon-1: autonomous system dataset
  - email-Enron: Enron email network
  - ca-AstroPh: Arxiv Astro Physics collaboration network
  - com-Youtube: social network dataset

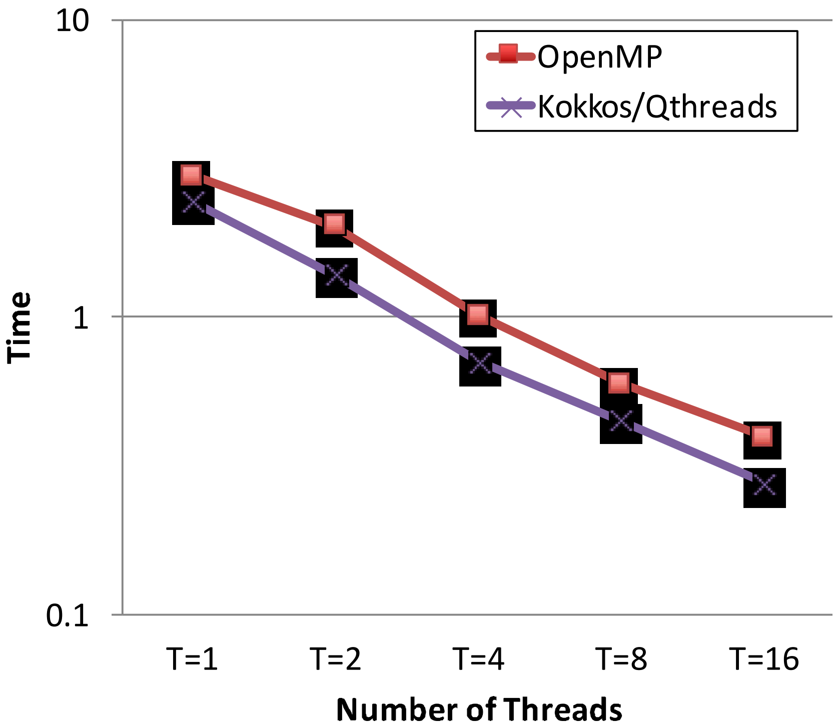
* University of Florida Sparse Matrix Collection

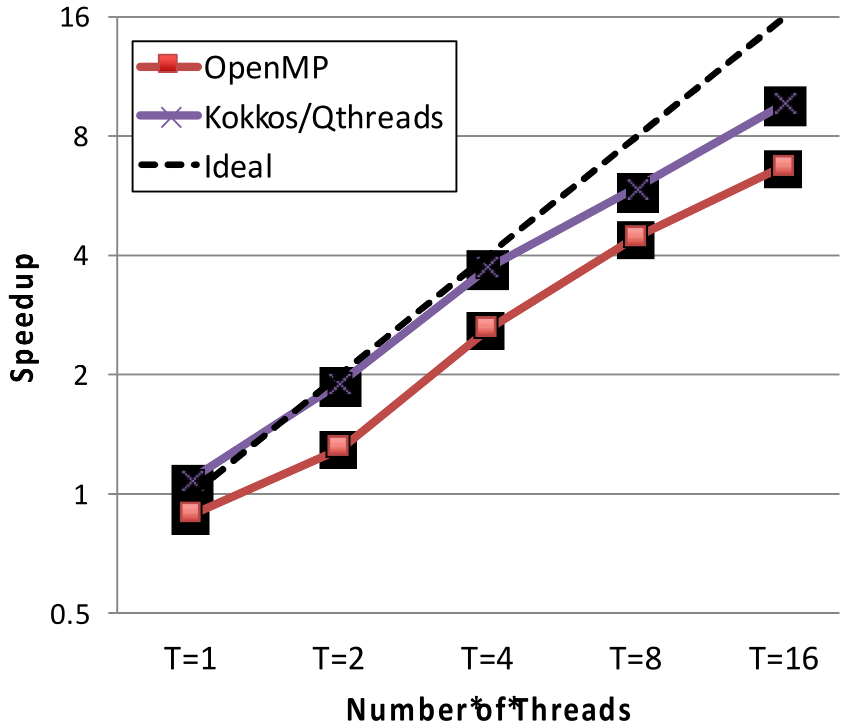
Oregon-1: Kokkos/Qthreads



Improvements in runtimes up to 16 threads for block sizes of 100

Oregon-1: Kokkos vs. OpenMP

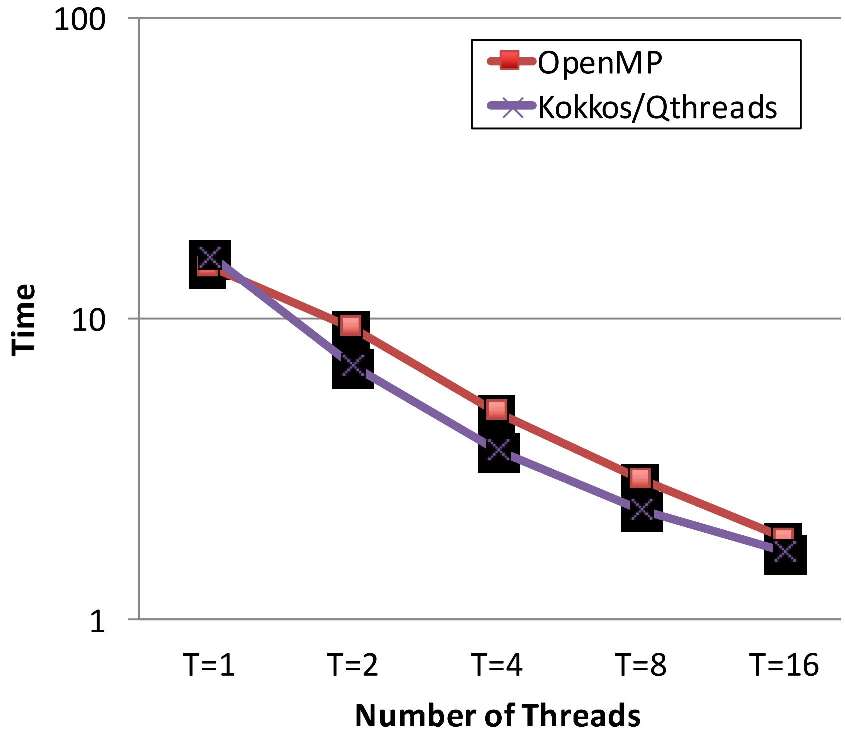


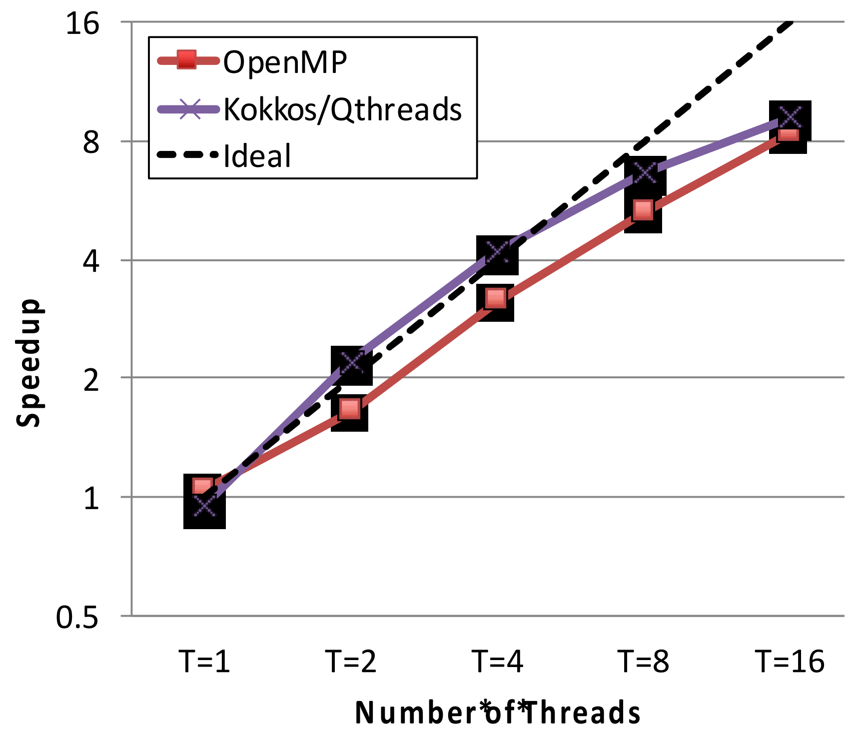


Kokkos/Qthreads performs significantly better than OpenMP

email-Enron: Kokkos vs. OpenMP

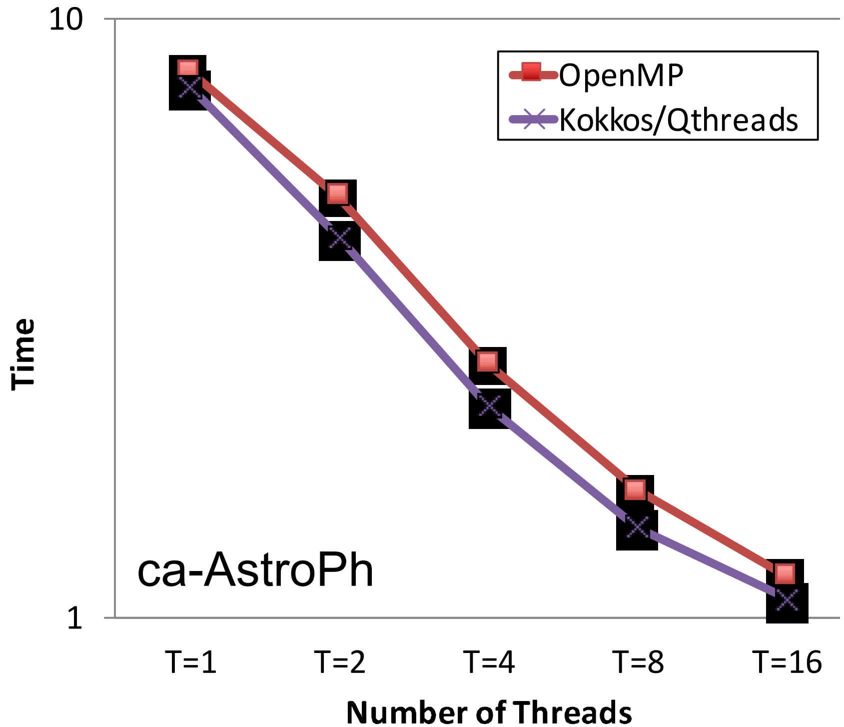
Sandia National Laboratories

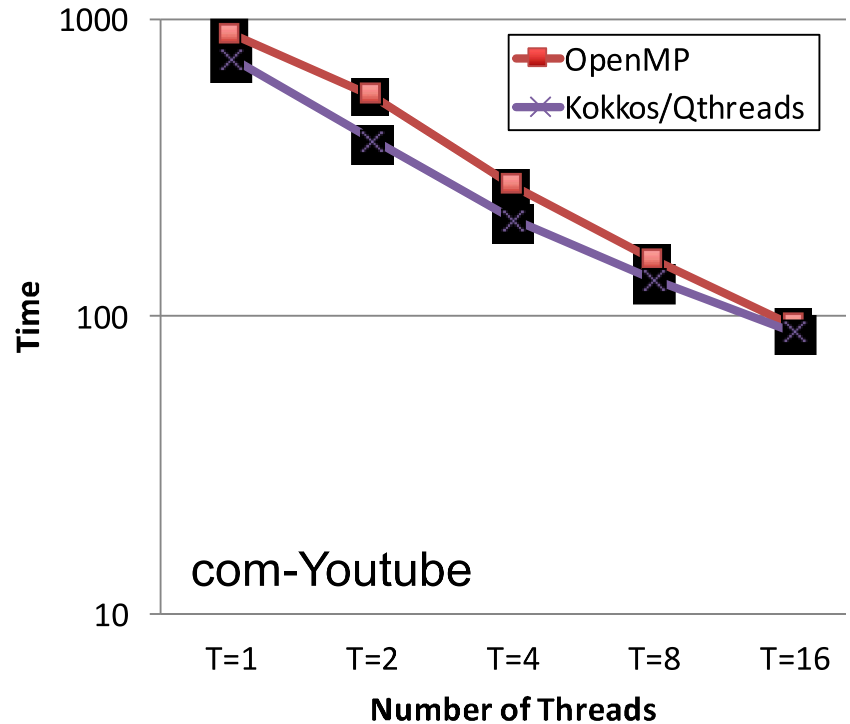




Kokkos/Qthreads performs slightly better than OpenMP

Kokkos vs. OpenMP: ca-AstroPh, com-Youtube





Kokkos/Qthreads performs slightly better than OpenMP

Outline

- Background
- miniTri
- Linear Algebra-Based miniTri (miniTriLA)
- Task Parallel Approach to miniTriLA
  - HPX
  - Kokkos/Qthreads
- Memory-Constrained Task Parallelism
- ➡ Summary

Summary/Conclusions

- Overview of new data analytics miniapp **miniTri**
  - Application relevant, released as part of Mantevo
- Presented linear algebra-based formulation of miniTri
  - miniTri in 4 compact linear algebra-based operations
  - Asynchronous, task parallel approach for constraining memory usage
- Described Kokkos/Qthreads task parallel implementation that outperforms data parallel (OpenMP) implementation
- Graph BLAS powerful approach for expressing graph algorithms
  - However, significant challenges exist for implementing certain graph applications efficiently (e.g., miniTri)
  - Linear algebra kernels should exploit **asynchrony** for more flexibility (e.g., solving larger problems) and better future performance – task parallelism helps here

Acknowledgements

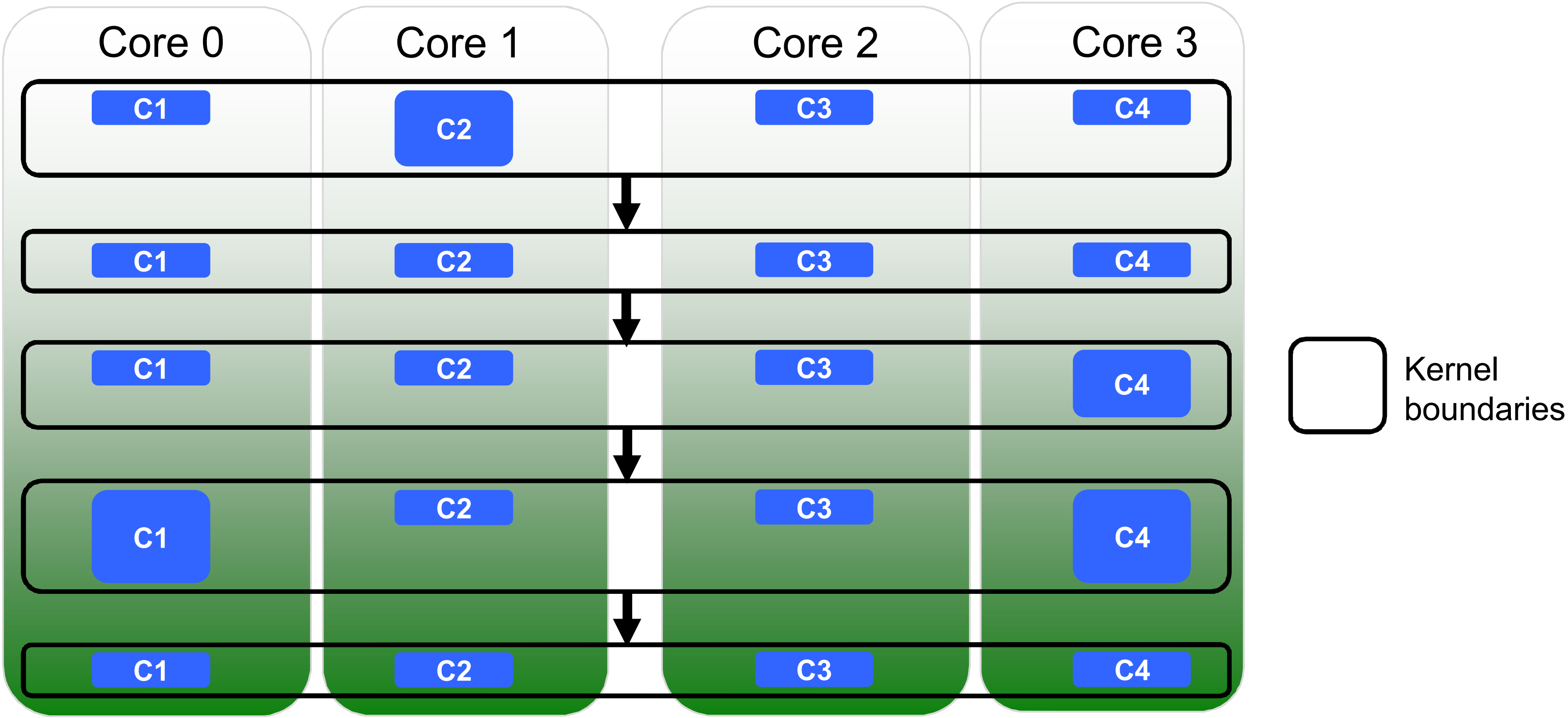
- Jon Berry (SNL)
  - Co-author of miniTri
- Carter Edwards (SNL)
  - Kokkos Lead
- Stephen Olivier (SNL)
  - Qthreads Lead
- Hartmut Kaiser, Daniel Bourgeois (LSU)
  - HPX

Additional Info/Resources

- miniTri now released under Mantevo repo
  - mantevo.org
  - Repo moving to github: **https://github.com/mantevo**
- Related publications
  - Wolf, Berry, Stark: "A Task-Based Linear Algebra Building Blocks Approach for Scalable Graph Analytics," *2015 IEEE HPEC*.
  - Wolf, Edwards, Olivier: "Kokkos/Qthreads Task-Parallel Approach to Linear Algebra Based Graph Analytics," *2016 IEEE HPEC* (to appear)
- Kokkos
  - https://github.com/kokkos
- Qthreads
  - https://github.com/Qthreads/qthreads

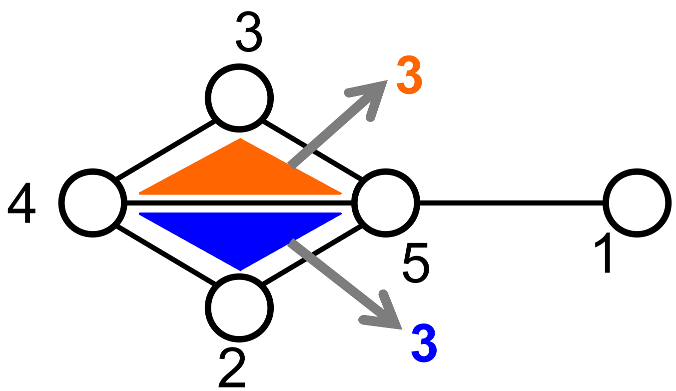
Extra

Traditional Data Parallelism



- **Data distributed across cores**
- **Global barriers between kernels**

kcount



k:

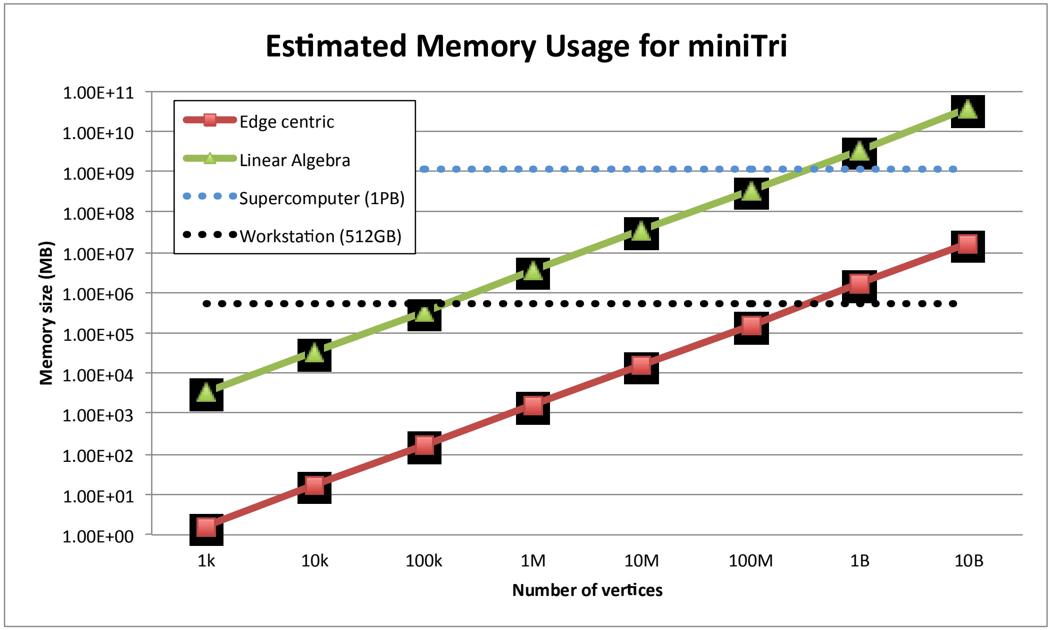
$$\arg\max_k \{(\min_{v \in t} t_v \geq \binom{k-1}{2}) \cap (\min_{e \in t} t_e \geq k-2)\}$$

- Upper bound on largest clique in graph = largest *c* such that Comb(*c*,3) triangles have *k*-counts at least *c*
  - Any v of *k*-clique, incident on Comb(*k*-1,2) triangles of that clique
  - Any e of *k*-clique, incident on *k*-2 triangles of that clique
  - argmax selects the largest *k* satisfying these condition (largest clique containing triangle)

miniTriLA: Challenges

miniTri

$C = A * B$
$t_e = C * 1$
$t_v = C^T * 1$
$kcount(C, t_e, t_v)$



- Challenge 1: Computation difficult to load-balance
- Challenge 2: Graph BLAS approach forms C, which means storing all triangles in graph
  - Worst case: $O(|E|^{3/2})$ triangles in graph, typical: 100-1000 triangles/edge
  - Severely limits size of graph

A = adjacency matrix for graph, B = incidence matrix for graph,
1 = vector of ones

Summary

- Overview of new data analytics miniapp **miniTri**
  - Will be released soon as part of Mantevo
- Presented linear algebra-based formulation of miniTri
  - miniTri in 4 compact linear algebra-based operations
  - Graph Algorithm Building Block (GABB) for triangle enumeration
  - GABBs for calculating triangle vertex and edge degree
- miniTri poses challenges for Graph BLAS-like implementations
  - Load balancing, Memory usage
- Presented task parallel approach that addresses these challenges
  - Asynchrony is key
  - Can use task priorities to constrain memory usage