SAND2018-13026C

Fast Triangle Counting Using Cilk

IEEE High Performance Extreme Computing Conference, HPEC'18
September 26, 2018

Abdurrahman Yaşar[1,2], Sivasankaran Rajamanickam[1], Michael Wolf[1], Jonathan Berry[1] and Ümit V. Çatalyürek[2]

[1]*Center for Computing Research, Sandia National Laboratories, Albuquerque*
[2]*Computational Science and Engineering, Georgia Institute of Technology*

. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and 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-NA-0003525.

Key Takeaways

KKTri was one of the "2017 Graph Challenge" champions. This work (KKTri-Cilk) is based on Cilk.

- KKTri-Cilk **surpasses 10^9 for the rate measure on a single multicore node** for the first time. Typical results for the fastest submissions in 2017 Graph Challenge were on the order of 10^8
  - Resolves scalability drawbacks of KKTri's OpenMP implementation for larger problems and hyperthreads usage.
- KKTri-Cilk is **faster than OpenMP implementation on 63 of 78 instances**
- KKTri-Cilk is also **faster than state-of-the-art graph based implementation** (up to 7x)
- We show that the **scalability of the triangle counting is bounded by O(n)** when the 4/3-moment is bounded
- We show correlation between the high rates achieved and the **conductance of the graph**

Overview

1. Approach
2. Experimental Evaluation
3. Conclusion

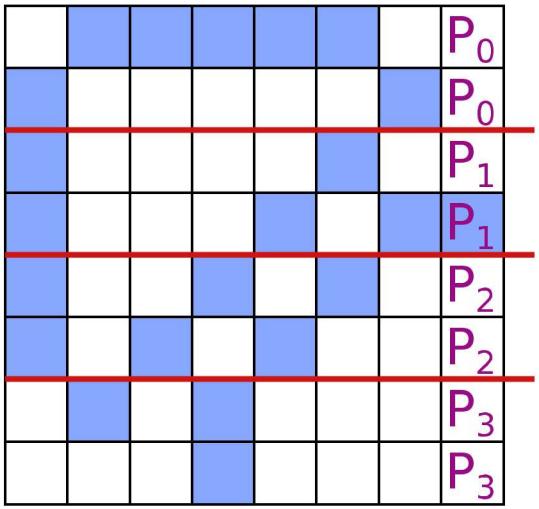
KKTri-Cilk Algorithm

Parallelization strategy and the runtime is the main difference between KKTri-Cilk and KKTri.

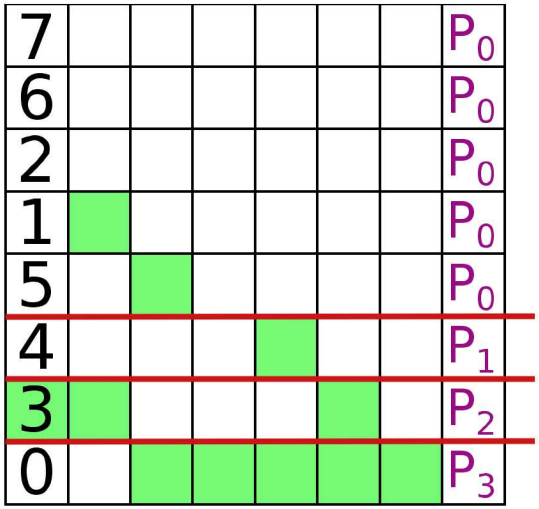
The outline of the KKTri-Cilk algorithm:

1. **Rows are ordered** based on number of non-zeros depending on the algorithm:
   - LL: Rows are sorted in decreasing order of degree
   - LU: Rows are sorted in increasing order of degree
2. Partitions' borders are decided by trying to **balance number of non-zeros** within them.
3. Each **partition is spawned (in parallel) as a task**.
   - Partitions are disjoint; they can be executed in parallel.
   - A task runs matrix matrix multiplication to count the number of triangles within a partition.

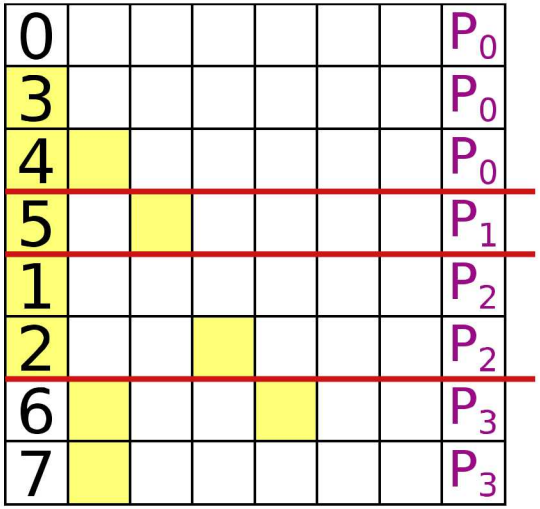
Matrix form



LL Algorithm (L matrix)



LU Algorithm (U matrix)



KKTri-Cilk Algorithm: Tasks and matrix multiplication

Each task does a matrix multiplication on its partition.

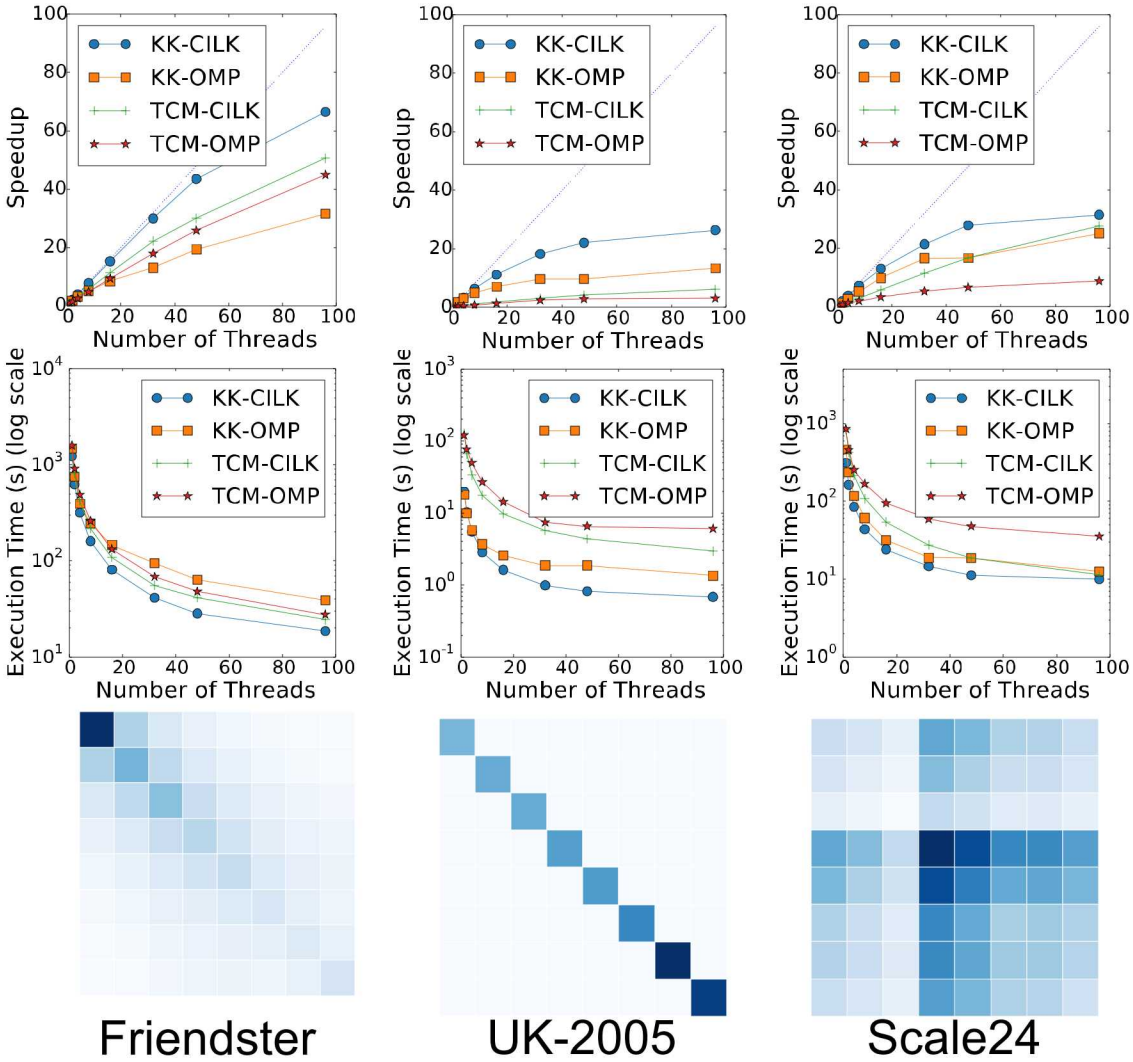
- **Linear algebra-based formulation**: sparse matrix-matrix multiplication followed by an element-wise matrix multiplication
- Using the **rightmost matrix as a mask** for the SpGEMM avoids explicitly storing all the non-zeros resulting from the multiplication.
- **Two linear-algebra based formulations** of triangle counting are implemented: $LU = (L \cdot U) .* L$ and $LL = (L \cdot L) .* L$.
- KKTri-Cilk implements an **in-place masking strategy** to reduce number of operations and also memory movement.

Experimental Setup

Three architectures with multicore processors. Intel compiler (icpc) version 18.1.163 is used to compile both KKTri and TCM codes. 2 hyperthreads for Skylake and Haswell architectures are used.

	Server 1	Server 2	Server 3
Code Name	Skylake	Haswell	KNL
Model	Intel Xeon Platinum 8160	Intel Xeon E7-4850 v3	Intel Xeon Phi 7250
Cores/Freq.	2 × 24/2.1GHz	4 × 14/2.2GHz	4 × 68/1.4GHz
Cache/Mem	33MB/196GB	35MB/2TB	1(MB)/16GB

Strong Scaling Experiments



KKTri-Cilk scales the best in all three problems

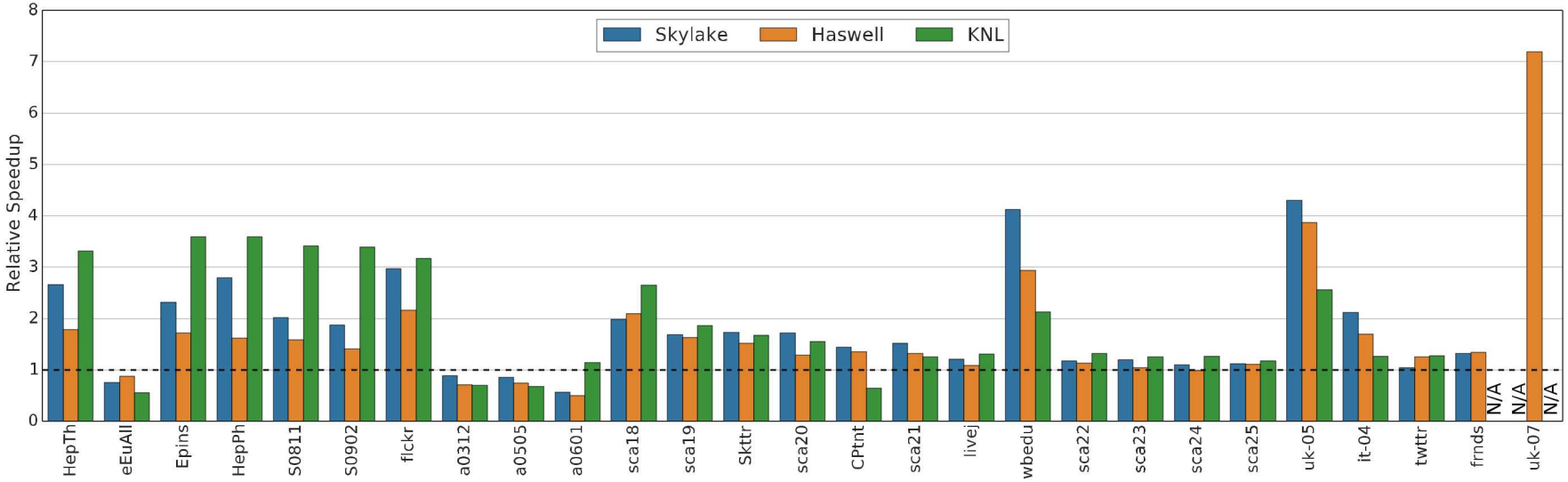
uk-2005 graph has a very good ordering: highly local computations (best rate).

scale24 graph has a very random distribution of the non-zeros. (Poor cache usage)

Friendster graph's distribution is in between (best scalability).

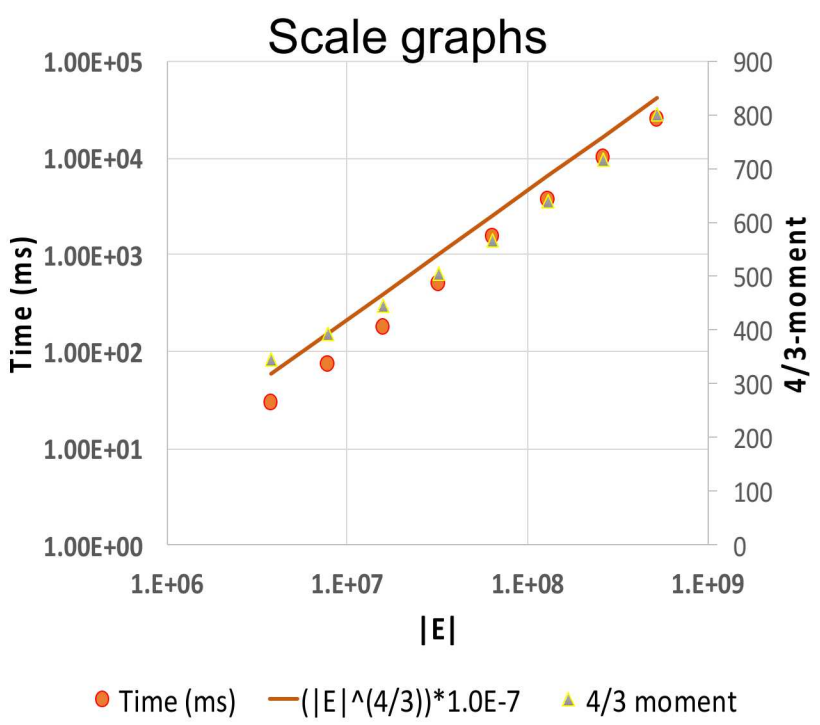
Scaling is with respect to the best sequential execution time.

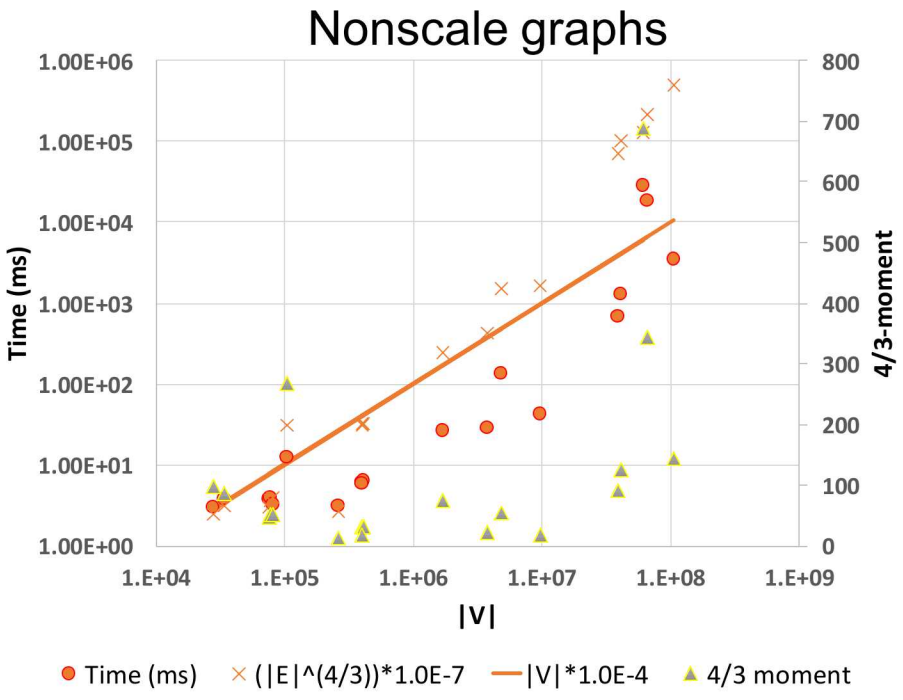
Relative Speedup Experiments



- Relative speedup in 3 architectures compared to TCM.
- KKTri outperforms TCM in 23 of 27 cases.
- KKTri can achieve up to 7x speedup on graphs that have a good natural ordering such as wb-edu, uk-2005, and uk-2007

Scalability of Triangle Counting





Correlation coefficient between triangle counting time per vertex/edge and the $E^{4/3}$ or 4/3-moment

Graph Type	Time-per-vertex		Time-per-edge	
	$E^{4/3}$	***4/3-moment***	$E^{4/3}$	***4/3-moment***
Scale graphs	0.90	***0.98***	0.89	***0.98***
Nonscale graphs	0.26	***0.95***	0.02	***0.76***

Conclusion

- KKTri-Cilk **surpasses 10^9 for the rate measure on a single multicore node** for the first time
- KKTri-Cilk is **faster than OpenMP implementation on 63 of 78 instances**
- KKTri-Cilk is also **faster than state-of-the-art graph based implementation** (up to 7x)
- We show that the **scalability of the triangle counting is bounded by O(n)** when the 4/3-moment is bounded
- We show correlation between the high rates achieved and the **conductance of the graph**

We plan to explore the use of the Cilk implementation for other algorithms such as “k-truss” computation.

Thanks!

- For more information
  - Email ayasar@gatech.edu srajama@sandia.gov
  - Visit tda.gatech.edu, www.sandia.gov/~srajama

Backup

Conductance and 4/3-moment

- Conductance
  - Conductance is defined as the ratio between the number of non-zeros in a partition where rows of their column indices appear in different partitions, and total number of non-zeros within that partition.
  - We use the average of the conductance of 16-way 7 partition to study the locality of a graph

- 4/3-moment
  - E[$d_v{}^{4/3}$] = $\frac{1}{n}\sum_v d_v{}^{4/3}$, where d_v is the degree of a vertex v, and *n* the number of vertices