Productive and extreme-scale graph computations enabled by GraphBLAS

Ariful Azad and Aydın Buluç, Computational Research Division, Lawrence Berkeley National Laboratory

Abstract

We develop several classes of graph algorithms using linear-algebraic (GraphBLAS) primitives. In-house combinatorial BLAS library enabled rapid development of bipartite graph matching, reverse Cuthill-McKee ordering, triangle counting, connected components and Markov clustering algorithms that scale to thousands of cores on modern supercomputers. These algorithms in turn empower key science applications including protein family detection and sparse linear solvers.

Motivation

❄ **Why graphs?** Graph computation drives many applications in biology and scientific computing.

❄ **Why GraphBLAS?** The diversity and rapid evolution of applications, architectures & algorithms motivates us to isolate a small number of graph kernels entrusted with delivering high-performance.

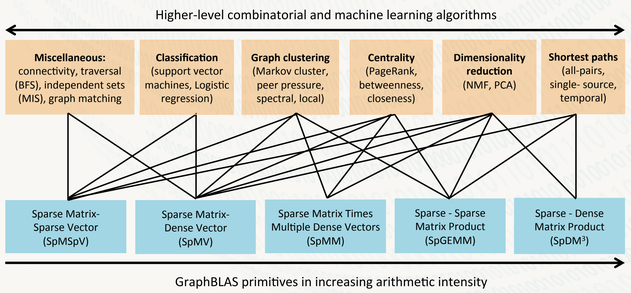
❄ **Expected impacts:** (a) better understanding of data and computational patterns, (b) rapid development of high-performance applications.

Approach

We develop graph and machine learning algorithms using linear-algebraic primitives. Two thrusts:

1. Develop communication-avoiding and work-efficient primitives (**Aydin's poster**)

2. Design algorithms using optimized primitives



Results

(1) Distributed-memory graph matching

- ✓ A suite of parallel algorithms developed. Sparse matrix-sparse vector multiply, inverted index used.
- ✓ Scales to several thousands of cores
- ✓ **Impact:** Remove the sequential-ordering bottleneck from SuperLU and STRUMPACK

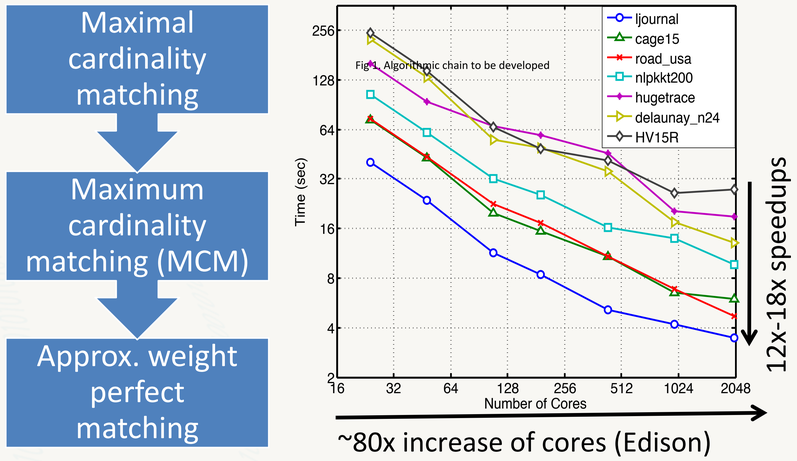
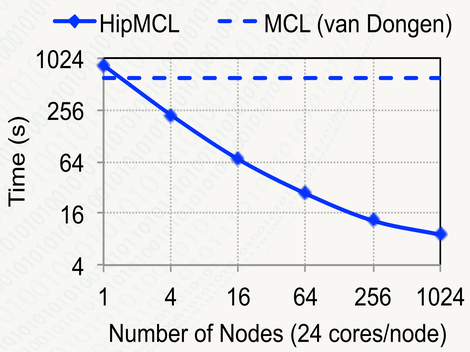


Fig. (a) Classes of algorithms, **(b)** performance of MCM

(2) Distributed-memory Markov clustering (HipMCL)

- ✓ Sparse matrix-matrix multiply, connected components, and k-select algorithms used.
- ✓ Scalable up to 136K cores on NERSC/Cori
- ✓ **Impact:** Reduced a 45-day clustering job to just an hour. Clustered massive networks with 70B edges.



|V|	|E|	HipMCL (cores)	MCL shm
69 M	12 B	1.66 hr (24K)	45 day
70 M	68 B	2.41 hr (136K)	X
282 M	37 B	3.23 hr (136K)	X

Fig. Performance of HipMCL w.r.t. a previous approach

Conclusions and Future Work

- ✓ GraphBLAS-based approach separates computational kernels from high-level algorithms & applications.
- ✓ Boosts the productivity of applications significantly.
- ✓ Resulted in highly-scalable algorithms for matching, ordering, connected components, maximal independent set and graph clustering.

Future directions and impacts:

❄ **Short term:** A GraphBLAS-compliant library with optimized in-node performance. Explore new domains.

❄ **Medium term:** Communication-avoiding algorithms for GraphBLAS primitives targeting future exascale systems. Enable new high-level algorithms.

❄ **Long term:** Provide easy-to-use graph libraries for biology, scientific computing & machine learning.

Areas in which we can help

Areas needing high-performance graph computation:

- ✓ Machine Learning
- ✓ Computational Biology
- ✓ Scientific computing
- ✓ Quantum computing

Areas in which we need help

Areas needing high-performance graph computation:

- ✓ Biology & other domains to understand applications
- ✓ Programming language and libraries (UPC, GASNet)
- ✓ Efficient FILE I/O (HDF5)

References

1. Azad, Buluç and Pothen. Computing maximum cardinality matchings in parallel on bipartite graphs via tree-grafting. TPDS, 2017.
2. Azad, Jacquelin, Buluç, and Ng. The reverse Cuthill-McKee algorithm in distributed-memory. IPDPS, 2017.
3. Azad, Ballard, Buluç, Demmel, Grigori, Schwartz, Toledo, & Williams. Exploiting multiple levels of parallelism in sparse matrix-matrix multiplication. SIAM Journal on Scientific Computing, 2016.
4. Azad and Buluç. Distributed-memory algorithms for maximum cardinality matching in bipartite graphs. IPDPS, 2016.
5. Azad, Buluç and Gilbert. Parallel triangle counting and enumeration using matrix algebra. IPDPSW 2015.