Progress Report on the Early-Career Research Project: Energy-Efficient Parallel Graph and Data Mining Algorithms

Principal Investigator: Dr. Aydın Buluç
Lawrence Berkeley National Laboratory
One Cyclotron Road, MS 50A-1148
Berkeley, CA 94720
abuluc@lbl.gov

Reporting Period: 10/14-9/15

Data are fundamental sources of insight for experimental and computational sciences. The Department of Energy acknowledges the challenges posed by fast-growing scientific data sets and more complex data. The graph abstraction provides a natural way to represent relationships among complex fast-growing scientific data sets. On future exascale systems, power consumption is of primary concern yet existing graph algorithms consume too much energy per useful operation due to their high communication costs, lack of locality, and inability to exploit hierarchy. This project explores methods to increase the energy efficiency of parallel graph algorithms and data mining tasks. A new family of algorithms will be developed to drastically reduce the energy footprint and running time of the graph and sparse matrix computations that form the basis of various data mining techniques. This project will also exploit the well-known duality between graph and sparse matrices to develop communication-avoiding graph algorithms that consume significantly less power. This project is relevant to DOE mission- critical science including bioinformatics and genomics with particular emphasis on plant genomics that can result in better biofuels through efficient genetic mapping, climate science where recent graph-based methods show increased accuracy in hurricane predictions, and combustion science where graph search techniques are used to analyze extreme-scale simulation data.

1 Summary

The early-career research project has made important progress during FY 2015. Specifically, our accomplishments include:

- Development of novel higher-level communication-avoiding matrix kernels for graph algorithms (detailed in Section 3.1).
- New formulations of graph algorithms in the language of linear algebra, as described in Section 3.2.
- Standardization of linear-algebraic building blocks for graph computations (GraphBLAS), as described in Section 3.3.
- Parallel data and graph analytics kernels for computational genomics. This is in collaboration with MANTISSA (ASCR - Applied Math) and DEGAS (ASCR - Computer Science) projects, and the details are explained in Section 3.4.

DOE relevance of each research accomplishment is mentioned within the corresponding section. Our next year plans to further advance both of these research areas, as well as our plans to expand our research into performing low-rank decompositions, such as non-negative matrix factorization, in parallel. Further details can be found in Section 4. Overall, our project so far resulted in six peer-reviewed publications [11, 13, 14, 2, 4, 3], and one invited article [17]. In addition, we have two papers under review in the top journals (IEEE Transactions on Parallel Computing and SIAM Journal of Scientific Computing).

The report concludes with lists of awards, software artifacts, presentations, community service, and references. In particular, Dr. Buluç was one of the three recipients of the *IEEE TCSC Award for Excellence for Early Career Research* (awarded by the IEEE Technical Committee on Scalable Computing) in 2015. Dr. Buluç was also an invited speaker in the IEEE HPEC conference.

2 Personnel

The project is lead by Aydın Buluç, who spends about 60% of his time on this project, on average. In addition, Ariful Azad, a postdoctoral fellow, works full time (100%) on this project. We also temporarily hired an undergraduate student, Chaitanya Aluru, from UC Berkeley, to help with the implementation of parallel algorithms for identifying heavy hitter (an important kernel used in the HipMER genome assembler pipeline (Section 3.4) to address the load imbalances that arise in the analysis of complex repetitive genomes). We plan to keep him part time during this academic year as well, in order to work with him on parallel non-negative matrix factorization (subject to the funding limitations).

3 Progress and Accomplishments

3.1 Communication-Avoiding Matrix Kernels for Graph Algorithms

Sparse matrix-matrix multiplication (SpGEMM) enables efficient parallelization of various graph algorithms. It is the workhorse of a scalable distributed-memory implementation of betweenness centrality, an algorithm that finds influential entities in networks. It is also the most time-consuming step (i.e. Galerkin triple product) of the algebraic multigrid setup phase at large concurrencies. The linear scaling electronic structure calculations, which are exponentially faster than their dense counterparts, also rely on SpGEMM [7].

Existing parallel algorithms for SpGEMM spend the majority of their time in inter-node communication on large concurrencies [10]. Earlier in this project, we had previously investigated communication-optimal algorithms for SpGEMM, and had presented new communication-optimal algorithms that both attain the communication lower bounds [5]. In FY15, we developed the high-performance implementation of a 3D SpGEMM algorithm, which outperforms all the existing implementations at large scale.

Our implementation exploits inter-node parallelism within a third processor grid dimension as well as thread-level parallelism within the node. It achieves higher performance compared to other available formulations of distributed-memory SpGEMM, without compromising flexibility in the numbers of processors that can be utilized. In particular, by

varying the third processor dimension. as well as the number of threads, one can run our algorithm on many processor counts.

The percentage of time spent in communication (data movement) is significantly lower in our new implementation compared to a 2D implementation. This is advantageous for multiple reasons. First, the bandwidth for data movement is expected to increase at a slower rate than other system components, providing a future bottleneck. Second, communication costs more energy than computation [18, Figure 5].

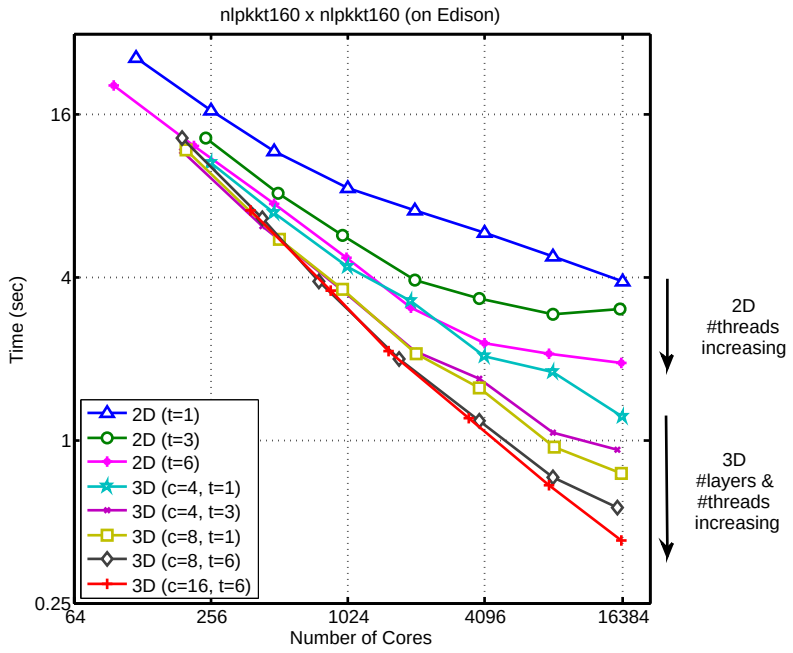

Figure 1: Weak scaling of sparse matrix-matrix multiplication on NERSC Edison, a Cray XC30 using the nlpkkt160 matrix from the University of Florida Sparse Matrix Collection. Performance benefits of the 3D algorithm and multithreading can be realized on higher concurrency.

Figure 1 shows strong scaling results for the 3D iterative SpGEMM algorithm on NERSC Edison when multiplying the nlpkkt160 matrix from the University of Florida Sparse Matrix Collection by itself. Due to the fill-in, the output matrix is denser. The number of grid layers, c, refers to the third dimension of the processor grid. For $c = 1$, the algorithm degenerates into a 2D algorithm. The number of threads used per MPI process is designated with t. The paper including these results are submitted to the SIAM Journal of Scientific Computing and its preprint is available online [1]. We expect to improve on these results via overlapping communication with computation.

The same paper also includes new multicore algorithms that exploit the increasing intra-node parallelism. A new implementation of the column-by-column algorithm of Buluç and Gilbert [9] shows good intra-node scaling. Figure 2 shows the scaling of this new implementation in comparison to MKL on inputs of (a) R-MAT graphs of scale 16 (i.e. matrices of dimensions $2^{16} \times 2^{16}$, each having approximately 16 nonzeros per column/row, following

the exact parameters of the Graph500 benchmark) on a single node of NERSC/Edison, and (b) a real sparse matrix from the Florida collection.

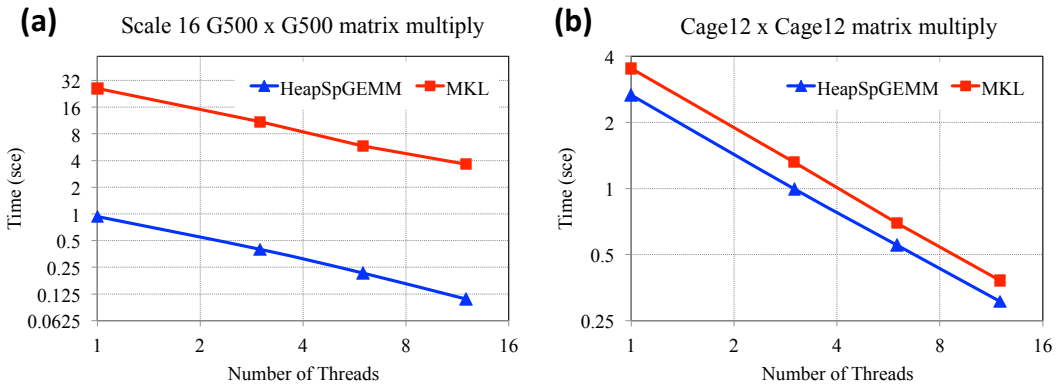

Figure 2: Scaling of multithreaded SpGEMM algorithm versus MKL

3.2 New graph algorithms in the language of linear algebra

Graph matching algorithms: We have been applying the techniques developed during our earlier parallel breadth-first search (BFS) work as described in the FY14 report, to more complex problems such as bipartite graph matching, with considerable success.

Given the limited scalability of existing graph matching algorithms, we investigated new algorithmic paradigms, such as utilizing the matrix-based primitives and using novel techniques that eliminate redundancies that are artifacts of algorithmic parallelism. We first developed a new multi-source maximum-cardinality matching algorithm that searches for a set of vertex-disjoint augmenting paths via specialized BFS searches from all unmatched vertices. This algorithm employs a novel *tree-grafting* method that eliminates most of the redundant edge traversals by grafting a part of a search tree that yields an augmenting path onto another search tree from which we have not found an augmenting path [4]. The shared memory implementation of the tree-grafting algorithm scales up to 80 threads of an Intel multiprocessor (Westmere-EX) and is 10 times faster than current state-of-the-art matching algorithms for several important classes of graphs.

Despite good scalability of matching algorithms on shared memory multiprocessors, it remains a challenge to achieve high performance on distributed platforms because these algorithms rely on searching for paths in the graph, and when few long paths pass through multiple processors, there is little concurrency. In this endeavor, we design matching algorithms on sparse graphs in terms of linear algebra routines such as sparse matrix-vector multiplication (SpMV) and other operations on sparse and dense vectors. We used existing functionalities of the Combinatorial BLAS (CombBLAS) library [8] and added several new functions to the library so that we can express the matching algorithms completely by a small number of functions from CombBLAS. By using these functions we implemented several cardinality matching algorithms, with impressive results [2]. The same matrix-algebraic framework is used to implement three different algorithms with minimal modifications to the underlying semiring operations and data structures. The strong scaling results of computing

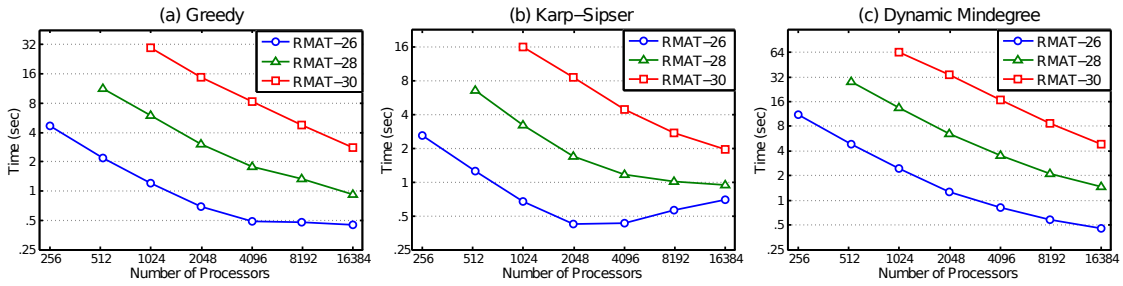

Figure 3: Strong scaling of maximal matching algorithms on RMAT graphs.

maximal cardinality matchings on synthetically generated graphs that follow a power-law degree distribution is shown in Figure 3.

In terms of DOE impact, this provides a first step towards a scalable solution to the distributed cardinality and weighted matching problems, because maximal cardinality matching is a subroutine used by both maximum cardinality and maximum weighted matching algorithms. In our primary application area of sparse linear solvers, the maximum cardinality matching can be applied to permute a matrix to its block triangular form, potentially leading to faster linear solvers.

Triangle counting algorithms: Triangle counting and enumeration are important kernels that are used to characterize graphs. They are also used to compute important statistics such as clustering coefficients. We provided a simple exact algorithm that is based on operations on sparse adjacency matrices. By parallelizing the individual sparse matrix operations, we achieved a parallel algorithm for triangle counting. The algorithm is generalizable to triangle enumeration by modifying the semiring that underlies the matrix algebra. We presented a new GraphBLAS primitive, *masked matrix multiplication*, that can be beneficial especially for the enumeration case. We provide results from an initial implementation for the counting case along with various optimizations for communication reduction and load balance. The preliminary results from this work (which we plan to extend in FY16) is published [3].

3.3 The GraphBLAS kernels

The GraphBLAS is an effort to define standard building blocks for graph algorithms in the language of linear algebra. Dr. Buluç co-leads a multi-institutional group of researchers that are working on defining the right primitives and providing initial reference implementations. So far, we have run monthly telecons among key 4-5 people until May 2015, which was followed by a face-to-face meeting at LBL on API design that was attended by ≈ 30 people on June 2015 [16]. Since then, Dr. Buluç has been leading weekly teleconferences with key players from MIT Lincoln Labs, IBM Research, Indiana University, Intel, Georgia Tech, and many others. We have successfully ran the first peer-reviewed Graph Algorithms Building Blocks (GABB) workshop at IPDPS'15. The call for papers for next version is currently out at `http://www.graphanalysis.org/workshop2016.html`.

GraphBLAS will include the reduced-communication algorithms described in this report. The GraphBLAS effort has implications for exascale as well: graph algorithms will need to be redesigned to address the joint issues of complexity, resilience, and scalability. The

translation of every individual graph algorithm to exascale platforms will be prohibitively difficult due to the complexity of hardware architectures and the diversity of graph operations. Primitives allow algorithm designers to think on a higher level of abstraction, and reduce duplication of implementation efforts. Our conjecture is that with the large body of experience with sparse linear algebra in extreme scale computing, the basic graph algorithm primitives we have proposed will be an effective way to manage the exascale challenge. In particular, if we can manage the exascale issues (scalability and reliability) inside the primitives, then graph algorithm developers can explore algorithms for these machines using the same approaches used on modest sized parallel systems. This would help assure that the graph-based software needed for these machines exists as exascale computers emerge.

More about our progress can be found in the GraphBLAS website [15]. Dr. Buluç has also co-authored an invited paper on case for GraphBLAS [17]. Our team develops new algorithms for GraphBLAS kernels and evaluates relative merits of different approaches. Sparse matrix-matrix multiplication (SpGEMM), which was previously described in this report, is a key GraphBLAS kernel and perhaps the most challenging one.

3.4 Genome assembly

We have used high-performance graph analysis and data engineering to address the challenges of large-scale genome assembly. Recent advances in sequencing technology have made the redundant sampling of genomes extremely cost-effective. Such a sampling consists mostly of shorts reads with low error rates most of which can be aligned to a reference genome in a straightforward way. De novo genome assemblers, on the other hand, reconstruct an unknown genome from a collection of short reads, without requiring a reference genome. De novo assembly is therefore more powerful and flexible than mapping-based approaches, since it can be applied to novel genomes and can in principle discover previously unknown sequences in species for which reference genomes do not exist. This advantage, however, comes at a cost of significantly increased runtime and memory requirements. If de novo assembly could be performed rapidly, it would be preferable to mapping based approaches for large-scale human genome sequencing, and other biomedical model and agricultural species.

The components of the de novo genome assembly pipelines have very different execution profiles and unique challenges. The initial processing and data reduction (k-mer analysis) is bandwidth-bound, the subsequent construction and traversal of the *de Bruijn* graph are latency-bound, and the sequence alignment is compute bound. Many of the graph processing techniques developed for low-diameter graphs are not applicable to de Bruijn graphs, which have very low degree and a high diameter.

Meraculous is a hybrid k-mer/read-based whole genome assembler that avoids explicit error correction steps. While Meraculous is one of the best assemblers available in terms of accuracy, it is known to be too slow for large genomes. HipMer is our high-performance re-engineering of Meraculous for distributed-memory supercomputers, designed and implemented from scratch using advanced data structures and Unified Parallel C (UPC)'s remote atomics capabilities. This year, we completed the HipMer pipeline beyond last year's advances in parallel construction and traversal of the de Bruijn graph. We provided the first complete parallelization of the contigs-to-reads alignment step, including the seed index

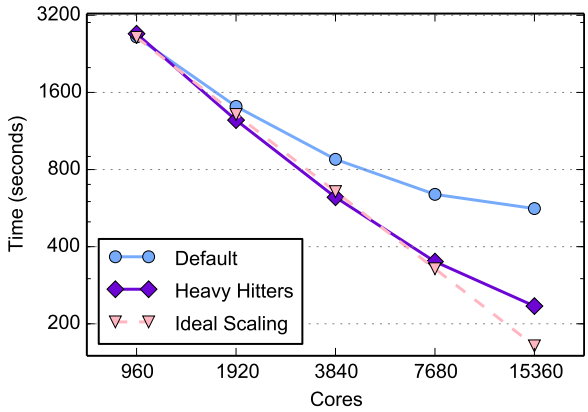

Figure 4: Strong scaling of k-mer analysis on wheat, showing the effect of the heavy hitters (high frequency k-mers) optimization.

(the database) construction step. This work has been published in IPDPS [14], a premier conference in parallel computing.

Recently, we completed the remaining steps of the HipMer pipeline. We also developed new methods that can efficiently handle complex repetitive genomes with skewed k-mer distributions that normally create load imbalance for naive approaches. We utilized state-of-the-art algorithmic ideas from big data analysis such as finding heavy hitters (very high-frequency items) with minimal communication in a single streaming pass. The effect of this optimization is shown in Figure 4.

HipMer can assemble large genomes such as human and wheat in the matter of minutes compared to days that original Meraculous took. For example, HipMer assembles the entire human genome end-to-end in about eight minutes using 15,360 computer processor cores [13]. With this tool, we estimate that the output from the world's biomedical sequencing capacity could be assembled using just a portion of NERSC's Edison supercomputer. Additional, as described in the same paper that will appear in SC'15 [13], we optimize the traversal of the de Bruijn graph of k-mers by employing a novel communication-avoiding parallel algorithm in a variety of practical use-case scenarios, such as the assembly of closely related genomes. Recall that the computational motif for de Bruijn graph traversal is to pick a random traversal seed and expand the connected components via consecutive lookups to the distributed hash table. Since we have no apriori knowledge about the graph structure, this leads to poor locality and fine grained remote accesses. In the extreme, the result can be a single very long chain (very high diameter graph), leading to very high communication costs. If we had an oracle partitioning function that could tell as ahead of time how the k-mers are placed into contigs, one could partition the k-mers in such a way that k-mers belonging eventually in the same contig are mapped to the same processor, which is a similar idea to graph partitioning. Our oracle is the genetic similarity between genomes that need to be assembled. For example, the human nucleotide diversity is estimated to be 0.1% to 0.4% of base pairs. This intuition is the base of our *communication-avoiding algorithm.*

We completed our end-to-end de-novo genome assembly pipeline, called HipMer, which also includes various performance improvements [13]. Figure 5 shows the end-to-end strong scaling performance of HipMer on the human (left) and the wheat (right) data sets. For the

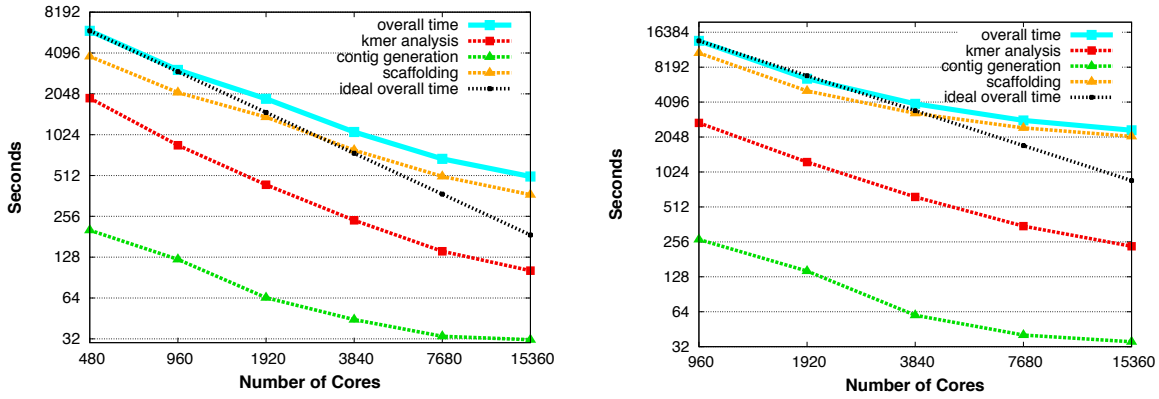

Figure 5: **End-to-end strong scaling** for (left) human genome and (right) wheat genome. Both axes are in log scale.

human dataset at 15,360 cores we achieve a speedup of 11.9× over our baseline execution (480 cores). At this extreme scale the human genome can be assembled from raw reads in just ≈ 8.4 minutes. On the complex wheat dataset, we achieve a speedup up to 5.9× over the baseline of 960 core execution, allowing us to perform the end-to-end assembly in 39 minutes when leveraging 15,360 cores. In the end-to-end experiments, a significant fraction of the overhead is spent in parallel scaffolding (e.g. 68% for human at 960 cores); k-mer analysis requires significantly less runtime (28% at 960 cores) and contig generation is the least expensive computational component (4% at 960 cores).

These advances on de novo genome assembly enabled the whole-genome assembling and anchoring of the hexaploid bread wheat genome, without using expensive chromosome separation technologies. This work [11] is considered a *breakthrough* because polyploid genomes (those that have more than two paired sets of chromosomes) were considered beyond the reach of methods that solely rely on whole-genome shotgun sequencing data. A review article of this work by independent authors praise this work by saying "The combination of whole genome shotgun sequencing and linkage mapping by skim sequencing produced a better genome assembly than both the chromosome arm-based assembly and a previously described whole genome".

4 Plans for the next fiscal year

Metagenome assembly: De novo assembly of large complex eukaryote genomes has been a computational challenge where our team had successfully applied the state-of-the-art techniques in high-performance parallel computing. A metagenome, a microbial community recovered directly from an environmental sample, on the other hand, is like a eukaryote with 10,000 very highly polymorphic chromosomes sampled at widely differing depth. This fact makes de novo assembly of metagenomes algorithmically more challenging as errors in high-depth regions can occur more frequently than true DNA samples in low-depth regions. If assembling a single genome is analogous to piecing together one novel, then assembling metagenomic data is like rebuilding a library. We plan to work on developing new parallel

algorithms for de novo assembly of metagenomic data sets, building on the techniques we developed for our HipMer assembler as much as possible. This research requires new algorithmic techniques and a change in the way assemblers currently run. We need new high-performance assemblers that can handle metagenomic samples with uneven depth.

Graph matching algorithms: Using the maximal cardinality matching algorithm as a building, we will develop successively more complex matching algorithms. This year, we plan to develop distributed-memory cardinality matching algorithms; which can later be used as a subroutine for bipartite maximum weighted-matching algorithms (MWM). MWM algorithms have tremendous importance in the scalability of sparse direct solvers.

Non-negative Matrix Factorization: Non-negative Matrix Factorization (NMF) is a dimension reduction technique used in a wide variety of fields ranging from chemometrics to bioinformatics to machine learning. NMF is a non-convex optimization problem, and as such it is computationally infeasible to compute a global solution. Many algorithms have been proposed to solve the problem, but none are able to handle the large scale datasets common in modern science. We consider a fast NMF algorithm based on the alternating non-negatively constrained least squares (ANLS). We will develop parallel versions of this algorithm and examine its runtime both theoretically and experimentally. We also plan to test this implementation on a large-scale genomic dataset from JGI, with the goal of separating contaminated (belongs to multiple organisms) contigs from clean (belongs to a single organism) contigs.

Awards

Aydın Buluç, IEEE TCSC Award for Excellence for Early Career Research by the IEEE Technical Committee on Scalable Computing, 2015

Software artifacts

Combinatorial BLAS: Both Dr. Buluç and Dr. Azad contributed significantly to the development of the Combinatorial BLAS (CombBLAS) library [8, 12] along both the functionality and the performance axes. We have not made a new release this year due to the unusually large number of new features added. Those features include complete in-node multithreading support, experimental support for 3D distributions, a new maximal matching algorithm implementation as well as the necessary auxiliary functions. We plan to release a new version by December 2015 after all the new functionality is thoroughly tested.

HipMer: HipMer is an end-to-end high performance de novo assembler designed to scale to massive concurrencies. It is based on the Meraculous assembler, which has been shown to produce high quality assembly results. Large-scale results on the Cray XC30 supercomputer demonstrate efficient performance and scalability on thousands of cores. The permission to distribute software has been given by LBL and will include (by SC'15 timeframe) both the full HipMer assembly pipeline, as well as an application performance benchmark implementing a limited subset of the latter functionality. More on HipMer can be found in the following news article: `http://cs.lbl.gov/news-media/news/2015/meraculous-deciphering-the-book-of-life-with-supercomputers/`.

Invited Talks

Faster Parallel Graph BLAS Kernels and New Graph Algorithms in Matrix Algebra

- IEEE High Performance Extreme Computing Conference (HPEC), Sep 16, 2015
- HP Labs (Host: Robert Schreiber), August 18, 2015

Scalable algorithms for complex genome assembly, alignment, and genetic mapping

- School of CSE (Host: Srinivas Aluru), Georgia Tech, Jan 16, 2015

Distributed-Memory Parallel Algorithms for Graph Traversal & Genome Assembly

- Dept. CS&E (Host: Murat Demirbas), University at Buffalo, Dec 5, 2014
- Dept. CS (Host: Petko Bogdanov), University at Albany, Dec 4, 2014
- Dept. CS (Host: Leman Akoglu), Stony Brook University, Dec 3, 2014

Community Service (PI only)

- *Associate Editor*: ACM Transactions on Parallel Computing
- *Guest Editor*: Parallel Computing, special issue on "Graph Analysis for Scientific Discovery"
- *Program Committee*: IEEE Int. Parallel & Dist. Processing Symp. (IPDPS), 2015
- *Program Committee*: International Conference on Supercomputing (ICS), 2015
- *Steering & Program Committee*: Graph Algorithms Building Blocks (GaBB), IPDPS workshop, 2015.

References

[1] A. Azad, G. Ballard, A. Buluç, J. Demmel, L. Grigori, O. Schwartz, S. Toledo, and S. Williams. Exploiting multiple levels of parallelism in sparse matrix-matrix multiplication. *arXiv preprint arXiv:1510.00844*, 2015.

[2] A. Azad and A. Buluç. Distributed-memory algorithms for maximal cardinality matching using matrix algebra. In *IEEE International Conference on Cluster Computing (CLUSTER)*, 2015. (full paper).

[3] A. Azad, A. Buluç, and J. R. Gilbert. Parallel triangle counting and enumeration using matrix algebra. In *Proceedings of the IPDPSW, Workshop on Graph Algorithm Building Blocks (GABB)*, 2015.

[4] A. Azad, A. Buluç, and A. Pothen. A parallel tree grafting algorithm for maximum cardinality matching in bipartite graphs. In *Proceedings of the IPDPS*, 2015.

[5] G. Ballard, A. Buluç, J. Demmel, L. Grigori, B. Lipshitz, O. Schwartz, and S. Toledo. Communication optimal parallel multiplication of sparse random matrices. In *SPAA 2013: The 25th ACM Symposium on Parallelism in Algorithms and Architectures*, Montreal, Canada, 2013.

[6] S. Beamer, A. Buluç, K. Asanović, and D. Patterson. Distributed memory breadth-first search revisited: Enabling bottom-up search. In *Proceedings of the IPDPSW*. IEEE Computer Society, 2013.

[7] U. Borštnik, J. VandeVondele, V. Weber, and J. Hutter. Sparse matrix multiplication: The distributed block-compressed sparse row library. *Parallel Computing*, 40(5):47–58, 2014.

[8] A. Buluç and J. R. Gilbert. The Combinatorial BLAS: Design, implementation, and applications. *The International Journal of High Performance Computing Applications*, 25(4):496 – 509, 2011.

[9] A. Buluç and J. R. Gilbert. On the representation and multiplication of hypersparse matrices. In *IPDPS'08: Proceedings of the 22nd IEEE International Symposium on Parallel&Distributed Processing*, pages 1–11. IEEE Computer Society, 2008.

[10] A. Buluç and J. R. Gilbert. Parallel sparse matrix-matrix multiplication and indexing: Implementation and experiments. *SIAM Journal of Scientific Computing*, 34(4):170 – 191, 2012.

[11] J. Chapman, M. Mascher, A. Buluç, K. Barry, E. Georganas, A. Session, V. Strnadova, J. Jenkins, S. Sehgal, L. Oliker, J. Schmutz, K. Yelick, U. Scholz, R. Waugh, J. Poland, G. Muehlbauer, N. Stein, and D. Rokhsar. A whole-genome shotgun approach for assembling and anchoring the hexaploid bread wheat genome. *Genome Biology*, 16(26), 2015.

[12] Combinatorial BLAS, release v1.4.0, January 2014. `http://gauss.cs.ucsb.edu/~aydin/CombBLAS`.

[13] E. Georganas, A. Buluç, J. Chapman, S. Hofmeyr, C. Aluru, R. Egan, L. Oliker, D. Rokhsar, and K. Yelick. HiPMer: An extreme-scale de novo genome assembler. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'15)*, 2015.

[14] E. Georganas, A. Buluç, J. Chapman, L. Oliker, D. Rokhsar, and K. Yelick. meraligner: A fully parallel sequence aligner. In *Proceedings of the IPDPS*, 2015.

[15] The Graph BLAS Forum, September 2015. `http://graphblas.org/`.

[16] GraphBLAS API meeting, June 2015. `https://sites.google.com/a/lbl.gov/graphblas-api-meeting/`.

[17] J. Kepner, D. Bader, A. Buluç, J. Gilbert, J. Kepner, T. Mattson, and H. Meyerhenke. Graphs, matrices, and the GraphBLAS: Seven good reasons. In *The International Conference on Computational Science (ICCS)*, 2015.

[18] P. Kogge and J. Shalf. Exascale computing trends: Adjusting to the. *Computing in Science & Engineering*, 15(6):16–26, 2013.

[19] N. Satish, N. Sundaram, M. A. Patwary, J. Seo, J. Park, M. A. Hassaan, S. Sengupta, Z. Yin, and P. Dubey. Navigating the maze of graph analytics frameworks using massive graph datasets. In *SIGMOD*, 2014.

[20] E. Werner, S. McMillan, and J. Chu. Patterns and practices for future architectures. Technical Report CMU/SEI-2014-TN-001, Software Engineering Institute, Carnegie Mellon University, August 2014.