2014 IEEE 28th International Parallel & Distributed Processing Symposium Workshops

Graph Algorithms Building Blocks (GABB'2014)

Tim Mattson, David Bader, Aydın Buluç, John Gilbert, Joseph Gonzalez, Jeremy Kepner

The Basic Linear Algebra Subprograms (BLAS), introduced over 30 years ago, had a transformative effect on linear algebra. By building Linear Algebra algorithms from a common set of highly optimized building blocks, researchers spend less time mapping algorithms onto specific hardware features and more time on interesting new algorithms.

Could the same transformation occur for Graph algorithms? Can Graph algorithm researchers converge around a core set of building blocks so we can focus more on algorithms and less on mapping software onto hardware?

Graph Algorithms Building Blocks workshop (GAB'14) will address these questions. The workshop will open with a pair of talks that define a candidate set of graph algorithm building blocks that we call the "Graph BLAS". With this context established, the reamining talks explore issues raised by these Graph BLAS, suggest alternative sets of low level building blocks, and finally consider lessons learned from past standards efforts. We will close with an interactive panel about our collective quest to standardize a set of core graph algorithm building blocks.

1. Motivation and mathematical foundations of the GraphBLAS - Tim Mattson (Intel Corp)

Graphs can be represented in terms of sparse incidence matrices where the row and column indices label vertices in the graph and the elements of the matrix indicate which edges connect pairs of vertices. The key insight behind this approach is that when a graph is represented by a sparse incidence matrix, Sparse Matrix-vector multiplication is the dual of Breadth First Search. By generalizing the pair of operations involved in the linear algebra computations to define a semiring, we can extend the range of these primitives to support a wide range of parallel graph algorithms. This work is the foundation of the new Graph BLAS initiative [1].

2. Examples and applications of graph algorithms in the language of Linear Algebra - John Gilbert (UC Santa Barbara)

What can you really do with the Graph BLAS? This talk will explore the range of algorithms that have been defined around the Combinatorial BLAS [2]; which served as the starting point for our work to define the Graph BLAS. We will consider the bold assertion frequently voiced by researchers working on "graphs in the language of Linear algebra" that "any graph algorithm I know how to map onto a distributed memory machine can be expressed using the sparse incidence matrices and algebraic semirings".

3. GraphX: Unifying Graphs and Tables Through Relational Algebra - Joseph Gonzalez (UC Berkeley)

Driven by the growing importance of graphs and graph structured data, a range of new systems [3–5] have emerged to support scalable graph computation on commodity hardware. These systems exploit specialized graph APIs to efficiently execute graph algorithms in multicore and distributed environments both in memory and on disk. While graph processing systems are well-suited for graph computation, they do not address the wider range of tasks required in real-world graph-analytics (e.g., graph construction, filtering, and data ingress, egress, and management). These more general tasks are typically accomplished using more general distributed data-processing systems. As a consequence practitioners are forced to alternate between systems leading to costly data movement and a more complex programming model.

To unify these disparate views of computation in a single system we introduce GraphX, which embeds graph computation (and GraphBLAS) within more general data processing systems by casting graph operations in relational algebra. Proposed over three decades ago by Codd [6], relational algebra has unified data management systems around a single highly expressive abstraction and helped create a several hundred billion dollar industry. However, while it is possible to directly express graph computation within relational algebra, this can lead to an inefficient execution as these systems are not optimized for graphs. To address this limitation, we translate advances in graph processing systems (e.g., distributed graph representation and data-movement patterns) to distributed join strategies and materialized view maintenance. We show that through these optimizations GraphX can achieve performance parity with widely used graph-processing systems while supporting computation that spans table and graphs.

4. Effective Graph-algorithmic Building Blocks for Graph Databases - David Mizell, Steven P. Reinhardt (YarcData LLC, A Cray Company)

The graph-algorithms building blocks (GABB) effort aims to accelerate the maturation of graph-analytic tools and their use by defining and encouraging implementation of a core set of building-block functions. In the authors opinion, GABBs need to be effective in the context of graph databases (GDBs), which provide persistent interfaces, typically usable by subject-matter experts, to provide greatest value. The GDB context presents constraints that may not exist elsewhere. First, we expect the primary user interface to GDBs will be declarative query languages, which will provide extensive general-purpose functionality but which will also need extension with specific high-value graph algorithms, which may be implemented by the developers of the GDB, by customers or users, or by third parties such as graph-analytic researchers. We see that the developers of these extension functions will initially be the primary audience for GABBs. Second, the performance cost of copying and reformatting data between data formats will continue to be prohibitive, and so it is vital that GABB enable implementations onto the native GDB data structure. Third, the native GDB data structure will be constrained by the query needs of users and need careful GABB interface design to enable high-performance implementations. Fourth, we envision domain-specific graph languages such as GreenMarl becoming the preferred vehicle for graph-algorithm development, below the level of the declarative query language and above the low-level procedural language (likely C++). Assuming that the initial API definition of GABBs is for C++, that API must also enable efficient implementation of graph-domain-specific languages.

5. Adjacency Matrices, Incidence Matrices, Database Schemas, and Associative Arrays - Jeremy Kepner and Vijay Gadepally (MIT Lincoln Laboratory)

Spreadsheets, databases, hash tables and dictionaries; these are the fundamental building blocks of big data storage, retrieval and processing. The MIT Dynamic Distributed Dimensional Data Model (D4M) is an interface to databases and spreadsheets that enable developers to write analytics in the language of linear algebra, significantly reducing the time and effort required. At the core of D4M are associative arrays that allow big data to be represented as large sparse matrices. This sparse matrix D4M schema has been adopted in a number of domains (most notably in the Apache Accumulo community). Linear algebraic graph algorithms work naturally with the D4M schema when such algorithms are cast in terms of edge incidence matrices. An incidence matrix, E, stores each edge in a graph as a row and every vertex as a column. Setting $E(i, v_1) = -1$ and $E(i, v_2) = 1$ is one convention to indicate that edge i starts at vertex v_1 and ends at vertex v_2. Incidence matrices naturally represent partite-directed weighted-multi-hyper graphs (i.e., graphs with many vertex classes, edge direction, weighted edges, multiple edges between vertices, and multiple vertices per edge). Incidence matrices naturally encompass the full richness of data that is found in many data sets (e.g., text documents, bioinformatics, network logs, weblinks, and health records). In contrast, in an adjacency matrix, A, each row and column represent vertices in the graph and setting $A(v_1, v_2) = 1$ denotes an edge from vertex v_1 to vertex v_2. In the above convention for E, the adjacency matrix and the incidence matrix are linked by the formula $A = |E' < 0||E > 0|$. In other words, A is the cross-correlation of E. Any algorithm that can be written using A can also be written using E via the above formula. However, because A is a projection of E, information is always lost in constructing A, and there are algorithms that can be written using E that cannot be constructed using A. This talk will describe the interrelationships between adjacency matrices, incidence matrices, database schemas, and associative arrays in the context of specific examples drawn from a number of real-world applications.

6. Linear algebra operator extensions for performance tuning of graph algorithms - Saeed Maleki, G. Carl Evans, and David Padua (University of Illinois)

Linear algebra operators can be used to represent many graph algorithms in a concise and clear manner. The use of this notation allows for rapid development of complex algorithms. This is potentially not only an immense benefit to developers but also can contribute to performance since linear algebra operations often translate directly in to parallel codes. Today, the power and flexibility of the linear algebra comes with some drawbacks. Standard sparse kernels can not take advantage of all the structure that is present in either the graphs or the algorithms and as a result fall short of the performance of custom implementations. To rectify this, we propose a multi-level system where a high-level linear algebra structure is used to construct a correct program and then low-level performance features possibly controlled by parameters are used to tune the generic kernels.

The key features we have identified so far are: data distribution, synchronization, selection, mirroring, and coalescing. The most obvious of these features, data distribution, is also important for standard parallel linear algebra. Therefore, it is no surprise that it is important in the case of graphs. The other features have proven important so far in the cases where iterative methods are used. Since these are very common in the linear algebraic representation of graph algorithms it is critical that they are supported to allow for maximum performance. When using iterative methods on graphs, there are many cases where most of the graph is unaffected by the iteration. By taking advantage of this, it is possible to obtain significant savings in communication costs. We propose handling this by allowing for local-only updates between synchronizations as well as by using sub graph selection to only apply an operation to a portion of a graph. This is augmented in many cases by mirroring of the partition borders. This can be done in either a push or pull model depending on the behavior of the algorithm. Finally, coalescing is used in conjunction with mirroring to allow updates to aggregate before they are sent remotely. By extending the traditional linear algebra operators with these features we have been able to produce linear algebra based versions of several problems including single source shortest path that preform close to custom implantations.

7. Communication-Avoiding Linear-Algebraic Primitives for Graph Analytics - Aydin Buluç(Berkeley Lab), Grey Ballard, James Demmel, John Gilbert, Laura Grigori, Ben Lipshitz, Adam Lugowski, Oded Schwartz, Edgar Solomonik, Sivan Toledo

Graph algorithms typically have very low computational intensities, hence their execution times are bounded by their communication requirements. In addition to improving the running time drastically, reducing communication will also help improve the energy consumption of graph algorithms. Many of the positive results for communication-avoiding algorithms come from numerical linear algebra. This suggests an immediate path forward for developing communication-avoiding graph algorithms in the language of linear algebra. Unfortunately, the algorithms that achieve communication optimality for asymptotically more available memory are the so-called 3D algorithms, yet the existing software for graph analytics is either 1D or 2D. In this talk, I will describe two new communication-avoiding kernels for graph computations, discuss how they can be integrated into an existing library like the Combinatorial BLAS and how they can be incorporated into the future Graph BLAS standard.

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. Existing parallel algorithms for SpGEMM spend the majority of their time in inter-node communication on large concurrencies. We investigated communication-optimal algorithms for SpGEMM. Our theoretical paper [7] proves new communication lower bounds, presents two new communication-optimal algorithms, and provides a unified communication analysis of existing and new algorithms. Here, I will also discuss the implications of input/output sparsity on the choice of stationary and replicated matrices.

All-pairs shortest paths is a computationally intensive graph algorithm. For dense graphs, we developed a distributed memory algorithm that is communication optimal. This work-efficient algorithm is based on the recursive version of Kleene's algorithm (as opposed to the more popular Floyd-Warshall algorithm), and uses the ideas from 3D algorithms that replicate the input data c times to reduce communication by a factor of $\sqrt{c}$. The algorithm enables much better strong scaling, and we were able to solve a 65K vertex dense APSP problem in about two minutes [8].

8. Standards: Lessons Learned - Andrew Lumsdaine (Indiana University)

Standardization works best when it happens at the right time in the evolution of a technology. There are several different models for technology evolution and standardization including those by Doi and by Shaw. The Doi model attempts to balance research interest in a technology with industrial adoption, proposing that standardization is best introduced in the gap between the decrease of new development and the increase in deployed implementations. The Shaw model is based on extensive studies of the evolution of best practices in multiple disciplines and the reflection of those processes into the software technology lifecycle. In this model, technology evolves in cycles consisting of new problems, ad hoc solutions, community folklore / best practices, codification, models / theory, and finally improved practice.

A slightly modified version of this latter model that substitutes standardization for codification and implementation for models / theory can well explain why a number of efforts with which we are all familiar either succeeded or failed. The original MPI standardization process, for example, took existing best practices and created a single unified interface for functionality that was well understood, already existed, and was being widely used. It is clear that this process went through the full cycle – standardizing MPI led to improved practices in scientific computing. Subsequent efforts to standardize what seemed like good ideas before there were significant examples of use were less successful. The story repeats itself in many venues including some with which this author has first hand experience – the "new" BLAS and C++ standardization.

There are important lessons to be learned from these past efforts. Accordingly, in this talk I will discuss the Doi and Shaw models and present case studies from MPI, new BLAS, and C++– covering both the successes and the disappointments. I will conclude by reflecting on how these lessons learned might apply to standardizing linear algebra based graph BLAS.

References

[1] http://istc-bigdata.org/GraphBlas/.

[2] A. Buluç and J. R. Gilbert, "The Combinatorial BLAS: Design, implementation, and applications," *Intl. Journal of High Perf. Comp. Appl. (IJHPCA)*, vol. 25, no. 4, pp. 496–509, 2011.

[3] G. Malewicz *et al.*, "Pregel: a system for large-scale graph processing," in *SIGMOD'10*, 2010.

[4] J. E. Gonzalez *et al.*, "PowerGraph: Distributed graph-parallel computation on natural graphs," in *OSDI '12*, 2012.

[5] A. Roy *et al.*, "X-Stream: Edge-centric graph processing using streaming partitions," in *SOSP '13*, 2013.

[6] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970.

[7] 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.

[8] E. Solomonik, A. Buluç, and J. Demmel, "Minimizing communication in all-pairs shortest paths," in *Proceedings of the IPDPS*. IEEE Computer Society, 2013.