

Application of Dependence Analysis and Data Flow Graph Scheduling at Runtime to Block Formulations of Matrix Computations

Addendum to Dissertation Proposal

Ernie Chan*

1 Introduction

Programmability is a fundamental problem that faces us with the advent of processors with multiple processing cores on a single chip. Developing simple yet effective techniques for exploiting parallelism will become paramount with the pervasive use of these multicore processors. As such, a parallel programming paradigm will not gain widespread use if it is too complex for commonplace developers to grasp and then implement. The proposed dissertation strives to advance the understanding of parallel programming, restricted to the domain of matrix computations, toward the goal of making it a science that can be understood and applied by many rather than an art practiced by a few. I focus on this problem domain because it provides a rich playground in which to experiment without becoming unmanageable. An additional contribution of this work will lie in the prototype libraries that have practical application in computational science.

2 Proposed Solution

The proposed solution is to combine abstraction and separation of concerns with theoretical analysis and empirical insight. The abstractions include viewing submatrices (blocks) of a matrix as the unit of data and formulating algorithms to compute with these units (algorithms-by-blocks). As a result, the computation can be expressed as a directed acyclic graph (DAG) of operations on blocks (tasks) that can be supplied to a runtime system that analyzes the DAG and executes it in parallel. By defining an appropriate API for encoding blocked matrices and implementing algorithms, the generation of a DAG can be attained

*Department of Computer Sciences, The University of Texas at Austin, Austin, TX 78712, echan@cs.utexas.edu

without expressing parallelism in the code, thus achieving the desired separation of concerns. Theoretical and practical issues will be pursued in an effort to achieve an efficient scheduling of the resulting DAG to multiple processors, cores, and/or threads.

3 Background

This research is related to four fields of computer science: abstractions for dense linear algebra library development, parallelizing matrix computations, scheduling directed acyclic graphs, and autotuning.

The Basic Linear Algebra Subprograms (BLAS) [17, 18, 34] is a standardized interface from which linear algebra operations can be implemented. Dense linear algebra libraries, such as libFLAME [36] and LAPACK [3], are implemented with blocked algorithms that call level-3 BLAS operations in order to obtain high performance on processors with multiple levels of memory hierarchy. To further enhance performance, different algorithmic variants for computing a linear algebra operation, which can be mechanically derived [4, 5] using the FLAME notation [22], can be employed on varying computer architectures. Blocked algorithms can also be formulated as algorithms-by-blocks that operate upon hierarchically stored matrices [10, 13, 14, 26, 27, 42, 51] for better spatial locality of submatrix blocks. I encode algorithms-by-blocks using the FLASH extension [37] to the FLAME API for the C programming language [6], which encapsulates hierarchical matrices as a recursive data structure.

A simple technique for parallelizing dense linear algebra operations is to link sequential algorithms to multithreaded BLAS libraries, but opportunities for parallelism are lost since inherent synchronization points exist between calls to multithreaded BLAS operations. In order to exploit parallelism at a coarser level of granularity, lookahead [1, 31, 48] and wavefront [12, 40, 38] scheduling techniques have been used where problem specific knowledge is used to expose parallelism between operations interleaved with complex data dependencies. While lookahead has been used to parallelize individual operations, such as the Cholesky factorization [2, 24], programmability was never addressed, so these techniques never gained a foothold because of the resulting obfuscated code. I borrow out-of-order execution techniques, which have been utilized by super-scalar computer architectures to exploit instruction-level parallelism [25, 49], and apply those ideas to algorithms-by-blocks, hence the name *SuperMatrix*. Expressing matrix computations as a DAG and scheduling tasks out-of-order completely subsume the concepts of lookahead and wavefront scheduling without the need for problem specific information.

Scheduling directed acyclic graphs with multiple processing elements in order to minimize the total execution time is an NP-hard problem [50]. Numerous scheduling heuristics exist to overcome this difficulty [28, 39]. For instance, critical path reduction heuristics [30, 32] typically aim to maximize load balancing while clustering heuristics [20, 41] generally seek to minimize data communication. The application of DAG scheduling to matrix computations has previously

focused on distributed-memory architectures with the implicit assumption that data communication occurs at a fixed cost between any two processors [35, 53]. This simplifying assumption enables the use of static scheduling methods [33] once a DAG is generated statically [29]. Using a workqueuing mechanism to dispatch tasks to threads [47], I leverage different heuristics, such as first in, first out (FIFO) order, to dynamically schedule tasks on multithreaded architectures assuming that data communication costs can vary. Even though many dense linear algebra operations are inherently recursive [23], there has not been work to systematically exploit the highly structured nature of DAGs formulated from matrix computations for the purpose of scheduling.

Autotuning was first applied by performing an exhaustive empirical search to find the best implementation on a particular computer architecture. This search process has been effective for optimizing discrete Fourier transforms [19, 43]. The application of autotuning for matrix computations has been an inexact science given the large search space of parameters [54]. As a result, the performance of automatically tuned dense linear algebra libraries [11, 15, 16, 52] has not approached that of handcoded libraries, such as the GotoBLAS [21]. I seek to decouple the dependence between autotuning and empirical search. For example, I can use a parameterized model based upon computational runtimes estimated with the floating point operation count of each task in a DAG as a tool for optimizing various parameters.

4 Scope

The applicability of the proposed solution depends on whether block formulations of matrix computations can be expressed as a DAG. Factorizations for solving linear systems and least squares problems such as the Cholesky [7], LU [44], and QR [45] factorizations, inversion algorithms [8], and matrix-matrix (level-3 BLAS-like) [9, 17] operations are all examples of linear algebra operations that SuperMatrix can parallelize. I define this class of operations as loop-based algorithms, which sweep through matrices exposing submatrices for computation.

While I will target operations for dense and banded matrices [46], I do not intend to investigate sparse matrices or iterative methods.

5 Expected Contributions to Science

My expected contributions to science include providing abstractions to enable the effective and intuitive development of parallel algorithms and implementations. I also intend to develop a theoretical framework for analyzing the parallel scheduling of computationally intensive applications, such as matrix computations, in an effort to advance insight towards the efficient use of multicore processors.

6 Proposed Work

Preliminary experience of ours has shown this to be an approach that continues to regularly spawn new research questions. Thus, I expect to pursue research beyond the stated goals as additional opportunities arise. As such, the proposed work for this dissertation includes:

- A) Designing a flexible API with a clear separation of concerns between the implementation of algorithms-by-blocks from the details of parallelization. I will augment the current interface with annotations that will provide semantic information to the runtime system, such as the input and output parameters of each task. I will also extend the interface to allow multiple levels of blocking which in turn may limit the time and space complexity of scheduling.
- B) Developing theoretical insight that motivates practical scheduling heuristics.
- C) Evaluating performance that is practically attained compared to performance that is predicted under idealized circumstances.
- D) Attempting to elevate autotuning to a science by creating a systematic methodology for identifying and optimizing different parameters that affect parallel performance.

Successful completion of these focus areas together will advance programmability in the domain of dense linear algebra algorithms, with possible impact beyond this narrow domain.

7 Measure of Success

The minimum set of requirements for this dissertation will be the completion of all the proposed items. The measures of success for each are:

- A) Anecdotal evidence that the API shields the library developer from details related to parallel computing.
- B) Ability to identify bottlenecks to computational efficiency on different multithreaded platforms, ranging from standard multicore processors to more unique architectures such as the Cell B.E. and Multi-GPU processors, and then apply scheduling heuristics to overcome those barriers under idealized conditions.
- C) Validation of the model of computation through scheduling algorithms that exhibit the same performance characteristics when compared to reference implementations, such as a FIFO workqueuing mechanism under idealized conditions and a sequential algorithm linked with multithreaded BLAS libraries in practice.

- D) Capability to produce near-optimal parameter combinations given a particular operation and problem size without the need of an exhaustive empirical search.

In addition, an outcome of this dissertation will be a prototype of the SuperMatrix runtime system that embodies the successful completion of the proposed work.

References

- [1] C. Addison, Y. Ren, and M. van Waveren. OpenMP issues arising in the development of parallel BLAS and LAPACK libraries. *Scientific Programming*, 11(2):95–104, 2003.
- [2] R. C. Agarwal and F. G. Gustavson. Vector and parallel algorithms for Cholesky factorization on IBM 3090. In *SC '89: Proceedings of the 1989 ACM/IEEE Conference on Supercomputing*, pages 225–233, Reno, NV, USA, November 1989.
- [3] E. Anderson, Z. Bai, J. Demmel, J. E. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. E. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, 1992.
- [4] Paolo Bientinesi. *Mechanical Derivation and Systematic Analysis of Correct Linear Algebra Algorithms*. PhD thesis, The University of Texas at Austin, 2006.
- [5] Paolo Bientinesi, John A. Gunnels, Margaret E. Myers, Enrique S. Quintana-Ortí, and Robert A. van de Geijn. The science of deriving dense linear algebra algorithms. *ACM Transactions on Mathematical Software*, 31(1):1–26, March 2005.
- [6] Paolo Bientinesi, Enrique S. Quintana-Ortí, and Robert A. van de Geijn. Representing linear algebra algorithms in code: The FLAME application programming interfaces. *ACM Transactions on Mathematical Software*, 31(1):27–59, March 2005.
- [7] Ernie Chan, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí, and Robert van de Geijn. SuperMatrix out-of-order scheduling of matrix operations for SMP and multi-core architectures. In *SPAA '07: Proceedings of the Nineteenth ACM Symposium on Parallelism in Algorithms and Architectures*, pages 116–125, San Diego, CA, USA, June 2007.
- [8] Ernie Chan, Field G. Van Zee, Paolo Bientinesi, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí, and Robert van de Geijn. SuperMatrix: A multithreaded runtime scheduling system for algorithms-by-blocks. In *PPoPP '08: Proceedings of the Thirteenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 123–132, Salt Lake City, UT, USA, February 2008.

- [9] Ernie Chan, Field G. Van Zee, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí, and Robert van de Geijn. Satisfying your dependencies with SuperMatrix. In *Cluster '07: Proceedings of the 2007 IEEE International Conference on Cluster Computing*, pages 91–99, Austin, TX, USA, September 2007.
- [10] S. Chatterjee, A. R. Lebeck, P. K. Patnala, and M. Thottethodi. Recursive array layouts and fast matrix multiplication. *IEEE Transactions on Parallel and Distributed Systems*, 13(11):1105–1123, 2002.
- [11] Zizhong Chen, Jack Dongarra, Piotr Luszczek, and Kenneth Roche. Self-adapting software for numerical linear algebra and LAPACK for clusters. *Parallel Computing*, 29(11-12):1723–1743, 2003.
- [12] José M. Claver. Parallel wavefront algorithms solving Lyapunov equations for the Cholesky factor on message passing multiprocessors. *The Journal of Supercomputing*, 13(2):171–189, 1999.
- [13] Timothy Collins and James C. Browne. Matrix++: An object-oriented environment for parallel high-performance matrix computations. In *HICSS '95: Proceedings of the 28th Annual Hawaii International Conference on System Sciences*, pages 202–211, Maui, HI, USA, January 1995.
- [14] Timothy Scott Collins. *Efficient Matrix Computations through Hierarchical Type Specifications*. PhD thesis, The University of Texas at Austin, 1996.
- [15] Javier Cuenca, Domingo Giménez, and José González. Architecture of an automatically tuned linear algebra library. *Parallel Computing*, 30(2):187–210, 2004.
- [16] Javier Cuenca, Domingo Giménez, and Juan-Pedro Martínez. Heuristics for work distribution of a homogeneous parallel dynamic programming scheme on heterogeneous systems. *Parallel Computing*, 31(7):711–735, 2005.
- [17] Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Iain Duff. A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software*, 16(1):1–17, March 1990.
- [18] Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Richard J. Hanson. An extended set of Fortran basic linear algebra subprograms. *ACM Transactions on Mathematical Software*, 14(1):1–17, March 1988.
- [19] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.
- [20] Apostolos Gerasoulis and Tao Yang. On the granularity and clustering of directed acyclic task graphs. *IEEE Transactions on Parallel and Distributed Systems*, 4(6):686–701, 1993.

- [21] Kazushige Goto and Robert A. van de Geijn. Anatomy of a high-performance matrix multiplication. *ACM Transactions on Mathematical Software*, 34(3):1–25, May 2008.
- [22] John A. Gunnels, Fred G. Gustavson, Greg M. Henry, and Robert A. van de Geijn. FLAME: Formal linear algebra methods environment. *ACM Transactions on Mathematical Software*, 27(4):422–455, December 2001.
- [23] Fred G. Gustavson. Recursion leads to automatic variable blocking for dense linear-algebra algorithms. *IBM Journal of Research and Development*, 41(6):737–756, November 1997.
- [24] Fred G. Gustavson, Lars Karlsson, and Bo Kagstrom. Three algorithms for Cholesky factorization on distributed memory using packed storage. *Applied Parallel Computing. State of the Art in Scientific Computing*, pages 550–559, Springer Berlin / Heidelberg, September 2007.
- [25] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, San Francisco, 2003.
- [26] Greg Henry. BLAS based on block data structures. Theory Center Technical Report CTC92TR89, Cornell University, February 1992.
- [27] José Ramón Herrero. *A Framework for Efficient Execution of Matrix Computations*. PhD thesis, Polytechnic University of Catalonia, Spain, 2006.
- [28] Shiyuan Jin, Guy Schiavone, and Damla Turgut. A performance study of multiprocessor task scheduling algorithms. *The Journal of Supercomputing*, 43(1):77–97, 2008.
- [29] Ajita John and James C. Browne. Compilation of constraint programs with noncyclic and cyclic dependencies to procedural parallel programs. *International Journal of Parallel Programming*, 26(1):65–119, February 1998.
- [30] S. J. Kim and J. C. Browne. A general approach to mapping of parallel computation upon multiprocessor architectures. In *ICPP '88: Proceedings of the 1988 International Conference on Parallel Processing*, pages 1–8, University Park, PA, USA, August 1988.
- [31] Jakub Kurzak and Jack Dongarra. Implementing linear algebra routines on multi-core processors with pipelining and a look ahead. In *Applied Parallel Computing. State of the Art in Scientific Computing*, pages 147–156. Springer Berlin / Heidelberg, September 2007.
- [32] Yu-Kwong Kwok and Ishfaq Ahmad. Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 7(5):506–521, 1996.
- [33] Yu-Kwong Kwok and Ishfaq Ahmad. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Computing Surveys*, 31(4):406–471, 1999.

- [34] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. Basic linear algebra subprograms for Fortran usage. *ACM Transactions on Mathematical Software*, 5(3):308–323, September 1979.
- [35] Heejo Lee, Jong Kim, Sung Je Hong, and Sunggu Lee. Task scheduling using a block dependency DAG for block-oriented sparse Cholesky factorization. *Parallel Computing*, 29(1):135–159, 2003.
- [36] libFLAME. <http://www.cs.utexas.edu/users/flame>.
- [37] Tze Meng Low and Robert van de Geijn. An API for manipulating matrices stored by blocks. FLAME Working Note #12 TR-2004-15, The University of Texas at Austin, Department of Computer Sciences, May 2004.
- [38] Naraig Manjikian and Tarek S. Abdelrahman. Scheduling of wavefront parallelism on scalable shared-memory multiprocessors. In *ICPP '96: Proceedings of the 1996 International Conference on Parallel Processing*, pages 122–131, Bloomington, IL, USA, August 1996.
- [39] C. L. McCreary, A. A. Khan, J. J. Thompson, and M. E. McArdle. A comparison of heuristics for scheduling DAGs on multiprocessors. In *IPPS '94: Proceedings of the Eighth International Parallel Processing Symposium*, pages 446–451, Cancun, Mexico, April 1994.
- [40] Dianne P. O’Leary and G. W. Stewart. Data-flow algorithms for parallel matrix computation. *Communications of the ACM*, 28(8):840–853, 1985.
- [41] Michael A. Palis, Jing-Chiou Liou, and David S. L. Wei. Task clustering and scheduling for distributed memory parallel architectures. *IEEE Transactions on Parallel and Distributed Systems*, 7(1):46–55, 1996.
- [42] N. Park, B. Hong, and V. K. Prasanna. Tiling, block data layout, and memory hierarchy performance. *IEEE Transactions on Parallel and Distributed Systems*, 14(7):640–654, 2003.
- [43] Markus Püschel, José M. F. Moura, Jeremy Johnson, David Padua, Manuela Veloso, Bryan Singer, Jianxin Xiong, Franz Franchetti, Aca Gacic, Yevgen Voronenko, Kang Chen, Robert W. Johnson, and Nicholas Rizzolo. SPIRAL: Code generation for DSP transforms. *Proceedings of the IEEE, Special Issue on Program Generation, Optimization, and Adaptation*, 93(2):232–275, 2005.
- [44] Gregorio Quintana-Ortí, Enrique S. Quintana-Ortí, Ernie Chan, Robert van de Geijn, and Field G. Van Zee. Design and scheduling of an algorithm-by-blocks for LU factorization on multithreaded architectures. In *MTAAP '08: Proceedings of the 2008 Workshop on Multithreaded Architectures and Applications*, Miami, FL, USA, April 2008.

- [45] Gregorio Quintana-Ortí, Enrique S. Quintana-Ortí, Ernie Chan, Robert A. van de Geijn, and Field G. Van Zee. Scheduling of QR factorization algorithms on SMP and multi-core architectures. In *PDP '08: Proceedings of the Sixteenth Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, pages 301–310, Toulouse, France, February 2008.
- [46] Gregorio Quintana-Ortí, Enrique S. Quintana-Ortí, Alfredo Remon, and Robert A. van de Geijn. An algorithm-by-blocks for SuperMatrix band Cholesky factorization. In *VECPAR '08: Proceedings of the Eighth International Meeting on High Performance Computing for Computational Science*, Toulouse, France, June 2008.
- [47] Sanjiv Shah, Grant Haab, Paul Petersen, and Joe Throop. Flexible control structures for parallelism in OpenMP. *Concurrency: Practice and Experience*, 12(12):1219–1239, 2000.
- [48] Peter Strazdins. A comparison of lookahead and algorithmic blocking techniques for parallel matrix factorization. *International Journal of Parallel and Distributed Systems and Networks*, 4(1):26–35, June 2001.
- [49] R. M. Tomasulo. An efficient algorithm for exploiting multiple arithmetic units. *IBM Journal of Research and Development*, 11(1):25–33, January 1967.
- [50] Jeffrey D. Ullman. NP-complete scheduling problems. *Journal of Computer and System Sciences*, 10(3):384–393, 1975.
- [51] Vinod Valsalam and Anthony Skjellum. A framework for high-performance matrix multiplication based on hierarchical abstractions, algorithms and optimized low-level kernels. *Concurrency and Computation: Practice and Experience*, 14(10):805–840, 2002.
- [52] R. Clint Whaley and Jack J. Dongarra. Automatically tuned linear algebra software. In *SC '98: Proceedings of the 1998 ACM/IEEE Conference on Supercomputing*, pages 1–27, San Jose, CA, USA, 1998.
- [53] Tao Yang. *Scheduling and Code Generation for Parallel Architectures*. PhD thesis, Rutgers, The State University of New Jersey, 1993.
- [54] Kamen Yotov, Keshav Pingali, and Paul Stodghill. Think globally, search locally. In *ICS '05: Proceedings of the Nineteenth Annual International Conference on Supercomputing*, pages 141–150, New York, NY, USA, 2005.