References

Dantzig1949 [1] G. Dantzig, "Programming of interdependent activities: II mathematical model," English, *Econometrica*, vol. 17, no. 3/4, pp. 200–211, 1949, ISSN: 0012-9682.

Markowitz1957 [2] H. M. Markowitz, "The elimination form of the inverse and its application to linear programming," *Management Science*, vol. 3, no. 3, pp. 255–269, Apr. 1957, ISSN: 0025-1909. DOI: 10.1287/mnsc.3.3.255.

Flynn1972 [3] M. J. Flynn, "Some computer organizations and their effectiveness," *IEEE Transactions on Computers*, vol. C-21, no. 9, pp. 948–960, Sep. 1972, ISSN: 0018-9340. DOI: 10.1109/TC.1972.5009071.

Forrest1972 [4] J. J. H. Forrest and J. A. Tomlin, "Updated triangular factors of the basis to maintain sparsity in the product form simplex method," *Mathematical Programming*, vol. 2, no. 1, pp. 263–278, 1972, ISSN: 1436-4646. DOI: 10.1007/BF01584548.

Tarjan1972 [5] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, Jun. 1972.

apadimitriou1982 [6] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1982, ISBN: 0-13-152462-3.

Gay1985 [7] D. M. Gay, "Electronic mail distribution of linear programming test problems," *Mathematical Programming Society COAL Newsletter*, vol. 13, pp. 10–12, 1985.

Kincaid1989 [8] D. R. Kincaid, T. C. Oppe, and D. M. Young, *ITPACKV 2D user's guide*, May 1989. [Online]. Available: http://www.netlib.no/netlib/itpack/index.html.

Mehrotra1992 [9] S. Mehrotra, "On the implementation of a primal-dual interior point method," *SIAM Journal on Optimization*, vol. 2, no. 4, pp. 575–601, 1992. DOI: 10.1137/0802028,

Abstract: This paper gives an approach to implementing a second-order primal-dual interior point method. It uses a Taylor polynomial of second order to approximate a primal-dual trajectory. The computations for the second derivative are combined with the computations for the centering direction. Computations in this approach do not require that primal and dual solutions be feasible. Expressions are given to compute all the higher-order derivatives of the trajectory of interest. The implementation ensures that a suitable potential function is reduced by a constant amount at each iteration.

There are several salient features of this approach. An adaptive heuristic for estimating the centering parameter is given. The approach used to compute the step length is also adaptive. A new practical approach to compute the starting point is given. This approach treats primal and dual problems symmetrically.

Computational results on a subset of problems available from netlib are given. On mutually tested problems the results show that the proposed method requires approximately 40 percent fewer iterations than the implementation proposed in Lustig, Marsten, and Shanno [Tech. Rep. TR J-89-11, Georgia Inst. of Technology, Atlanta, 1989]. It requires approximately 50 percent fewer iterations than the dual affine scaling method in Adler, Karmarkar, Resende, and Veiga [Math. Programming, 44 (1989), pp. 297-336], and 35 percent fewer iterations than the second-order dual affine scaling method in the same paper. The new approach for estimating the centering parameter and finding the step length and the starting point have contributed to the reduction in the number of iterations. However, the contribution due to the use of second derivative is most significant.

On the tested problems, on the average the implementation shown was found to be approximately two times faster than OBl (version 02/90) described in Lustig, Marsten, and Shanno and 2.5 times faster than MINOS 5.3 described in Murtagh and Saunders [Tech. Rep. SOL 83-20, Dept. of Operations Research, Stanford Univ., Stanford, CA, 1983].

Temam1992 [10] O. Temam and W. Jalby, "Characterizing the behaviour of sparse algorithms on caches," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '92, Minneapolis, MN, USA, 1992, pp. 578–587,

Abstract: A methodology is presented for modeling the irregular references of sparse codes using probabilistic methods. The behavior on cache of one of the most frequent primitives, SpMxV sparse matrix vector multiply, is analyzed. A model of its references is built, and performance bottlenecks of SpMxV are analyzed using the model and simulations. The main parameters are identified and their role is explained and quantified. This analysis is then used to discuss optimizations of SpMxV. A blocking technique which takes into account the specifics of sparse codes is proposed.

Alvarado1993 [11] F. L. Alvarado, A. Pothen, and R. Schreiber, "Highly parallel sparse triangular solution," English, in *Graph Theory and Sparse Matrix Computation*, ser. The IMA Volumes in Mathematics and its Applications, A. George, J. R. Gilbert, and J. W. H. Liu, Eds., vol. 56, Springer New York, 1993, pp. 141–157, ISBN: 978-1-4613-8371-0. DOI: 10.1007/978-1-4613-8369-7_7,
Abstract: In this paper we survey a recent approach for solving sparse triangular systems of equations on highly parallel computers. This approach employs a partitioned representation of the inverse of the triangular matrix so that the solution can be computed by matrix-vector multiplication. The number of factors in the partitioned inverse is proportional to the number of general communication steps (router steps on a CM-2) required in a highly parallel algorithm. We describe partitioning algorithms that minimize the number of factors in the partitioned inverse over all symmetric permutations of the triangular matrix such that the permuted matrix continues to be triangular. For a Cholesky factor we describe an O(n) time and space algorithm to solve the partitioning problem above, where n is the order of the matrix. Our computational results on a CM-2 demonstrate the potential superiority of the partitioned inverse approach over the conventional substitution algorithm for highly parallel sparse triangular solution. Finally we describe current and future extensions of these results.

Osier1993 [12] J. Osier, *GNU gprof*, Free Software Foundation, Inc., Jan. 1993. [Online]. Available: https://ftp.gnu.org/old-gnu/Manuals/gprof-2.9.1/html_chapter/gprof_1.html.

Peyton1993 [13] B. W. Peyton, A. Pothen, and X. Yuan, "Partitioning a chordal graph into transitive subgraphs for parallel sparse triangular solution," *Linear Algebra and its Applications*, vol. 192, no. 0, pp. 329–353, 1993, ISSN: 0024-3795. DOI: 10.1016/0024-3795(93)90248-M. [Online]. Available: http://www.sciencedirect.com/science/article/pii/002437959390248M,
Abstract: A recent approach for solving sparse triangular systems of equations on massively parallel computers employs a factorization of the triangular coefficient matrix to obtain a representation of its inverse in product form. The number of general communication steps required by this approach is proportional to the number of factors in the factorization. The triangular matrix can be symmetrically permuted to minimize the number of factors over suitable classes of permutations, and thereby the complexity of the parallel algorithm can be minimized. Algorithms for minimizing the number of factors over several classes of permutations have been considered in earlier work. Let $F = L + L^T$ denote the symmetric filled matrix corresponding to a Cholesky factor L, and let G_F denote the adjacency graph of F. We consider the problem of minimizing the number of factors over all permutations which preserve the structure of G_F. The graph model of this problem is to partition the vertices G_F into the fewest transitively closed subgraphs over all perfect elimination orderings while satisfying a certain precedence relationship. The solution to this chordal-graph partitioning problem can be described by a greedy scheme which eliminates a largest permissible subgraph at each step. Further, the subgraph eliminated at each step can be characterized in terms of lengths of chordless paths in the current elimination graph. This solution relies on several results concerning transitive perfect elimination orderings introduced in this paper. We describe a partitioning algorithm with $O(|V| + |E|)$ time and space complexity.

Saad1993 [14] Y. Saad, "A flexible inner-outer preconditioned gmres algorithm," *SIAM Journal on Scientific Computing*, vol. 14, no. 2, pp. 461–469, 1993. DOI: 10.1137/0914028,
Abstract: A variant of the GMRES algorithm is presented that allows changes in the preconditioning at every step. There are many possible applications of the new algorithm, some of which are briefly discussed. In particular, a result of the flexibility of the new variant is that any iterative method can be used as a preconditioner. For example, the standard GMRES algorithm itself can be used as a preconditioner, as can CGNR (or CGNE), the conjugate gradient method applied

to the normal equations. However, the more appealing utilization of the method is in conjunction with relaxation techniques, possibly multilevel techniques. The possibility of changing preconditioners may be exploited to develop efficient iterative methods and to enhance robustness. A few numerical experiments are reported to illustrate this fact.

Smith1993 [15] J. R. Smith, *The Design and Analysis of Parallel Algorithms*. Oxford University Press, 1993, ISBN: 0195078810,
Abstract: This text for students and professionals in computer science provides a valuable overview of current knowledge concerning parallel algorithms. These computer operations have recently acquired increased importance due to their ability to enhance the power of computers by permitting multiple processors to work on different parts of a problem independently and simultaneously. This approach has led to solutions of difficult problems in a number of vital fields, including artificial intelligence, image processing, and differential equations. As the first up-to-date summary of the topic, this book will be sought after by researchers, computer science professionals, and advanced students involved in parallel computing and parallel algorithms.

Suhl1993 [16] L. M. Suhl and U. H. Suhl, "A fast LU update for linear programming," *Annals of Operations Research*, vol. 43, no. 1, pp. 33–47, 1993, ISSN: 0254-5330. DOI: 10.1007/BF02025534,
Abstract: This paper discusses sparse matrix kernels of simplex-based linear programming software. State-of-the-art implementations of the simplex method maintain an LU factorization of the basis matrix which is updated at each iteration. The LU factorization is used to solve two sparse sets of linear equations at each iteration. We present new implementation techniques for a modified Forrest-Tomlin LU update which reduce the time complexity of the update and the solution of the associated sparse linear systems. We present numerical results on Netlib and other real-life LP models.

Knijnenburg1994 [17] P. M. W. Knijnenburg and H. A. G. Wijshoff, "On improving data locality in sparse matrix computations," High Performance Computing Division, Dept. of Computer Science, Leiden University, Tech. Rep., 1994. [Online]. Available: http://liacs.leidenuniv.nl/assets/PDF/TechRep/tr94-15.pdf,
Abstract: Sparse matrix computations and irregular type computations show poor data locality behavior. Recently compiler optimizations techniques have been proposed to improve the data locality for regular type loop structures. Sparse matrix computations do not fall into this categorie of computations and the issue of compiler optimizations for sparse computations is merely understood. In this paper we describe how compiler optimizations based on pattern matching techniques can be used to improve the data locality behavior for sparse computations.

Shewchuk1994 [18] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep., 1994. [Online]. Available: http://www.cs.cmu.edu/~./quake-papers/painless-conjugate-gradient.pdf,
Abstract: The Conjugate Gradient Method is the most prominent iterative method for solving sparse systems of linear equations. Unfortunately, many textbook treatments of the topic are written so that even their own authors would be mystified, if they bothered to read their own writing. For this reason, an understanding of the method has been reserved for the elite brilliant few who have painstakingly decoded the mumblings of their forebears. Nevertheless, the Conjugate Gradient Method is a composite of simple, elegant ideas that almost anyone can understand. Of course, a reader as intelligent as yourself will learn them almost effortlessly. The idea of quadratic forms is introduced and used to derive the methods of Steepest Descent, Conjugate Directions, and Conjugate Gradients. Eigenvectors are explained and used to examine the convergence of the Jacobi Method, Steepest Descent, and Conjugate Gradients. Other topics include preconditioning and the nonlinear Conjugate Gradient Method. I have taken pains to make this article easy to read. Sixty-two illustrations are provided. Dense prose is avoided. Concepts are explained in several different ways. Most equations are coupled with an intuitive interpretation.

Blumofe1995 [19] R. D. Blumofe, C. F. Joerg, B. C. Kuszmaul, C. E. Leiserson, K. H. Randall, and Y. Zhou, "Cilk: An efficient multithreaded runtime system," in *Proceedings of the 5th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPOPP

'95, Santa Barbara, California, USA: ACM, 1995, pp. 207–216, ISBN: 0-89791-700-6. DOI: 10.1145/209936.209958,
Abstract: Cilk (pronounced "silk") is a C-based runtime system for multi-threaded parallel programming. In this paper, we document the efficiency of the Cilk work-stealing scheduler, both empirically and analytically. We show that on real and synthetic applications, the "work" and "critical path" of a Cilk computation can be used to accurately model performance. Consequently, a Cilk programmer can focus on reducing the work and critical path of his computation, insulated from load balancing and other runtime scheduling issues. We also prove that for the class of "fully strict" (well-structured) programs, the Cilk scheduler achieves space, time and communication bounds all within a constant factor of optimal.The Cilk runtime system currently runs on the Connection Machine CM5 MPP, the Intel Paragon MPP, the Silicon Graphics Power Challenge SMP, and the MIT Phish network of workstations. Applications written in Cilk include protein folding, graphic rendering, backtrack search, and the *Socrates chess program, which won third prize in the 1994 ACM International Computer Chess Championship.

Lain1995 [20] A. Lain and P. Banerjee, "Exploiting spatial regularity in irregular iterative applications," in *Proceedings of the 9th International Symposium on Parallel Processing*, ser. IPPS '95, Santa Barbara, CA, USA: IEEE Computer Society, 1995, pp. 820–826, ISBN: 0-8186-7074-6. [Online]. Available: http://dl.acm.org/citation.cfm?id=645605.663225&preflayout=tabs,
Abstract: The increasing gap between the speed of microprocessors and memory subsystems makes it imperative to exploit locality of reference in sequential irregular applications. The parallelization of such applications requires special considerations. Current RTS (Run-Time Support) for irregular computations fails to exploit the fine grain regularity present in these applications, producing unnecessary time and memory overheads. PILAR (Parallel Irregular Library with Application of Regularity) is a new RTS for irregular computations that provides a variety of internal representations of communication patterns based on their regularity; allowing for the efficient support of a wide spectrum of regularity under a common framework. Experimental results on the IBM SP-1 and Intel Paragon demonstrate the validity of our approach.

Lawson1995 [21] C. Lawson and R. Hanson, *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, 1995. DOI: 10.1137/1.9781611971217. eprint: http://epubs.siam.org/doi/pdf/10.1137/1.9781611971217.

Mukherjee1995 [22] S. S. Mukherjee, S. D. Sharma, M. D. Hill, J. R. Larus, A. Rogers, and J. Saltz, "Efficient support for irregular applications on distributed-memory machines," in *Proceedings of the Fifth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '95, Santa Barbara, CA, USA: ACM, 1995, pp. 68–79, ISBN: 0-89791-700-6. DOI: 10.1145/209936.209945,
Abstract: Irregular computation problems underlie many important scientific applications. Although these problems are computationally expensive, and so would seem appropriate for parallel machines, their irregular and unpredictable run-time behavior makes this type of parallel program difficult to write and adversely affects run-time performance.This paper explores three issues partitioning, mutual exclusion, and data transfer crucial to the efficient execution of irregular problems on distributed-memory machines. Unlike previous work, we studied the same programs running in three alternative systems on the same hardware base (a Thinking Machines CM-5): the CHAOS irregular application library, Transparent Shared Memory (TSM), and eXtensible Shared Memory (XSM). CHAOS and XSM performed equivalently for all three applications. Both systems were somewhat (13%) to significantly faster (991%) than TSM.

Shin1995 [23] K. Shin and X. Cui, "Computing time delay and its effects on real-time control systems," *IEEE Transactions on Control Systems Technology*, vol. 3, no. 2, pp. 218–224, 1995, ISSN: 1063-6536. DOI: 10.1109/87.388130,
Abstract: The reliability of a real-time digital control computer depends not only on the reliability of the hardware and software used, but also on the time delay in computing the control output, because of the negative effects of computing time delay on control system performance. For a given fixed sampling interval, the effects of computing time delay are classified into the delay and loss

problems. The delay problem occurs when the computing time delay is nonzero but smaller than the sampling interval, while the loss problem occurs when the computing time delay is greater than, or equal to, the sampling interval, i.e., loss of the control output. These two problems are analyzed as a means of evaluating real-time control systems. First, a generic analysis of the effects of computing time delay is presented along with necessary conditions for system stability. Then, we present both qualitative and quantitative analyses of the computing time delay effects on a robot control system, deriving upper bounds of the computing time delay with respect to system stability and system performance.

Golub1996 [24] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: Johns Hopkins University Press, 1996, ISBN: 0-8018-5414-8.

Wunderling1996 [25] R. Wunderling, "Paralleler und objektorientierter simplex-algorithmus," German, PhD thesis, Technische Universität Berlin, 1996.

Boisvert1997 [26] R. F. Boisvert, R. Pozo, K. Remington, R. F. Barrett, and J. J. Dongarra, "Matrix market: A web resource for test matrix collections," in *Proceedings of the IFIP TC2/WG2.5 Working Conference on Quality of Numerical Software: Assessment and Enhancement*, Oxford, UK: Chapman & Hall, Ltd., 1997, pp. 125–137, ISBN: 0-412-80530-8. [Online]. Available: `http://dl.acm.org/citation.cfm?id=265834.265854`.

Davis1997 [27] T. A. Davis and I. S. Duff, "An unsymmetric-pattern multifrontal method for sparse LU factorization," *SIAM Journal on Matrix Analysis and Applications*, vol. 18, no. 1, pp. 140–158, 1997. DOI: `10.1137/S0895479894246905`,

Abstract: Sparse matrix factorization algorithms for general problems are typically characterized by irregular memory access patterns that limit their performance on parallel-vector supercomputers. For symmetric problems, methods such as the multifrontal method avoid indirect addressing in the innermost loops by using dense matrix kernels. However, no efficient LU factorization algorithm based primarily on dense matrix kernels exists for matrices whose pattern is very unsymmetric. We address this deficiency and present a new unsymmetric-pattern multifrontal method based on dense matrix kernels. As in the classical multifrontal method, advantage is taken of repetitive structure in the matrix by factorizing more than one pivot in each frontal matrix, thus enabling the use of Level 2 and Level 3 BLAS. The performance is compared with the classical multifrontal method and other unsymmetric solvers on a CRAY C-98.

Dongarra1998 [28] J. Dongarra, I. Duff, D. Sorensen, and H. van der Vorst, *Numerical Linear Algebra for High-Performance Computers*. Society for Industrial and Applied Mathematics, 1998. DOI: `10.1137/1.9780898719611`. eprint: `http://epubs.siam.org/doi/pdf/10.1137/1.9780898719611`.

Fraguela1998 [29] B. B. Fraguela, R. Doallo, and E. L. Zapata, "Modeling set associative caches behavior for irregular computations," in *Proceedings of the 1998 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '98/PERFORMANCE '98, Madison, Wisconsin, USA: ACM, 1998, pp. 192–201, ISBN: 0-89791-982-3. DOI: `10.1145/277851.277910`,

Abstract: While much work has been devoted to the study of cache behavior during the execution of codes with regular access patterns, little attention has been paid to irregular codes. An important portion of these codes are scientific applications that handle compressed sparse matrices. In this work a probabilistic model for the prediction of the number of misses on a K-way associative cache memory considering sparse matrices with a uniform or banded distribution is presented. Two different irregular kernels are considered: the sparse matrix-vector product and the transposition of a sparse matrix. The model was validated with simulations on synthetic uniform matrices and banded matrices from the Harwell-Boeing collection.

Mitchell1998 [30] M. Mitchell, *An introduction to genetic algorithms*. The MIT Press, 1998.

Anderson1999 [31] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999, ISBN: 0-89871-447-8 (paperback).

Bertsekas1999 [32] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999, ISBN: 1-886529-00-0.

Davis1999 [33] T. A. Davis and I. S. Duff, "A combined unifrontal/multifrontal method for unsymmetric sparse matrices," *ACM Transactions on Mathematical Software*, vol. 25, no. 1, pp. 1–20, Mar. 1999, ISSN: 0098-3500. DOI: 10.1145/305658.287640,
Abstract: We discuss the organization of frontal matrices in multifrontal methods for the solution of large sparse sets of unsymmetric linear equations. In the multifrontal method, work on a frontal matrix can be suspended, the frontal matrix can be stored for later reuse, and a new frontal matrix can be generated. There are thus several frontal matrices stored during the factorization, and one or more of these are assembled (summed) when creating a new frontal matrix. Although this means that arbitrary sparsity patterns can be handled efficiently, extra work is required to sum the frontal matrices together and can be costly because indirect addressing is requred. The (uni)frontal method avoids this extra work by factorizing the matrix with a single frontal matrix. Rows and columns are added to the frontal matrix, and pivot rows and columns are removed. Data movement is simpler, but higher fill-in can result if the matrix cannot be permuted into a variable-band form with small profile. We consider a combined unifrontal/multifrontal algorithm to enable general fill-in reduction orderings to be applied without the data movement of previous multifrontal approaches. We discuss this technique in the context of a code designed for the solution of sparse systems with unsymmetric pattern.

Ding1999 [34] C. Ding and K. Kennedy, "Improving cache performance in dynamic applications through data and computation reorganization at run time," *ACM SIGPLAN Notices*, vol. 34, no. 5, pp. 229–241, May 1999, ISSN: 0362-1340. DOI: 10.1145/301631.301670,
Abstract: With the rapid improvement of processor speed, performance of the memory hierarchy has become the principal bottleneck for most applications. A number of compiler transformations have been developed to improve data reuse in cache and registers, thus reducing the total number of direct memory accesses in a program. Until now, however, most data reuse transformations have been static—applied only at compile time. As a result, these transformations cannot be used to optimize irregular and dynamic applications, in which the data layout and data access patterns remain unknown until run time and may even change during the computation.In this paper, we explore ways to achieve better data reuse in irregular and dynamic applications by building on the inspector-executor method used by Saltz for run-time parallelization. In particular, we present and evaluate a dynamic approach for improving both computation and data locality in irregular programs. Our results demonstrate that run-time program transformations can substantially improve computation and data locality and, despite the complexity and cost involved, a compiler can automate such transformations, eliminating much of the associated run-time overhead.

Karypis1999 [35] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, vol. 20, no. 1, pp. 359–392, 1999. DOI: 10.1137/S1064827595287997,
Abstract: Recently, a number of researchers have investigated a class of graph partitioning algorithms that reduce the size of the graph by collapsing vertices and edges, partition the smaller graph, and then uncoarsen it to construct a partition for the original graph [Bui and Jones, Proc. of the 6th SIAM Conference on Parallel Processing for Scientific Computing, 1993, 445–452; Hendrickson and Leland, A Multilevel Algorithm for Partitioning Graphs, Tech. report SAND 93-1301, Sandia National Laboratories, Albuquerque, NM, 1993]. From the early work it was clear that multilevel techniques held great promise; however, it was not knownif they can be made to consistently produce high quality partitions for graphs arising in a wide range of application domains. We investigate the effectiveness of many different choices for all three phases: coarsening, partition of the coarsest graph, and refinement. In particular, we present a new coarsening heuristic (called heavy-edge heuristic) for which the size of the partition of the coarse graph is within a small factor of the size of the final partition obtained after multilevel refinement. We also present a much faster variation of the Kernighan–Lin (KL) algorithm for refining during uncoarsening. We test our scheme on a large number of graphs arising in various domains including finite element methods, linear programming, VLSI, and transportation. Our experiments show that our scheme produces partitions that are consistently better than those produced by spectral partitioning schemes in

substantially smaller time. Also, when our scheme is used to compute fill-reducing orderings for sparse matrices, it produces orderings that have substantially smaller fill than the widely used multiple minimum degree algorithm.

Byrd2000 [36] R. H. Byrd, J. C. Gilbert, and J. Nocedal, "A trust region method based on interior point techniques for nonlinear programming," *Mathematical Programming*, vol. 89, no. 1, pp. 149–185, Nov. 1, 2000, ISSN: 1436-4646. DOI: 10.1007/PL00011391,
Abstract: An algorithm for minimizing a nonlinear function subject to nonlinear inequality constraints is described. It applies sequential quadratic programming techniques to a sequence of barrier problems, and uses trust regions to ensure the robustness of the iteration and to allow the direct use of second order derivatives. This framework permits primal and primal-dual steps, but the paper focuses on the primal version of the new algorithm. An analysis of the convergence properties of this method is presented.

Notay2000 [37] Y. Notay, "Flexible conjugate gradients," *SIAM Journal on Scientific Computing*, vol. 22, no. 4, pp. 1444–1460, 2000. DOI: 10.1137/S1064827599362314,
Abstract: We analyze the conjugate gradient (CG) method with preconditioning slightly variable from one iteration to the next. To maintain the optimal convergence properties, we consider a variant proposed by Axelsson that performs an explicit orthogonalization of the search directions vectors. For this method, which we refer to as flexible CG, we develop a theoretical analysis that shows that the convergence rate is essentially independent of the variations in the preconditioner as long as the latter are kept sufficiently small. We further discuss the real convergence rate on the basis of some heuristic arguments supported by numerical experiments. Depending on the eigenvalue distribution corresponding to the fixed reference preconditioner, several situations have to be distinguished. In some cases, the convergence is as fast with truncated versions of the algorithm or even with the standard CG method, whereas quite large variations are allowed without too much penalty. In other cases, the flexible variant effectively outperforms the standard method, while the need for truncation limits the size of the variations that can be reasonably allowed.

Saad2010 [38] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h, "A deflated version of the conjugate gradient algorithm," *SIAM Journal on Scientific Computing*, vol. 21, no. 5, pp. 1909–1926, 2000. DOI: 10.1137/S1064829598339761. [Online]. Available: https://doi.org/10.1137/S1064829598339761,
Abstract: We present a deflated version of the conjugate gradient algorithm for solving linear systems. The new algorithm can be useful in cases when a small number of eigenvalues of the iteration matrix are very close to the origin. It can also be useful when solving linear systems with multiple right-hand sides, since the eigenvalue information gathered from solving one linear system can be recycled for solving the next systems and then updated.

Blackford2001 [39] L. S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, and R. C. Whaley, "An updated set of basic linear algebra subprograms (BLAS)," *ACM Transactions on Mathematical Software*, vol. 28, pp. 135–151, 2001. DOI: 10.1145/567806.567807.

Braun2001 [40] T. D. Braun, H. J. Siegel, and A. A. Maciejewski, "Heterogeneous computing: Goals, methods, and open problems," English, in *High Performance Computing (HiPC 2001)*, ser. Lecture Notes in Computer Science, B. Monien, V. K. Prasanna, and S. Vajapeyam, Eds., vol. 2228, Springer Berlin Heidelberg, Dec. 2001, pp. 307–318, ISBN: 978-3-540-43009-4. DOI: 10.1007/3-540-45307-5_27,
Abstract: This paper discusses the material tob e presented by H. J. Siegel in his keynote talk. Distributed high-performance heterogeneous computing (HC) environments are composed of machines with varied computational capabilities interconnected by high-speed links. These environments are well suited to meet the computational demands of large, diverse groups of applications. One key factor in achieving the best performance possible from HC environments is the ability to assign effectively the applications to machines and schedule their execution. Several factors must be considered during this assignment. A conceptual model for the automatic decomposition

of an application into tasks and assignment of tasks to machines is presented. An example of a static matching and scheduling approach for an HC environment is summarized. Some examples of current HC technology and open research problems are discussed.

llor-Crummey2001 [41] J. Mellor-Crummey, D. Whalley, and K. Kennedy, "Improving memory hierarchy performance for irregular applications using data and computation reorderings," *International Journal of Parallel Programming*, vol. 29, no. 3, pp. 217–247, Jun. 2001, ISSN: 0885-7458. DOI: 10.1023/A:1011119519789,
Abstract: The performance of irregular applications on modern computer systems is hurt by the wide gap between CPU and memory speeds because these applications typically under-utilize multi-level memory hierarchies, which help hide this gap. This paper investigates using data and computation reorderings to improve memory hierarchy utilization for irregular applications. We evaluate the impact of reordering on data reuse at different levels in the memory hierarchy. We focus on coordinated data and computation reordering based on space-filling curves and we introduce a new architecture-independent multi-level blocking strategy for irregular applications. For two particle codes we studied, the most effective reorderings reduced overall execution time by a factor of two and four, respectively. Preliminary experience with a scatter benchmark derived from a large unstructured mesh application showed that careful data and computation ordering reduced primary cache misses by a factor of two compared to a random ordering.

Strout2001 [42] M. M. Strout, L. Carter, and J. Ferrante, "Rescheduling for locality in sparse matrix computations," in *Proceedings of the 2001 International Conference on Computational Science*, V. N. Alexandrov, J. J. Dongarra, B. A. Juliano, R. S. Renner, and C. J. K. Tan, Eds., ser. ICCS '01. San Francisco, CA, USA: Springer Berlin Heidelberg, 2001, pp. 137–146, ISBN: 978-3-540-45545-5. DOI: 10.1007/3-540-45545-0_23,
Abstract: In modern computer architecture the use of memory hierarchies causes a program's data locality to directly affect performance. Data locality occurs when a piece of data is still in a cache upon reuse. For dense matrix computations, loop transformations can be used to improve data locality. However, sparse matrix computations have non-affine loop bounds and indirect memory references which prohibit the use of compile time loop transformations. This paper describes an algorithm to tile at runtime called serial sparse tiling. We test a runtime tiled version of sparse Gauss-Seidel on 4 different architectures where it exhibits speedups of up to 2.7. The paper also gives a static model for determining tile size and outlines how overhead affects the overall speedup.

Bixby2002 [43] R. E. Bixby, "Solving real-world linear programs: A decade and more of progress," *Operations Research*, vol. 50, no. 1, pp. 3–15, Jan. 2002. DOI: 10.1287/opre.50.1.3.17780,
Abstract: This paper is an invited contribution to the 50th anniversary issue of the journalOperations Research, published by the Institute of Operations Research and Management Science (INFORMS). It describes one person's perspective on the development of computational tools for linear programming. The paper begins with a short personal history, followed by historical remarks covering the some 40 years of linear-programming developments that predate my own involvement in this subject. It concludes with a more detailed look at the evolution of computational linear programming since 1987.

Stoer2002 [44] J. Stoer and R. Bulirsch, *Introduction to numerical analysis*, ser. Texts in applied mathematics. New York: Springer, 2002, ISBN: 0-387-95452-X.

Vuduc2002 [45] R. Vuduc, S. Kamil, J. Hsu, R. Nishtala, J. W. Demmel, and K. A. Yelick, "Automatic performance tuning and analysis of sparse triangular solve," in *Proceeding of the Workshop on Performance Optimization of High-level Languages and Libraries (POHLL) at the ACM International Conference on Supercomputing*, ser. ICS '02, Venice, IT, Jun. 2002. [Online]. Available: http://bebop.cs.berkeley.edu/pubs/vuduc2002-sts-bounds.pdf.

Biswas2003 [46] R. Biswas, L. Oliker, and H. Shan, "Parallel computing strategies for irregular algorithms," LBNL, Tech. Rep. 53115, 2003,

Abstract: Parallel computing promises several orders of magnitude increase in our ability to solve realistic computationally intensive problems, but relies on their efficient mapping and execution on large-scale multiprocessor architectures. Unfortunately, many important applications are irregular and dynamic in nature, making their effective parallel implementation a daunting task. Moreover, with the proliferation of parallel architectures and programming paradigms, the typical scientist is faced with a plethora of questions that must be answered in order to obtain an acceptable parallel implementation of the solution algorithm. In this paper, we consider three representative irregular applications: unstructured remeshing, sparse matrix computations, and N-body problems, and parallelize them using various popular programming paradigms on a wide spectrum of computing platforms ranging from state-of-the-art supercomputers to PC clusters. We present the underlying problems, the solution algorithms, and the parallel implementation strategies. Smart load-balancing, partitioning, and ordering techniques are used to enhance parallel performance. Overall results demonstrate the complexity of efficiently parallelizing irregular algorithms.

Bolz2003 [47] J. Bolz, I. Farmer, E. Grinspun, and P. Schröder, "Sparse matrix solvers on the GPU: Conjugate gradients and multigrid," in *ACM SIGGRAPH 2003 Papers*, ser. SIGGRAPH '03, San Diego, California: ACM, 2003, pp. 917–924, ISBN: 1-58113-709-5. DOI: 10.1145/1201775.882364,

Abstract: Many computer graphics applications require high-intensity numerical simulation. We show that such computations can be performed efficiently on the GPU, which we regard as a full function streaming processor with high floating-point performance. We implemented two basic, broadly useful, computational kernels: a sparse matrix conjugate gradient solver and a regular-grid multigrid solver. Real time applications ranging from mesh smoothing and parameterization to fluid solvers and solid mechanics can greatly benefit from these, evidence our example applications of geometric flow and fluid simulation running on NVIDIA's GeForce FX.

Kamvar2003 [48] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub, "Exploiting the block structure of the web for computing pagerank," InfoLab, Stanford University, Tech. Rep. 2003–17, 2003. [Online]. Available: http://ilpubs.stanford.edu:8090/579/,

Abstract: The web link graph has a nested block structure: the vast majority of hyperlinks link pages on a host to other pages on the same host, and many of those that do not link pages within the same domain. We show how to exploit this structure to speed up the computation of PageRank by a 3-stage algorithm whereby (1) the local PageRanks of pages for each host are computed independently using the link structure of that host, (2) these local PageRanks are then weighted by the "importance" of the corresponding host, and (3) the standard PageRank algorithm is then run using as its starting vector the weighted aggregate of the local PageRanks. Empirically, this algorithm speeds up the computation of PageRank by a factor of 2 in realistic scenarios. Further, we develop a variant of this algorithm that efficiently computes many different "personalized" PageRanks, and a variant that efficiently recomputes PageRank after node updates.

Maros2003 [49] I. Maros, *Computational Techniques of the Simplex Method*, Springer, Ed., ser. International Series in Operations Research & Management Science. Kluwer Academic, 2003, vol. 61.

Saad2003 [50] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Society for Industrial and Applied Mathematics, 2003. DOI: 10.1137/1.9780898718003.

Aykanat2004 [51] C. Aykanat, A. Pinar, and Ü. V. Çatalyürek, "Permuting sparse rectangular matrices into block-diagonal form," *SIAM Journal on Scientific Computing*, vol. 25, no. 6, pp. 1860–1879, 2004. DOI: 10.1137/S1064827502401953,

Abstract: We investigate the problem of permuting a sparse rectangular matrix into block-diagonal form. Block-diagonal form of a matrix grants an inherent parallelism for solving the deriving problem, as recently investigated in the context of mathematical programming, LU factorization, and QR factorization. To represent the nonzero structure of a matrix, we propose bipartite graph and hypergraph models that reduce the permutation problem to those of graph partitioning by vertex separator and hypergraph partitioning, respectively. Our experiments on a wide range of matrices, using the state-of-the-art graph and hypergraph partitioning tools MeTiS and PaToH,

revealed that the proposed methods yield very effective solutions both in terms of solution quality and runtime.

Bixby2004 [52] R. E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling, "Mixed-integer programming: A progress report," in *The Sharpest Cut*, M. Grötschel, Ed., ser. MOS-SIAM Series on Optimization. 2004, ch. 18, pp. 309–325. DOI: 10.1137/1.9780898718805.ch18. eprint: http://epubs.siam.org/doi/pdf/10.1137/1.9780898718805.ch18,
Abstract: Over the last several decades, from the early 1970s to as recently as 1998, the underlying solution technology in commercial mixed-integer programming codes remained essentially unchanged. In spite of important advances in the theory, many of these advances have clear computational value. In the last several years, that situation has changed. The result has been a major step forward in our ability to solve real-world mixed-integer programming problems.

Forrest2004 [53] J. Forrest, D. de la Nuez, and R. Lougee-Heimer, *CLP user guide*, IBM Research, 2004. [Online]. Available: http://www.coin-or.org/Clp/userguide/index.html.

Gabriel2004 [54] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Proceedings of the 11th European PVM/MPI Users' Group Meeting*, Budapest, HUN, Sep. 2004, pp. 97–104.

Spielman2004 [55] D. A. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *Journal of the ACM*, vol. 51, no. 3, pp. 385–463, May 2004, ISSN: 0004-5411. DOI: 10.1145/990308.990310,
Abstract: We introduce the smoothed analysis of algorithms, which continuously interpolates between the worst-case and average-case analyses of algorithms. In smoothed analysis, we measure the maximum over inputs of the expected performance of an algorithm under small random perturbations of that input. We measure this performance in terms of both the input size and the magnitude of the perturbations. We show that the simplex algorithm has smoothed complexity polynomial in the input size and the standard deviation of Gaussian perturbations.

Bosch2005 [56] R. Bosch and M. Tick, "Integer programming," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, E. K. Burke and G. Kendall, Eds. Springer, US, 2005, ch. 9, pp. 69–95. DOI: 10.1007/0-387-28356-0_3,
Abstract: Over the last 20 years, the combination of faster computers, more reliable data, and improved algorithms has resulted in the near-routine solution of many integer programs of practical interest. Integer programming models are used in a wide variety of applications, including scheduling, resource assignment, planning, supply chain design, auction design, and many, many others. In this tutorial, we outline some of the major themes involved in creating and solving integer programming models.

Diestel2005 [57] R. Diestel, *Graph Theory*, 3rd ed., ser. Graduate Texts in Mathematics. Springer–Verlag Heidelberg, Aug. 2005, vol. 173, ISBN: 3540261826.

Gay2005 [58] D. M. Gay. (Aug. 2005). Netlib LP benchmark documentation. Accessed on 7 November 2014, [Online]. Available: http://www.netlib.org/lp/data/readme.

Gregor2005 [59] D. Gregor and A. Lumsdaine, "The parallel BGL: A generic library for distributed graph computations," in *Parallel Object-Oriented Scientific Computing*, ser. POOSC '05, Jul. 2005. [Online]. Available: http://www.osl.iu.edu/publications/prints/2005/Gregor:POOSC:2005.pdf,
Abstract: This paper presents the Parallel BGL, a generic C++ library for distributed graph computation. Like the sequential Boost Graph Library (BGL) upon which it is based, the Parallel BGL applies the paradigm of generic programming to the domain of graph computations. Emphasizing efficient generic algorithms and the use of concepts to specify the requirements on type parameters, the Parallel BGL also provides flexible supporting data structures such as distributed adjacency lists and external property maps. The generic programming approach simultaneously stresses flexibility and efficiency, resulting in a parallel graph library that can adapt to various

data structures and communication models while retaining the efficiency of equivalent hand-coded programs. Performance data for selected algorithms are provided demonstrating the efficiency and scalability of the Parallel BGL.

Scholtes2005 [60] C. Scholtes, "A method to derive the cache performance of irregular applications on machines with direct mapped caches," *International Journal of Computational Science and Engineering*, vol. 1, no. 2-4, pp. 157–174, May 2005, ISSN: 1742-7185. DOI: 10.1504/IJCSE.2005.009700,
Abstract: A probabilistic method is presented to derive the cache performance of irregular applications on machines with direct mapped caches from inspection of the source code. The method has been applied to analyse both a program to multiply a sparse matrix with a dense matrix and a program for the Cholesky-factorisation of a sparse matrix. The resulting predictions are compared with measurements of the respective programs.

Asanovic2006 [61] K. Asanovic, B. C. Catanzaro, D. a. Patterson, and K. Yelick, "The landscape of parallel computing research : A view from berkeley," *Communications of the ACM*, vol. 52, no. 10, pp. 56–67, 2006, ISSN: 0001-0782. DOI: 10.1145/1562764.1562783,
Abstract: Writing programs that scale with increasing numbers of cores should be as easy as writing programs for sequential computers.

Bayliss2006 [62] S. Bayliss, C.-S. Bouganis, G. A. Constantinides, and W. Luk, "An FPGA implementation of the simplex algorithm," in *Proceedings of the IEEE International Conference on Field Programmable Technology*, ser. FPT '06, Bangkok, TH, Dec. 2006, pp. 49–56. DOI: 10.1109/FPT.2006.270294,
Abstract: Linear programming is applied to a large variety of scientific computing applications and industrial optimization problems. The Simplex algorithm is widely used for solving linear programs due to its robustness and scalability properties. However, application of the current software implementations of the Simplex algorithm to real-life optimization problems are time consuming when used as the bounding engine within an integer linear programming framework. This work aims to accelerate the Simplex algorithm by proposing a novel parameterizable hardware implementation of the algorithm on an FPGA. Evaluation of the proposed design using real problems demonstrates a speedup of up to 20 times over a highly optimized commercial software implementation running on a 3.4GHz Pentium 4 processor, which is itself 100 times faster than one of the main public domain solvers.

Bleris2006 [63] L. Bleris, P. Vouzis, M. Arnold, and M. Kothare, "A co-processor FPGA platform for the implementation of real-time model predictive control," in *American Control Conference*, ser. ACC '06, Minneapolis, MN, USA, Jun. 2006, 6 pp. DOI: 10.1109/ACC.2006.1656499,
Abstract: In order to effectively control nonlinear and multivariable models, and to incorporate constraints on system states, inputs and outputs (bounds, rate of change), a suitable (sometimes necessary) controller is model predictive control (MPC). MPC is an optimization-based control scheme that requires abundant matrix operations for the calculation of the optimal control moves. In this work we propose a mixed software and hardware embedded MPC implementation. Using a codesign step and based on profiling results, we decompose the optimization algorithm into two parts: one that fits into a host processor and one that fits into a custom made unit that performs the computationally demanding arithmetic operations. The profiling results and information on the co-processor design are provided.

Choi2006 [64] S. Choi, "Iterative methods for singular linear equations and least-square problems," PhD thesis, Stanford University, 2006. [Online]. Available: https://web.stanford.edu/group/SOL/dissertations/sou-cheng-choi-thesis.pdf,
Abstract: CG, MINRES, and SYMMLQ are Krylov subspace methods for solving large symmetric systems of linear equations. CG (the conjugate-gradient method) is reliable on positive-definite systems, while MINRES and SYMMLQ are designed for indefinite systems. When these methods are applied to an inconsistent system (that is, a singular symmetric least-squares problem), CG could break down and SYMMLQ's solution could explode, while MINRES would give a least-squares solution but not necessarily the minimum-length solution (often called the pseudoinverse solution).

This understanding motivates us to design a MINRES-like algorithm to compute minimum-length solutions to singular symmetric systems.
MINRES uses QR factors of the tridiagonal matrix from the Lanczos process (where R is upper-tridiagonal). Our algorithm uses a QLP decomposition (where rotations on the right reduce R to lower-tridiagonal form), and so we call it MINRES-QLP. On singular or nonsingular systems, MINRES-QLP can give more accurate solutions than MINRES or SYMMLQ. We derive preconditioned MINRES-QLP, new stopping rules, and better estimates of the solution and residual norms, the matrix norm and condition number.
For a singular matrix of arbitrary shape, we observe that null vectors can be obtained by solving least-squares problems involving the transpose of the matrix. For sparse rectangular matrices, this suggests an application of the iterative solver LSQR. In the square case, MINRES, MINRES-QLP, or LSQR are applicable. Results are given for solving homogeneous systems, computing the stationary probability vector for Markov Chain models, and finding null vectors for sparse systems arising in helioseismology.

Davis2006 [65] T. A. Davis, *Direct Methods for Sparse Linear Systems*, ser. Fundamentals of Algorithms. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2006, vol. 2, ISBN: 0898716136.

Gilbert2006 [66] J. R. Gilbert, S. Reinhardt, and V. B. Shah, "High-performance graph algorithms from parallel sparse matrices," in *Proceedings of the 8th International Conference on Applied Parallel Computing: State of the Art in Scientific Computing*, ser. PARA '06, Umeå, SE: Springer-Verlag, 2006, pp. 260–269, ISBN: 978-3-540-75754-2. [Online]. Available: `http://dl.acm.org/citation.cfm?id=1775097`,
Abstract: Large-scale computation on graphs and other discrete structures is becoming increasingly important in many applications, including computational biology, web search, and knowledge discovery. High-performance combinatorial computing is an infant field, in sharp contrast with numerical scientific computing.
We argue that many of the tools of high-performance numerical computing - in particular, parallel algorithms and data structures for computation with sparse matrices - can form the nucleus of a robust infrastructure for parallel computing on graphs. We demonstrate this with an implementation of a graph analysis benchmark using the sparse matrix infrastructure in Star-P, our parallel dialect of the MATLAB programming language.

Han2006 [67] H. Han and C.-W. Tseng, "Exploiting locality for irregular scientific codes," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 7, pp. 606–618, Jul. 2006, ISSN: 1045-9219. DOI: `10.1109/TPDS.2006.88`,
Abstract: Irregular scientific codes experience poor cache performance due to their irregular memory access patterns. In this paper, we present two new locality improving techniques for irregular scientific codes. Our techniques exploit geometric structures hidden in data access patterns and computation structures. Our new data reordering (GPART) finds the graph structure within data accesses and applies hierarchical clustering. Quality partitions are constructed quickly by clustering multiple neighbor nodes with priority on nodes with high degree and repeating a few passes. Overhead is kept low by clustering multiple nodes in each pass and considering only edges between partitions. Our new computation reordering (Z-SORT) treats the values of index arrays as coordinates and reorders corresponding computations in Z-curve order. Applied to dense inputs, Z-SORT achieves performance close to data reordering combined with other computation reordering but without the overhead involved in data reordering. Experiments on irregular scientific codes for a variety of meshes show locality optimization techniques are effective for both sequential and parallelized codes, improving performance by 60-87 percent. GPART achieved within 1-2 percent of the performance of more sophisticated partitioning algorithms, but with one third of the overhead. Z-SORT also yields the performance improvement of 64 percent for dense inputs, which is comparable with data reordering combined with computation reordering.

Nocedal2006 [68] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd. New York: Springer, 2006.

Waechter2006 [69] A. Waechter and L. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, May 2006, ISSN: 0025-5610. DOI: 10.1007/s10107-004-0559-y,
Abstract: We present a primal-dual interior-point algorithm with a filter line-search method for nonlinear programming. Local and global convergence properties of this method were analyzed in previous work. Here we provide a comprehensive description of the algorithm, including the feasibility restoration phase for the filter method, second-order corrections, and inertia correction of the KKT matrix. Heuristics are also considered that allow faster performance. This method has been implemented in the IPOPT code, which we demonstrate in a detailed numerical study based on 954 problems from the CUTEr test set. An evaluation is made of several line-search options, and a comparison is provided with two state-of-the-art interior-point codes for nonlinear programming.

Wolter2006 [70] K. Wolter, "Implementation of cutting plane separators for mixed integer programs," PhD thesis, Technische Universität Berlin, 2006.

Zhang2006 [71] C. Zhang, C. Ding, M. Ogihara, Y. Zhong, and Y. Wu, "A hierarchical model of data locality," in *Conference Record of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. POPL '06, Charleston, SC, USA: ACM, 2006, pp. 16–29, ISBN: 1-59593-027-2. DOI: 10.1145/1111037.1111040,
Abstract: In POPL 2002, Petrank and Rawitz showed a universal result—finding optimal data placement is not only NP-hard but also impossible to approximate within a constant factor if P $\neq$ NP. Here we study a recently published concept called reference affinity, which characterizes a group of data that are always accessed together in computation. On the theoretical side, we give the complexity for finding reference affinity in program traces, using a novel reduction that converts the notion of distance into satisfiability. We also prove that reference affinity automatically captures the hierarchical locality in divide-and-conquer computations including matrix solvers and N-body simulation. The proof establishes formal links between computation patterns in time and locality relations in space.On the practical side, we show that efficient heuristics exist. In particular, we present a sampling method and show that it is more effective than the previously published technique, especially for data that are often but not always accessed together. We show the effect on generated and real traces. These theoretical and empirical results demonstrate that effective data placement is still attainable in general-purpose programs because common (albeit not all) locality patterns can be precisely modeled and efficiently analyzed.

Cafieri2007 [72] S. Cafieri, M. D'Apuzzo, V. De Simone, and D. di Serafino, "On the iterative solution of KKT systems in potential reduction software for large-scale quadratic problems," *Computational Optimization and Applications*, vol. 38, no. 1, pp. 27–45, 2007, ISSN: 1573-2894. DOI: 10.1007/s10589-007-9035-y,
Abstract: Iterative solvers appear to be very promising in the development of efficient software, based on Interior Point methods, for large-scale nonlinear optimization problems. In this paper we focus on the use of preconditioned iterative techniques to solve the KKT system arising at each iteration of a Potential Reduction method for convex Quadratic Programming. We consider the augmented system approach and analyze the behaviour of the Constraint Preconditioner with the Conjugate Gradient algorithm. Comparisons with a direct solution of the augmented system and with MOSEK show the effectiveness of the iterative approach on large-scale sparse problems.

Grigori2007 [73] L. Grigori, J. Demmel, and X. Li, "Parallel symbolic factorization for sparse LU with static pivoting," *SIAM Journal on Scientific Computing*, vol. 29, no. 3, pp. 1289–1314, 2007. DOI: 10.1137/050638102,
Abstract: This paper presents the design and implementation of a memory scalable parallel symbolic factorization algorithm for general sparse unsymmetric matrices. Our parallel algorithm uses a graph partitioning approach, applied to the graph of $|A| + |A|^T$, to partition the matrix in such a way that is good for sparsity preservation as well as for parallel factorization. The partitioning yields a so-called separator tree which represents the dependencies among the computations. We use the separator tree to distribute the input matrix over the processors using a block cyclic approach and a subtree to subprocessor mapping. The parallel algorithm performs a bottom-up

traversal of the separator tree. With a combination of right-looking and left-looking partial factorizations, the algorithm obtains one column structure of L and one row structure of U at each step. The algorithm is implemented in C and MPI. From a performance study on large matrices, we show that the parallel algorithm significantly reduces the memory requirement of the symbolic factorization step, as well as the overall memory requirement of the parallel solver. It also often reduces the runtime of the sequential algorithm, which is already relatively small. In general, the parallel algorithm prevents the symbolic factorization step from being a time or memory bottleneck of the parallel solver.

Knyazev2007 [74] A. V. Knyazev and I. Lashuk, "Steepest descent and conjugate gradient methods with variable preconditioning," *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 4, pp. 1267–1280, 2007. DOI: 10.1137/060675290,
Abstract: We analyze the conjugate gradient (CG) method with variable preconditioning for solving a linear system with a real symmetric positive definite (SPD) matrix of coefficients A. We assume that the preconditioner is SPD on each step, and that the condition number of the preconditioned system matrix is bounded above by a constant independent of the step number. We show that the CG method with variable preconditioning under this assumption may not give improvement, compared to the steepest descent (SD) method. We describe the basic theory of CG methods with variable preconditioning with the emphasis on worst case scenarios, and provide complete proofs of all facts not available in the literature. We give a new elegant geometric proof of the SD convergence rate bound. Our numerical experiments, comparing the preconditioned SD and CG methods, not only support and illustrate our theoretical findings, but also reveal two surprising and potentially practically important effects. First, we analyze variable preconditioning in the form of inner-outer iterations. In previous such tests, the unpreconditioned CG inner iterations are applied to an artificial system with some fixed preconditioner as a matrix of coefficients. We test a different scenario, where the unpreconditioned CG inner iterations solve linear systems with the original system matrix A. We demonstrate that the CG-SD inner-outer iterations perform as well as the CG-CG inner-outer iterations in these tests. Second, we compare the CG methods using a two-grid preconditioning with fixed and randomly chosen coarse grids, and observe that the fixed preconditioner method is twice as slow as the method with random preconditioning.

Achterberg2008 [75] T. Achterberg, "Constraint integer programming," PhD thesis, Technische Universität Berlin, 2008. [Online]. Available: https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/1112,
Abstract: This thesis introduces the novel paradigm of constraint integer programming (CIP), which integrates constraint programming (CP) and mixed integer programming (MIP) modeling and solving techniques. It is supplemented by the software SCIP, which is a solver and framework for constraint integer programming that also features SAT solving techniques. SCIP is freely available in source code for academic and non-commercial purposes.
Our constraint integer programming approach is a generalization of MIP that allows for the inclusion of arbitrary constraints, as long as they turn into linear constraints on the continuous variables after all integer variables have been fixed. The constraints, may they be linear or more complex, are treated by any combination of CP and MIP techniques: the propagation of the domains by constraint specific algorithms, the generation of a linear relaxation and its solving by LP methods, and the strengthening of the LP by cutting plane separation.
The current version of SCIP comes with all of the necessary components to solve mixed integer programs. In the thesis, we cover most of these ingredients and present extensive computational results to compare different variants for the individual building blocks of a MIP solver. We focus on the algorithms and their impact on the overall performance of the solver.
In addition to mixed integer programming, the thesis deals with chip design verification, which is an important topic of electronic design automation. Chip manufacturers have to make sure that the logic design of a circuit conforms to the specification of the chip. Otherwise, the chip would show an erroneous behavior that may cause failures in the device where it is employed. An important subproblem of chip design verification is the property checking problem, which is to verify whether a circuit satisfies a specified property. We show how this problem can be modeled as constraint

integer program and provide a number of problem-specific algorithms that exploit the structure of the individual constraints and the circuit as a whole. Another set of extensive computational benchmarks compares our CIP approach to the current state-of-the-art SAT methodology and documents the success of our method.

Baskaran2008 [76] M. M. Baskaran and R. Bordawekar, "Optimizing sparse matrix-vector multiplication on GPUs using compile-time and run-time strategies," IBM Research Division, Tech. Rep., 2008. [Online]. Available: http://domino.watson.ibm.com/library/CyberDig.nsf/1e4115aea78b6e7c85256b360066f0d4/1d32f6d23b99f78985257522006183390penDocument,
Abstract: We are witnessing the emergence of Graphics Processor units (GPUs) as powerful massively parallel systems. Furthermore, the introduction of new APIs for general-purpose computations on GPUs, namely CUDA from NVIDIA, Stream SDK from AMD, and OpenCL, makes GPUs an attractive choice for high-performance numerical and scientific computing. Sparse Matrix-Vector multiplication (SpMV) is one of the most important and heavily used kernels in scientific computing. However with indirect and irregular memory accesses resulting in more memory accesses per floating point operation, optimization of SpMV kernel is a significant challenge in any architecture. In this paper, we evaluate the various challenges in developing a high-performance SpMV kernel on NVIDIA GPUs using the CUDA programming model and propose a framework that employs both compile-time and run-time optimizations. The compile-time optimizations include: (1) exploiting synchronization-free parallelism, (2) optimized thread mapping based on the affinity towards optimal memory access pattern, (3) optimized off-chip memory access to tolerate the high latency, and (4) exploiting data reuse. The runtime optimizations involve a runtime inspection of the sparse matrix to determine dense non-zero sub-blocks, which facilitate the reuse of input vector elements while execution. We propose a new blocked storage format for storing and accessing elements of a sparse matrix in an optimized manner from the GPU memories. We evaluate our optimizations over two classes of NVIDIA GPU chips, namely, GeForce 8800 GTX and GeForce GTX 280, and we compare the performance of our approach with that of existing parallel SpMV implementations such as the one from NVIDIA's CUDPP library and the one implemented using optimal segmented scan primitive. Our approach outperforms the other existing implementations by a factor of 2 to 4. Using our framework, we achieve a peak SpMV performance that is 70% of the performance observed for SpMV computations using dense matrices stored in sparse format.

Boland2008 [77] D. Boland and G. Constantinides, "An FPGA-based implementation of the MINRES algorithm," in *Proceedings of the International Conference on Field Programmable Logic and Applications*, ser. FPL '08, Heidelberg, DEU, Sep. 2008, pp. 379–384. DOI: 10.1109/FPL.2008.4629967,
Abstract: Due to continuous improvements in the resources available on FPGAs, it is becoming increasingly possible to accelerate floating point algorithms. The solution of a system of linear equations forms the basis of many problems in engineering and science, but its calculation is highly time consuming. The minimum residual algorithm (MINRES) is one method to solve this problem, and is highly effective provided the matrix exhibits certain characteristics. This paper examines an IEEE 754 single precision floating point implementation of the MINRES algorithm on an FPGA. It demonstrates that through parallelisation and heavy pipelining of all floating point components it is possible to achieve a sustained performance of up to 53 GFLOPS on the Virtex5-330T. This compares favourably to other hardware implementations of floating point matrix inversion algorithms, and corresponds to an improvement of nearly an order of magnitude compared to a software implementation.

Cong2008 [78] G. Cong, S. Kodali, S. Krishnamoorthy, D. Lea, V. Saraswat, and T. Wen, "Solving large, irregular graph problems using adaptive work-stealing," in *Proceedings of the 37th International Conference on Parallel Processing*, ser. ICPP '08, Portland, OR, USA, Sep. 2008, pp. 536–545. DOI: 10.1109/ICPP.2008.88,
Abstract: Solving large, irregular graph problems efficiently is challenging. Current software systems and commodity multiprocessors do not support fine-grained, irregular parallelism well. We present XWS, the X10 work stealing framework, an open-source runtime for the parallel programming language X10 and a library to be used directly by application writers. XWS extends the Cilk work-

stealing framework with several features necessary to efficiently implement graph algorithms, viz., support for improperly nested procedures, global termination detection, and phased computation. We also present a strategy to adaptively control the granularity of parallel tasks in the work-stealing scheme, depending on the instantaneous size of the work queue. We compare the performance of the XWS implementations of spanning tree algorithms with that of the hand-written C and Cilk implementations using various graph inputs. We show that XWS programs (written in Java) scale and exhibit comparable or better performance.

Garland2008 [79] M. Garland, S. Le Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Y. Zhang, and V. Volkov, "Parallel computing experiences with CUDA," *IEEE Micro*, vol. 28, no. 4, pp. 13–27, Jul. 2008, ISSN: 0272-1732. DOI: 10.1109/MM.2008.57,
Abstract: The CUDA programming model provides a straightforward means of describing inherently parallel computations, and NVIDIA's Tesla GPU architecture delivers high computational throughput on massively parallel problems. This article surveys experiences gained in applying CUDA to a diverse set of problems and the parallel speedups over sequential codes running on traditional CPU architectures attained by executing key computations on the GPU.

Jung2008 [80] J. H. Jung and D. P. O'Leary, "Implementing an interior point method for linear programs on a CPU-GPU system.," eng, *Electronic Transactions on Numerical Analysis*, vol. 28, pp. 174–189, 2008. [Online]. Available: http://eudml.org/doc/117656,
Abstract: Graphics processing units (GPUs), present in every laptop and desktop computer, are potentially powerful computational engines for solving numerical problems. We present a mixed precision CPU-GPU algorithm for solving linear programming problems using interior point methods. This algorithm, based on the rectangular-packed matrix storage scheme of Gunnels and Gustavson, uses the GPU for computationally intensive tasks such as matrix assembly, Cholesky factorization, and forward and back substitution. Comparisons with a CPU implementation demonstrate that we can improve performance by using the GPU for sufficiently large problems. Since GPU architectures and programming languages are rapidly evolving, we expect that GPUs will be an increasingly attractive tool for matrix computation in the future.

OpenMP2008 [81] *OpenMP application program interface*, 3rd ed., OpenMP Architecture Review Board, May 2008. [Online]. Available: http://www.openmp.org/mp-documents/spec30.pdf.

Rodriguez2008 [82] G. Rodriguez, *Algoritmi numerici*. Pitagora Editrice, 2008, p. 117, ISBN: 88-371-1714-0.

Ryoo2008 [83] S. Ryoo, C. I. Rodrigues, S. S. Stone, S. S. Baghsorkhi, S.-Z. Ueng, J. A. Stratton, and W.-m. W. Hwu, "Program optimization space pruning for a multithreaded GPU," in *Proceedings of the 6th Annual IEEE/ACM International Symposium on Code Generation and Optimization*, ser. CGO '08, Boston, MA, USA: ACM, 2008, pp. 195–204, ISBN: 978-1-59593-978-4. DOI: 10.1145/1356058.1356084,
Abstract: Program optimization for highly-parallel systems has historically been considered an art, with experts doing much of the performance tuning by hand. With the introduction of inexpensive, single-chip, massively parallel platforms, more developers will be creating highly-parallel applications for these platforms, who lack the substantial experience and knowledge needed to maximize their performance. This creates a need for more structured optimization methods with means to estimate their performance effects. Furthermore these methods need to be understandable by most programmers. This paper shows the complexity involved in optimizing applications for one such system and one relatively simple methodology for reducing the workload involved in the optimization process.
This work is based on one such highly-parallel system, the GeForce 8800 GTX using CUDA. Its flexible allocation of resources to threads allows it to extract performance from a range of applications with varying resource requirements, but places new demands on developers who seek to maximize an application's performance. We show how optimizations interact with the architecture in complex ways, initially prompting an inspection of the entire configuration space to find the optimal configuration. Even for a seemingly simple application such as matrix multiplication, the optimal configuration can be unexpected. We then present metrics derived from static code that capture the first-order factors of performance. We demonstrate how these metrics can be used to

prune many optimization configurations, down to those that lie on a Pareto-optimal curve. This reduces the optimization space by as much as 98% and still finds the optimal configuration for each of the studied applications.

Sager2008 [84] S. Sager, C. Kirches, and H. G. Bock, "Fast solution of periodic optimal control problems in automobile test-driving with gear shifts," in *Proceedings of the 47th IEEE Conference on Decision and Control*, ser. CDC '08, Cancun, MEX, Dec. 2008, pp. 1563–1568. DOI: 10.1109/CDC.2008.4739014,
Abstract: Optimal control problems involving time-dependent decisions from a finite set have gained much interest lately, as they occur in practical applications with a high potential for optimization. A typical application is automobile driving with gear shifts. Recent work [7], [8], [9] lead to a tremendous speedup in computational times to obtain optimal solutions, allowing for more complex scenarios. In this paper we extend a benchmark mixed-integer optimal control problem to a more complicated case in which a periodic solution on a closed track is considered. Our generic solution approach is based on a convexification and relaxation of the integer control constraint. It may also be used for other objectives, such as energy minimization. Using the direct multiple shooting method we solve the new benchmark problem and present numerical results.

Agullo2009 [85] E. Agullo, J. Demmel, J. Dongarra, B. Hadri, J. Kurzak, J. Langou, H. Ltaief, P. Luszczek, and S. Tomov, "Numerical linear algebra on emerging architectures: The PLASMA and MAGMA projects," *Journal of Physics: Conference Series*, vol. 180, no. 1, p. 012 037, 2009. [Online]. Available: http://stacks.iop.org/1742-6596/180/i=1/a=012037,
Abstract: The emergence and continuing use of multi-core architectures and graphics processing units require changes in the existing software and sometimes even a redesign of the established algorithms in order to take advantage of now prevailing parallelism. Parallel Linear Algebra for Scalable Multi-core Architectures (PLASMA) and Matrix Algebra on GPU and Multics Architectures (MAGMA) are two projects that aims to achieve high performance and portability across a wide range of multi-core architectures and hybrid systems respectively. We present in this document a comparative study of PLASMA's performance against established linear algebra packages and some preliminary results of MAGMA on hybrid multi-core and GPU systems.

Bakhoda2009 [86] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*, ser. ISPASS '09, Boston, MA, USA, Apr. 2009, pp. 163–174. DOI: 10.1109/ISPASS.2009.4919648,
Abstract: Modern Graphic Processing Units (GPUs) provide sufficiently flexible programming models that understanding their performance can provide insight in designing tomorrow's manycore processors, whether those are GPUs or otherwise. The combination of multiple, multithreaded, SIMD cores makes studying these GPUs useful in understanding tradeoffs among memory, data, and thread level parallelism. While modern GPUs offer orders of magnitude more raw computing power than contemporary CPUs, many important applications, even those with abundant data level parallelism, do not achieve peak performance. This paper characterizes several non-graphics applications written in NVIDIA's CUDA programming model by running them on a novel detailed microarchitecture performance simulator that runs NVIDIA's parallel thread execution (PTX) virtual instruction set. For this study, we selected twelve non-trivial CUDA applications demonstrating varying levels of performance improvement on GPU hardware (versus a CPU-only sequential version of the application). We study the performance of these applications on our GPU performance simulator with configurations comparable to contemporary high-end graphics cards. We characterize the performance impact of several microarchitecture design choices including choice of interconnect topology, use of caches, design of memory controller, parallel workload distribution mechanisms, and memory request coalescing hardware. Two observations we make are (1) that for the applications we study, performance is more sensitive to interconnect bisection bandwidth rather than latency, and (2) that, for some applications, running fewer threads concurrently than on-chip resources might otherwise allow can improve performance by reducing contention in the memory system.

Bell2009 [87] N. Bell and M. Garland, "Implementing sparse matrix-vector multiplication on throughput-oriented processors," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, ser. SC '09, Portland, Oregon: ACM, 2009, pp. 1–11, ISBN: 978-1-60558-744-8. DOI: 10.1145/1654059.1654078,
Abstract: Sparse matrix-vector multiplication (SpMV) is of singular importance in sparse linear algebra. In contrast to the uniform regularity of dense linear algebra, sparse operations encounter a broad spectrum of matrices ranging from the regular to the highly irregular. Harnessing the tremendous potential of throughput-oriented processors for sparse operations requires that we expose substantial fine-grained parallelism and impose sufficient regularity on execution paths and memory access patterns. We explore SpMV methods that are well-suited to throughput-oriented architectures like the GPU and which exploit several common sparsity classes. The techniques we propose are efficient, successfully utilizing large percentages of peak bandwidth. Furthermore, they deliver excellent total throughput, averaging 16 GFLOP/s and 10 GFLOP/s in double precision for structured grid and unstructured mesh matrices, respectively, on a GeForce GTX 285. This is roughly 2.8 times the throughput previously achieved on Cell BE and more than 10 times that of a quad-core Intel Clovertown system.

Buatois2009 [88] L. Buatois, G. Caumon, and B. Lévy, "Concurrent number cruncher: A GPU implementation of a general sparse linear solver," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 24, no. 3, pp. 205–223, 2009. DOI: 10.1080/17445760802337010,
Abstract: A wide class of numerical methods needs to solve a linear system, where the matrix pattern of non-zero coefficients can be arbitrary. These problems can greatly benefit from highly multithreaded computational power and large memory bandwidth available on graphics processor units (GPUs), especially since dedicated general purpose APIs such as close-to-metal (CTM) (AMD–ATI) and compute unified device architecture (CUDA) (NVIDIA) have appeared. CUDA even provides a BLAS implementation, but only for dense matrices (CuBLAS). Other existing linear solvers for the GPU are also limited by their internal matrix representation. This paper describes how to combine recent GPU programming techniques and new GPU dedicated APIs with high performance computing strategies (namely block compressed row storage (BCRS), register blocking and vectorization), to implement a sparse general-purpose linear solver. Our implementation of the Jacobi-preconditioned conjugate gradient algorithm outperforms by up to a factor of 6.0× leading-edge CPU counterparts, making it attractive for applications which are content with single precision.

Buttari2009 [89] A. Buttari, J. Langou, J. Kurzak, and J. Dongarra, "A class of parallel tiled linear algebra algorithms for multicore architectures," *Parallel Computing*, vol. 35, no. 1, pp. 38–53, 2009, ISSN: 0167-8191. DOI: 10.1016/j.parco.2008.10.002,
Abstract: As multicore systems continue to gain ground in the high performance computing world, linear algebra algorithms have to be reformulated or new algorithms have to be developed in order to take advantage of the architectural features on these new processors. Fine grain parallelism becomes a major requirement and introduces the necessity of loose synchronization in the parallel execution of an operation. This paper presents algorithms for the Cholesky, LU and QR factorization where the operations can be represented as a sequence of small tasks that operate on square blocks of data. These tasks can be dynamically scheduled for execution based on the dependencies among them and on the availability of computational resources. This may result in out of order execution of tasks which will completely hide the presence of intrinsically sequential tasks in the factorization. Performance comparisons are presented with LAPACK algorithms where parallelism can only be exploited at the level of the BLAS operations and vendor implementations.

Christen2009 [90] M. Christen, O. Schenk, E. Neufeld, P. Messmer, and H. Burkhart, "Parallel data-locality aware stencil computations on modern micro-architectures," in *Proceedings of the 2009 IEEE International Symposium on Parallel Distributed Processing*, ser. IPDPS '09, Rome, IT, May 2009, pp. 1–10. DOI: 10.1109/IPDPS.2009.5161031,
Abstract: Novel micro-architectures including the Cell Broadband Engine Architecture and graphics processing units are attractive platforms for compute-intensive simulations. This paper focuses on stencil computations arising in the context of a biomedical simulation and presents performance

benchmarks on both the Cell BE and GPUs and contrasts them with a benchmark on a traditional CPU system. Due to the low arithmetic intensity of stencil computations, typically only a fraction of the peak performance of the compute hardware is reached. An algorithm is presented, which reduces the bandwidth requirements and thereby improves performance by exploiting temporal locality of the data. We report on performance improvements over CPU implementations.

Hong2009 [91] S. Hong and H. Kim, "An analytical model for a GPU architecture with memory-level and thread-level parallelism awareness," in *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ser. ISCA '09, Austin, TX, USA: ACM, 2009, pp. 152–163, ISBN: 978-1-60558-526-0. DOI: 10.1145/1555754.1555775,
Abstract: GPU architectures are increasingly important in the multi-core era due to their high number of parallel processors. Programming thousands of massively parallel threads is a big challenge for software engineers, but understanding the performance bottlenecks of those parallel programs on GPU architectures to improve application performance is even more difficult. Current approaches rely on programmers to tune their applications by exploiting the design space exhaustively without fully understanding the performance characteristics of their applications.
To provide insights into the performance bottlenecks of parallel applications on GPU architectures, we propose a simple analytical model that estimates the execution time of massively parallel programs. The key component of our model is estimating the number of parallel memory requests (we call this the memory warp parallelism) by considering the number of running threads and memory bandwidth. Based on the degree of memory warp parallelism, the model estimates the cost of memory requests, thereby estimating the overall execution time of a program. Comparisons between the outcome of the model and the actual execution time in several GPUs show that the geometric mean of absolute error of our model on micro-benchmarks is 5.4% and on GPU computing applications is 13.3%. All the applications are written in the CUDA programming language.

Kulkarni2009a [92] M. Kulkarni, M. Burtscher, C. Cascaval, and K. Pingali, "Lonestar: A suite of parallel irregular programs," in *Proceedings of the 2009 IEEE International Symposium on Performance Analysis of Systems and Software*, ser. ISPASS '09, Boston, MA, USA, Apr. 2009, pp. 65–76. DOI: 10.1109/ISPASS.2009.4919639,
Abstract: Until recently, parallel programming has largely focused on the exploitation of data-parallelism in dense matrix programs. However, many important application domains, including meshing, clustering, simulation, and machine learning, have very different algorithmic foundations: they require building, computing with, and modifying large sparse graphs. In the parallel programming literature, these types of applications are usually classified as irregular applications, and relatively little attention has been paid to them. To study and understand the patterns of parallelism and locality in sparse graph computations better, we are in the process of building the Lonestar benchmark suite. In this paper, we characterize the first five programs from this suite, which target domains like data mining, survey propagation, and design automation. We show that even such irregular applications often expose large amounts of parallelism in the form of amorphous data-parallelism. Our speedup numbers demonstrate that this new type of parallelism can successfully be exploited on modern multi-core machines.

Kulkarni2009 [93] M. Kulkarni, M. Burtscher, R. Inkulu, K. Pingali, and C. Caşcaval, "How much parallelism is there in irregular applications?" In *Proceedings of the 14th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '09, Raleigh, NC, USA: ACM, 2009, pp. 3–14, ISBN: 978-1-60558-397-6. DOI: 10.1145/1504176.1504181,
Abstract: Irregular programs are programs organized around pointer-based data structures such as trees and graphs. Recent investigations by the Galois project have shown that many irregular programs have a generalized form of data-parallelism called amorphous data-parallelism. However, in many programs, amorphous data-parallelism cannot be uncovered using static techniques, and its exploitation requires runtime strategies such as optimistic parallel execution. This raises a natural question: how much amorphous data-parallelism actually exists in irregular programs?
In this paper, we describe the design and implementation of a tool called ParaMeter that produces parallelism profiles for irregular programs. Parallelism profiles are an abstract measure of the amount of amorphous data-parallelism at different points in the execution of an algorithm,

independent of implementation-dependent details such as the number of cores, cache sizes, load-balancing, etc. ParaMeter can also generate constrained parallelism profiles for a fixed number of cores. We show parallelism profiles for seven irregular applications, and explain how these profiles provide insight into the behavior of these applications.

Mayer2009 [94] J. Mayer, "Parallel algorithms for solving linear systems with sparse triangular matrices," *Computing*, vol. 86, no. 4, pp. 291–312, Sep. 2009, ISSN: 0010-485X. DOI: 10.1007/s00607-009-0066-3,
Abstract: In this article, we present two new algorithms for solving given triangular systems in parallel on a shared memory architecture. Multilevel incomplete LU factorization based preconditioners, which have been very successful for solving linear systems iteratively, require these triangular solves. Hence, the algorithms presented here can be seen as parallelizing the application of these preconditioners. The first algorithm solves the triangular matrix by block anti-diagonals. The drawback of this approach is that it can be difficult to choose an appropriate block structure. On the other hand, if a good block partition can be found, this algorithm can be quite effective. The second algorithm takes a hybrid approach by solving the triangular system by block columns and anti-diagonals. It is usually as effective as the first algorithm, but the block structure can be chosen in a nearly optimal manner. Although numerical results indicate that the speed-up can be fairly good, systems with matrices having a strong diagonal structure or narrow bandwidth cannot be solved effectively in parallel. Hence, for these matrices, the results are disappointing. On the other hand, the results are better for matrices having a more uniform distribution of non-zero elements. Although not discussed in this article, these algorithms can possibly be adapted for distributed memory architectures.

Muller2009 [95] C. Müller, S. Frey, M. Strengert, C. Dachsbacher, and T. Ertl, "A compute unified system architecture for graphics clusters incorporating data locality," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 4, pp. 605–617, Jul. 2009, ISSN: 1077–2626. DOI: 10.1109/TVCG.2008.188,
Abstract: We present a development environment for distributed GPU computing targeted for multi-GPU systems, as well as graphics clusters. Our system is based on CUDA and logically extends its parallel programming model for graphics processors to higher levels of parallelism, namely, the PCI bus and network interconnects. While the extended API mimics the full function set of current graphics hardware-including the concept of global memory-on all distribution layers, the underlying communication mechanisms are handled transparently for the application developer. To allow for high scalability, in particular for network-interconnected environments, we introduce an automatic GPU-accelerated scheduling mechanism that is aware of data locality. This way, the overall amount of transmitted data can be heavily reduced, which leads to better GPU utilization and faster execution. We evaluate the performance and scalability of our system for bus and especially network-level parallelism on typical multi-GPU systems and graphics clusters.

NvidiaFermi2009 [96] "NVIDIA's next generation CUDA compute architecture: Fermi," NVIDIA Corporation, Tech. Rep., 2009. [Online]. Available: http://www.nvidia.it/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf.

Shimai2009 [97] Y. Shimai, J. Tani, H. Noguchi, H. Kawaguchi, and M. Yoshimoto, "FPGA implementation of mixed integer quadratic programming solver for mobile robot control," in *Proceedings of the 2009 International Conference on Field-Programmable Technology*, ser. FPT '09, Sidney, AUS, Dec. 2009, pp. 447–450. DOI: 10.1109/FPT.2009.5377635,
Abstract: We propose a high-speed mixed integer quadratic programming (MIQP) solver on an FPGA. The MIQP solver can be applied to various optimizing applications including real-time robot control. In order to rapidly solve the MIQP problem, we implement reusing a first solution (first point), pipeline architecture, and multi-core architecture on the single FPGA. By making use of them, we confirmed that 79.5% of the cycle times are reduced, compared with straightforward sequential processing. The operating frequency is 67 MHz, although a core 2 duo PC requires 3.16 GHz in processing the same size problem. The power consumption of the MIQP solver is 4.2 W.

Spampinato2009 [98] D. D. Spampinato and A. C. Elster, "Linear optimization on modern GPUs," in *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing*, ser. IPDPS '09, Rome, IT, May 2009, pp. 1–8. DOI: 10.1109/IPDPS.2009.5161106,
Abstract: Optimization algorithms are becoming increasingly more important in many areas, such as finance and engineering. Typically, real problems involve several hundreds of variables, and are subject to as many constraints. Several methods have been developed trying to reduce the theoretical time complexity. Nevertheless, when problems exceed reasonable sizes they end up being very computationally intensive. Heterogeneous systems composed by coupling commodity CPUs and GPUs are becoming relatively cheap, highly performing systems. Recent developments of GPGPU technologies give even more powerful control over them. In this paper, we show how we use a revised simplex algorithm for solving linear programming problems originally described by Dantzig for both our CPU and GPU implementations. Previously, this approach has showed not to scale beyond around 200 variables. However, by taking advantage of modern libraries such as ATLAS for matrix-matrix multiplication, and the NVIDIA CUDA programming library on recent GPUs, we show that we can scale to problem sizes up to at least 2000 variables in our experiments for both architectures. On the GPU, we also achieve an appreciable precision on large problems with thousands of variables and constraints while achieving between 2× and 2.5× speed-ups over the serial ATLAS-based CPU version. With further tuning of both the algorithm and its implementations, even better results should be achievable for both the CPU and GPU versions.

Wu2009 [99] C.-H. Wu, S. O. Memik, and S. Merothra, "FPGA implementation of the interior-point algorithm with applications to collision detection," in *Proceedings of the 17th IEEE Symposium on Field Programmable Custom Computing Machines*, ser. FCCM '09, Napa Valley, CA, USA, Apr. 2009, pp. 295–298. DOI: 10.1109/FCCM.2009.38,
Abstract: The interior-point algorithm is a powerful method for solving a Linear Program (LP). A variety of optimization problems can be formulated as LPs. Often times the limiting factor of deploying an algorithm to solve LPs in a high performance system is the run-time efficiency. In this paper, we present the FPGA implementation of an affine interior-point algorithm that is designed to solve LPs. Specifically, we present the application of this algorithm to solving the LP for the real-time collision detection. The most important feature that distinguishes this particular algorithm from other collision detection methods is its superior ability to perform detection between pairs of objects undergoing fast rotational and translational motions.

Zhang2009 [100] Y. Zhang, Y. H. Shalabi, R. Jain, K. K. Nagar, and J. D. Bakos, "FPGA vs. GPU for sparse matrix vector multiply," in *Proceedings of the 2009 International Conference on Field-Programmable Technology*, ser. FPT '09, Sydney, NSW, AUS, Dec. 2009, pp. 255–262. DOI: 10.1109/FPT.2009.5377620,
Abstract: Sparse matrix-vector multiplication (SpMV) is a common operation in numerical linear algebra and is the computational kernel of many scientific applications. It is one of the original and perhaps most studied targets for FPGA acceleration. Despite this, GPUs, which have only recently gained both general-purpose programmability and native support for double precision floating-point arithmetic, are viewed by some as a more effective platform for SpMV and similar linear algebra computations. In this paper, we present an analysis comparing an existing GPU SpMV implementation to our own, novel FPGA implementation. In this analysis, we describe the challenges faced by any SpMV implementation, the unique approaches to these challenges taken by both FPGA and GPU implementations, and their relative performance for SpMV.

Bazaraa2010 [101] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*. Wiley, 2010.

Bieling2010 [102] J. Bieling, P. Peschlow, and P. Martini, "An efficient GPU implementation of the revised simplex method," in *Proceedings of the 2011 IEEE International Parallel and Distributed Processing Symposium*, ser. IPDPS '10, Atlanta, GE, USA, 2010, pp. 1–8. DOI: 10.1109/IPDPSW.2010.5470831,
Abstract: The computational power provided by the massive parallelism of modern graphics processing units (GPUs) has moved increasingly into focus over the past few years. In particular, general purpose computing on GPUs (GPGPU) is attracting attention among researchers and

practitioners alike. Yet GPGPU research is still in its infancy, and a major challenge is to rearrange existing algorithms so as to obtain a significant performance gain from the execution on a GPU. In this paper, we address this challenge by presenting an efficient GPU implementation of a very popular algorithm for linear programming, the revised simplex method. We describe how to carry out the steps of the revised simplex method to take full advantage of the parallel processing capabilities of a GPU. Our experiments demonstrate considerable speedup over a widely used CPU implementation, thus underlining the tremendous potential of GPGPU.

Buss2010 [103] A. Buss, Harshvardhan, I. Papadopoulos, O. Pearce, T. Smith, G. Tanase, N. Thomas, X. Xu, M. Bianco, N. M. Amato, and L. Rauchwerger, "STAPL: Standard template adaptive parallel library," in *Proceedings of the 3rd Annual Haifa Experimental Systems Conference*, ser. SYSTOR '10, Haifa, IL: ACM, 2010, 14:1–14:10, ISBN: 978-1-60558-908-4. DOI: 10.1145/1815695.1815713,
Abstract: The Standard Template Adaptive Parallel Library (stapl) is a high-productivity parallel programming framework that extends C++ and stl with unified support for shared and distributed memory parallelism. stapl provides distributed data structures (pContainers) and parallel algorithms (pAlgorithms) and a generic methodology for extending them to provide customized functionality. The stapl runtime system provides the abstraction for communication and program execution. In this paper, we describe the major components of stapl and present performance results for both algorithms and data structures showing scalability up to tens of thousands of processors.

Calgaro2010 [104] C. Calgaro, J.-P. Chehab, and Y. Saad, "Incremental incomplete LU factorizations with applications," *Numerical Linear Algebra with Applications*, vol. 17, no. 5, pp. 811–837, 2010, ISSN: 1099–1506. DOI: 10.1002/nla.756,
Abstract: This paper addresses the problem of computing preconditioners for solving linear systems of equations with a sequence of slowly varying matrices. This problem arises in many important applications. For example, a common situation in computational fluid dynamics, is when the equations change only slightly, possibly in some parts of the physical domain. In such situations it is wasteful to recompute entirely any LU or ILU factorizations computed for the previous coefficient matrix. A number of techniques for computing incremental ILU factorizations are examined. For example we consider methods based on approximate inverses as well as alternating techniques for updating the factors L and U of the factorization.

Cope2010 [105] B. Cope, P. Y. K. Cheung, W. Luk, and L. Howes, "Performance comparison of graphics processors to reconfigurable logic: A case study," *IEEE Transactions on Computers*, vol. 59, no. 4, pp. 433–448, Apr. 2010. DOI: 10.1109/TC.2009.179,
Abstract: A systematic approach to the comparison of the graphics processor (GPU) and reconfigurable logic is defined in terms of three throughput drivers. The approach is applied to five case study algorithms, characterized by their arithmetic complexity, memory access requirements, and data dependence, and two target devices: the nVidia GeForce 7900 GTX GPU and a Xilinx Virtex-4 field programmable gate array (FPGA). Two orders of magnitude speedup, over a general-purpose processor, is observed for each device for arithmetic intensive algorithms. An FPGA is superior, over a GPU, for algorithms requiring large numbers of regular memory accesses, while the GPU is superior for algorithms with variable data reuse. In the presence of data dependence, the implementation of a customized data path in an FPGA exceeds GPU performance by up to eight times. The trends of the analysis to newer and future technologies are analyzed.

Davis2010 [106] T. A. Davis and E. Palamadai Natarajan, "Algorithm 907: KLU, a direct sparse solver for circuit simulation problems," *ACM Transactions on Mathematical Software*, vol. 37, no. 3, pp. 1–17, Sep. 2010, ISSN: 0098-3500. DOI: 10.1145/1824801.1824814,
Abstract: KLU is a software package for solving sparse unsymmetric linear systems of equations that arise in circuit simulation applications. It relies on a permutation to Block Triangular Form (BTF), several methods for finding a fill-reducing ordering (variants of approximate minimum degree and nested dissection), and Gilbert/Peierls' sparse left-looking LU factorization algorithm to factorize each block. The package is written in C and includes a MATLAB interface. Performance results comparing KLU with SuperLU, Sparse 1.3, and UMFPACK on circuit simulation matrices

are presented. KLU is the default sparse direct solver in the XyceTMcircuit simulation package developed by Sandia National Laboratories.

GLPK2010 [107] *GNU linear programming kit reference manual*, Free Software Foundation, 2010. [Online]. Available: http://kam.mff.cuni.cz/~elias/glpk.pdf.

Hall2010 [108] J. Hall, "Towards a practical parallelisation of the simplex method," English, *Computational Management Science*, vol. 7, no. 2, pp. 139–170, 2010, ISSN: 1619–697X. DOI: 10.1007/s10287-008-0080-5,
Abstract: The simplex method is frequently the most efficient method of solving linear programming (LP) problems. This paper reviews previous attempts to parallelise the simplex method in relation to efficient serial simplex techniques and the nature of practical LP problems. For the major challenge of solving general large sparse LP problems, there has been no parallelisation of the simplex method that offers significantly improved performance over a good serial implementation. However, there has been some success in developing parallel solvers for LPs that are dense or have particular structural properties. As an outcome of the review, this paper identifies scope for future work towards the goal of developing parallel implementations of the simplex method that are of practical value.

Hong2010 [109] S. Hong and H. Kim, "An integrated GPU power and performance model," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ser. ISCA '10, Saint-Malo, France: ACM, 2010, pp. 280–289, ISBN: 978-1-4503-0053-7. DOI: 10.1145/1815961.1815998,
Abstract: GPU architectures are increasingly important in the multi-core era due to their high number of parallel processors. Performance optimization for multi-core processors has been a challenge for programmers. Furthermore, optimizing for power consumption is even more difficult. Unfortunately, as a result of the high number of processors, the power consumption of many-core processors such as GPUs has increased significantly.
Hence, in this paper, we propose an integrated power and performance (IPP) prediction model for a GPU architecture to predict the optimal number of active processors for a given application. The basic intuition is that when an application reaches the peak memory bandwidth, using more cores does not result in performance improvement.
We develop an empirical power model for the GPU. Unlike most previous models, which require measured execution times, hardware performance counters, or architectural simulations, IPP predicts execution times to calculate dynamic power events. We then use the outcome of IPP to control the number of running cores. We also model the increases in power consumption that resulted from the increases in temperature.
With the predicted optimal number of active cores, we show that we can save up to 22.09%of runtime GPU energy consumption and on average 10.99% of that for the five memory bandwidth-limited benchmarks.

Hugues2010 [110] M. R. Hugues and S. G. Petiton, "Sparse matrix formats evaluation and optimization on a GPU," in *Proceedings of the 12th IEEE International Conference on High Performance Computing and Communications*, ser. HPCC '10, Melbourne, AUS, Sep. 2010, pp. 122–129. DOI: 10.1109/HPCC.2010.85,
Abstract: The data parallel programming model comes back with massive multicore architectures. The GPU is one of these and offers important possibilities to accelerate linear algebra. However, the irregular structure of sparse matrix operations generates problems with this programming model to obtain efficient performance. This depends on the used format to store values and the matrix structure. The sparse matrix-vector product (SpMV) is one of the most used kernel in scientific computing and is the main performance source of iterative methods. We propose an evaluation and optimization of several sparse formats for the SpMV kernel which have succeeded at the time of data parallel computer. This study is realized by analyzing the performances following the distribution of the non zeros values in the matrix to determine the best and the worst reachable value. The results show that all sparse formats converge to the same efficiency and perform poorly with a strong distribution of elements.

Jones2010 [111] D. H. Jones, A. Powell, and C.-S. B. P. Y. K. Cheung, "GPU versus FPGA for high productivity computing," in *Proceedings of the 2010 International Conference on Field Programmable Logic and Applications*, ser. FPL '10, Milan, IT: IEEE Computer Society, 2010, pp. 119–124, ISBN: 978-0-7695-4179-2. DOI: 10.1109/FPL.2010.32,
Abstract: Heterogeneous or co-processor architectures are becoming an important component of high productivity computing systems (HPCS). In this work the performance of a GPU based HPCS is compared with the performance of a commercially available FPGA based HPC. Contrary to previous approaches that focussed on specific examples, a broader analysis is performed by considering processes at an architectural level. A set of benchmarks is employed that use different process architectures in order to exploit the benefits of each technology. These include the asynchronous pipelines common to "map" tasks, a partially synchronous tree common to "reduce" tasks and a fully synchronous, fully connected mesh. We show that the GPU is more productive than the FPGA architecture for most of the benchmarks and conclude that FPGA-based HPCS is being marginalised by GPUs.

Lee2010 [112] V. W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, R. Singhal, and P. Dubey, "Debunking the 100x GPU vs. CPU myth: An evaluation of throughput computing on CPU and GPU," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ser. ISCA '10, Saint-Malo, FR: ACM, 2010, pp. 451–460, ISBN: 978-1-4503-0053-7. DOI: 10.1145/1815961.1816021,
Abstract: Recent advances in computing have led to an explosion in the amount of data being generated. Processing the ever-growing data in a timely manner has made throughput computing an important aspect for emerging applications. Our analysis of a set of important throughput computing kernels shows that there is an ample amount of parallelism in these kernels which makes them suitable for today's multi-core CPUs and GPUs. In the past few years there have been many studies claiming GPUs deliver substantial speedups (between 10× and 1000×) over multi-core CPUs on these kernels. To understand where such large performance difference comes from, we perform a rigorous performance analysis and find that after applying optimizations appropriate for both CPUs and GPUs the performance gap between an Nvidia GTX280 processor and the Intel Core i7-960 processor narrows to only 2.5× on average. In this paper, we discuss optimization techniques for both CPU and GPU, analyze what architecture features contributed to performance differences between the two architectures, and recommend a set of architectural features which provide significant improvement in architectural efficiency for throughput kernels.

Malewicz2010 [113] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '10, Indianapolis, IN, USA: ACM, 2010, pp. 135–146, ISBN: 978-1-4503-0032-2. DOI: 10.1145/1807167.1807184,
Abstract: Many practical computing problems concern large graphs. Standard examples include the Web graph and various social networks. The scale of these graphs - in some cases billions of vertices, trillions of edges - poses challenges to their efficient processing. In this paper we present a computational model suitable for this task. Programs are expressed as a sequence of iterations, in each of which a vertex can receive messages sent in the previous iteration, send messages to other vertices, and modify its own state and that of its outgoing edges or mutate graph topology. This vertex-centric approach is flexible enough to express a broad set of algorithms. The model has been designed for efficient, scalable and fault-tolerant implementation on clusters of thousands of commodity computers, and its implied synchronicity makes reasoning about programs easier. Distribution-related details are hidden behind an abstract API. The result is a framework for processing large graphs that is expressive and easy to program.

Monakov2010 [114] A. Monakov, A. Lokhmotov, and A. Avetisyan, "Automatically tuning sparse matrix-vector multiplication for GPU architectures," in *High Performance Embedded Architectures and Compilers, HiPEAC '10*, Y. N. Patt, P. Foglia, E. Duesterwald, P. Faraboschi, and X. Martorell, Eds., ser. Lecture Notes in Computer Science, Pisa, IT: Springer Berlin Heidelberg,

2010, pp. 111–125, ISBN: 978-3-642-11515-8. DOI: 10.1007/978-3-642-11515-8_10. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-11515-8_10,
Abstract: Graphics processors are increasingly used in scientific applications due to their high computational power, which comes from hardware with multiple-level parallelism and memory hierarchy. Sparse matrix computations frequently arise in scientific applications, for example, when solving PDEs on unstructured grids. However, traditional sparse matrix algorithms are difficult to efficiently parallelize for GPUs due to irregular patterns of memory references. In this paper we present a new storage format for sparse matrices that better employs locality, has low memory footprint and enables automatic specialization for various matrices and future devices via parameter tuning. Experimental evaluation demonstrates significant speedups compared to previously published results.

OpenCl2010 [115] A. Munshi, Ed., *The OpenCL specification*, version 1.1, Khronos OpenCL Working Group, 2010. [Online]. Available: https://www.khronos.org/registry/OpenCL/specs/opencl-1.1.pdf.

NvidiaGt430 [116] *NVIDIA GeForce GT 430*, Online, Accessed 13 October 2016, NVIDIA corporation, 2010. [Online]. Available: http://www.nvidia.com/object/product-geforce-gt-430-oem-us.html.

Shahzad2010 [117] A. Shahzad, E. C. Kerrigan, and G. A. Constantinides, "A warm-start interior-point method for predictive control," in *Proceedings of the 6th UKACC International Conference on Control*, ser. UKACC '10, Coventry, UK, 2010. DOI: 10.1137/S1052623400369235,
Abstract: We study the situation in which, having solved a linear program with an interior-point method, we are presented with a new problem instance whose data is slightly perturbed from the original. We describe strategies for recovering a "warm-start" point for the perturbed problem instance from the iterates of the original problem instance. We obtain worst-case estimates of the number of iterations required to converge to a solution of the perturbed instance from the warm-start points, showing that these estimates depend on the size of the perturbation and on the conditioning and other properties of the problem instances.

Soman2010 [118] J. Soman, K. Kishore, and P. J. Narayanan, "A fast GPU algorithm for graph connectivity," in *Proceedings of the 2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*, ser. IPDPSW '10, Atlanta, GA, US, Apr. 2010, pp. 1–8. DOI: 10.1109/IPDPSW.2010.5470817,
Abstract: Graphics processing units provide a large computational power at a very low price which position them as an ubiquitous accelerator. General purpose programming on the graphics processing units (GPGPU) is best suited for regular data parallel algorithms. They are not directly amenable for algorithms which have irregular data access patterns such as list ranking, and finding the connected components of a graph, and the like. In this work, we present a GPU-optimized implementation for finding the connected components of a given graph. Our implementation tries to minimize the impact of irregularity, both at the data level and functional level. Our implementation achieves a speed up of 9 to 12 times over the best sequential CPU implementation. For instance, our implementation finds connected components of a graph of 10 million nodes and 60 million edges in about 500 milliseconds on a GPU, given a random edge list. We also draw interesting observations on why PRAM algorithms, such as the Shiloach-Vishkin algorithm may not be a good fit for the GPU and how they should be modified.

Stone2010 [119] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A parallel programming standard for heterogeneous computing systems," *Computing in Science & Engineering*, vol. 12, no. 3, pp. 66–73, May 2010, ISSN: 0740-7475. DOI: 10.1109/MCSE.2010.69,
Abstract: The OpenCL standard offers a common API for program execution on systems composed of different types of computational devices such as multicore CPUs, GPUs, or other accelerators.

Tomov2010 [120] S. Tomov, J. Dongarra, and M. Baboulin, "Towards dense linear algebra for hybrid GPU accelerated manycore systems," *Parallel Computing*, vol. 36, no. 5–6, pp. 232–240, Jun. 2010, ISSN: 0167-8191. DOI: 10.1016/j.parco.2009.12.005,

Abstract: We highlight the trends leading to the increased appeal of using hybrid multicore + GPU systems for high performance computing. We present a set of techniques that can be used to develop efficient dense linear algebra algorithms for these systems. We illustrate the main ideas with the development of a hybrid LU factorization algorithm where we split the computation over a multicore and a graphics processor, and use particular techniques to reduce the amount of pivoting and communication between the hybrid components. This results in an efficient algorithm with balanced use of a multicore processor and a graphics processor.

Vuduc2010 [121] R. Vuduc, A. Chandramowlishwaran, J. Choi, M. Guney, and A. Shringarpure, "On the limits of GPU acceleration," in *Proceedings of the 2nd USENIX Conference on Hot Topics in Parallelism*, ser. HotPar'10, Berkeley, CA, USA: USENIX Association, 2010, pp. 13–13. [Online]. Available: http://dl.acm.org/citation.cfm?id=1863086.1863099,
Abstract: This paper throws a small "wet blanket" on the hot topic of GPGPU acceleration, based on experience analyzing and tuning both multithreaded CPU and GPU implementations of three computations in scientific computing. These computations–(a) iterative sparse linear solvers; (b) sparse Cholesky factorization; and (c) the fast multipole method–exhibit complex behavior and vary in computational intensity and memory reference irregularity. In each case, algorithmic analysis and prior work might lead us to conclude that an idealized GPU can deliver better performance, but we find that for at least equal-effort CPU tuning and consideration of realistic workloads and calling-contexts, we can with two modern quad-core CPU sockets roughly match one or two GPUs in performance.
Our conclusions are not intended to dampen interest in GPU acceleration; on the contrary, they should do the opposite: they partially illuminate the boundary between CPU and GPU performance, and ask architects to consider application contexts in the design of future coupled on-die CPU/GPU processors.

Wilson2010 [122] G. Wilson and W. Banzhaf, "Deployment of parallel linear genetic programming using GPUs on PC and video game console platforms," English, *Genetic Programming and Evolvable Machines*, vol. 11, no. 2, pp. 147–184, 2010, ISSN: 1389–2576. DOI: 10.1007/s10710-010-9102-5,
Abstract: We present a general method for deploying parallel linear genetic programming (LGP) to the PC and Xbox 360 video game console by using a publicly available common framework for the devices called XNA (for XNA's Not Acronymed). By constructing the LGP within this framework, we effectively produce an LGP game for PC and XBox 360 that displays results as they evolve. We use the GPU of each device to parallelize fitness evaluation and the mutation operator of the LGP algorithm, thus providing a general LGP implementation suitable for parallel computation on heterogeneous devices. While parallel GP implementations on PCs are now common, both the implementation of GP on a video game console using GPU and the construction of a GP around a framework for heterogeneous devices are novel contributions. The objective of this work is to describe how to implement the parallel execution of LGP in order to use the underlying hardware (especially GPU) on the different platforms while still maintaining loyalty to the general methodology of the LGP algorithm built for the common framework. We discuss the implementation of texture-based data structures and the sequential and parallel algorithms built for their use on both CPU and GPU. Following the description of the general algorithm, the particular tailoring of the implementations for each hardware platform is described. Sequential (CPU) and parallel (GPU-based) algorithm performance is compared on both PC and video game platforms using the metrics of GP operations per second, actual time elapsed, speedup of parallel over sequential implementation, and percentage of execution time used by the GPU versus CPU.

Wong2010 [123] H. Wong, M. M. Papadopoulou, M. Sadooghi-Alvandi, and A. Moshovos, "Demystifying GPU microarchitecture through microbenchmarking," in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems Software*, ser. ISPASS '10, Mar. 2010, pp. 235–246. DOI: 10.1109/ISPASS.2010.5452013,
Abstract: Graphics processors (GPU) offer the promise of more than an order of magnitude speedup over conventional processors for certain non-graphics computations. Because the GPU is often presented as a C-like abstraction (e.g., Nvidia's CUDA), little is known about the characteristics of

the GPU's architecture beyond what the manufacturer has documented. This work develops a microbechmark suite and measures the CUDA-visible architectural characteristics of the Nvidia GT200 (GTX280) GPU. Various undisclosed characteristics of the processing elements and the memory hierarchies are measured. This analysis exposes undocumented features that impact program performance and correctness. These measurements can be useful for improving performance optimization, analysis, and modeling on this architecture and offer additional insight on the decisions made in developing this GPU.

Carneiro2011 [124] T. Carneiro, A. E. Muritiba, M. Negreiros, and G. A. L. de Campos, "A new parallel schema for branch-and-bound algorithms using GPGPU," in *Proceedings of the 23rd International Symposium on Computer Architecture and High Performance Computing*, ser. SBAC-PAD '11, Vitória, Espírito Santo, BR, Oct. 2011, pp. 41–47. DOI: 10.1109/SBAC-PAD.2011.20,
Abstract: This work presents a new parallel procedure designed to process combinatorial B&B algorithms using GPGPU. In our schema we dispatch a number of threads that treats intelligently the massively parallel processors of NVIDIA GeForce graphical units. The strategy is to build sequentially a series of initial searches that can map a subspace of the B&B tree by starting a number of limited threads after achieving a specific level of the tree. The search is then processed massively by DFS. The whole subspace is optimized accordingly to memory and limits of threads and blocks available by the GPU. We compare our results with its OpenMP and Serial versions of the same search schema using explicitly enumeration (all possible solutions) to the Asymmetrical Travelling Salesman Problem's instances. We also show the great superiority of our GPGPU based method.

CPLEX2011 [125] *CPLEX user's manual*, 12.4, Accessed 30 March 2014, IBM corporation, 2011. [Online]. Available: http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r4/topic/ilog.odms.studio.help/pdf/usrcplex.pdf.

Davis2011 [126] T. A. Davis and Y. Hu, "The university of florida sparse matrix collection," *ACM Transactions on Mathematical Software*, vol. 38, no. 1, pp. 1–25, Dec. 2011, ISSN: 0098-3500. DOI: 10.1145/2049662.2049663,
Abstract: We describe the University of Florida Sparse Matrix Collection, a large and actively growing set of sparse matrices that arise in real applications. The Collection is widely used by the numerical linear algebra community for the development and performance evaluation of sparse matrix algorithms. It allows for robust and repeatable experiments: robust because performance results with artificially generated matrices can be misleading, and repeatable because matrices are curated and made publicly available in many formats. Its matrices cover a wide spectrum of domains, include those arising from problems with underlying 2D or 3D geometry (as structural engineering, computational fluid dynamics, model reduction, electromagnetics, semiconductor devices, thermodynamics, materials, acoustics, computer graphics/vision, robotics/kinematics, and other discretizations) and those that typically do not have such geometry (optimization, circuit simulation, economic and financial modeling, theoretical and quantum chemistry, chemical process simulation, mathematics and statistics, power networks, and other networks and graphs). We provide software for accessing and managing the Collection, from MATLAB, Mathematica, Fortran, and C, as well as an online search capability. Graph visualization of the matrices is provided, and a new multilevel coarsening scheme is proposed to facilitate this task.

Elteir2011 [127] M. Elteir, H. Lin, and W.-c. Feng, "Performance characterization and optimization of atomic operations on AMD GPUs," in *Proceedings of the 2011 IEEE International Conference on Cluster Computing*, ser. CLUSTER '11, Austin, TX, US, Sep. 2011, pp. 234–243. DOI: 10.1109/CLUSTER.2011.34,
Abstract: Atomic operations are important building blocks in supporting general-purpose computing on graphics processing units (GPUs). For instance, they can be used to coordinate execution between concurrent threads, and in turn, assist in constructing complex data structures such as hash tables or implementing GPU-wide barrier synchronization. While the performance of atomic operations has improved substantially on the latest NVIDIA Fermi-based GPUs, system-provided atomic operations still incur significant performance penalties on AMD GPUs. A memory-bound kernel on an AMD GPU, for example, can suffer severe performance degradation when including

an atomic operation, even if the atomic operation is never executed. In this paper, we first quantify the performance impact of atomic instructions to application kernels on AMD GPUs. We then propose a novel software-based implementation of atomic operations that can significantly improve the overall kernel performance. We evaluate its performance against the system-provided atomic using two micro-benchmarks and four real applications. The results show that using our software based atomic operations on an AMD GPU can speedup an application kernel by 67-fold over the same application kernel but with the (default) system-provided atomic operations.

Feo2011 [128] J. Feo, O. Villa, A. Tumeo, and S. Secchi, Eds., *Proceedings of the First Workshop on Irregular Applications: Architectures and Algorithms*, IAAA '11, Seattle, WA, USA: ACM, 2011, pp. 1–2, ISBN: 978-1-4503-1121-2. DOI: 10.1145/2089142.2089144,
Abstract: Irregular applications are characterized by irregular data structures, control and communication patterns. Novel irregular high performance applications which deal with large data sets and require have recently appeared. Unfortunately, current high performance systems and software infrastructures executes irregular algorithms poorly. Only coordinated efforts by end user, area specialists and computer scientists that consider both the architecture and the software stack may be able to provide solutions to the challenges of modern irregular applications.

Fong2011 [129] D. C.-L. Fong and M. Saunders, "LSMR: An iterative algorithm for sparse least-squares problems," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2950–2971, 2011. DOI: 10.1137/10079687X. [Online]. Available: https://doi.org/10.1137/10079687X,
Abstract: An iterative method LSMR is presented for solving linear systems $Ax = b$ and least-squares problems $\min \|Ax - b\|_2$, with A being sparse or a fast linear operator. LSMR is based on the Golub–Kahan bidiagonalization process. It is analytically equivalent to the MINRES method applied to the normal equation $A^TAx = A^Tb$, so that the quantities $\|A^Tr_k\|$ are monotonically decreasing (where $r_k = b - Ax_k$ is the residual for the current iterate x_k). We observe in practice that $\|r_k\|$ also decreases monotonically, so that compared to LSQR (for which only $\|r_k\|$ is monotonic) it is safer to terminate LSMR early. We also report some experiments with reorthogonalization.

Hamzic2011 [130] A. Hamzić, A. Huseinović, and N. Nosović, "Implementation and performance analysis of the simplex algorithm adapted to run on commodity OpenCL enabled graphics processors," in *Proceedings of the XXIII International Symposium on Information Communication and Automation Technologies*, ser. ICAT '11, Sarajevo, BIH, Oct. 2011, pp. 1–7. DOI: 10.1109/ICAT.2011.6102135,
Abstract: The Simplex algorithm is commonly used for solving Linear Optimization problems. Linear Optimization methods are used to solve problems in areas such as Economics, Business, Planning and Engineering. Developments of hardware platforms have allowed the use of Linear Optimization methods on problems that presented serious computational challenges in the past. However, solving large optimization problems can be time consuming, which has to be taken into consideration for time-critical applications. With the invention of the GPU assisted computing the situation in this field has progressed. In this paper, implementation and performance analysis of the Simplex algorithm adapted to take the advantage of modern graphics processors versus traditional CPU adapted implementation is presented.

Hennessy2011 [131] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 5th. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011, ISBN: 012383872X.

IntelI3 [132] *Intel core i3-2130 processor*, Online, Accessed 13 October 2016, Intel Corporation, 2011. [Online]. Available: http://ark.intel.com/products/53428/Intel-Core-i3-2130-Processor-3M-Cache-3_40-GHz.

Jang2011 [133] B. Jang, D. Schaa, P. Mistry, and D. Kaeli, "Exploiting memory access patterns to improve memory performance in data-parallel architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 1, pp. 105–118, Jan. 2011, ISSN: 1045–9219. DOI: 10.1109/TPDS.2010.107,

Abstract: The introduction of General-Purpose computation on GPUs (GPGPUs) has changed the landscape for the future of parallel computing. At the core of this phenomenon are massively multithreaded, data-parallel architectures possessing impressive acceleration ratings, offering low-cost supercomputing together with attractive power budgets. Even given the numerous benefits provided by GPGPUs, there remain a number of barriers that delay wider adoption of these architectures. One major issue is the heterogeneous and distributed nature of the memory subsystem commonly found on data-parallel architectures. Application acceleration is highly dependent on being able to utilize the memory subsystem effectively so that all execution units remain busy. In this paper, we present techniques for enhancing the memory efficiency of applications on data-parallel architectures, based on the analysis and characterization of memory access patterns in loop bodies; we target vectorization via data transformation to benefit vector-based architectures (e.g., AMD GPUs) and algorithmic memory selection for scalar-based architectures (e.g., NVIDIA GPUs). We demonstrate the effectiveness of our proposed methods with kernels from a wide range of benchmark suites. For the benchmark kernels studied, we achieve consistent and significant performance improvements (up to 11.4× and 13.5× over baseline GPU implementations on each platform, respectively) by applying our proposed methodology.

Jerez2011a [134] J. L. Jerez, G. A. Constantinides, and E. C. Kerrigan, "An FPGA implementation of a sparse quadratic programming solver for constrained predictive control," in *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '11, Monterey, CA, USA: ACM, 2011, pp. 209–218, ISBN: 978-1-4503-0554-9. DOI: 10.1145/1950413.1950454,
Abstract: Model predictive control (MPC) is an advanced industrial control technique that relies on the solution of a quadratic programming (QP) problem at every sampling instant to determine the input action required to control the current and future behaviour of a physical system. Its ability in handling large multiple input multiple output (MIMO) systems with physical constraints has led to very successful applications in slow processes, where there is sufficient time for solving the optimization problem between sampling instants. The application of MPC to faster systems, which adds the requirement of greater sampling frequencies, relies on new ways of finding faster solutions to QP problems. Field-programmable gate arrays (FPGAs) are specially well suited for this application due to the large amount of computation for a small amount of I/O. In addition, unlike a software implementation, an FPGA can provide the precise timing guarantees required for interfacing the controller to the physical system. We present a high-throughput floating-point FPGA implementation that exploits the parallelism inherent in interior-point optimization methods. It is shown that by considering that the QPs come from a control formulation, it is possible to make heavy use of the sparsity in the problem to save computations and reduce memory requirements by 75%. The implementation yields a 6.5× improvement in latency and a 51× improvement in throughput for large problems over a software implementation running on a general purpose microprocessor.

Jerez2011 [135] J. L. Jerez, G. A. Constantinides, E. C. Kerrigan, and K.-V. Ling, "Parallel MPC for real-time FPGA-based implementation," in *Proceedings of the 18th IFAC World Congress*, vol. 18, Milan, IT, Sep. 2011, pp. 1338–1345. DOI: 10.3182/20110828-6-IT-1002.01392,
Abstract: The succesful application of model predictive control (MPC) in fast embedded systems relies on faster and more energy efficient ways of solving complex optimization problems. A custom quadratic programming (QP) solver implementation on a field-programmable gate array (FPGA) can provide substantial acceleration by exploiting the parallelism inherent in some optimization algorithms, apart from providing novel computational opportunities arising from deep pipelining. This paper presents a new MPC algorithm based on multiplexed MPC that can take advantage of the full potential of an existing FPGA design by utilizing the provided free parallel computational channels arising from such pipelining. The result is greater acceleration over a conventional MPC implementation and reduced silicon usage. The FPGA implementation is shown to be approximately 200× more energy efficient than a high performance general purpose processor (GPP) for large control problems.

Kelman2011 [136] A. Kelman and F. Borrelli, "Bilinear model predictive control of a hvac system using sequen-

tial quadratic programming," in *Proceedings of the 18th IFAC World Congress*, ser. IFAC '11, Milan, IT, 2011. DOI: 10.3182/20110828-6-IT-1002.03811,
Abstract: We study the problem of heating, ventilation, and air conditioning (HVAC) control in a typical commercial building. We propose a model predictive control (MPC) approach which minimizes energy use while satisfying occupant comfort constraints. A sequential quadratic programming algorithm is used to efficiently solve the resulting bilinear optimization problem. This paper presents the control design approach and the procedure for computing its solution. Extensive numerical simulations show the effectiveness of the proposed approach. In particular, the MPC is able to systematically reproduce a variety of well-known commercial solutions for energy savings, which include demand response, economizer mode and precooling/preheating.

Koch2011 [137] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter, "MIPLIB 2010," *Mathematical Programming Computation*, vol. 3, no. 2, pp. 103–163, 2011. DOI: 10.1007/s12532-011-0025-9,
Abstract: This paper reports on the fifth version of the Mixed Integer Programming Library. The miplib 2010 is the first miplib release that has been assembled by a large group from academia and from industry, all of whom work in integer programming. There was mutual consent that the concept of the library had to be expanded in order to fulfill the needs of the community. The new version comprises 361 instances sorted into several groups. This includes the main benchmark test set of 87 instances, which are all solvable by today's codes, and also the challenge test set with 164 instances, many of which are currently unsolved. For the first time, we include scripts to run automated tests in a predefined way. Further, there is a solution checker to test the accuracy of provided solutions using exact arithmetic.

Lalami2011 [138] M. E. Lalami, D. El-Baz, and V. Boyer, "Multi GPU implementation of the simplex algorithm," in *Proceedings of the 13th IEEE International Conference on High Performance Computing and Communications*, ser. HPCC '11, Banff, AB, CA, Sep. 2011, pp. 179–186. DOI: 10.1109/HPCC.2011.32,
Abstract: The Simplex algorithm is a well known method to solve linear programming (LP) problems. In this paper, we propose an implementation via CUDA of the Simplex method on a multi GPU architecture. Computational tests have been carried out on randomly generated instances for non-sparse LP problems. The tests show a maximum speedup of 24.5 with two Tesla C2050 boards.

Lalami2011a [139] M. E. Lalami, V. Boyer, and D. El-Baz, "Efficient implementation of the simplex method on a CPU–GPU system," in *Proceedings of the 2011 IEEE International Parallel and Distributed Processing Symposium*, ser. IPDPS '11, Anchorage, AL, USA, May 2011, pp. 1999–2006, ISBN: 978-1-61284-425-1. DOI: 10.1109/IPDPS.2011.362,
Abstract: The Simplex algorithm is a well known method to solve linear programming (LP) problems. In this paper, we propose a parallel implementation of the Simplex on a CPU-GPU systems via CUDA. Double precision implementation is used in order to improve the quality of solutions. Computational tests have been carried out on randomly generated instances for non-sparse LP problems. The tests show a maximum speedup of 12:5 on a GTX 260 board.

Li2011 [140] J. Li, R. Lv, X. Hu, and Z. Jiang, "A GPU-based parallel algorithm for large scale linear programming problem," in *Intelligent Decision Technologies*, ser. Smart Innovation, Systems and Technologies, J. Watada, G. Phillips-Wren, L. Jain, and R. J. Howlett, Eds., vol. 10, Springer Berlin Heidelberg, 2011, pp. 37–46, ISBN: 978-3-642-22193-4. DOI: 10.1007/978-3-642-22194-1_4,
Abstract: A GPU-based parallel algorithm to solve large scale linear programming problem is proposed in this research. It aims to improve the computing efficiency when the linear programming problem becomes sufficiently large scale or more complicated. This parallel algorithm, based on Gaussian elimination, uses the GPU (Graphics Processing Unit) for computationally intensive tasks such as basis matrix operation, canonical form transformation and entering variable selection. At the same time, CPU is used to control the iteration. Experimental results show that the algorithm

is competitive with CPU algorithm and can greatly reduce the computing time, so the GPU-based parallel algorithm is an effective way to solve large scale linear programming problem.

Meyer2011 [141] X. Meyer, P. Albuquerque, and B. Chopard, "A multi-GPU implementation and performance model for the standard simplex method," in *Proceedings of the 1st International Symposium and 10th Balkan Conference on Operational Research*, ser. BALCOR '11, Thessaloniki, GR, Sep. 2011, pp. 312–319. [Online]. Available: http://spc.unige.ch/lib/exe/fetch.php?media=pub:sgpu_europar2011.pdf,
Abstract: The standard simplex method is a well-known optimization algorithm for solving linear programming models in the field of operational research. It is part of software that is often employed by businesses for solving scheduling or assignment problems. But their always increasing complexity and size drives the demand for more computational power.In the past few years, GPUs have gained a lot of popularity as they offer an opportunity to accelerate many algorithms. In this paper we present a mono and a multi-GPU implementation of the standard simplex method, which is based on CUDA. Measurements show that it outperforms the CLP solver provided the problem size is large enough. We also derive a performance model and establish its accurateness. To our knowledge, only the revised simplex method has so far been implemented on a GPU.

Munawar2011 [142] A. Munawar, M. Wahib, M. Munetomo, and K. Akama, "Advanced genetic algorithm to solve MINLP problems over GPU," in *Proceedings of the IEEE Congress on Evolutionary Computation*, ser. CEC '11, New Orleans, LA, USA, Jun. 2011, pp. 318–325. DOI: 10.1109/CEC.2011.5949635,
Abstract: In this paper we propose a many-core implementation of evolutionary computation for GPGPU (General-Purpose Graphic Processing Unit) to solve non-convex Mixed Integer Non-Linear Programming (MINLP) and non-convex Non Linear Programming (NLP) problems using a stochastic algorithm. Stochastic algorithms being random in their behavior are difficult to implement over GPU like architectures. In this paper we not only succeed in implementation of a stochastic algorithm over GPU but show considerable speedups over CPU implementations. The stochastic algorithm considered for this paper is an adaptive resolution approach to genetic algorithm (arGA), developed by the authors of this paper. The technique uses the entropy measure of each variable to adjust the intensity of the genetic search around promising individuals. Performance is further improved by hybridization with adaptive resolution local search (arLS) operator. In this paper, we describe the challenges and design choices involved in parallelization of this algorithm to solve complex MINLPs over a commodity GPU using Compute Unified Device Architecture (CUDA) programming model. Results section shows several numerical tests and performance measurements obtained by running the algorithm over an nVidia Fermi GPU. We show that for difficult problems we can obtain a speedup of up to 20× with double precision and up to 42× with single precision.

Naumov2011 [143] M. Naumov, "Parallel solution of sparse triangular linear systems in the preconditioned iterative methods on the GPU," NVIDIA Corporation, Technical report 2011–001, Jun. 2011. [Online]. Available: http://research.nvidia.com/sites/default/files/publications/nvr-2011-001.pdf,
Abstract: A novel algorithm for solving in parallel a sparse triangular linear system on a graphical processing unit is proposed. It implements the solution of the triangular system in two phases. First, the analysis phase builds a dependency graph based on the matrix sparsity pattern and groups the independent rows into levels. Second, the solve phase obtains the full solution by iterating sequentially across the constructed levels. The solution elements corresponding to each single level are obtained at once in parallel. The numerical experiments are also presented and it is shown that the incomplete-LU and Cholesky preconditioned iterative methods, using the parallel sparse triangular solve algorithm, can achieve on average more than 2× speedup on graphical processing units (GPUs) over their CPU implementation.

Pingali2011 [144] K. Pingali, D. Nguyen, M. Kulkarni, M. Burtscher, M. A. Hassaan, R. Kaleem, T.-H. Lee, A. Lenharth, R. Manevich, M. Méndez-Lojo, D. Prountzos, and X. Sui, "The tao of parallelism in algorithms," *ACM SIGPLAN Notices*, vol. 46, no. 6, pp. 12–25, Jun. 2011, ISSN: 0362-1340. DOI: 10.1145/1993316.1993501,

Abstract: For more than thirty years, the parallel programming community has used the dependence graph as the main abstraction for reasoning about and exploiting parallelism in "regular" algorithms that use dense arrays, such as finite-differences and FFTs. In this paper, we argue that the dependence graph is not a suitable abstraction for algorithms in new application areas like machine learning and network analysis in which the key data structures are "irregular" data structures like graphs, trees, and sets.
To address the need for better abstractions, we introduce a data-centric formulation of algorithms called the operator formulation in which an algorithm is expressed in terms of its action on data structures. This formulation is the basis for a structural analysis of algorithms that we call tao-analysis. Tao-analysis can be viewed as an abstraction of algorithms that distills out algorithmic properties important for parallelization. It reveals that a generalized form of data-parallelism called amorphous data-parallelism is ubiquitous in algorithms, and that, depending on the tao-structure of the algorithm, this parallelism may be exploited by compile-time, inspector-executor or optimistic parallelization, thereby unifying these seemingly unrelated parallelization techniques. Regular algorithms emerge as a special case of irregular algorithms, and many application-specific optimization techniques can be generalized to a broader context.
These results suggest that the operator formulation and tao-analysis of algorithms can be the foundation of a systematic approach to parallel programming.

Sadrieh2011 [145] A. Sadrieh and P. A. Bahri, "Optimal control of the process systems using graphic processing units," in *Proceedings of the 18th IFAC World Congress*, ser. IFAC '11, Milan, IT, 2011. DOI: 10.3182/20110828-6-IT-1002.01804,
Abstract: In this paper the Graphic Processing Unit (GPU) is applied in order to improve the computational performance of process systems optimal control calculations. To apply GPU massive parallel architecture, a simplified version of interior point optimisation algorithm was selected and modified to fulfil special hardware requirements of GPU architecture. In this algorithm, a damped nonlinear Newton with preconditioned iterative linear solver was used to solve Dual-Primal equations. The comparison of results between this implementation and standard CPU-based implementation shows considerable improvement in computational performance.

Scarpino2011 [146] M. Scarpino, *OpenCL in action*. Manning Publications Co., 2011.

Smith2011 [147] B. Smith and H. Zhang, "Sparse triangular solves for ILU revisited: Data layout crucial to better performance," *International Journal of High Performance Computing Applications*, vol. 25, no. 4, pp. 386–391, Nov. 2011. DOI: 10.1177/1094342010389857,
Abstract: A key to good processor utilization for sparse matrix computations is storing the data in the format that is most conducive to fast access by the memory system. In particular, for sparse matrix triangular solves the traditional compressed sparse matrix format is poor, and minor adjustments to the data structure can increase the processor utilization dramatically. Such adjustments involve storing the L and U factors separately and storing the U rows backwards so that they are accessed in a simple streaming fashion during the triangular solves. Changes to the PETSc libraries to use this modified storage format resulted in over twice the floating-point rate for some matrices. This improvement can be accounted for by a decrease in the cache misses and TLB (transaction lookaside buffer) misses in the modified code.

Thoman2011 [148] P. Thoman, K. Kofler, H. Studt, J. Thomson, and T. Fahringer, "Automatic OpenCL device characterization: Guiding optimized kernel design," in *Proceedings of the 17th International Conference on Parallel Processing*, ser. Euro-Par '11, Bordeaux, FR: Springer-Verlag, 2011, pp. 438–452, ISBN: 978-3-642-23396-8. DOI: 10.1007/978-3-642-23397-5_43,
Abstract: The OpenCL standard allows targeting a large variety of CPU, GPU and accelerator architectures using a single unified programming interface and language. While the standard guarantees portability of functionality for complying applications and platforms, performance portability on such a diverse set of hardware is limited. Devices may vary significantly in memory architecture as well as type, number and complexity of computational units. To characterize and compare the OpenCL performance of existing and future devices we propose a suite of microbenchmarks, uCLbench.

We present measurements for eight hardware architectures four GPUs, three CPUs and one accelerator and illustrate how the results accurately reflect unique characteristics of the respective platform. In addition to measuring quantities traditionally benchmarked on CPUs like arithmetic throughput or the bandwidth and latency of various address spaces, the suite also includes code designed to determine parameters unique to OpenCL like the dynamic branching penalties prevalent on GPUs. We demonstrate how our results can be used to guide algorithm design and optimization for any given platform on an example kernel that represents the key computation of a linear multigrid solver. Guided manual optimization of this kernel results in an average improvement of 61% across the eight platforms tested.

Whitehead2011 [149] N. Whitehead and A. Fit-Florea, "Precision & performance: Floating point and IEEE 754 compliance for nvidia GPUs," NVIDIA Corporation, Tech. Rep. 1, 2011. [Online]. Available: http://docs.nvidia.com/cuda/floating-point/#axzz4gbha7lKQ.

Wolf2011 [150] M. M. Wolf, M. A. Heroux, and E. G. Boman, "Factors impacting performance of multithreaded sparse triangular solve," in *Proceedings of the 9th International Conference on High Performance Computing for Computational Science*, ser. VECPAR '10, Berkeley, CA, USA: Springer-Verlag, 2011, pp. 32–44, ISBN: 978-3-642-19327-9. [Online]. Available: http://dl.acm.org/citation.cfm?id=1964246,
Abstract: As computational science applications grow more parallel with multi-core supercomputers having hundreds of thousands of computational cores, it will become increasingly difficult for solvers to scale. Our approach is to use hybrid MPI/threaded numerical algorithms to solve these systems in order to reduce the number of MPI tasks and increase the parallel efficiency of the algorithm. However, we need efficient threaded numerical kernels to run on the multi-core nodes in order to achieve good parallel efficiency. In this paper, we focus on improving the performance of a multithreaded triangular solver, an important kernel for preconditioning. We analyze three factors that affect the parallel performance of this threaded kernel and obtain good scalability on the multi-core nodes for a range of matrix sizes.

Amd2012 [151] "AMD graphics cores next (GCN) architecture," AMD Corporation, Online, 2012, Accessed on 21 June 2017. [Online]. Available: https://www.amd.com/Documents/GCN_Architecture_whitepaper.pdf.

Boukedjar2012 [152] A. Boukedjar, M. E. Lalami, and D. El-Baz, "Parallel branch and bound on a CPU–GPU system," in *Proceedings of the 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, ser. PDP '12, Garching, DE, Feb. 2012, pp. 392–398. DOI: 10.1109/PDP.2012.23,
Abstract: Hybrid implementation via CUDA of a branch and bound method for knapsack problems is proposed. Branch and bound computations can be carried out either on the CPU or on the GPU according to the size of the branch and bound list, i.e. the number of nodes. Tests are carried out on a Tesla C2050 GPU. A first series of computational results showing a substantial speedup is displayed and analyzed.

Chakroun2012 [153] I. Chakroun and N. Melab, "An adaptive multi-GPU based branch-and-bound. a case study: The flow-shop scheduling problem," in *Proceedings of the 14th IEEE International Conference on High Performance Computing and Communications*, ser. HPCC '12, Liverpool, UK, Jun. 2012, pp. 389–395. DOI: 10.1109/HPCC.2012.59,
Abstract: Solving exactly Combinatorial Optimization Problems (COPs) using a Branch-and-Bound (B&B) algorithm requires a huge amount of computational resources. Therefore, we recently investigated designing B&B algorithms on top of graphics processing units (GPUs) using a parallel bounding model. The proposed model assumes parallelizing the evaluation of the lower bounds on pools of sub-problems. The results demonstrated that the size of the evaluated pool has a significant impact on the performance of B&B and that it depends strongly on the problem instance being solved. In this paper, we design an adaptative parallel B&B algorithm for solving permutation-based combinatorial optimization problems such as FSP (Flow-shop Scheduling Problem) on GPU accelerators. To do so, we propose a dynamic heuristic for parameter auto-tuning at runtime. Another challenge of this pioneering work is to exploit larger degrees of parallelism

by using the combined computational power of multiple GPU devices. The approach has been applied to the permutation flow-shop problem. Extensive experiments have been carried out on well-known FSP benchmarks using an Nvidia Tesla S1070 Computing System equipped with two Tesla T10 GPUs. Compared to a CPU-based execution, accelerations up to 105 are achieved for large problem instances.

Chakroun2012a [154] I. Chakroun, M. Mezmaz, N. Melab, and A. Bendjoudi, "Reducing thread divergence in a GPU-accelerated branch-and-bound algorithm," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 8, pp. 1121–1136, Sep. 2012. DOI: 10.1002/cpe.2931,
Abstract: In this paper, we address the design and implementation of graphical processing unit (GPU)-accelerated branch-and-bound algorithms (B&B) for solving flow-shop scheduling optimization problems (FSP). Such applications are CPU-time consuming and highly irregular. On the other hand, GPUs are massively multithreaded accelerators using the single instruction multiple data model at execution. A major issue that arises when executing on GPU, a B&B applied to FSP is thread or branch divergence. Such divergence is caused by the lower bound function of FSP that contains many irregular loops and conditional instructions. Our challenge is therefore to revisit the design and implementation of B&B applied to FSP dealing with thread divergence. Extensive experiments of the proposed approach have been carried out on well-known FSP benchmarks using an Nvidia Tesla (C2050 GPU card). Compared with a CPU-based execution, accelerations up to $\times$ 77.46 are achieved for large problem instances.

Doggett2012 [155] M. Doggett, "Texture caches," *IEEE Micro*, vol. 32, no. 3, pp. 136–141, May 2012, ISSN: 0272-1732. DOI: 10.1109/MM.2012.44,
Abstract: This column examines the texture cache, an essential component of modern GPUs that plays an important role in achieving real-time performance when generating realistic images. GPUs have many components and the texture cache is only one of them. But it has a real impact on the performance of the GPU if rasterization and memory tiling are set up correctly.

Domahidi2012 [156] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *Proceedings of the 51st IEEE Conference on Decision and Control*, ser. CDC '12, Maui, HI, USA, Dec. 2012, pp. 668–674. DOI: 10.1109/CDC.2012.6426855,
Abstract: Receding horizon control requires the solution of an optimization problem at every sampling instant. We present efficient interior point methods tailored to convex multistage problems, a problem class which most relevant MPC problems with linear dynamics can be cast in, and specify important algorithmic details required for a high speed implementation with superior numerical stability. In particular, the presented approach allows for quadratic constraints, which is not supported by existing fast MPC solvers. A categorization of widely used MPC problem formulations into classes of different complexity is given, and we show how the computational burden of certain quadratic or linear constraints can be decreased by a low rank matrix forward substitution scheme. Implementation details are provided that are crucial to obtain high speed solvers. We present extensive numerical studies for the proposed methods and compare our solver to three well-known solver packages, outperforming the fastest of these by a factor 2-5 in speed and 3-70 in code size. Moreover, our solver is shown to be very efficient for large problem sizes and for quadratically constrained QPs, extending the set of systems amenable to advanced MPC formulations on low-cost embedded hardware.

Gleixner2012a [157] A. M. Gleixner, "Factorization and update of a reduced basis matrix for the revised simplex method," eng, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr, 7, 14195 Berlin, Tech. Rep. 12–36, 2012. [Online]. Available: https://opus4.kobv.de/opus4-zib/files/1634/ZR-12-36.pdf,
Abstract: In this paper, we describe a method to enhance the FTRAN and BTRAN operations in the revised simplex algorithm by using a reduced basis matrix defined by basic columns and nonbasic rows. This submatrix of the standard basis matrix is potentially much smaller, but may change its dimension dynamically from iteration to iteration.
For the classical product form update (eta updates), the idea has been noted already by Zoutendijk, but only preliminarily tested by Powell in the early 1970s. We extend these ideas to Forrest-Tomlin

type update formulas for an LU factorization of the reduced basis matrix, which are suited for efficient implementation within a state-of-the-art simplex solver. The computational advantages of the proposed method apply to pure LP solving as well as to LP-based branch-cut-and-price algorithms. It can easily be integrated into existing simplex codes.

Gleixner2012 [158] A. M. Gleixner, D. E. Steffy, and K. Wolter, "Improving the accuracy of linear programming solvers with iterative refinement," in *Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*, ser. ISSAC '12, Grenoble, FR: ACM, Jul. 2012, pp. 187–194, ISBN: 978-1-4503-1269-1. DOI: 10.1145/2442829.2442858,
Abstract: We describe an iterative refinement procedure for computing extended precision or exact solutions to linear programming problems (LPs). Arbitrarily precise solutions can be computed by solving a sequence of closely related LPs with limited precision arithmetic. The LPs solved share the same constraint matrix as the original problem instance and are transformed only by modification of the objective function, right-hand side, and variable bounds. Exact computation is used to compute and store the exact representation of the transformed problems, while numeric computation is used for solving LPs. At all steps of the algorithm the LP bases encountered in the transformed problems correspond directly to LP bases in the original problem description.
We demonstrate that this algorithm is effective in practice for computing extended precision solutions and that this leads to direct improvement of the best known methods for solving LPs exactly over the rational numbers.

Gonzalez2012 [159] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "PowerGraph: Distributed graph-parallel computation on natural graphs," in *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI '12, Hollywood, CA, USA: USENIX Association, 2012, pp. 17–30, ISBN: 978-1-931971-96-6. [Online]. Available: http://dl.acm.org/citation.cfm?id=2387880,
Abstract: Large-scale graph-structured computation is central to tasks ranging from targeted advertising to natural language processing and has led to the development of several graph-parallel abstractions including Pregel and GraphLab. However, the natural graphs commonly found in the real-world have highly skewed power-law degree distributions, which challenge the assumptions made by these abstractions, limiting performance and scalability.
In this paper, we characterize the challenges of computation on natural graphs in the context of existing graph-parallel abstractions. We then introduce the PowerGraph abstraction which exploits the internal structure of graph programs to address these challenges. Leveraging the PowerGraph abstraction we introduce a new approach to distributed graph placement and representation that exploits the structure of power-law graphs. We provide a detailed analysis and experimental evaluation comparing PowerGraph to two popular graph-parallel systems. Finally, we describe three different implementation strategies for PowerGraph and discuss their relative merits with empirical evaluations on large-scale real-world problems demonstrating order of magnitude gains.

Hall2012 [160] J. Hall and Q. Huangfu, "A high performance dual revised simplex solver," English, in *Parallel Processing and Applied Mathematics*, ser. Lecture Notes in Computer Science, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Waśniewski, Eds., vol. 7203, Springer Berlin Heidelberg, 2012, pp. 143–151, ISBN: 978-3-642-31463-6. DOI: 10.1007/978-3-642-31464-3_15,
Abstract: When solving families of related linear programming (LP) problems and many classes of single LP problems, the simplex method is the preferred computational technique. Hitherto there has been no efficient parallel implementation of the simplex method that gives good speed-up on general, large sparse LP problems. This paper presents a variant of the dual simplex method and a prototype parallelisation scheme. The resulting implementation, ParISS, is efficient when run in serial and offers modest speed-up for a range of LP test problems.

Hong2012 [161] S. Hong, H. Chafi, E. Sedlar, and K. Olukotun, "Green-Marl: A DSL for easy and efficient graph analysis," in *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS XVII, London, UK: ACM, 2012, pp. 349–362, ISBN: 978-1-4503-0759-8. DOI: 10.1145/2150976.2151013,

Abstract: The increasing importance of graph-data based applications is fueling the need for highly efficient and parallel implementations of graph analysis software. In this paper we describe Green-Marl, a domain-specific language (DSL) whose high level language constructs allow developers to describe their graph analysis algorithms intuitively, but expose the data-level parallelism inherent in the algorithms. We also present our Green-Marl compiler which translates high-level algorithmic description written in Green-Marl into an efficient C++ implementation by exploiting this exposed data-level parallelism. Furthermore, our Green-Marl compiler applies a set of optimizations that take advantage of the high-level semantic knowledge encoded in the Green-Marl DSL. We demonstrate that graph analysis algorithms can be written very intuitively with Green-Marl through some examples, and our experimental results show that the compiler-generated implementation out of such descriptions performs as well as or better than highly-tuned hand-coded implementations.

IntelI7 [162] *Intel core i7-3770 processor*, Online, Accessed 13 October 2016, Intel Corporation, 2012. [Online]. Available: `http://ark.intel.com/products/65719/Intel-Core-i7-3770-Processor-8M-Cache-up-to-3_90-GHz`.

Jerez2012a [163] J. Jerez, K.-V. Ling, G. C. E.C., and Kerrigan, "Model predictive control for deeply pipelined field-programmable gate array implementation: Algorithms and circuitry," *Control Theory Applications, IET*, vol. 6, no. 8, pp. 1029–1041, May 2012, ISSN: 1751-8644. DOI: `10.1049/iet-cta.2010.0441`,

Abstract: Model predictive control (MPC) is an optimisation-based scheme that imposes a real-time constraint on computing the solution of a quadratic programming (QP) problem. The implementation of MPC in fast embedded systems presents new technological challenges. In this paper we present a parameterised field-programmable gate array implementation of a customised QP solver for optimal control of linear processes with constraints, which can achieve substantial acceleration over a general purpose microprocessor, especially as the size of the optimisation problem grows. The focus is on exploiting the structure and accelerating the computational bottleneck in a primal-dual interior-point method. We then introduce a new MPC formulation that can take advantage of the novel computational opportunities, in the form of parallel computational channels, offered by the proposed pipelined architecture to improve performance even further. This highlights the importance of the interaction between the control theory and digital system design communities for the success of MPC in fast embedded systems.

Jerez2012c [164] J. L. Jerez, G. A. Constantinides, and E. C. Kerrigan, "Fixed point lanczos: Sustaining TFLOP-equivalent performance in FPGAs for scientific computing," in *Proceedings of the 20th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, ser. FCCM '12, Toronto, ON, CA, Apr. 2012, pp. 53–60. DOI: `10.1109/FCCM.2012.19`,

Abstract: We consider the problem of enabling fixed-point implementations of linear algebra kernels to match the strengths of the field-programmable gate array (FPGA). Algorithms for solving linear equations, finding eigen values or finding singular values are typically nonlinear and recursive making the problem of establishing analytical bounds on variable dynamic range non-trivial. Current approaches fail to provide tight bounds for this type of algorithms. We use as a case study one of the most important kernels in scientific computing, the Lanczos iteration, which lies at the heart of well known methods such as conjugate gradient and minimum residual, and we show how we can modify the algorithm to allow us to apply standard linear algebra analysis to prove tight analytical bounds on all variables of the process, regardless of the properties of the original matrix. It is shown that the numerical behaviour of fixed-point implementations of the modified problem can be chosen to be at least as good as a double precision floating point implementation. Using this approach it is possible to get sustained FPGA performance very close to the peak general-purpose graphics processing unit (GPGPU) performance in FPGAs of comparable size when solving a single problem. If there are several independent problems to solve simultaneously it is possible to exceed the peak floating-point performance of a GPGPU, obtaining approximately 1, 2 or 4 TFLOPs for error tolerances of 10^-7, 10^-5 and 10^-3, respectively, in a large Virtex 7 FPGA.

Jerez2012 [165] ——, "Towards a fixed point QP solver for predictive control," in *Proceedings of the 51st IEEE Annual Conference on Decision and Control*, ser. CDC '12, Maui, HI, USA, Dec. 2012, pp. 675–680. DOI: 10.1109/CDC.2012.6427015,
Abstract: There is a need for high speed, low cost and low energy solutions for convex quadratic programming to enable model predictive control (MPC) to be implemented in a wider set of applications than is currently possible. For most quadratic programming (QP) solvers the computational bottleneck is the solution of systems of linear equations, which we propose to solve using a fixed-point implementation of an iterative linear solver to allow for fast and efficient computation in parallel hardware. However, fixed point arithmetic presents additional challenges, such as having to bound peak values of variables and constrain their dynamic ranges. For these types of algorithms the problems cannot be automated by current tools. We employ a preconditioner in a novel manner to allow us to establish tight analytical bounds on all the variables of the Lanczos process, the heart of modern iterative linear solving algorithms. The proposed approach is evaluated through the implementation of a mixed precision interior-point controller for a Boeing 747 aircraft. The numerical results show that there does not have to be a loss of control quality by moving from floating-point to fixed-point.

Jerez2012b [166] J. L. Jerez, E. C. Kerrigan, and G. A. Constantinides, "A sparse and condensed QP formulation for predictive control of LTI systems," *Automatica*, vol. 48, no. 5, pp. 999–1002, 2012, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2012.03.010,
Abstract: The computational burden that model predictive control (MPC) imposes depends to a large extent on the way the optimal control problem is formulated as an optimization problem. We present a formulation where the input is expressed as an affine function of the state such that the closed-loop dynamics matrix becomes nilpotent. Using this approach and removing the equality constraints leads to a compact and sparse optimization problem to be solved at each sampling instant. The problem can be solved with a cost per interior-point iteration that is linear with respect to the horizon length, when this is bigger than the controllability index of the plant. The computational complexity of existing condensed approaches grow cubically with the horizon length, whereas existing non-condensed and sparse approaches also grow linearly, but with a greater proportionality constant than with the method presented here.

Koch2012 [167] T. Koch, T. Ralphs, and Y. Shinano, "Could we use a million cores to solve an integer program?" *Mathematical Methods of Operations Research*, vol. 76, no. 1, pp. 67–93, 2012, ISSN: 1432-5217. DOI: 10.1007/s00186-012-0390-9,
Abstract: Given the steady increase in cores per CPU, it is only a matter of time before supercomputers will have a million or more cores. In this article, we investigate the opportunities and challenges that will arise when trying to utilize this vast computing power to solve a single integer linear optimization problem. We also raise the question of whether best practices in sequential solution of ILPs will be effective in massively parallel environments.

Lalami2012 [168] M. E. Lalami and D. El-Baz, "GPU implementation of the branch and bound method for knapsack problems," in *Proceedings of the 26th IEEE International Parallel and Distributed Processing Symposium Workshops and PhD Forum*, ser. IPDPSW '12, Shanghai, CN, 2012, pp. 1769–1777. DOI: 10.1109/IPDPSW.2012.219,
Abstract: In this paper, we propose an efficient implementation of the branch and bound method for knapsack problems on a CPU-GPU system via CUDA. Branch and bound computations can be carried out either on the CPU or on a GPU according to the size of the branch and bound list. A better management of GPUs memories, less GPU-CPU communications and better synchronization between GPU threads are proposed in this new implementation in order to increase efficiency. Indeed, a series of computational results is displayed and analyzed showing a substantial speedup on a Tesla C2050 GPU.

Liu2012 [169] L. Liu, L. Liu, and G. Yang, "A higly efficient CPU-GPU hybrid parallel implementation of sparse LU factorization," *Chinese Journal of Electronics*, vol. 21, no. 1, pp. 7–12, Jan. 2012. [Online]. Available: http://www.ejournal.org.cn/Jweb_cje/CN/abstract/abstract916.shtml,

Abstract: In this paper, we try to accelerate sparse LU factorization on GPU. We present a tiled storage format and a parallel algorithm to improve the memory access pattern, and a register blocking method to compress the on-chip working set. The OPENMP implementation of our algorithm gives more stable performance over different matrices, and outperforms SuperLU and KLU by 1.88 6 times on an Intel 8-core CPU (Central processing unit) for matrices from the Florida matrix collection. Based on this algorithm, we further propose a GPU-CPU hybrid pipelined scheme to overlap computations on CPU with computations on GPU. Compared to the better of SuperLU and KLU on an Intel 8-core CPU, our algorithm achieves 1.1 19.7-fold speedup on GPU for double precision. Compared to the OPENMP implementation of our algorithm on an Intel 8-core CPU, our GPU implementation gets a 2-fold speedup for the best cases.

Low2012 [170] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed GraphLab: A framework for machine learning and data mining in the cloud," *Proc. VLDB Endow.*, vol. 5, no. 8, pp. 716–727, Apr. 2012, ISSN: 2150-8097. DOI: 10.14778/2212351.2212354,

Abstract: While high-level data parallel frameworks, like MapReduce, simplify the design and implementation of large-scale data processing systems, they do not naturally or efficiently support many important data mining and machine learning algorithms and can lead to inefficient learning systems. To help fill this critical void, we introduced the GraphLab abstraction which naturally expresses asynchronous, dynamic, graph-parallel computation while ensuring data consistency and achieving a high degree of parallel performance in the shared-memory setting. In this paper, we extend the GraphLab framework to the substantially more challenging distributed setting while preserving strong data consistency guarantees.

We develop graph based extensions to pipelined locking and data versioning to reduce network congestion and mitigate the effect of network latency. We also introduce fault tolerance to the GraphLab abstraction using the classic Chandy-Lamport snapshot algorithm and demonstrate how it can be easily implemented by exploiting the GraphLab abstraction itself. Finally, we evaluate our distributed implementation of the GraphLab abstraction on a large Amazon EC2 deployment and show 1-2 orders of magnitude performance gains over Hadoop-based implementations.

Ma2012 [171] Y. Ma, F. Borrelli, B. Hencey, B. Coffey, S. Bengea, and P. Haves, "Model predictive control for the operation of building cooling systems," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 3, pp. 796–803, May 2012. DOI: 10.1109/TCST.2011.2124461,

Abstract: This brief presents a model-based predictive control (MPC) approach to building cooling systems with thermal energy storage. We focus on buildings equipped with a water tank used for actively storing cold water produced by a series of chillers. First, simplified models of chillers, cooling towers, thermal storage tanks, and buildings are developed and validated for the purpose of model-based control design. Then an MPC for the chilling system operation is proposed to optimally store the thermal energy in the tank by using predictive knowledge of building loads and weather conditions. This brief addresses real-time implementation and feasibility issues of the MPC scheme by using a simplified hybrid model of the system, a periodic robust invariant set as terminal constraints, and a moving window blocking strategy. The controller is experimentally validated at the University of California, Merced. The experiments show a reduction in the central plant electricity cost and an improvement of its efficiency.

Mattingley2012 [172] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," English, *Optimization Engineering*, vol. 13, no. 1, pp. 1–27, 2012, ISSN: 1389-4420. DOI: 10.1007/s11081-011-9176-9,

Abstract: CVXGEN is a software tool that takes a high level description of a convex optimization problem family, and automatically generates custom C code that compiles into a reliable, high speed solver for the problem family. The current implementation targets problem families that can be transformed, using disciplined convex programming techniques, to convex quadratic programs of modest size. CVXGEN generates simple, flat, library-free code suitable for embedding in real-time applications. The generated code is almost branch free, and so has highly predictable run-time behavior. The combination of regularization (both static and dynamic) and iterative refinement in the search direction computation yields reliable performance, even with poor quality data. In

this paper we describe how CVXGEN is implemented, and give some results on the speed and reliability of the automatically generated solvers.

Melab2012 [173] N. Melab, I. Chakroun, M. Mezmaz, and D. Tuyttens, "A GPU-accelerated branch-and-bound algorithm for the flow-shop scheduling problem," in *IEEE Int. Conf. Cluster Computing*, ser. CLUSTER '12, Beijing, CN, Sep. 2012, pp. 10–17. DOI: 10.1109/CLUSTER.2012.18,
Abstract: Branch-and-Bound (B&B) algorithms are time-intensive tree-based exploration methods for solving to optimality combinatorial optimization problems. In this paper, we investigate the use of GPU computing as a major complementary way to speed up those methods. The focus is put on the bounding mechanism of B&B algorithms, which is the most time consuming part of their exploration process. We propose a parallel B&B algorithm based on a GPU-accelerated bounding model. The proposed approach concentrate on optimizing data access management to further improve the performance of the bounding mechanism which uses large and intermediate data sets that do not completely fit in GPU memory. Extensive experiments of the contribution have been carried out on well-known FSP benchmarks using an Nvidia Tesla C2050 GPU card. We compared the obtained performances to a single and a multithreaded CPU-based execution. Accelerations up to $\times 100$ are achieved for large problem instances.

Merrill2012 [174] D. Merrill, M. Garland, and A. Grimshaw, "Scalable GPU graph traversal," *ACM SIGPLAN Notices*, vol. 47, no. 8, pp. 117–128, Feb. 2012, ISSN: 0362-1340. DOI: 10.1145/2370036.2145832,
Abstract: Breadth-first search (BFS) is a core primitive for graph traversal and a basis for many higher-level graph analysis algorithms. It is also representative of a class of parallel computations whose memory accesses and work distribution are both irregular and data-dependent. Recent work has demonstrated the plausibility of GPU sparse graph traversal, but has tended to focus on asymptotically inefficient algorithms that perform poorly on graphs with non-trivial diameter.
We present a BFS parallelization focused on fine-grained task management constructed from efficient prefix sum that achieves an asymptotically optimal $O(|V| + |E|)$ work complexity. Our implementation delivers excellent performance on diverse graphs, achieving traversal rates in excess of 3.3 billion and 8.3 billion traversed edges per second using single and quad-GPU configurations, respectively. This level of performance is several times faster than state-of-the-art implementations both CPU and GPU platforms.

CudaK20B [175] "NVIDIA tesla K20-K20X GPU accelerators – benchmarks," NVIDIA Corporation, Tech. Rep., 2012, Accessed 20 July 2016. [Online]. Available: http://www.nvidia.com/docs/IO/122874/K20-and-K20X-application-performance-technical-brief.pdf.

Reguly2012 [176] I. Reguly and M. Giles, "Efficient sparse matrix-vector multiplication on cache-based GPUs," in *Proceedings of Innovative Parallel Computing*, ser. InPar '12, San Jose, CA, USA, May 2012, pp. 1–12. DOI: 10.1109/InPar.2012.6339602,
Abstract: Sparse matrix-vector multiplication is an integral part of many scientific algorithms. Several studies have shown that it is a bandwidth-limited operation on current hardware. On cache-based architectures the main factors that influence performance are spatial locality in accessing the matrix, and temporal locality in re-using the elements of the vector. This paper discusses efficient implementations of sparse matrix-vector multiplication on NVIDIA's Fermi architecture, the first to introduce conventional L1 caches to GPUs. We focus on the compressed sparse row (CSR) format for developing general purpose code. We present a parametrised algorithm, show the effects of parameter tuning on performance and introduce a method for determining the nearoptimal set of parameters that incurs virtually no overhead. On a set of sparse matrices from the University of Florida Sparse Matrix Collection we show an average speed-up of 2.1 times over NVIDIA's CUSPARSE 4.0 library in single precision and 1.4 times in double precision. Many algorithms require repeated evaluation of sparse matrix-vector products with the same matrix, so we introduce a dynamic run-time auto-tuning system which improves performance by 10-15% in seven iterations. The CSR format is compared to alternative ELLPACK and HYB formats and the cost of conversion is assessed using CUSPARSE. Sparse matrix-vector multiplication performance is also

analysed when solving a finite element problem with the conjugate gradient method. We show how problemspecific knowledge can be used to improve performance by up to a factor of two.

Ren2012 [177] L. Ren, X. Chen, Y. Wang, C. Zhang, and H. Yang, "Sparse LU factorization for parallel circuit simulation on GPU," in *Proceedings of the 49th ACM/EDAC/IEEE Design Automation Conference*, ser. DAC '12, San Francisco, CA, US, Jun. 2012, pp. 1125–1130, ISBN: 978-1-4503-1199-1. [Online]. Available: http://ieeexplore.ieee.org/document/6241646/,
Abstract: Sparse solver has become the bottleneck of SPICE simulators. There has been few work on GPU-based sparse solver because of the high data-dependency. The strong data-dependency determines that parallel sparse LU factorization runs efficiently on shared-memory computing devices. But the number of CPU cores sharing the same memory is often limited. The state of the art Graphic Processing Units (GPU) naturally have numerous cores sharing the device memory, and provide a possible solution to the problem. In this paper, we propose a GPU-based sparse LU solver for circuit simulation. We optimize the work partitioning, the number of active thread groups, and the memory access pattern, based on GPU architecture. On matrices whose factorization involves many floating-point operations, our GPU-based sparse LU factorization achieves 7.90× speedup over 1-core CPU and 1.49× speedup over 8-core CPU. We also analyze the scalability of parallel sparse LU factorization and investigate the specifications on CPUs and GPUs that most influence the performance.

Shinano2012 [178] Y. Shinano, T. Achterberg, T. Berthold, S. Heinz, and T. Koch, "ParaSCIP: A parallel extension of SCIP," in *Competence in High Performance Computing*, C. Bischof, H.-G. Hegering, W. Nagel, and G. Wittum, Eds., 2012, pp. 135–148. DOI: 10.1007/978-3-642-24025-6_12,
Abstract: Mixed integer programming (MIP) has become one of the most important techniques in Operations Research and Discrete Optimization. SCIP (Solving Constraint Integer Programs) is currently one of the fastest non-commercial MIP solvers. It is based on the branch-and-bound procedure in which the problem is recursively split into smaller subproblems, thereby creating a so-called branching tree. We present ParaSCIP, an extension of SCIP, which realizes a parallelization on a distributed memory computing environment. ParaSCIP uses SCIP solvers as independently running processes to solve subproblems (nodes of the branching tree) locally. This makes the parallelization development independent of the SCIP development. Thus, ParaSCIP directly profits from any algorithmic progress in future versions of SCIP. Using a first implementation of ParaSCIP, we were able to solve two previously unsolved instances from MIPLIB2003, a standard test set library for MIP solvers. For these computations, we used up to 2048 cores of the HLRN II supercomputer.

Smith2012 [179] E. Smith, J. Gondzio, and J. Hall, "GPU acceleration of the matrix-free interior point method," in *Parallel Processing and Applied Mathematics*, ser. Lecture Notes in Computer Science, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Waśniewski, Eds., vol. 7203, Springer Berlin Heidelberg, 2012, pp. 681–689, ISBN: 978-3-642-31463-6. DOI: 10.1007/978-3-642-31464-3_69,
Abstract: The matrix-free technique is an iterative approach to interior point methods (IPM), so named because both the solution procedure and the computation of an appropriate preconditioner require only the results of the operations Ax and ATy, where A is the matrix of constraint coefficients. This paper demonstrates its overwhelmingly superior performance on two classes of linear programming (LP) problems relative to both the simplex method and to IPM with equations solved directly. It is shown that the reliance of this technique on sparse matrix-vector operations enables further, significant performance gains from the use of a GPU, and from multi-core processors.

Suchoski2012 [180] B. Suchoski, C. Severn, M. Shantharam, and P. Raghavan, "Adapting sparse triangular solution to GPUs," in *Proceeding of the 41st International Conference on Parallel Processing Workshops*, ser. ICPPW '12, Los Alamitos, CA, USA, Sep. 2012, pp. 140–148. DOI: 10.1109/ICPPW.2012.23,
Abstract: High performance computing systems are increasingly incorporating hybrid CPU/GPU nodes to accelerate the rate at which floating point calculations can be performed for scientific applications. Currently, a key challenge is adapting scientific applications to such systems when

the underlying computations are sparse, such as sparse linear solvers for the simulation of partial differential equation models using semi-implicit methods. Now, a key bottleneck is sparse triangular solution for solvers such as preconditioned conjugate gradients (PCG). We show that sparse triangular solution can be effectively mapped to GPUs by extracting very large degrees of fine-grained parallelism using graph coloring. We develop simple performance models to predict these effects at intersection of the data and hardware attributes and we evaluate our scheme on a Nvidia Tesla M2090 GPU relative to the level set scheme developed at NVIDIA. Our results indicate that our approach significantly enhances the available fine-grained parallelism to speed-up PCG iteration time compared to the NVIDIA scheme, by a factor with a geometric mean of 5.41 on a single GPU, with speedups as high as 63 in some cases.

Ubal2012 [181] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli, "Multi2sim: A simulation framework for CPU-GPU computing," in *21st International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '12, Sep. 2012, pp. 335–344,
Abstract: Accurate simulation is essential for the proper design and evaluation of any computing platform. Upon the current move toward the CPU-GPU heterogeneous computing era, researchers need a simulation framework that can model both kinds of computing devices and their interaction. In this paper, we present Multi2Sim, an open-source, modular, and fully configurable toolset that enables ISA-level simulation of an x86 CPU and an AMD Evergreen GPU. Focusing on a model of the AMD Radeon 5870 GPU, we address program emulation correctness, as well as architectural simulation accuracy, using AMD's OpenCL benchmark suite. Simulation capabilities are demonstrated with a preliminary architectural exploration study, and workload characterization examples. The project source code, benchmark packages, and a detailed user's guide are publicly available at www.multi2sim.org.

Unkule2012 [182] S. Unkule, C. Shaltz, and A. Qasem, "Automatic restructuring of GPU kernels for exploiting inter-thread data locality," in *Proceedings of the 21st International Conference on Compiler Construction*, ser. CC '12, Tallinn, EE: Springer-Verlag, 2012, pp. 21–40, ISBN: 978-3-642-28651-3. DOI: 10.1007/978-3-642-28652-0_2,
Abstract: Hundreds of cores per chip and support for fine-grain multithreading have made GPUs a central player in today's HPC world. For many applications, however, achieving a high fraction of peak on current GPUs, still requires significant programmer effort. A key consideration for optimizing GPU code is determining a suitable amount of work to be performed by each thread. Thread granularity not only has a direct impact on occupancy but can also influence data locality at the register and shared-memory levels. This paper describes a software framework to analyze dependencies in parallel GPU threads and perform source-level restructuring to obtain GPU kernels with varying thread granularity. The framework supports specification of coarsening factors through source-code annotation and also implements a heuristic based on estimated register pressure that automatically recommends coarsening factors for improved memory performance. We present preliminary experimental results on a select set of CUDA kernels. The results show that the proposed strategy is generally able to select profitable coarsening factors. More importantly, the results demonstrate a clear need for automatic control of thread granularity at the software level for achieving higher performance.

Belotti2013 [183] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, "Mixed-integer nonlinear optimization," *Acta Numerica*, vol. 22, pp. 1–131, May 2013, ISSN: 1474-0508. DOI: 10.1017/S0962492913000032,
Abstract: Many optimal decision problems in scientific, engineering, and public sector applications involve both discrete decisions and nonlinear system dynamics that affect the quality of the final design or plan. These decision problems lead to mixed-integer nonlinear programming (MINLP) problems that combine the combinatorial difficulty of optimizing over discrete variable sets with the challenges of handling nonlinear functions. We review models and applications of MINLP, and survey the state of the art in methods for solving this challenging class of problems.
Most solution methods for MINLP apply some form of tree search. We distinguish two broad classes of methods: single-tree and multitree methods. We discuss these two classes of methods first in the case where the underlying problem functions are convex. Classical single-tree methods include non-

linear branch-and-bound and branch-and-cut methods, while classical multitree methods include outer approximation and Benders decomposition. The most efficient class of methods for convex MINLP are hybrid methods that combine the strengths of both classes of classical techniques.
Non-convex MINLPs pose additional challenges, because they contain non-convex functions in the objective function or the constraints; hence even when the integer variables are relaxed to be continuous, the feasible region is generally non-convex, resulting in many local minima. We discuss a range of approaches for tackling this challenging class of problems, including piecewise linear approximations, generic strategies for obtaining convex relaxations for non-convex functions, spatial branch-and-bound methods, and a small sample of techniques that exploit particular types of non-convex structures to obtain improved convex relaxations.
We finish our survey with a brief discussion of three important aspects of MINLP. First, we review heuristic techniques that can obtain good feasible solution in situations where the search-tree has grown too large or we require real-time solutions. Second, we describe an emerging area of mixed-integer optimal control that adds systems of ordinary differential equations to MINLP. Third, we survey the state of the art in software for MINLP.

Berkelaar2013 [184] M. Berkelaar, K. Eikland, and P. Notebaert, *lp_solve reference guide*, 5.5.2.0, Accessed 30 March 2014, Apr. 2013. [Online]. Available: `http://lpsolve.sourceforge.net/5.5/`.

Boechat2013 [185] M.-A. Boéchat, J. Liu, H. Peyril, A. Zanarini, and T. Besselmann, "An architecture for solving quadratic programs with the fast gradient method on a field programmable gate array," in *Proceedings of the 21st Mediterranean Conference on Control Automation*, ser. MED '13, Platanias-Chania, GR, Jun. 2013, pp. 1557–1562. DOI: `10.1109/MED.2013.6608929`,
Abstract: In this paper an architecture for the implementation of gradient-based optimisation methods on a Field Programmable Gate Array (FPGA) is proposed. Combining the algorithmic advantages of gradient-based algorithms with the computational strengths of a tailored FPGA implementation allows to solve quadratic programs occurring, for example, in Model Predictive Control (MPC) applications in the microsecond range. The experimental comparisons show a computational advantage of the proposed FPGA implementation against parallel software versions ranging between one and two orders of magnitude. The proposed FPGA-based solution can broaden the applicability of MPC to problems that were considered out-of-reach till recent years.

Boyer2013 [186] V. Boyer and D. El Baz, "Recent advances on GPU computing in operations research," in *Proceedings of the 27th IEEE International Symposium on Parallel & Distributed Processing Workshops and PhD Forum*, ser. IPDPSW '13, Boston, MA, USA: IEEE Computer Society, May 2013, pp. 1778–1787. DOI: `10.1109/IPDPSW.2013.45`,
Abstract: In the last decade, Graphics Processing Units(GPUs) have gained an increasing popularity as accelerators for High Performance Computing (HPC) applications. Recent GPUs are not only powerful graphics engines but also highly threaded parallel computing processors that can achieve sustainable speedup as compared with CPUs. In this context, researchers try to exploit the capability of this architecture to solve difficult problems in many domains in science and engineering. In this article, we present recent advances on GPU Computing in Operations Research. We focus in particular on Integer Programming and Linear Programming.

Chakroun2013 [187] I. Chakroun, N. Melab, M. Mezmaz, and D. Tuyttens, "Combining multi-core and GPU computing for solving combinatorial optimization problems," *Journal of Parallel and Distributed computing*, vol. 73, no. 12, pp. 1563–1577, Dec. 2013, ISSN: 0743-7315. DOI: `10.1016/j.jpdc.2013.07.023`,
Abstract: In this paper, we revisit the design and implementation of Branch-and-Bound (B&B) algorithms for solving large combinatorial optimization problems on GPU-enhanced multi-core machines. B&B is a tree-based optimization method that uses four operators (selection, branching, bounding and pruning) to build and explore a highly irregular tree representing the solution space. In our previous works, we have proposed a GPU-accelerated approach in which only a single CPU core is used and only the bounding operator is performed on the GPU device. Here, we extend the approach (LL-GB&B) in order to minimize the CPU-GPU communication latency and thread divergence. Such an objective is achieved through a GPU-based fine-grained parallelization of the branching and pruning operators in addition to the bounding one. The second contribution

consists in investigating the combination of a GPU with multi-core processing. Two scenarios have been explored leading to two approaches: a concurrent (RLL-GB&B) and a cooperative one (PLL-GB&B). In the first one, the exploration process is performed concurrently by the GPU and the CPU cores. In the cooperative approach, the CPU cores prepare and off-load to GPU pools of tree nodes using data streaming while the GPU performs the exploration. The different approaches have been extensively experimented on the Flowshop scheduling problem. Compared to a single CPU-based execution, LL-GB&B allows accelerations up to (×160) for large problem instances. Moreover, when combining multi-core and GPU, we figure out that using RLL-GB&B is not beneficial while PLL-GB&B enables an improvement up to 36% compared to LL-GB&B.

Che2013 [188] S. Che, B. M. Beckmann, S. K. Reinhardt, and K. Skadron, "Pannotia: Understanding irregular GPGPU graph applications," in *Proceedings of the IEEE International Symposium on Workload Characterization*, ser. IISWC '13, Portland, OR, USA, Sep. 2013, pp. 185–195. DOI: 10.1109/IISWC.2013.6704684,
Abstract: GPUs have become popular recently to accelerate general-purpose data-parallel applications. However, most existing work has focused on GPU-friendly applications with regular data structures and access patterns. While a few prior studies have shown that some irregular workloads can also achieve speedups on GPUs, this domain has not been investigated thoroughly. Graph applications are one such set of irregular workloads, used in many commercial and scientific domains. In particular, graph mining -as well as web and social network analysis- are promising applications that GPUs could accelerate. However, implementing and optimizing these graph algorithms on SIMD architectures is challenging because their data-dependent behavior results in significant branch and memory divergence. To address these concerns and facilitate research in this area, this paper presents and characterizes a suite of GPGPU graph applications, Pannotia, which is implemented in OpenCL and contains problems from diverse and important graph application domains. We perform a first-step characterization and analysis of these benchmarks and study their behavior on real hardware. We also use clustering analysis to illustrate the similarities and differences of the applications in the suite. Finally, we make architectural and scheduling suggestions that will improve their execution efficiency on GPUs.

CUDA2013 [189] *CUDA toolkit documentation*, English, 5.5, Accessed 1 March 2014, Nvidia corporation, Oct. 2013. [Online]. Available: http://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html.

Demidov2013 [190] D. Demidov, K. Ahnert, K. Rupp, and P. Gottschling, "Programming CUDA and OpenCL: A case study using modern C++ libraries," *SIAM Journal on Scientific Computing*, vol. 35, no. 5, pp. C453–C472, 2013. DOI: 10.1137/120903683,
Abstract: We present a comparison of several modern C++ libraries providing high-level interfaces for programming multi- and many-core architectures on top of CUDA or OpenCL. The comparison focuses on the solution of ordinary differential equations (ODEs) and is based on odeint, a framework for the solution of systems of ODEs. Odeint is designed in a very flexible way and may be easily adapted for effective use of libraries such as MTL4, VexCL, or ViennaCL, using CUDA or OpenCL technologies. We found that CUDA and OpenCL work equally well for problems of large sizes, while OpenCL has higher overhead for smaller problems. Furthermore, we show that modern high-level libraries allow us to effectively use the computational resources of many-core GPUs or multicore CPUs without much knowledge of the underlying technologies.

Gottschling2013 [191] P. Gottschling. (2013). CUDA-MTL4 manual. Accessed on 10 May 2017, SimuNova UG, [Online]. Available: http://www.simunova.com/node/300,
Abstract: Many things can be realized on a computer very elegantly and efficiently today thanks to progress in software and programming languages. One thing that cannot be done elegantly on a computer is computing. At least not computing fast.
In the Matrix Template Library 4 we aim for a natural mathematical notation without sacrifying performance. You can write an expression like $x = y * z$ and the library will perform the according operation: scaling a vector, multiplying a sparse matrix with a dense vector or two sparse matrices. Some operations like dense matrix product use tuned BLAS implementation. In parallel, all described operations in this manual are also realized in C++ so that the library can be used

without BLAS and is not limited to types supported by BLAS. For short, general applicability is combined with maximal available performance. We developed new techniques to allow for (1) Unrolling of dynamicly sized data with user-define block and tile sizes; (2) Combining multiple vector assignments in a single statement (and more importingly perform them in one single loop); (3) Storing matrices recursively in a never-before realized generality; (4) Performing operations on recursive and non-recursive matrices recursively; (5) Filling compressed sparse matrices efficiently; and much more.
The manual still not covers all features and techniques of the library. But it should give you enough information to get started.

Hogg2013 [192] J. Hogg, "A fast dense triangular solve in cuda," *SIAM Journal on Scientific Computing*, vol. 35, no. 3, pp. C303–C322, 2013. DOI: 10.1137/12088358X,
Abstract: The level 2 BLAS operation _trsv performs a dense triangular solve and is often used in the solve phase of a direct solver following a matrix factorization. With the advent of manycore architectures reducing the cost of compute-bound parts of the computation, memory-bound operations such as this kernel become increasingly important. This is particularly noticeable in sparse direct solvers used for optimization applications where multiple memory-bound solves follow each (traditionally expensive) compute-bound factorization. In this paper, a high performance implementation of the triangular solve is developed through an analysis of theoretical and practical bounds on its run time. This implementation outperforms the CUBLAS by a factor of 5–15.

Hogg2013a [193] J. Hogg and J. Scott, "New parallel sparse direct solvers for multicore architectures," *Algorithms*, vol. 6, no. 4, pp. 702–725, 2013, ISSN: 1999-4893. DOI: 10.3390/a6040702. [Online]. Available: http://www.mdpi.com/1999-4893/6/4/702,
Abstract: At the heart of many computations in science and engineering lies the need to efficiently and accurately solve large sparse linear systems of equations. Direct methods are frequently the method of choice because of their robustness, accuracy and potential for use as black-box solvers. In the last few years, there have been many new developments, and a number of new modern parallel general-purpose sparse solvers have been written for inclusion within the HSL mathematical software library. In this paper, we introduce and briefly review these solvers for symmetric sparse systems. We describe the algorithms used, highlight key features (including bit-compatibility and out-of-core working) and then, using problems arising from a range of practical applications, we illustrate and compare their performances. We demonstrate that modern direct solvers are able to accurately solve systems of order 106 in less than 3 minutes on a 16-core machine.

Altera2013 [194] *Implementing FPGA design with the OpenCL standard*, Whitepaper, Nov. 2013. [Online]. Available: http://www.altera.co.uk/literature/wp/wp-01173-opencl.pdf.

Jerez2013 [195] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded predictive control on an FPGA using the fast gradient method," in *Proceedings of the 2013 European Control Conference*, ser. ECC '13, Zurich, CH, Jul. 2013, pp. 3614–3620, ISBN: 978-3-033-03962-9. [Online]. Available: http://ieeexplore.ieee.org/document/6669598/,
Abstract: Model predictive control (MPC) in resource-constrained embedded platforms requires faster, cheaper and more power-efficient solvers for convex programs than is currently offered by software-based solutions. In this paper we present the first field programmable gate array (FPGA) implementation of a fast gradient solver for linear-quadratic MPC problems with input constraints. We use fixed-point arithmetic to exploit the characteristics of the computing platform and provide analytical guarantees ensuring no overflow errors occur during operation. We further prove that the arithmetic errors due to round-off can lead only to reduced accuracy, but not instability, of the fast gradient method. The results are demonstrated on a model of an industrial atomic force microscope (AFM) where we show that, on a low-end FPGA, satisfactory control performance at a sample rate beyond 1 MHz is achievable, opening up new possibilities for the application of MPC.

Kayiran2013 [196] O. Kayıran, A. Jog, M. T. Kandemir, and C. R. Das, "Neither more nor less: Optimizing thread-level parallelism for GPGPUs," in *Proceedings of the 22nd International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '13, Edinburgh, UK:

IEEE Press, 2013, pp. 157–166, ISBN: 978-1-4799-1021-2. [Online]. Available: http://dl.acm.org/citation.cfm?id=2523721.2523745,
Abstract: General-purpose graphics processing units (GPGPUs) are at their best in accelerating computation by exploiting abundant thread-level parallelism (TLP) offered by many classes of HPC applications. To facilitate such high TLP, emerging programming models like CUDA and OpenCL allow programmers to create work abstractions in terms of smaller work units, called cooperative thread arrays (CTAs). CTAs are groups of threads and can be executed in any order, thereby providing ample opportunities for TLP. The state-of-the-art GPGPU schedulers allocate maximum possible CTAs per-core (limited by available on-chip resources) to enhance performance by exploiting TLP. However, we demonstrate in this paper that executing the maximum possible number of CTAs on a core is not always the optimal choice from the performance perspective. High number of concurrently executing threads might cause more memory requests to be issued, and create contention in the caches, network and memory, leading to long stalls at the cores. To reduce resource contention, we propose a dynamic CTA scheduling mechanism, called DYNCTA, which modulates the TLP by allocating optimal number of CTAs, based on application characteristics. To minimize resource contention, DYNCTA allocates fewer CTAs for applications suffering from high contention in the memory sub-system, compared to applications demonstrating high throughput. Simulation results on a 30-core GPGPU platform with 31 applications show that the proposed CTA scheduler provides 28% average improvement in performance compared to the existing CTA scheduler.

Kurzak2013 [197] J. Kurzak, P. Luszczek, M. Faverge, and J. Dongarra, "LU factorization with partial pivoting for a multicore system with accelerators," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 8, pp. 1613–1621, Aug. 2013, ISSN: 1045-9219. DOI: 10.1109/TPDS.2012.242,
Abstract: LU factorization with partial pivoting is a canonical numerical procedure and the main component of the high performance LINPACK benchmark. This paper presents an implementation of the algorithm for a hybrid, shared memory, system with standard CPU cores and GPU accelerators. The difficulty of implementing the algorithm for such a system lies in the disproportion between the computational power of the CPUs, compared to the GPUs, and in the meager bandwidth of the communication link between their memory systems. An additional challenge comes from the complexity of the memory-bound and synchronization-rich nature of the panel factorization component of the block LU algorithm, imposed by the use of partial pivoting. The challenges are tackled with the use of a data layout geared toward complex memory hierarchies, autotuning of GPU kernels, fine-grain parallelization of memory-bound CPU operations and dynamic scheduling of tasks to different devices. Performance in excess of one TeraFLOPS is achieved using four AMD Magny Cours CPUs and four NVIDIA Fermi GPUs.

Li2013 [198] R. Li and Y. Saad, "GPU-accelerated preconditioned iterative linear solvers," *Journal of Supercomputing*, vol. 63, no. 2, pp. 443–466, Feb. 2013, ISSN: 0920-8542. DOI: 10.1007/s11227-012-0825-3,
Abstract: This work is an overview of our preliminary experience in developing a high-performance iterative linear solver accelerated by GPU coprocessors. Our goal is to illustrate the advantages and difficulties encountered when deploying GPU technology to perform sparse linear algebra computations. Techniques for speeding up sparse matrix-vector product (SpMV) kernels and finding suitable preconditioning methods are discussed. Our experiments with an NVIDIA TESLA M2070 show that for unstructured matrices SpMV kernels can be up to 8 times faster on the GPU than the Intel MKL on the host Intel Xeon X5675 Processor. Overall performance of the GPU-accelerated Incomplete Cholesky (IC) factorization preconditioned CG method can outperform its CPU counterpart by a smaller factor, up to 3, and GPU-accelerated The incomplete LU (ILU) factorization preconditioned GMRES method can achieve a speed-up nearing 4. However, with better suited preconditioning techniques for GPUs, this performance can be further improved.

limateChange2013 [199] "Meeting carbon budgets: 2013 progress report to parliament," The Committee on Climate Change, Tech. Rep., 2013, ch. 3. [Online]. Available: http://www.theccc.org.uk/wp-content/uploads/2013/06/CCC-Prog-Rep_Chap3_singles_web_1.pdf.

Meyer2013 [200] X. Meyer, B. Chopard, and P. Albuquerque, "Linear programming on a GPU: A case study," in *Designing Scientific Applications on GPUs*, R. Couturier, Ed., ser. Numerical Analysis and Scientific Computing. Chapman & Hall/CRC Press, 2013, ch. 10, pp. 215–249, ISBN: 9781466571624.

Mocanu2013 [201] A. Mocanu and N. Ţăpuş, "Sparse matrix permutations to a block triangular form in a distributed environment," in *IEEE International Conference on Intelligent Computer Communication and Processing*, ser. ICCP '13, Lyon, FR, Sep. 2013, pp. 331–338. DOI: 10.1109/ICCP.2013.6646131,
Abstract: Arranging the sparse circuit matrix into a diagonal block upper triangular form is the first step of the KLU algorithm. This paper presents the two steps of the parallel algorithm, running in a distributed environment, that performs unsymmetric and symmetric permutations of the matrix's rows. First, using the [Duff] maximum transversal algorithm and performing asymmetrical permutations, the matrix is shaped to achieve a zero free diagonal. Then, searching the strongly connected components of the associated matrix's graph, and performing symmetric permutation, the sparse matrix is shaped in a diagonal block upper triangular form. Both algorithm and architecture are presented.

Nasre2013 [202] R. Nasre, M. Burtscher, and K. Pingali, "Atomic-free irregular computations on GPUs," in *Proceedings of the 6th Workshop on General Purpose Processor Using Graphics Processing Units*, ser. GPGPU-6, Houston, Texas, USA: ACM, 2013, pp. 96–107, ISBN: 978-1-4503-2017-7. DOI: 10.1145/2458523.2458533,
Abstract: Atomic instructions are a key ingredient of codes that operate on irregular data structures like trees and graphs. It is well known that atomics can be expensive, especially on massively parallel GPUs, and are often on the critical path of a program. In this paper, we present two high-level methods to eliminate atomics in irregular programs. The first method advocates synchronous processing using barriers. We illustrate how to exploit synchronous processing to avoid atomics even when the threads' memory accesses conflict with each other. The second method is based on exploiting algebraic properties of algorithms to elide atomics. Specifically, we focus on three key properties: monotonicity, idempotency and associativity, and show how each of them enables an atomic-free implementation. We illustrate the generality of the two methods by applying them to five irregular graph applications: breadth-first search, single-source shortest paths computation, Delaunay mesh refinement, pointer analysis and survey propagation, and show that both methods provide substantial speedup in each case on different GPUs.

Nasre2013a [203] ——, "Data-driven versus topology-driven irregular computations on GPUs," in *Proceedings of the 27th IEEE International Symposium on Parallel and Distributed Processing*, ser. IPDPS '13, Boston, MA, US: IEEE Computer Society, 2013, pp. 463–474, ISBN: 978-0-7695-4971-2. DOI: 10.1109/IPDPS.2013.28,
Abstract: Irregular algorithms are algorithms with complex main data structures such as directed and undirected graphs, trees, etc. A useful abstraction for many irregular algorithms is its operator formulation in which the algorithm is viewed as the iterated application of an operator to certain nodes, called active nodes, in the graph. Each operator application, called an activity, usually touches only a small part of the overall graph, so nonoverlapping activities can be performed in parallel. In topology-driven implementations, all nodes are assumed to be active so the operator is applied everywhere in the graph even if there is no work to do at some nodes. In contrast, in data-driven implementations the operator is applied only to nodes at which there might be work to do. Multicore implementations of irregular algorithms are usually data-driven because current multicores only support small numbers of threads and work-efficiency is important. Conversely, many irregular GPU implementations use a topology-driven approach because work inefficiency can be counterbalanced by the large number of GPU threads. In this paper, we study data-driven and topology-driven implementations of six important graph algorithms on GPUs. Our goal is to understand the tradeoffs between these implementations and how to optimize them. We find that data-driven versions are generally faster and scale better despite the cost of maintaining a worklist. However, topology-driven versions can be superior when certain algorithmic properties

are exploited to optimize the implementation. These results led us to devise hybrid approaches that combine the two techniques and outperform both of them.

Nguyen2013 [204] D. Nguyen, A. Lenharth, and K. Pingali, "A lightweight infrastructure for graph analytics," in *Proceedings of the 24th ACM Symposium on Operating Systems Principles*, ser. SOSP '13, Farminton, PA, USA: ACM, 2013, pp. 456–471, ISBN: 978-1-4503-2388-8. DOI: 10.1145/2517349.2522739,
Abstract: Several domain-specific languages (DSLs) for parallel graph analytics have been proposed recently. In this paper, we argue that existing DSLs can be implemented on top of a general-purpose infrastructure that (i) supports very fine-grain tasks, (ii) implements autonomous, speculative execution of these tasks, and (iii) allows application-specific control of task scheduling policies. To support this claim, we describe such an implementation called the Galois system.
We demonstrate the capabilities of this infrastructure in three ways. First, we implement more sophisticated algorithms for some of the graph analytics problems tackled by previous DSLs and show that end-to-end performance can be improved by orders of magnitude even on power-law graphs, thanks to the better algorithms facilitated by a more general programming model. Second, we show that, even when an algorithm can be expressed in existing DSLs, the implementation of that algorithm in the more general system can be orders of magnitude faster when the input graphs are road networks and similar graphs with high diameter, thanks to more sophisticated scheduling. Third, we implement the APIs of three existing graph DSLs on top of the common infrastructure in a few hundred lines of code and show that even for power-law graphs, the performance of the resulting implementations often exceeds that of the original DSL systems, thanks to the lightweight infrastructure.

Nvidia2013 [205] *NVIDIA professional graphics solution*, Online, NVIDIA Corporation, Jul. 2013. [Online]. Available: http://www.nvidia.co.uk/content/PDF/line_card/6660-nv-prographicssolutions-linecard-july13-final-lr.pdf.

Ploskas2013 [206] N. Ploskas and N. Samaras, "A computational comparison of basis updating schemes for the simplex algorithm on a CPU-GPU system," *American Journal of Operations Research*, vol. 3, no. 6, pp. 497–505, Nov. 2013. DOI: 10.4236/ajor.2013.36048,
Abstract: The computation of the basis inverse is the most time-consuming step in simplex type algorithms. This inverse does not have to be computed from scratch at any iteration, but updating schemes can be applied to accelerate this calculation. In this paper, we perform a computational comparison in which the basis inverse is computed with five different updating schemes. Then, we propose a parallel implementation of two updating schemes on a CPU-GPU System using MATLAB and CUDA environment. Finally, a computational study on randomly generated full dense linear programs is presented to establish the practical value of GPU-based implementation.

Shinano2013 [207] Y. Shinano, S. Heinz, S. Vigerske, and M. Winkler, "FiberSCIP: A shared memory parallelization of SCIP," ZIB, Takustr.7, 14195 Berlin, Tech. Rep. 13–55, 2013,
Abstract: Recently, parallel computing environments have become significantly popular. In order to obtain the benefit of using parallel computing environments, we have to deploy our programs for these effectively. This paper focuses on a parallelization of SCIP (Solving Constraint Integer Programs), which is a MIP solver and constraint integer programming framework available in source code. There is a parallel extension of SCIP named ParaSCIP, which parallelizes SCIP on massively parallel distributed memory computing environments. This paper describes FiberSCIP, which is yet another parallel extension of SCIP to utilize multi-threaded parallel computation on shared memory computing environments, and has the following contributions: First, the basic concept of having two parallel extensions and the relationship between them and the parallelization framework provided by UG (Ubiquity Generator) is presented, including an implementation of deterministic parallelization. Second, the difficulties to achieve a good performance that utilizes all resources on an actual computing environment and the difficulties of performance evaluation of the parallel solvers are discussed. Third, a way to evaluate the performance of new algorithms and parameter settings of the parallel extensions is presented. Finally, current performance of FiberSCIP for solving mixed-integer linear programs (MIPs) and mixed-integer non-linear programs (MINLPs) in parallel is demonstrated.

Shun2013 [208] J. Shun and G. E. Blelloch, "Ligra: A lightweight graph processing framework for shared memory," in *Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '13, Shenzhen, CN: ACM, 2013, pp. 135–146, ISBN: 978-1-4503-1922-5. DOI: 10.1145/2442516.2442530,
Abstract: There has been significant recent interest in parallel frameworks for processing graphs due to their applicability in studying social networks, the Web graph, networks in biology, and unstructured meshes in scientific simulation. Due to the desire to process large graphs, these systems have emphasized the ability to run on distributed memory machines. Today, however, a single multicore server can support more than a terabyte of memory, which can fit graphs with tens or even hundreds of billions of edges. Furthermore, for graph algorithms, shared-memory multicores are generally significantly more efficient on a per core, per dollar, and per joule basis than distributed memory systems, and shared-memory algorithms tend to be simpler than their distributed counterparts.
In this paper, we present a lightweight graph processing framework that is specific for shared-memory parallel/multicore machines, which makes graph traversal algorithms easy to write. The framework has two very simple routines, one for mapping over edges and one for mapping over vertices. Our routines can be applied to any subset of the vertices, which makes the framework useful for many graph traversal algorithms that operate on subsets of the vertices. Based on recent ideas used in a very fast algorithm for breadth-first search (BFS), our routines automatically adapt to the density of vertex sets. We implement several algorithms in this framework, including BFS, graph radii estimation, graph connectivity, betweenness centrality, PageRank and single-source shortest paths. Our algorithms expressed using this framework are very simple and concise, and perform almost as well as highly optimized code. Furthermore, they get good speedups on a 40-core machine and are significantly more efficient than previously reported results using graph frameworks on machines with many more cores.

Smith2013 [209] E. Smith, "Parallel solution of linear programs," PhD thesis, University of Edinburgh, 2013. [Online]. Available: http://hdl.handle.net/1842/8833,
Abstract: The factors limiting the performance of computer software periodically undergo sudden shifts, resulting from technological progress, and these shifts can have profound implications for the design of high performance codes. At the present time, the speed with which hardware can execute a single stream of instructions has reached a plateau. It is now the number of instruction streams that may be executed concurrently which underpins estimates of compute power, and with this change, a critical limitation on the performance of software has come to be the degree to which it can be parallelised. The research in this thesis is concerned with the means by which codes for linear programming may be adapted to this new hardware. For the most part, it is codes implementing the simplex method which will be discussed, though these have typically lower performance for single solves than those implementing interior point methods. However, the ability of the simplex method to rapidly re-solve a problem makes it at present indispensable as a subroutine for mixed integer programming. The long history of the simplex method as a practical technique, with applications in many industries and government, has led to such codes reaching a great level of sophistication. It would be unexpected in a research project such as this one to match the performance of top commercial codes with many years of development behind them. The simplex codes described in this thesis are, however, able to solve real problems of small to moderate size, rather than being confined to random or otherwise artificially generated instances. The remainder of this thesis is structured as follows. The rest of this chapter gives a brief overview of the essential elements of modern parallel hardware and of the linear programming problem. Both the simplex method and interior point methods are discussed, along with some of the key algorithmic enhancements required for such systems to solve real-world problems. Some background on the parallelisation of both types of code is given. The next chapter describes two standard simplex codes designed to exploit the current generation of hardware. i6 is a parallel standard simplex solver capable of being applied to a range of real problems, and showing exceptional performance for dense, square programs. i8 is also a parallel, standard simplex solver, but now implemented for graphics processing units (GPUs).

Tang2013 [210] W. T. Tang, W. J. Tan, R. Ray, Y. W. Wong, W. Chen, S.-h. Kuo, R. S. M. Goh, S. J. Turner, and W.-F. Wong, "Accelerating sparse matrix-vector multiplication on GPUs using bit-representation-optimized schemes," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '13, Denver, CO, USA: ACM, 2013, 26:1–26:12, ISBN: 978-1-4503-2378-9. DOI: 10.1145/2503210.2503234,
Abstract: The sparse matrix-vector (SpMV) multiplication routine is an important building block used in many iterative algorithms for solving scientific and engineering problems. One of the main challenges of SpMV is its memory-boundedness. Although compression has been proposed previously to improve SpMV performance on CPUs, its use has not been demonstrated on the GPU because of the serial nature of many compression and decompression schemes. In this paper, we introduce a family of bit-representation-optimized (BRO) compression schemes for representing sparse matrices on GPUs. The proposed schemes, BRO-ELL, BRO-COO, and BRO-HYB, perform compression on index data and help to speed up SpMV on GPUs through reduction of memory traffic. Furthermore, we formulate a BRO-aware matrix reordering scheme as a data clustering problem and use it to increase compression ratios. With the proposed schemes, experiments show that average speedups of 1.5× compared to ELLPACK and HYB can be achieved for SpMV on GPUs.

Wu2013 [211] G. Wu, W. Xu, Y. Zhang, and Y. Wei, "A preconditioned conjugate gradient algorithm for GeneRank with application to microarray data mining," *Data Mining and Knowledge Discovery*, vol. 26, no. 1, pp. 27–56, 2013, ISSN: 1573-756X. DOI: 10.1007/s10618-011-0245-7,
Abstract: The problem of identifying key genes is of fundamental importance in biology and medicine. The GeneRank model explores connectivity data to produce a prioritization of the genes in a microarray experiment that is less susceptible to variation caused by experimental noise than the one based on expression levels alone. The GeneRank algorithm amounts to solving an unsymmetric linear system. However, when the matrix in question is very large, the GeneRank algorithm is inefficient and even can be infeasible. On the other hand, the adjacency matrix is symmetric in the GeneRank model, while the original GeneRank algorithm fails to exploit the symmetric structure of the problem in question. In this paper, we discover that the GeneRank problem can be rewritten as a symmetric positive definite linear system, and propose a preconditioned conjugate gradient algorithm to solve it. Numerical experiments support our theoretical results, and show superiority of the novel algorithm.

Yoo2013 [212] R. M. Yoo, C. J. Hughes, C. Kim, Y.-K. Chen, and C. Kozyrakis, "Locality-aware task management for unstructured parallelism: A quantitative limit study," in *Proceedings of the 25th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, ser. SPAA '13, Montréal, QB, Canada: ACM, 2013, pp. 315–325, ISBN: 978-1-4503-1572-2. DOI: 10.1145/2486159.2486175,
Abstract: As we increase the number of cores on a processor die, the on-chip cache hierarchies that support these cores are getting larger, deeper, and more complex. As a result, non-uniform memory access effects are now prevalent even on a single chip. To reduce execution time and energy consumption, data access locality should be exploited. This is especially important for task-based programming systems, where a scheduler decides when and where on the chip the code segments, i.e., tasks, should execute. Capturing locality for structured task parallelism has been done effectively, but the more difficult case, unstructured parallelism, remains largely unsolved - little quantitative analysis exists to demonstrate the potential of locality-aware scheduling, and to guide future scheduler implementations in the most fruitful direction.
This paper quantifies the potential of locality-aware scheduling for unstructured parallelism on three different many-core processors. Our simulation results of 32-core systems show that locality-aware scheduling can bring up to 2.39× speedup over a randomized schedule, and 2.05× speedup over a state-of-the-art baseline scheduling scheme. At the same time, a locality-aware schedule reduces average energy consumption by 55% and 47%, relative to the random and the baseline schedule, respectively. In addition, our 1024-core simulation results project that these benefits will only increase: Compared to 32-core executions, we see up to 1.83× additional locality benefits. To

capture such potentials in a practical setting, we also perform a detailed scheduler design space exploration to quantify the impact of different scheduling decisions. We also highlight the importance of locality-aware stealing, and demonstrate that a stealing scheme can exploit significant locality while performing load balancing. Over randomized stealing, our proposed scheme shows up to 2.0× speedup for stolen tasks.

Virtex7 [213] *7 series FPGA overview*, 1.16.1, Product Specification, Dec. 2014. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf.

Amd2014 [214] *AMD firepro w5000*, Online, AMD Corporation, 2014. [Online]. Available: http://www.amd.com/documents/2795_W5000_DataSheet_R3.pdf.

Anderson2014 [215] M. Anderson, "A framework for composing high-performance OpenCL from python descriptions," UCB/EECS-2014-177, PhD thesis, University of California at Berkeley, Nov. 2014. [Online]. Available: http://digitalassets.lib.berkeley.edu/etd/ucb/text/Anderson_berkeley_0028E_14773.pdf,
Abstract: Parallel processors have become ubiquitous; most programmers today have access to parallel hardware such as multi-core processors and graphics processors. This has created an implementation gap, where efficiency programmers with knowledge of hardware details can attain high performance by exploiting parallel hardware, while productivity programmers with application-level knowledge may not understand low-level performance trade-offs. Ideally, we would like to be able to write programs in productivity languages such as Python or MATLAB, and achieve performance comparable to the best hand-tuned code.
One approach toward achieving this ideal is to write libraries that get high efficiency on certain operations, and call these libraries from the productivity environment. We propose a framework that addresses two problems with this approach: that it fails to fuse operations for efficiency, and that it may not consider runtime information such as shapes and sizes of data structures. With our framework, efficiency programmers write and/or generate customized OpenCL snippets at runtime and the framework automatically fuses, compiles, and executes these operations based on a Python description.
We evaluate the framework with case studies of two very different applications: space-time adaptive radar processing and optical flow. For a space-time adaptive radar processing application, our framework's implementation is competitive with a hand-coded implementation that uses a vendor-optimized library. For optical flow, a computer vision application, the framework achieves frame rates that are between 0.5× and 0.97× hand-coded OpenCL performance.

Choo2014 [216] K. Choo, W. Panlener, and B. Jang, "Understanding and optimizing GPU cache memory performance for compute workloads," in *Proceedings of the 13th IEEE International Symposium on Parallel and Distributed Computing*, ser. ISPDC '14, Lymassol, CY: IEEE Computer Society, 2014, pp. 189–196, ISBN: 978-1-4799-5919-8. DOI: 10.1109/ISPDC.2014.29,
Abstract: Processing elements such as CPUs and GPUs depend on cache technology to bridge the classic processor memory subsystem performance gap. As GPUs evolve into general purpose co-processors with CPUs sharing the load, good cache design and use becomes increasingly important. While both CPUs and GPUs must cooperate and perform well, their memory access patterns are very different. On CPUs only a few threads access memory simultaneously. On GPUs, there is significantly higher memory access contention among thousands of threads. Despite such different behavior, there is little research that investigates the behavior and performance of GPU caches in depth. In this paper, we present our extensive study on the characterization and improvement of GPU cache behavior and performance for general-purpose workloads using a cycle-accurate ISA level GPU architectural simulator that models one of the latest GPU architectures, Graphics Core Next (GCN) from AMD. Our study makes the following observations and improvements. First, we observe that L1 vector data cache hit rate is substantially lower when compared to CPU caches. The main culprit is compulsory misses caused by lack of data reuse among massively simultaneous threads. Second, there is significant memory access contention in shared L2 data cache, accounting for up to 19% of total access for some benchmarks. This high contention remains a main performance barrier in L2 data cache even though its hit rate is high. Third, we demonstrate

that memory access coalescing plays a critical role in reducing memory traffic. Finally we found that there exists inter-workgroup locality which can affect the cache behavior and performance. Our experimental results show memory performance can be improved by 1) shared L1 vector data cache where multiple compute units share a single cache to exploit inter-workgroup locality and increase data reusability, and 2) clustered workgroup scheduling where workgroups with consecutive IDs are assigned on the same compute unit.

Cuda65P [217] "CUDA 6.5 performance report," NVIDIA Corporation, Tech. Rep., Sep. 2014, Accessed 20 July 2016. [Online]. Available: http://developer.download.nvidia.com/compute/cuda/6_5/rel/docs/CUDA_6.5_Performance_Report.pdf.

Demmel2014 [218] J. Demmel and H. Diep Nguyen, "Parallel reproducible summation," *IEEE Transactions on Computers*, vol. 64, pp. 1–1, Jan. 2014. DOI: 10.1109/TC.2014.2345391,
Abstract: Reproducibility, i.e. getting bitwise identical floating point results from multiple runs of the same program, is a property that many users depend on either for debugging or correctness checking in many codes [10]. However, the combination of dynamic scheduling of parallel computing resources, and floating point nonassociativity, makes attaining reproducibility a challenge even for simple reduction operations like computing the sum of a vector of numbers in parallel. We propose a technique for floating point summation that is reproducible independent of the order of summation. Our technique uses Rump's algorithm for error-free vector transformation [7], and is much more efficient than using (possibly very) high precision arithmetic. Our algorithm reproducibly computes highly accurate results with an absolute error bound of $ncdot2^{-28} \cdot macheps \cdot max_i|v_i|$ at a cost of $7n$ FLOPs and a small constant amount of extra memory usage. Higher accuracies are also possible by increasing the number of error-free transformations. As long as all operations are performed in to-nearest rounding mode, results computed by the proposed algorithms are reproducible for any run on any platform. In particular, our algorithm requires the minimum number of reductions, i.e. one reduction of an array of six double precision floating point numbers per sum, and hence is well suited for massively parallel environments.

Harshvardhan2014 [219] Harshvardhan, A. Fidel, N. M. Amato, and L. Rauchwerger, "KLA: A new algorithmic paradigm for parallel graph computations," in *Proceedings of the 23rd International Conference on Parallel Architectures and Compilation*, ser. PACT '14, Edmonton, AB, CA: ACM, 2014, pp. 27–38, ISBN: 978-1-4503-2809-8. DOI: 10.1145/2628071.2628091,
Abstract: This paper proposes a new algorithmic paradigm - k-level asynchronous (KLA) - that bridges level-synchronous and asynchronous paradigms for processing graphs. The KLA paradigm enables the level of asynchrony in parallel graph algorithms to be parametrically varied from none (level-synchronous) to full (asynchronous). The motivation is to improve execution times through an appropriate trade-off between the use of fewer, but more expensive global synchronizations, as in level-synchronous algorithms, and more, but less expensive local synchronizations (and perhaps also redundant work), as in asynchronous algorithms. We show how common patterns in graph algorithms can be expressed in the KLA pardigm and provide techniques for determining k, the number of asynchronous steps allowed between global synchronizations. Results of an implementation of KLA in the STAPL Graph Library show excellent scalability on up to 96K cores and improvements of 10× or more over level-synchronous and asynchronous versions for graph algorithms such as breadth-first search, PageRank, k-core decomposition and others on certain classes of real-world graphs.

Hartley2014 [220] E. N. Hartley, J. L. Jerez, A. Suardi, J. M. Macjeowski, E. C. Kerrigan, and G. A. Constantinides, "Predictive control using an FPGA with application to aircraft control," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 3, pp. 1006–1017, May 2014, ISSN: 1063-6536. DOI: 10.1109/TCST.2013.2271791,
Abstract: Alternative and more efficient computational methods can extend the applicability of model predictive control (MPC) to systems with tight real-time requirements. This paper presents a system-on-a-chip MPC system, implemented on a field-programmable gate array (FPGA), consisting of a sparse structure-exploiting primal dual interior point (PDIP) quadratic program (QP) solver for MPC reference tracking and a fast gradient QP solver for steady-state target calculation. A parallel reduced precision iterative solver is used to accelerate the solution of the set of linear

equations forming the computational bottleneck of the PDIP algorithm. A numerical study of the effect of reducing the number of iterations highlights the effectiveness of the approach. The system is demonstrated with an FPGA-in-the-loop testbench controlling a nonlinear simulation of a large airliner. This paper considers many more manipulated inputs than any previous FPGA-based MPC implementation to date, yet the implementation comfortably fits into a midrange FPGA, and the controller compares well in terms of solution quality and latency to state-of-the-art QP solvers running on a standard PC.

IntelXeonE5 [221] *Intel Xeon processor E5-2640 v3*, Online, Accessed 13 October 2016, Intel Corporation, 2014. [Online]. Available: `http://ark.intel.com/products/83359/Intel-Xeon-Processor-E5-2640-v3-20M-Cache-2_60-GHz`.

Jacomy2014 [222] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian, "ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software," *PLoS ONE*, vol. 9, no. 6, Jun. 2014. DOI: `10.1371/journal.pone.0098679`,
Abstract: Gephi is a network visualization software used in various disciplines (social network analysis, biology, genomics). One of its key features is the ability to display the spatialization process, aiming at transforming the network into a map, and ForceAtlas2 is its default layout algorithm. The latter is developed by the Gephi team as an all-around solution to Gephi users' typical networks (scale-free, 10 to 10,000 nodes). We present here for the first time its functioning and settings. ForceAtlas2 is a force-directed layout close to other algorithms used for network spatialization. We do not claim a theoretical advance but an attempt to integrate different techniques such as the Barnes Hut simulation, degree-dependent repulsive force, and local and global adaptive temperatures. It is designed for the Gephi user experience (it is a continuous algorithm), and we explain which constraints it implies. The algorithm benefits from much feedback and is developed in order to provide many possibilities through its settings. We lay out its complete functioning for the users who need a precise understanding of its behaviour, from the formulas to graphic illustration of the result. We propose a benchmark for our compromise between performance and quality. We also explain why we integrated its various features and discuss our design choices.

Jerez2014 [223] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded online optimization for model predictive control at megahertz rates," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3238–3251, Dec. 2014, ISSN: 0018-9286. DOI: `10.1109/TAC.2014.2351991`,
Abstract: Faster, cheaper, and more power efficient optimization solvers than those currently possible using general-purpose techniques are required for extending the use of model predictive control (MPC) to resource-constrained embedded platforms. We propose several custom computational architectures for different first-order optimization methods that can handle linear-quadratic MPC problems with input, input-rate, and soft state constraints. We provide analysis ensuring the reliable operation of the resulting controller under reduced precision fixed-point arithmetic. Implementation of the proposed architectures in FPGAs shows that satisfactory control performance at a sample rate beyond 1 MHz is achievable even on low-end devices, opening up new possibilities for the application of MPC on embedded systems.

Khorasani2014 [224] F. Khorasani, K. Vora, R. Gupta, and L. N. Bhuyan, "CuSha: Vertex-centric graph processing on GPUs," in *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, ser. HPDC '14, Vancouver, BC, CA: ACM, 2014, pp. 239–252, ISBN: 978-1-4503-2749-7. DOI: `10.1145/2600212.2600227`,
Abstract: Vertex-centric graph processing is employed by many popular algorithms (e.g., PageRank) due to its simplicity and efficient use of asynchronous parallelism. The high compute power provided by SIMT architecture presents an opportunity for accelerating these algorithms using GPUs. Prior works of graph processing on a GPU employ Compressed Sparse Row (CSR) form for its space-efficiency; however, CSR suffers from irregular memory accesses and GPU underutilization that limit its performance. In this paper, we present CuSha, a CUDA-based graph processing framework that overcomes the above obstacle via use of two novel graph representations: G-Shards and Concatenated Windows (CW). G-Shards uses a concept recently introduced for non-GPU systems that organizes a graph into autonomous sets of ordered edges called shards.

CuSha's mapping of GPU hardware resources on to shards allows fully coalesced memory accesses. CW is a novel representation that enhances the use of shards to achieve higher GPU utilization for processing sparse graphs. Finally, CuSha fully utilizes the GPU power by processing multiple shards in parallel on GPU's streaming multiprocessors. For ease of programming, CuSha allows the user to define the vertex-centric computation and plug it into its framework for parallel processing of large graphs. Our experiments show that CuSha provides significant speedups over the state-of-the-art CSR-based virtual warp-centric method for processing graphs on GPUs.

Kreutzer2014 [225] M. Kreutzer, G. Hazen, G. Wellein, H. Fehske, and A. R. Bishop, "A unified sparse matrix data format for efficient general sparse matrix-vector multiplication on modern processors with wide SIMD units," *SIAM Journal on Scientific Computing*, vol. 36, no. 5, pp. C401–C423, 2014. DOI: 10.1137/130930352,

Abstract: Sparse matrix-vector multiplication (spMVM) is the most time-consuming kernel in many numerical algorithms and has been studied extensively on all modern processor and accelerator architectures. However, the optimal sparse matrix data storage format is highly hardware-specific, which could become an obstacle when using heterogeneous systems. Also, it is as yet unclear how the wide single instruction multiple data (SIMD) units in current multi- and many-core processors should be used most efficiently if there is no structure in the sparsity pattern of the matrix. We suggest SELL-C-σ, a variant of Sliced ELLPACK, as a SIMD-friendly data format which combines long-standing ideas from general-purpose graphics processing units and vector computer programming. We discuss the advantages of SELL-C-σ compared to established formats like Compressed Row Storage and ELLPACK and show its suitability on a variety of hardware platforms (Intel Sandy Bridge, Intel Xeon Phi, and Nvidia Tesla K20) for a wide range of test matrices from different application areas. Using appropriate performance models we develop deep insight into the data transfer properties of the SELL-C-σ spMVM kernel. SELL-C-σ comes with two tuning parameters whose performance impact across the range of test matrices is studied and for which reasonable choices are proposed. This leads to a hardware-independent ("catch-all") sparse matrix format, which achieves very high efficiency for all test matrices across all hardware platforms.

Lee2014 [226] S.-Y. Lee and C.-J. Wu, "CAWS: Criticality-aware warp scheduling for GPGPU workloads," in *Proceedings of the 23rd International Conference on Parallel Architectures and Compilation*, ser. PACT '14, Edmonton, AB, CA: ACM, 2014, pp. 175–186, ISBN: 978-1-4503-2809-8. DOI: 10.1145/2628071.2628107,

Abstract: The ability to perform fast context-switching and massive multi-threading is the forte of modern GPU architectures, which have emerged as an efficient alternative to traditional chip-multiprocessors for parallel workloads. One of the main benefits of such architecture is its latency-hiding capability. However, the efficacy of GPU's latency-hiding varies significantly across GPGPU applications.

To investigate this, this paper first proposes a new algorithm that profiles execution behavior of GPGPU applications. We characterize latencies caused by various pipeline hazards, memory accesses, synchronization primitives, and the warp scheduler. Our results show that the current round-robin warp scheduler works well in overlapping various latency stalls with the execution of other available warps for only a few GPGPU applications. For other applications, there is an excessive latency stall that cannot be hidden by the scheduler effectively. With the latency characterization insight, we observe a significant execution time disparity for warps within the same thread block, which causes sub-optimal performance, called the warp criticality problem.

To tackle the warp criticality problem, we design a family of criticality-aware warp scheduling (CAWS) policies by scheduling the critical warp(s) more frequently than other warps. Our results on the breadth-first-search, B+tree search, two point angular correlation function, and K-means clustering show that, with oracle knowledge of warp criticality, our best-performing scheduling policy can improve GPGPU applications' performance by 17% on average. With our designed criticality predictor, the various scheduling policies can improve performance by 10-21% on breadth-first-search. To our knowledge, this is the first paper to characterize warp criticality and explore different criticality-aware warp scheduling policies for GPGPU workloads.

Liu2014 [227] J. Liu, H. Peyril, A. Burg, and G. A. Constantinides, "FPGAimplementation of an interior point method for high-speed model predictive control," in *Proceedings of the 24th International Conference on Field Programmable Logic and Applications*, ser. FPL '14, Munich, GER, Sep. 2014, pp. 1–8. DOI: 10.1109/FPL.2014.6927473,
Abstract: In this paper, we present a hardware architecture for implementing an interior point method for model predictive control (MPC) on field programmable gate arrays (FPGA). The FPGA implementation allows the solution of quadratic programs occurring in MPC at very high speed. Experiments show that our hardware implementation is able to outperform an software implementation running on a high-end CPU while consuming significantly less power making it well-suited for embedded industrial control applications. In contrast to existing FPGA implementations, the proposed solution exploits the MPC-specific problem structure with the direct linear equation solver and uses an efficient predictor-corrector algorithm. Moreover, the modular design of the architecture simplifies customization or extension to special control problem classes. The proposed FPGA solution can broaden the applicability of solving complex or large MPC problems in embedded computing platforms that were so far considered out of reach.

Nelson2014 [228] J. Nelson, B. Holt, B. Myers, P. Briggs, L. Ceze, S. Kahan, and M. Oskin, "Grappa: A latency-tolerant runtime for large-scale irregular applications," in *Proceedings of the International Workshop on Rack-Scale Computing*, ser. WRSC '14, Amsterdam, NL, Apr. 2014. [Online]. Available: ftp://trout.cs.washington.edu/tr/2014/02/UW-CSE-14-02-01.PDF,
Abstract: Grappa is a runtime system for commodity clusters of multicore computers that presents a massively parallel, single address space abstraction to applications. Grappa's purpose is to enable scalable performance of irregular parallel applications, such as branch and bound optimization, SPICE circuit simulation, and graph processing. Poor data locality, imbalanced parallel work and complex communication patterns make scaling these applications difficult.
Grappa serves both as a C++ user library and as a foundation for higher level languages. Grappa tolerates delays to remote memory by multiplexing thousands of lightweight workers to each processor core, balances load via fine-grained distributed work-stealing, increases communication throughput by aggregating smaller data requests into large ones, and provides efficient synchronization and remote operations. We present a detailed description of the Grappa system and performance comparisons on several irregular benchmarks to hand-optimized MPI code and to the Cray XMT, a custom system used to target the real-time graph-analytics market. We find Grappa to be 9× faster than MPI on a random access microbenchmark, between 3.5× and 5.4× slower than MPI on applications, and between 2.6× faster and 4.4× slower than the XMT.

NvidiaK80 [229] *NVIDIA tesla GPU accelerators*, Online, Accessed 13 October 2016, NVIDIA corporation, 2014. [Online]. Available: http://international.download.nvidia.com/pdf/kepler/TeslaK80-datasheet.pdf.

Pedram2014 [230] A. Pedram, A. Gerstlauer, and R. A. van de Geijn, "Algorithm, architecture, and floating-point unit codesign of a matrix factorization accelerator," *IEEE Transactions on Computers*, vol. 63, no. 8, 2014, ISSN: 0018-9340. DOI: 10.1109/TC.2014.2315627,
Abstract: This paper examines the mapping of algorithms encountered when solving dense linear systems and linear least-squares problems to a custom Linear Algebra Processor. Specifically, the focus is on Cholesky, LU (with partial pivoting), and QR factorizations and their blocked algorithms. As part of the study, we expose the benefits of redesigning floating point units and their surrounding data-paths to support these complicated operations. We show how adding moderate complexity to the architecture greatly alleviates complexities in the algorithm. We study design tradeoffs and the effectiveness of architectural modifications to demonstrate that we can improve power and performance efficiency to a level that can otherwise only be expected of full-custom ASIC designs. A feasibility study of inner kernels is extended to blocked level and shows that, at block level, the Linear Algebra Core (LAC) can achieve high efficiencies with up to 45 GFLOPS/W for both Cholesky and LU factorization, and over 35 GFLOPS/W for QR factorization. While maintaining such efficiencies, our extensions to the MAC units can achieve up to 10, 12, and 20 percent speedup for the blocked algorithms of Cholesky, LU, and QR factorization, respectively.

Ploskas2014 [231] N. Ploskas and N. Samaras, "GPU accelerated pivoting rules for the simplex algorithm," *Journal of Systems and Software*, vol. 96, pp. 1–9, 2014, ISSN: 0164-1212. DOI: 10.1016/j.jss.2014.04.047. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121214001174,

Abstract: Simplex type algorithms perform successive pivoting operations (or iterations) in order to reach the optimal solution. The choice of the pivot element at each iteration is one of the most critical step in simplex type algorithms. The flexibility of the entering and leaving variable selection allows to develop various pivoting rules. In this paper, we have proposed some of the most well-known pivoting rules for the revised simplex algorithm on a CPU-GPU computing environment. All pivoting rules have been implemented in MATLAB and CUDA. Computational results on randomly generated optimal dense linear programs and on a set of benchmark problems (Netlib-optimal, Kennington, Netlib-infeasible, Maros) are also presented. These results showed that the proposed GPU implementations of the pivoting rules outperform the corresponding CPU implementations.

Rupp2014 [232] K. Rupp. (Mar. 2014). CPU, GPU and MIC hardware characteristics over time. Accessed 21 July 2016, [Online]. Available: https://www.karlrupp.net/2013/06/cpu-gpu-and-mic-hardware-characteristics-over-time/.

Salihoglu2014 [233] S. Salihoglu and J. Widom, "HelP: High-level primitives for large-scale graph processing," in *Proceedings of Workshop on GRAph Data Management Experiences and Systems*, ser. GRADES '14, Snowbird, UT, USA: ACM, 2014, 3:1–3:6, ISBN: 978-1-4503-2982-8. DOI: 10.1145/2621934.2621938,

Abstract: Large-scale graph processing systems typically expose a small set of functions, such as the compute() function of Pregel, or the gather(), apply(), and scatter() functions of PowerGraph. For some computations, these APIs are too low-level, yielding long and complex programs, but with shared coding patterns. Similar issues with the MapReduce framework have led to widely-used languages such as Pig Latin and Hive, which introduce higher-level primitives. We take an analogous approach for graph processing: we propose HelP, a set of high-level primitives that capture commonly appearing operations in large-scale graph computations. Using our primitives we have implemented a large suite of algorithms, some of which we previously implemented with the APIs of existing systems. Our experience has been that implementing algorithms using our primitives is more intuitive and much faster than using the APIs of existing distributed systems. All of our primitives and algorithms are fully implemented as a library on top of the open-source GraphX system.

Spral2014 [234] (2014). The sparse parallel robust algorithms library (SPRAL), Numerical Analysis Group, Rutherford Appleton Laboratory, [Online]. Available: http://www.numerical.rl.ac.uk/spral/.

SDAccel2014 [235] *The Xilinx SDAccel development environment*, 2014. [Online]. Available: http://www.xilinx.com/publications/prod_mktg/sdnet/sdaccel-backgrounder.pdf.

Totoni2014 [236] E. Totoni, M. T. Health, and L. V. Kale, "Structure-adaptive parallel solution of sparse triangular linear systems," *Parallel Computing*, vol. 40, no. 9, pp. 454–470, Oct. 2014. DOI: 10.1016/j.parco.2014.06.006,

Abstract: Solving sparse triangular systems of linear equations is a performance bottleneck in many methods for solving more general sparse systems. Both for direct methods and for many iterative preconditioners, it is used to solve the system or improve an approximate solution, often across many iterations. Solving triangular systems is notoriously resistant to parallelism, however, and existing parallel linear algebra packages appear to be ineffective in exploiting significant parallelism for this problem.

We develop a novel parallel algorithm based on various heuristics that adapt to the structure of the matrix and extract parallelism that is unexploited by conventional methods. By analyzing and reordering operations, our algorithm can often extract parallelism even for cases where most of the nonzero matrix entries are near the diagonal. Our main parallelism strategies are: (1) identify independent rows, (2) send data earlier to achieve greater overlap, and (3) process dense off-

diagonal regions in parallel. We describe the implementation of our algorithm in Charm++ and MPI and present promising experimental results on up to 512 cores of BlueGene/P, using numerous sparse matrices from real applications.

Oracle2014 [237] *X86 assembly language reference manual*, Online, Accessed 4 June 2017, Oracle, 2014. [Online]. Available: http://docs.oracle.com/cd/E36784_01/html/E36859/docinfo.html#scrolltoc.

Zhong2014 [238] J. Zhong and B. He, "Medusa: Simplified graph processing on gpus," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 6, pp. 1543–1552, Jun. 2014, ISSN: 1045-9219. DOI: 10.1109/TPDS.2013.111,
Abstract: Graphs are common data structures for many applications, and efficient graph processing is a must for application performance. Recently, the graphics processing unit (GPU) has been adopted to accelerate various graph processing algorithms such as BFS and shortest paths. However, it is difficult to write correct and efficient GPU programs and even more difficult for graph processing due to the irregularities of graph structures. To simplify graph processing on GPUs, we propose a programming framework called Medusa which enables developers to leverage the capabilities of GPUs by writing sequential C/C++ code. Medusa offers a small set of user-defined APIs and embraces a runtime system to automatically execute those APIs in parallel on the GPU. We develop a series of graph-centric optimizations based on the architecture features of GPUs for efficiency. Additionally, Medusa is extended to execute on multiple GPUs within a machine. Our experiments show that 1) Medusa greatly simplifies implementation of GPGPU programs for graph processing, with many fewer lines of source code written by developers and 2) the optimization techniques significantly improve the performance of the runtime system, making its performance comparable with or better than manually tuned GPU graph operations.

Alvarez2015 [239] A. M. Alvarez, L. Wehenkel, and Q. Louveaux, "Machine learning to balance the load in parallel branch-and-bound," Department of Electrical Engineering and Computer Science, University of Liege, Tech. Rep., Mar. 2015. [Online]. Available: http://www.optimization-online.org/DB_FILE/2015/03/4832.pdf,
Abstract: We describe in this paper a new approach to parallelize branch-and-bound on a certain number of processors. We propose to split the optimization of the original problem into the optimization of several subproblems that can be optimized separately with the goal that the amount of work that each processor carries out is balanced between the processors, while achieving interesting speedups. The main innovation of our approach consists in the use of machine learning to create a function able to estimate the difficulty (number of nodes) of a subproblem of the original problem. We also present a set of features that we developed in order to characterize the encountered subproblems. These features are used as input of the function learned with machine learning in order to estimate the difficulty of a subproblem. The estimates of the numbers of nodes are then used to decide how to partition the original optimization tree into a given number of subproblems, and to decide how to distribute them among the available processors. The experiments that we carry out show that our approach succeeds in balancing the amount of work between the processors, and that interesting speedups can be achieved with little effort.

Bouwmeester2015 [240] H. Bouwmeester, B. Dougherty, and A. Knyazev, "Nonsymmetric preconditioning for conjugate gradient and steepest descent methods," *Procedia Computer Science*, vol. 51, pp. 276–285, 2015, ISSN: 1877-0509. DOI: 10.1016/j.procs.2015.05.241,
Abstract: We analyze a possibility of turning off post-smoothing (relaxation) in geometric multigrid when used as a preconditioner in preconditioned conjugate gradient (PCG) linear and eigenvalue solvers for the 3D Laplacian. The geometric Semicoarsening Multigrid (SMG) method is provided by the hypre parallel software package. We solve linear systems using two variants (standard and flexible) of PCG and preconditioned steepest descent (PSD) methods. The eigen-value problems are solved using the locally optimal block preconditioned conjugate gradient (LOBPCG) method available in hypre through BLOPEX software. We observe that turning off the post-smoothing in SMG dramatically slows down the standard PCG-SMG. For flexible PCG and LOBPCG, our numerical tests show that removing the post-smoothing results in overall 40–50 percent acceleration, due to the high costs of smoothing and relatively insignificant decrease in convergence speed. We

demonstrate that PSD-SMG and flexible PCG-SMG converge similarly if SMG post-smoothing is off. A theoretical justification is provided.

Candel2015 [241] F. Candel, S. Petit, J. Sahuquillo, and J. Duato, "Accurately modeling the GPU memory subsystem," in *Proceedings of the 2015 International Conference on High Performance Computing Simulation*, ser. HPCS '15, Amsterdam, NL, Jul. 2015, pp. 179–186. DOI: 10.1109/HPCSim.2015.7237038,

Abstract: Nowadays, research on GPU processor architecture is extraordinarily active since these architectures offer much more performance per watt than CPU architectures. This is the main reason why massive deployment of GPU multiprocessors is considered one of the most feasible solutions to attain exascale computing capabilities. In this context, ongoing GPU architecture research is required to improve GPU programmability as well as to integrate CPU and GPU cores in the same die. One of the most important research topics in current GPUs, is the GPU memory hierarchy, since its design goals are very different from those of conventional CPU memory hierarchies. To explore novel designs to better support General Purpose computing in GPUs (GPGPU computing) as well as to improve the performance of GPU and CPU/GPU systems, researchers often require advanced microarchitectural simulators with detailed models of the memory subsystem. Nevertheless, due to fast speed at which current GPU architectures evolve, simulation accuracy of existing state-of-the-art simulators suffers. This paper focuses on accurately modeling the GPU memory subsystem. We identified three main aspects that should be modeled with more accuracy: i) miss status holding registers, ii) coalescing vector memory requests, and iii) non-blocking GPU stores. In this sense, we extend the Multi2Sim heterogeneous CPU/GPU processor simulator to model these aspects with enough accuracy. Experimental results show that if these aspects are not considered in the simulation framework, performance deviations can rise in some applications up to 70%, 75%, and 60%, respectively.

Cublas2015 [242] *CUBLAS library user guide*, 7.5, NVIDIA, Oct. 2015. [Online]. Available: http://docs.nvidia.com/cuda/pdf/CUBLAS_Library.pdf.

Cuda70P [243] "CUDA 7.0 performance report," NVIDIA Corporation, Tech. Rep., May 2015, Accessed 20 July 2016. [Online]. Available: http://developer.download.nvidia.com/compute/cuda/compute-docs/cuda-performance-report.pdf.

aProgramming2015 [244] *CUDA c programming guide*, 7.5, NVIDIA, Oct. 2015. [Online]. Available: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#axzz4mitW5BwQ.

Cusparse2015 [245] *CUSPARSE library*, 7.0, Accessed 1 September 2015., NVIDIA corporation, Mar. 2015. [Online]. Available: https://developer.nvidia.com/cuda-toolkit-70.

Daga2015 [246] M. Daga and J. L. Greathouse, "Structural agnostic SpMV: Adapting CSR-adaptive for irregular matrices," in *Proceedings of the 22nd IEEE International Conference on High Performance Computing*, ser. HiPC '15, Bengaluru, IN, Dec. 2015, pp. 64–74. DOI: 10.1109/HiPC.2015.55,

Abstract: Sparse matrix vector multiplication (SpMV) is an important linear algebra primitive. Recent research has focused on improving the performance of SpMV on GPUs when using compressed sparse row (CSR), the most frequently used matrix storage format on CPUs. Efficient CSR-based SpMV obviates the need for other GPU-specific storage formats, thereby saving runtime and storage overheads. However, existing CSR-based SpMV algorithms on GPUs perform poorly on irregular sparse matrices, limiting their usefulness. We propose a novel approach for SpMV on GPUs which works well for both regular and irregular matrices while keeping the CSR format intact. We start with CSR-Adaptive, which dynamically chooses between two SpMV algorithms depending on the length of each row. We then add a series of performance improvements, such as a more efficient reduction technique. Finally, we add a third algorithm which uses multiple parallel execution units when operating on irregular matrices with very long rows. Our implementation dynamically assigns the best algorithm to sets of rows in order to ensure that the GPU is efficiently utilized. We effectively double the performance of CSR-Adaptive, which had previously demonstrated better performance than algorithms that use other storage formats. In addition, our implementation is 36% faster than CSR5, the current state of the art for SpMV on GPUs.

Dieguez2015 [247] A. P. Diéguez, M. Amor, and R. Doallo, "New tridiagonal systems solvers on GPU architectures," in *Proceedings of the 22nd IEEE International Conference on High Performance Computing*, ser. HiPC '15, Bengaluru, IN, Dec. 2015, pp. 85–94. DOI: 10.1109/HiPC.2015.17,
Abstract: Modern GPUs (Graphics Processing Units) offer very high computing power at relatively low cost. Nevertheless, designing efficient algorithms for the GPUs usually requires additional time and effort, even for experienced programmers. On the other hand, tridiagonal systems solvers are an important building block for a wide range of applications. In this paper, we present a new tuning parallel proposal in order to generate new tridiagonal systems solvers. This proposal is based on the combination of a new reduction algorithm (Redundant Reduction-RR) with a tuning proposal to generate efficient parallel prefix algorithms on the GPU. Specifically, we present two new solvers combining RR with two GPU efficient parallel prefix patterns. The performance of the resulting proposals was analyzed using three different CUDA GPUs, obtaining an improvement of up to 20.5× over the CUSPARSE library and 28.9× over CUDPP.

Eaton2015 [248] J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring, *GNU octave version 4.0.0 manual: A high-level interactive language for numerical computations*, Accessed 20 July 2016, 2015. [Online]. Available: http://www.gnu.org/software/octave/doc/interpreter.

Ehrenberg2015 [249] N. Ehrenberg, "Heading for urban energy internets," *Pictures of the Future*, Jun. 2015, Accessed online on September 28th, 2015. [Online]. Available: http://www.siemens.com/innovation/en/home/pictures-of-the-future/infrastructure-and-finance/smart-cities-smart-buildings.html.

Feldman2015 [250] M. Feldman and A. Snell, "Accelerated computing: A tipping point for HPC," Intersect360 Research, Tech. Rep., Nov. 2015, Accessed 21 July 2016. [Online]. Available: http://images.nvidia.com/content/pdf/tesla/accelerated-computing-at-a-tipping-point.pdf.

Frick2015 [251] D. Frick, A. Domahidi, and M. Morari, "Embedded optimization for mixed logical dynamical systems," *Computers and Chemical Engineering*, vol. 72, no. 0, pp. 21–33, Jan. 2015, A Tribute to Ignacio E. Grossmann, ISSN: 0098-1354. DOI: http://dx.doi.org/10.1016/j.compchemeng.2014.06.005,
Abstract: Predictive control of hybrid systems is currently considered prohibitive using embedded computing platforms. To overcome this limitation for mixed logical dynamical systems of small to medium size, we propose to use (1) a standard branch-and-bound approach combined with a fast embedded interior point solver, (2) pre-processing heuristics, run once and offline, to significantly reduce the number of subproblems to be solved, and (3) relaxations of the original MPC problem that allow a trade off between computation time and closed-loop performance. A problem-specific ANSI C implementation of the proposed method can be automatically generated, and has a fixed memory footprint and a code size that is insignificantly larger than that of the subproblem solver. Two extensive numerical studies are presented, where problems with up to 60 binary variables are solved in less than 0.2 s with a performance deterioration of below 2% when compared to an optimal MPC scheme.

Gamrath2015 [252] G. Gamrath, T. Koch, A. Martin, M. Miltenberger, and D. Weninger, "Progress in presolving for mixed integer programming," *Mathematical Programming Computation*, vol. 7, no. 4, pp. 367–398, 2015, ISSN: 1867–2957. DOI: 10.1007/s12532-015-0083-5,
Abstract: This paper describes three presolving techniques for solving mixed integer programming problems (MIPs) that were implemented in the academic MIP solver SCIP. The task of presolving is to reduce the problem size and strengthen the formulation, mainly by eliminating redundant information and exploiting problem structures. The first method fixes continuous singleton columns and extends results known from duality fixing. The second analyzes and exploits pairwise dominance relations between variables, whereas the third detects isolated subproblems and solves them independently. The performance of the presented techniques is demonstrated on two MIP test sets. One contains all benchmark instances from the last three MIPLIB versions, while the other consists of real-world supply chain management problems. The computational results show that

the combination of all three presolving techniques almost halves the solving time for the considered supply chain management problems. For the MIPLIB instances we obtain a speedup of 20% on affected instances while not degrading the performance on the remaining problems.

Huangfu2015 [253] Q. Huangfu and J. Hall, "Novel update techniques for the revised simplex method," *Computational Optimization and Applications*, vol. 60, no. 3, pp. 587–608, 2015. DOI: 10.1007/s10589-014-9689-1,
Abstract: This paper introduces three novel techniques for updating the invertible representation of the basis matrix when solving practical sparse linear programming problems using a high performance implementation of the dual revised simplex method, being of particular value when suboptimization is used. Two are variants of the product form update and the other permits multiple Forrest-Tomlin updates to be performed. Computational results show that one of the product form variants is significantly more efficient than the traditional approach, with its performance approaching that of the Forrest-Tomlin update for some problems. The other is less efficient, but valuable in the context of the dual revised simplex method with suboptimization. Results show that the multiple Forrest-Tomlin updates are performed with no loss of serial efficiency.

Kabir2015 [254] K. Kabir, A. Haidar, S. Tomov, and J. Dongarra, "Performance analysis and optimisation of two-sided fractorization algorithms for heterogeneous platform," *Procedia Computer Science*, vol. 51, pp. 180–190, 2015. DOI: 10.1016/j.procs.2015.05.222,
Abstract: Many applications, ranging from big data analytics to nanostructure designs, require the solution of large dense singular value decomposition (SVD) or eigenvalue problems. A first step in the solution methodology for these problems is the reduction of the matrix at hand to condensed form by two-sided orthogonal transformations. This step is standardly used to significantly accelerate the solution process. We present a performance analysis of the main two-sided factorizations used in these reductions: the bidiagonalization, tridiagonalization, and the upper Hessenberg factorizations on heterogeneous systems of multicore CPUs and Xeon Phi coprocessors. We derive a performance model and use it to guide the analysis and to evaluate performance. We develop optimized implementations for these methods that get up to 80% of the optimal performance bounds. Finally, we describe the heterogeneous multicore and coprocessor development considerations and the techniques that enable us to achieve these high-performance results. The work here presents the first highly optimized implementation of these main factorizations for Xeon Phi coprocessors. Compared to the LAPACK versions optmized by Intel for Xeon Phi (in MKL), we achieve up to 50% speedup.

Kerrigan2015 [255] E. C. Kerrigan, "Feedback and time are important for the optimal control of computing systems," in *Proceedings of the 5th IFAC Conference on Nonlinear Model Predictive Control*, ser. NMPC '15, Seville, ES, Sep. 2015. DOI: 10.1016/j.ifacol.2015.11.309,
Abstract: The performance, reliability, cost, size and energy usage of computing systems can be improved by one or more orders of magnitude by the systematic use of modern control and optimization methods. Computing systems rely on the use of feedback algorithms to schedule tasks, data and resources, but the models that are used to design these algorithms are validated using open-loop metrics. By using closed-loop metrics instead, such as the gap metric developed in the control community, it should be possible to develop improved scheduling algorithms and computing systems that have not been over-engineered. Furthermore, scheduling problems are most naturally formulated as constraint satisfaction or mathematical optimization problems, but these are seldom implemented using state of the art numerical methods, nor do they explicitly take into account the fact that the scheduling problem itself takes time to solve. This paper makes the case that recent results in real-time model predictive control, where optimization problems are solved in order to control a process that evolves in time, are likely to form the basis of scheduling algorithms of the future. We therefore outline some of the research problems and opportunities that could arise by explicitly considering feedback and time when designing optimal scheduling algorithms for computing systems.

LiC2015 [256] C. Li, S. L. Song, H. Dai, A. Sidelnik, S. K. S. Hari, and H. Zhou, "Locality-driven dynamic GPU cache bypassing," in *Proceedings of the 29th ACM International Conference*

on Supercomputing, ser. ICS '15, Newport Beach, CA, USA: ACM, 2015, pp. 67–77, ISBN: 978-1-4503-3559-1. DOI: 10.1145/2751205.2751237,
Abstract: This paper presents novel cache optimizations for massively parallel, throughput-oriented architectures like GPUs. L1 data caches (L1 D-caches) are critical resources for providing high-bandwidth and low-latency data accesses. However, the high number of simultaneous requests from single-instruction multiple-thread (SIMT) cores makes the limited capacity of L1 D-caches a performance and energy bottleneck, especially for memory-intensive applications. We observe that the memory access streams to L1 D-caches for many applications contain a significant amount of requests with low reuse, which greatly reduce the cache efficacy. Existing GPU cache management schemes are either based on conditional/reactive solutions or hit-rate based designs specifically developed for CPU last level caches, which can limit overall performance.
To overcome these challenges, we propose an efficient locality monitoring mechanism to dynamically filter the access stream on cache insertion such that only the data with high reuse and short reuse distances are stored in the L1 D-cache. Specifically, we present a design that integrates locality filtering based on reuse characteristics of GPU workloads into the decoupled tag store of the existing L1 D-cache through simple and cost-effective hardware extensions. Results show that our proposed design can dramatically reduce cache contention and achieve up to 56.8% and an average of 30.3% performance improvement over the baseline architecture, for a range of highly-optimized cache-unfriendly applications with minor area overhead and better energy efficiency. Our design also significantly outperforms the state-of-the-art CPU and GPU bypassing schemes (especially for irregular applications), without generating extra L2 and DRAM level contention.

LiD2015 [257] D. Li, H. Wu, and M. Becchi, "Nested parallelism on GPU: Exploring parallelization templates for irregular loops and recursive computations," in *Proceedings of the 44th International Conference on Parallel Processing*, ser. ICPP '15, Beijing, CN, Sep. 2015, pp. 979–988. DOI: 10.1109/ICPP.2015.107,
Abstract: The effective deployment of applications exhibiting irregular nested parallelism on GPUs is still an open problem. A naive mapping of irregular code onto the GPU hardware often leads to resource underutilization and, thereby, limited performance. In this work, we focus on two computational patterns exhibiting nested parallelism: irregular nested loops and parallel recursive computations. In particular, we focus on recursive algorithms operating on trees and graphs. We propose different parallelization templates aimed to increase the GPU utilization of these codes. Specifically, we investigate mechanisms to effectively distribute irregular work to streaming multiprocessors and GPU cores. Some of our parallelization templates rely on dynamic parallelism, a feature recently introduced by Nvidia in their Kepler GPUs and announced as part of the Open CL 2.0 standard. We propose mechanisms to maximize the work performed by nested kernels and minimize the overhead due to their invocation. Our results show that the use of our parallelization templates on applications with irregular nested loops can lead to a 2-6× speedup over baseline GPU codes that do not include load balancing mechanisms. The use of nested parallelism-based parallelization templates on recursive tree traversal algorithms can lead to substantial speedups (up to 15-24×) over optimized CPU implementations. However, the benefits of nested parallelism are still unclear in the presence of recursive applications operating on graphs, especially when recursive code variants require expensive synchronization. In these cases, a flat parallelization of iterative versions of the considered algorithms may be preferable.

Liu2015 [258] W. Liu and B. Vinter, "CSR5: An efficient storage format for cross-platform sparse matrix-vector multiplication," in *Proceedings of the 29th ACM International Conference on Supercomputing*, ser. ICS '15, Newport Beach, CA, USA: ACM, 2015, pp. 339–350, ISBN: 978-1-4503-3559-1. DOI: 10.1145/2751205.2751209,
Abstract: Sparse matrix-vector multiplication (SpMV) is a fundamental building block for numerous applications. In this paper, we propose CSR5 (Compressed Sparse Row 5), a new storage format, which offers high-throughput SpMV on various platforms including CPUs, GPUs and Xeon Phi. First, the CSR5 format is insensitive to the sparsity structure of the input matrix. Thus the single format can support an SpMV algorithm that is efficient both for regular matrices and for irregular matrices. Furthermore, we show that the overhead of the format conversion from the CSR

to the CSR5 can be as low as the cost of a few SpMV operations.
We compare the CSR5-based SpMV algorithm with 11 state-of-the-art formats and algorithms on four mainstream processors using 14 regular and 10 irregular matrices as a benchmark suite. For the 14 regular matrices in the suite, we achieve comparable or better performance over the previous work. For the 10 irregular matrices, the CSR5 obtains average performance improvement of 17.6%, 28.5%, 173.0% and 293.3% (up to 213.3%, 153.6%, 405.1% and 943.3%) over the best existing work on dual-socket Intel CPUs, an nVidia GPU, an AMD GPU and an Intel Xeon Phi, respectively. For real-world applications such as a solver with only tens of iterations, the CSR5 format can be more practical because of its low-overhead for format conversion.

Mujtaba2015 [259] H. Mujtaba. (Nov. 2015). NVIDIA pascal GPU's double precision performance rated at over 4 TFLOPs, 16nm FinFET architecture confirmed – volta GPU peaks at over 7 TFLOPs, 1.2 TB/s HBM2. Accessed 20 July 2016, [Online]. Available: http://wccftech.com/nvidia-pascal-volta-gpus-sc15/.

Munz2015 [260] U. Münz, M. Metzger, A. Szabo, M. Reischböck, F. Steinke, P. Wolfrum, R. Sollacher, D. Obradovic, M. Buhl, T. Lehmann, M. Duckheim, and S. Langemeyer, "Overview of recent control technologies for future power systems: An industry perspective," *Automatisierungstechnik*, pp. 869–882, 11 Nov. 2015. DOI: 10.1515/auto-2015-0047.

Nakamura2015 [261] T. Nakamura and T. Nodera, "The flexible ILU preconditioning for solving large nonsymmetric linear systems of equations," in *Proceedings of the International Workshop on Eigenvalue Problems: Algorithms, Software and Applications in Petascale Computing*, T. Sakurai, S.-L. Zhang, T. Imamura, Y. Yamamoto, Y. Kuramashi, and T. Hoshi, Eds., ser. EPASA 2015, Springer International Publishing, 2015, pp. 51–61, ISBN: 978-3-319-62426-6,
Abstract: The ILU factorization is one of the most popular preconditioners for the Krylov subspace method, alongside the GMRES. Properties of the preconditioner derived from the ILU factorization are relayed onto the dropping rules. Recently, Zhang et al. (Numer Linear Algebra Appl 19:555–569, 2011) proposed a Flexible incomplete Cholesky (IC) factorization for symmetric linear systems. This paper is a study of the extension of the IC factorization to the nonsymmetric case. The new algorithm is called the Crout version of the flexible ILU factorization, and attempts to reduce the number of nonzero elements in the preconditioner and computation time during the GMRES iterations. Numerical results show that our approach is effective and useful.

Ploskas2015 [262] N. Ploskas and N. Samaras, "Efficient GPU-based implementations of simplex type algorithms," *Applied Mathematics and Computation*, vol. 250, pp. 552–570, 2015, ISSN: 0096-3003. DOI: 10.1016/j.amc.2014.10.096,
Abstract: Recent hardware advances have made it possible to solve large scale Linear Programming problems in a short amount of time. Graphical Processing Units (GPUs) have gained a lot of popularity and have been applied to linear programming algorithms. In this paper, we propose two efficient GPU-based implementations of the Revised Simplex Algorithm and a Primal-Dual Exterior Point Simplex Algorithm. Both parallel algorithms have been implemented in MATLAB using MATLAB's Parallel Computing Toolbox. Computational results on randomly generated optimal sparse and dense linear programming problems and on a set of benchmark problems (netlib, kennington, Maros) are also presented. The results show that the proposed GPU implementations outperform MATLAB's interior point method.

Poulson2015 [263] J. Poulson, *Elemental manual*, version 0.84, 2015. [Online]. Available: http://libelemental.org/documentation/elem-0.85.pdf.

Nvprof2015 [264] *Profiler user's guide*, 7.0, Accessed 12 July 2017., NVIDIA corporation, Mar. 2015. [Online]. Available: http://docs.nvidia.com/cuda/profiler-users-guide/index.html#axzz4mitW5BwQ.

Prountzos2015 [265] D. Prountzos, R. Manevich, and K. Pingali, "Synthesizing parallel graph programs via automated planning," in *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '15, Portland, OR, USA: ACM, 2015, pp. 533–544, ISBN: 978-1-4503-3468-6. DOI: 10.1145/2737924.2737953,

Abstract: We describe a system that uses automated planning to synthesize correct and efficient parallel graph programs from high-level algorithmic specifications. Automated planning allows us to use constraints to declaratively encode program transformations such as scheduling, implementation selection, and insertion of synchronization. Each plan emitted by the planner satisfies all constraints simultaneously, and corresponds to a composition of these transformations. In this way, we obtain an integrated compilation approach for a very challenging problem domain. We have used this system to synthesize parallel programs for four graph problems: triangle counting, maximal independent set computation, preflow-push maxflow, and connected components. Experiments on a variety of inputs show that the synthesized implementations perform competitively with hand-written, highly-tuned code.

Rafique2015 [266] A. Rafique, G. A. Constantinides, and N. Kapre, "Communication optimization of iterative sparse matrix-vector multiply on GPUs and FPGAs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 1, pp. 24–34, Jan. 2015, ISSN: 1045-9219. DOI: 10.1109/TPDS.2014.6,
Abstract: Trading communication with redundant computation can increase the silicon efficiency of FPGAs and GPUs in accelerating communication-bound sparse iterative solvers. While k iterations of the iterative solver can be unrolled to provide O(k) reduction in communication cost, the extent of this unrolling depends on the underlying architecture, its memory model, and the growth in redundant computation. This paper presents a systematic procedure to select this algorithmic parameter k, which provides communication-computation tradeoff on hardware accelerators like FPGA and GPU. We provide predictive models to understand this tradeoff and show how careful selection of k can lead to performance improvement that otherwise demands significant increase in memory bandwidth. On an Nvidia C2050 GPU, we demonstrate a 1.9×-42.6× speedup over standard iterative solvers for a range of benchmarks and that this speedup is limited by the growth in redundant computation. In contrast, for FPGAs, we present an architecture-aware algorithm that limits off-chip communication but allows communication between the processing cores. This reduces redundant computation and allows large k and hence higher speedups. Our approach for FPGA provides a 0.3×-4.4× speedup over same-generation GPU devices where k is picked carefully for both architectures for a range of benchmark.

Intel2015 [267] "The compute architecture of intel processor graphics Gen9," Intel Corporation, Tech. Rep., 2015. [Online]. Available: https://software.intel.com/sites/default/files/managed/c5/9a/The-Compute-Architecture-of-Intel-Processor-Graphics-Gen9-v1d0.pdf.

Vielma2015 [268] J. P. Vielma, "Mixed integer linear programming formulation techniques," *SIAM Review*, vol. 57, no. 1, pp. 3–57, Feb. 2015. DOI: 10.1137/130915303,
Abstract: A wide range of problems can be modeled as Mixed Integer Linear Programming (MIP) problems using standard formulation techniques. However, in some cases the resulting MIP can be either too weak or too large to be effectively solved by state of the art solvers. In this survey we review advanced MIP formulation techniques that result in stronger and/or smaller formulations for a wide class of problems.

Vilches2015 [269] A. Vilches, R. Asenjo, A. navarro, F. Corbera, R. Gran, and M. Garzarán, "Adaptive partitioning for irregular applications on heterogeneous CPU-GPU chips," *Procedia Computer Science*, vol. 51, pp. 140–149, 2015. DOI: 10.1016/j.procs.2015.05.213,
Abstract: Commodity processors are comprised of several CPU cores and one integrated GPU. To fully exploit this type of architectures, one needs to automatically determine how to partition the workload between both devices. This is specially challenging for irregular workloads, where each iteration's work is data dependent and shows control and memory divergence. In this paper, we present a novel adaptive partitioning strategy specially designed for irregular applications running on heterogeneous CPU-GPU chips. The main novelty of this work is that the size of the workload assigned to the GPU and CPU adapts dynamically to maximize the GPU and CPU utilization while balancing the workload among the devices. Our experimental results on an Intel Haswell architecture using a set of irregular benchmarks show that our approach outperforms exhaustive static and adaptive state-of-the-art approaches in terms of performance and energy consumption.

Zhao2015 [270] C. Zhao, S. Dong, F. Li, and Y. Song, "Optimal home energy management system with mixed types of loads," *CSEE Journal of Power and Energy Systems*, PP, vol. 1, no. 4, pp. 29–37, Dec. 2015. DOI: 10.17775/CSEEJPES.2015.00045,
Abstract: This paper presents a novel home area energy management system (HEMS) for smart homes with different load profiles installed with photovoltaic generation, energy storage, and DC demand. The developed HEMS is shown to optimize the utilization of local renewables while minimizing energy waste between AC and DC conversions and between storage charging and discharging. Previous studies on HEMS have not considered the impact of load types. In this study, the performance of the proposed HEMS is demonstrated on different smart homes with and without electric heating. A comparative study is carried out to investigate battery behavior, characteristics of AC and DC conversion, and benefits that customers receive. A sensitivity analysis is also conducted to discuss the effects from varied energy storage capacities, AC/DC conversion efficiencies, and PV output. The results show that cost reduction in energy bills can be greatly impacted by load profiles, and customers with electric heating load coupled with sufficiently large energy storage would receive the most reduction in their energy bills.

Azad2016 [271] 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," *SIAM Journal on Scientific Computing*, vol. 38, no. 6, pp. C624–C651, 2016. DOI: 10.1137/15M104253X,
Abstract: Sparse matrix-matrix multiplication (or SpGEMM) is a key primitive for many high-performance graph algorithms as well as for some linear solvers, such as algebraic multigrid. The scaling of existing parallel implementations of SpGEMM is heavily bound by communication. Even though 3D (or 2.5D) algorithms have been proposed and theoretically analyzed in the flat MPI model on Erdős–Rényi matrices, those algorithms had not been implemented in practice and their complexities had not been analyzed for the general case. In this work, we present the first implementation of the 3D SpGEMM formulation that exploits multiple (intranode and internode) levels of parallelism, achieving significant speedups over the state-of-the-art publicly available codes at all levels of concurrencies. We extensively evaluate our implementation and identify bottlenecks that should be subject to further research.

Brandes2016 [272] T. Brandes, E. Schricker, and T. Soddemann, "The LAMA approach for writing portable applications on heterogenous architectures," Fraunhofer Institute for Algorithms and Scientific Computing, Sankt Augustin, Germany, Tech. Rep., 2016. [Online]. Available: http://www.libama.org/assets/lamawhitepaper.pdf,
Abstract: Ensuring longevity and maintainability of modern software applications is mandatory for a proper return on investment. Since the hardware landscape is changing rapidly and will continue to do so, it is imperative to take on those topics also in the HPC domain where applications traditionally have a long live-span. For recent years, we have observed a trend towards more and more heterogeneous systems. Realizing the performance promises of the hardware vendors is a huge challenge to the software developer. Portability is the second challenge to be met in this context. In this paper we present our library LAMA. We created this library to address both challenges successfully in the realm of linear algebra and numerical mathematics. We introduce our solutions to heterogeneous memory and kernel management as well as our solutions to task parallelism. In the end we do performance and scalability benchmarks drawing a comparison to PETSc for the example of a CG solver.

Chen2016 [273] Z. Chen, H. Liu, and B. Yang. (2016). Parallel triangular solvers on GPU. eprint: arXiv:1606.00541,
Abstract: In this paper, we investigate GPU based parallel triangular solvers systematically. The parallel triangular solvers are fundamental to incomplete LU factorization family preconditioners and algebraic multigrid solvers. We develop a new matrix format suitable for GPU devices. Parallel lower triangular solvers and upper triangular solvers are developed for this new data structure. With these solvers, ILU preconditioners and domain decomposition preconditioners are developed. Numerical results show that we can speed triangular solvers around seven times faster.

Chen2016a [274] ——, "Parallel triangular solvers on GPU," The Computing Research Repository, Tech. Rep., 2016. [Online]. Available: http://arxiv.org/abs/1606.00541,
Abstract: In this paper, we investigate GPU based parallel triangular solvers systematically. The parallel triangular solvers are fundamental to incomplete LU factorization family preconditioners and algebraic multigrid solvers. We develop a new matrix format suitable for GPU devices. Parallel lower triangular solvers and upper triangular solvers are developed for this new data structure. With these solvers, ILU preconditioners and domain decomposition preconditioners are developed. Numerical results show that we can speed triangular solvers around seven times faster.

Clsparse2016 [275] (2016). clSPARSE documentation. version 0.10.0.0. Accessed on 10 May 2017, [Online]. Available: http://clmathlibraries.github.io/clSPARSE/.

Galassi2016 [276] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, F. Rossi, and R. Ulerich, *GNU scientific library*, 2.3 for GSL vesion 2.3, 2016. [Online]. Available: https://www.gnu.org/software/gsl/manual/gsl-ref.pdf.

Ghose2016 [277] A. Ghose, S. Dey, P. Mitra, and M. Chaudhuri, "Divergence aware automated partitioning of OpenCL workloads," in *Proceedings of the 9th India Software Engineering Conference*, ser. ISEC '16, Goa, India: ACM, 2016, pp. 131–135, ISBN: 978-1-4503-4018-2. DOI: 10.1145/2856636.2856639,
Abstract: Heterogeneous partitioning is a key step for efficient mapping and scheduling of data parallel applications on multi-core computing platforms involving both CPUs and GPUs. Over the last few years, several automated partitioning methodologies, both static as well as dynamic, have been proposed for this purpose. The present work provides an in-depth analysis of control flow divergence and its impact on the quality of such program partitions. We characterize the amount of divergence in a program as an important performance feature and train suitable Machine Learning (ML) based classifiers which statically decide the partitioning of an OpenCL workload for a heterogeneous platform involving a single CPU and a single GPU. Our approach reports improved partitioning results with respect to timing performance when compared with existing approaches for ML based static partitioning of data parallel workloads.

Hogg2016 [278] J. D. Hogg, E. Ovtchinnikov, and J. A. Scott, "A sparse symmetric indefinite direct solver for GPU architectures," *ACM Transactions on Mathematical Software*, vol. 42, no. 1, 1:1–1:25, Jan. 2016, ISSN: 0098-3500. DOI: 10.1145/2756548,
Abstract: In recent years, there has been considerable interest in the potential for graphics processing units (GPUs) to speed up the performance of sparse direct linear solvers. Efforts have focused on symmetric positive-definite systems for which no pivoting is required, while little progress has been reported for the much harder indefinite case. We address this challenge by designing and developing a sparse symmetric indefinite solver SSIDS. This new library-quality LDL^T factorization is designed for use on GPU architectures and incorporates threshold partial pivoting within a multifrontal approach. Both the factorize and the solve phases are performed using the GPU. Another important feature is that the solver produces bit-compatible results. Numerical results for indefinite problems arising from a range of practical applications demonstrate that, for large problems, SSIDS achieves performance improvements of up to a factor of 4.6× compared with a state-of-the-art multifrontal solver on a multicore CPU.

Kaleem2016 [279] R. Kaleem, A. Venkat, S. Pai, M. Hall, and K. Pingali, "Synchronization trade-offs in GPU implementations of graph algorithms," in *Proceedings of the 30th International Parallel and Distributed Processing Symposium*, ser. IPDPS '16, Chicago, IL, USA, May 2016, pp. 514–523. DOI: 10.1109/IPDPS.2016.106,
Abstract: Although there is an extensive literature on GPU implementations of graph algorithms, we do not yet have a clear understanding of how implementation choices impact performance. As a step towards this goal, we studied how the choice of synchronization mechanism affects the end-to-end performance of complex graph algorithms, using stochastic gradient descent (SGD) as an exemplar. We implemented seven synchronization strategies for this application and evaluated them on two GPU platforms, using both road networks and social network graphs as inputs. Our

experiments showed that although none of the seven strategies dominates the rest, it is possible to use properties of the platform and input graph to predict the best strategy.

Kreutzer2016 [280] M. Kreutzer, J. Thies, M. Röhrig-Zöllner, A. Pieper, F. Shahzad, M. Galgon, A. Basermann, H. Fehske, G. Hager, and G. Wellein, "GHOST: Building blocks for high performance sparse linear algebra on heterogeneous systems," *International Journal of Parallel Programming*, pp. 1–27, 2016, ISSN: 1573-7640. DOI: 10.1007/s10766-016-0464-z,
Abstract: While many of the architectural details of future exascale-class high performance computer systems are still a matter of intense research, there appears to be a general consensus that they will be strongly heterogeneous, featuring standard as well as accelerated resources. Today, such resources are available as multicore processors, graphics processing units (GPUs), and other accelerators such as the Intel Xeon Phi. Any software infrastructure that claims usefulness for such environments must be able to meet their inherent challenges: massive multi-level parallelism, topology, asynchronicity, and abstraction. The General, Hybrid, and Optimized Sparse Toolkit (GHOST) is a collection of building blocks that targets algorithms dealing with sparse matrix representations on current and future large-scale systems. It implements the MPI+X paradigm, has a pure C interface, and provides hybrid-parallel numerical kernels, intelligent resource management, and truly heterogeneous parallelism for multicore CPUs, Nvidia GPUs, and the Intel Xeon Phi. We describe the details of its design with respect to the challenges posed by modern heterogeneous supercomputers and recent algorithmic developments. Implementation details which are indispensable for achieving high efficiency are pointed out and their necessity is justified by performance measurements or predictions based on performance models. We also provide instructions on how to make use of GHOST in existing software packages, together with a case study which demonstrates the applicability and performance of GHOST as a component within a larger software stack. The library code and several applications are available as open source.

Langr2016 [281] D. Langr and P. Tvrdik, "Evaluation criteria for sparse matrix storage formats," *IEEE-Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 428–440, Feb. 2016, ISSN: 1045-9219. DOI: 10.1109/TPDS.2015.2401575,
Abstract: When authors present new storage formats for sparse matrices, they usually focus mainly on a single evaluation criterion, which is the performance of sparse matrix-vector multiplication (SpMV) in FLOPS. Though such an evaluation is essential, it does not allow to directly compare the presented format with its competitors. Moreover, in case that matrices are within an HPC application constructed in different formats, this criterion alone is not sufficient for the key decision whether or not to convert them into the presented format for the SpMV-based application phase. We establish ten evaluation criteria for sparse matrix storage formats, discuss their advantages and disadvantages, and provide general suggestions for format authors/evaluators to make their work more valuable for the HPC community.

Li2016 [282] S. Li, Y. Wang, W. Wen, Y. Wang, Y. Chen, and H. Li, "A data locality-aware design framework for reconfigurable sparse matrix-vector multiplication kernel," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Austin, TX, US, Nov. 2016, pp. 1–6. DOI: 10.1145/2966986.2966987,
Abstract: Sparse matrix-vector multiplication (SpMV) is an important computational kernel in many applications. For performance improvement, software libraries designated for SpMV computation have been introduced, e.g., MKL library for CPUs and cuSPARSE library for GPUs. However, the computational throughput of these libraries is far below the peak floating-point performance offered by hardware platforms, because the efficiency of SpMV kernel is greatly constrained by the limited memory bandwidth and irregular data access patterns. In this work, we propose a data locality-aware design framework for FPGA-based SpMV acceleration. We first include the hardware constraints in sparse matrix compression at software level to regularize the memory allocation and accesses. Moreover, a distributed architecture composed of processing elements is developed to improve the computation parallelism. We implement the reconfigurable SpMV kernel on Convey HC-2ex and conduct the evaluation by using the University of Florida sparse matrix collection. The experiments demonstrate an average computational efficiency of 48.2%, which is a lot better than those of CPU and GPU implementations. Our FPGA-based kernel has a com-

parable runtime as GPU, and achieves 2.1× reduction than CPU. Moreover, our design obtains substantial saving in energy consumption, say, 9.3× and 5.6× better than the implementations on CPU and GPU, respectively.

Mujtaba2016 [283] H. Mujtaba. (Feb. 2016). NVIDIA pascal GP100 GPU expected to feature 12 TFLOPs of single precision compute, 4 TFLOPs of double precision compute performance. Accessed 20 July 2016, [Online]. Available: http://wccftech.com/nvidia-pascal-gp100-gpu-compute-performance/.

Paralution2016 [284] *Paralution - user manual*, version 1.1.0, 2016. [Online]. Available: http://www.paralution.com/downloads/paralution-um.pdf.

Phist2016 [285] (2016). PHIST - a pipelined hybrid parallel iterative solver toolkit. Git repository, Accessed 10 May 2017, [Online]. Available: https://bitbucket.org/essex/phist/wiki/Getting_Started/Overview.

Picciau2016 [286] A. Picciau, G. E. Inggs, J. Wickerson, E. C. Kerrigan, and G. A. Constantinides, "Balancing locality and concurrency: Solving sparse triangular systems on GPUs," in *Proceedings of the 23rd IEEE International Conference on High Performance Computing*, ser. HiPC '16, Hyderabad, IN, Dec. 2016, pp. 183–192. DOI: 10.1109/HiPC.2016.030,
Abstract: Many numerical optimisation problems rely on fast algorithms for solving sparse triangular systems of linear equations (STLs). To accelerate the solution of such equations, two types of approaches have been used: on GPUs, concurrency has been prioritised to the disadvantage of data locality, while on multi-core CPUs, data locality has been prioritised to the disadvantage of concurrency. In this paper, we discuss the interaction between data locality and concurrency in the solution of STLs on GPUs, and we present a new algorithm that balances both. We demonstrate empirically that, subject to there being enough concurrency available in the input matrix, our algorithm outperforms Nvidia's concurrency-prioritising CUSPARSE algorithm for GPUs. Experimental results show a maximum speedup of 5.8-fold. Our solution algorithm, which we have implemented in OpenCL, requires a pre-processing phase that partitions the graph associated with the input matrix into sub-graphs, whose data can be stored in low-latency local memories. This preliminary analysis phase is expensive, but because it depends only on the input matrix, its cost can be amortised when solving for many different right-hand sides.

Picciau2016a [287] A. Picciau. (2016). Smart building optimisation problems. Git repository, Accessed 10 May 2017, [Online]. Available: https://github.com/andpic/smart-building-optimisation.

Rupp2016 [288] K. Rupp, P. Tillet, F. Rudolf, J. Weinbub, A. Morhammer, T. Grasser, A. Jüngel, and S. Selberherr, "ViennaCL - linear algebra library for multi- and many-core architectures," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, S412–S439, 2016. DOI: 10.1137/15M1026419,
Abstract: CUDA, OpenCL, and OpenMP are popular programming models for the multicore architectures of CPUs and many-core architectures of GPUs or Xeon Phis. At the same time, computational scientists face the question of which programming model to use to obtain their scientific results. We present the linear algebra library ViennaCL, which is built on top of all three programming models, thus enabling computational scientists to interface to a single library, yet obtain high performance for all three hardware types. Since the respective compute back end can be selected at runtime, one can seamlessly switch between different hardware types without the need for error-prone and time-consuming recompilation steps. We present new benchmark results for sparse linear algebra operations in ViennaCL, complementing results for the dense linear algebra operations in ViennaCL reported in earlier work. Comparisons with vendor libraries show that ViennaCL provides better overall performance for sparse matrix-vector and sparse matrix-matrix products. Additional benchmark results for pipelined iterative solvers with kernel fusion and preconditioners identify the respective sweet spots for CPUs, Xeon Phis, and GPUs.

Solomonik2016 [289] E. Solomonik, E. Carson, N. Knight, and J. Demmel, "Trade-offs between synchronization, communication, and computation in parallel linear algebra computations," *ACM Transaction on Parallel Computing*, vol. 3, no. 1, 3:1–3:47, Jan. 2016, ISSN: 2329-4949. DOI: 10.1145/2897188. [Online]. Available: http://doi.acm.org/10.1145/2897188,

Abstract: This article derives trade-offs between three basic costs of a parallel algorithm: synchronization, data movement, and computational cost. These trade-offs are lower bounds on the execution time of the algorithm that are independent of the number of processors but dependent on the problem size. Therefore, they provide lower bounds on the execution time of any parallel schedule of an algorithm computed by a system composed of any number of homogeneous processors, each with associated computational, communication, and synchronization costs. We employ a theoretical model that measures the amount of work and data movement as a maximum over that incurred along any execution path during the parallel computation. By considering this metric rather than the total communication volume over the whole machine, we obtain new insights into the characteristics of parallel schedules for algorithms with nontrivial dependency structures. We also present reductions from BSP and LogGP algorithms to our execution model, extending our lower bounds to these two models of parallel computation. We first develop our results for general dependency graphs and hypergraphs based on their expansion properties, and then we apply the theorem to a number of specific algorithms in numerical linear algebra, namely triangular substitution, Cholesky factorization, and stencil computations. We represent some of these algorithms as families of dependency graphs. We derive their communication lower bounds by studying the communication requirements of the hypergraph structures shared by these dependency graphs. In addition to these lower bounds, we introduce a new communication-efficient parallelization for stencil computation algorithms, which is motivated by results of our lower bound analysis and the properties of previously existing parallelizations of the algorithms.

Sutton2016 [290] M. Sutton, T. Ben-Nun, A. Barak, S. Pai, and K. Pingali. (2016). Adaptive work-efficient connected components on the GPU. arXiv: arXiv:1612.01178,
Abstract: This report presents an adaptive work-efficient approach for implementing the Connected Components algorithm on GPUs. The results show a considerable increase in performance (up to 6.8×) over current state-of-the-art solutions.

Sutton2016a [291] ——, "Adaptive work-efficient connected components on the GPU," The Computing Research Repository, Tech. Rep., 2016. [Online]. Available: http://arxiv.org/abs/1612.01178,
Abstract: This report presents an adaptive work-efficient approach for implementing the Connected Components algorithm on GPUs. The results show a considerable increase in performance (up to 6.8×) over current state-of-the-art solutions.

Wang2016 [292] Y. Wang, A. Davidson, Y. Pan, Y. Wu, A. Riffel, and J. D. Owens, "Gunrock: A high-performance graph processing library on the GPU," in *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '16, Barcelona, ES: ACM, 2016, 11:1–11:12, ISBN: 978-1-4503-4092-2. DOI: 10.1145/2851141.2851145,
Abstract: For large-scale graph analytics on the GPU, the irregularity of data access/control flow and the complexity of programming GPUs have been two significant challenges for developing a programmable high-performance graph library. "Gunrock," our high-level bulk-synchronous graph-processing system targeting the GPU, takes a new approach to abstracting GPU graph analytics: rather than designing an abstraction around computation, Gunrock instead implements a novel data-centric abstraction centered on operations on a vertex or edge frontier. Gunrock achieves a balance between performance and expressiveness by coupling high-performance GPU computing primitives and optimization strategies with a high-level programming model that allows programmers to quickly develop new graph primitives with small code size and minimal GPU programming knowledge. We evaluate Gunrock on five graph primitives (BFS, BC, SSSP, CC, and PageRank) and show that Gunrock has on average at least an order of magnitude speedup over Boost and PowerGraph, comparable performance to the fastest GPU hardwired primitives, and better performance than any other GPU high-level graph library.

Candel2017 [293] F. Candel, S. Petit, J. Sahuquillo, and Duato, "Accurately modeling the on-chip and off-chip GPU memory subsystem," *Future Generation Computer Systems*, 2017, ISSN: 0167-739X. DOI: 10.1016/j.future.2017.02.012,

Abstract: Research on GPU architecture is becoming pervasive in both the academia and the industry because these architectures offer much more performance per watt than typical CPU architectures. This is the main reason why massive deployment of GPU multiprocessors is considered one of the most feasible solutions to attain exascale computing capabilities.
The memory hierarchy of the GPU is a critical research topic, since its design goals widely differ from those of conventional CPU memory hierarchies. Researchers typically use detailed microarchitectural simulators to explore novel designs to better support GPGPU computing as well as to improve the performance of GPU and CPU-GPU systems. In this context, the memory hierarchy is a critical and continuously evolving subsystem.
Unfortunately, the fast evolution of current memory subsystems deteriorates the accuracy of existing state-of-the-art simulators. This paper focuses on accurately modeling the entire (both on-chip and off-chip) GPU memory subsystem. For this purpose, we identify four main memory related components that impact on the overall performance accuracy. Three of them belong to the on-chip memory hierarchy: (i) memory request coalescing mechanisms, (ii) miss status holding registers, and (iii) cache coherence protocol; while the fourth component refers to the memory controller and GDDR memory working activity.
To evaluate and quantify our claims, we accurately modeled the aforementioned memory components in an extended version of the state-of-the-art Multi2Sim heterogeneous CPU-GPU processor simulator. Experimental results show important deviations, which can vary the final system performance provided by the simulation framework up to a factor of three. The proposed GPU model has been compared and validated against the original framework and the results from a real AMD Southern-Islands 7870HD GPU.

Filippone2017 [294] S. Filippone, V. Cardellini, D. Barbieri, and A. Fanfarillo, "Sparse matrix-vector multiplication on GPGPUs," *ACM Transactions on Mathematical Software*, vol. 43, no. 4, 30:1–30:49, Jan. 2017, ISSN: 0098-3500. DOI: 10.1145/3017994,
Abstract: The multiplication of a sparse matrix by a dense vector (SpMV) is a centerpiece of scientific computing applications: it is the essential kernel for the solution of sparse linear systems and sparse eigenvalue problems by iterative methods. The efficient implementation of the sparse matrix-vector multiplication is therefore crucial and has been the subject of an immense amount of research, with interest renewed with every major new trend in high-performance computing architectures. The introduction of General-Purpose Graphics Processing Units (GPGPUs) is no exception, and many articles have been devoted to this problem.
With this article, we provide a review of the techniques for implementing the SpMV kernel on GPGPUs that have appeared in the literature of the last few years. We discuss the issues and tradeoffs that have been encountered by the various researchers, and a list of solutions, organized in categories according to common features. We also provide a performance comparison across different GPGPU models and on a set of test matrices coming from various application domains.

Flegar2017 [295] G. Flegar and H. Anzt, "Overcoming load imbalance for irregular sparse matrices," in *Proceedings of the Seventh Workshop on Irregular Applications: Architectures and Algorithms*, ser. IA3 2017, Nov. 2017, pp. 1–8. DOI: 10.1145/3149704.3149767,
Abstract: In this paper we propose a load-balanced GPU kernel for computing the sparse matrix vector (SpMV) product. Making heavy use of the latest GPU programming features, we also enable satisfying performance for irregular and unbalanced matrices. In a performance comparison using 400 test matrices we reveal the new kernel being superior to the most popular SpMV implementations.

Gurobi2017 [296] *Gurobi optimizer reference manual*, version 7.0, Gurobi Optimization, Dec. 2017. [Online]. Available: http://www.gurobi.com/documentation/7.0/refman.pdf.

Hbeika2017 [297] J. Hbeika and M. Kulkarni, "Locality-aware task-parallel execution on GPUs," in *Proceedings of the 29th International Workshop on Languages and Compilers for Parallel Computing*, C. Ding, J. Criswell, and P. Wu, Eds., ser. LCPC '16. Rochester, NY, USA: Springer International Publishing, 2017, pp. 250–264, ISBN: 978-3-319-52709-3. DOI: 10.1007/978-3-319-52709-3_19,

Abstract: GPGPUs deliver high speedup for regular applications while remaining energy efficient. In recent years, there has been much focus on tuning irregular, task-parallel applications and/or the GPU architecture in order to achieve similar benefits for irregular applications running on GPUs. While most of the previous works have focused on minimizing the effect of control and memory divergence, which are prominent in irregular applications and which degrade the performance, there has been less attention paid to decreasing cache pressure and hence improving performance of applications given the small cache sizes on GPUs.
In this paper we tackle two problems. First we extract data parallelism from irregular task parallel applications, which we do by subdividing each task into sub tasks at the CPU side and sending these sub tasks to the GPU for execution. By doing so we take advantage of the massive parallelism provided by the GPU. Second, to mitigate the memory demands of many tasks that access irregular data structures, we schedule these subtasks in a way to minimize the memory footprint of each warp running on the GPU. We use our framework with 3 task-parallel algorithms and show that we can achieve significant speedups over optimized GPU code.

ImperialHpc [298] *Imperial college high performance computing service*, 2017. [Online]. Available: `http://www.imperial.ac.uk/admin-services/ict/self-service/research-support/hpc/`.

Liu2017 [299] W. Liu, A. Li, J. D. Hogg, I. S. Duff, and B. Vinter, "Fast synchronization-free algorithms for parallel sparse triangular solves with multiple right-hand sides," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 21, e4244, 2017, e4244 cpe.4244. DOI: `10.1002/cpe.4244`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.4244`. [Online]. Available: `https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4244`,
Abstract: Summary The sparse triangular solve kernels, SpTRSV and SpTRSM, are important building blocks for a number of numerical linear algebra routines. Parallelizing SpTRSV and SpTRSM on today's manycore platforms, such as GPUs, is not an easy task since computing a component of the solution may depend on previously computed components, enforcing a degree of sequential processing. As a consequence, most existing work introduces a preprocessing stage to partition the components into a group of level-sets or colour-sets so that components within a set are independent and can be processed simultaneously during the subsequent solution stage. However, this class of methods requires a long preprocessing time as well as significant runtime synchronization overheads between the sets. To address this, we propose in this paper novel approaches for SpTRSV and SpTRSM in which the ordering between components is naturally enforced within the solution stage. In this way, the cost for preprocessing can be greatly reduced, and the synchronizations between sets are completely eliminated. To further exploit the data-parallelism, we also develop an adaptive scheme for efficiently processing multiple right-hand sides in SpTRSM. A comparison with a state-of-the-art library supplied by the GPU vendor, using 20 sparse matrices on the latest GPU device, shows that the proposed approach obtains an average speedup of over two for SpTRSV and up to an order of magnitude speedup for SpTRSM. In addition, our method is up to two orders of magnitude faster for the preprocessing stage than existing SpTRSV and SpTRSM methods.

Matlab2017a [300] MATLAB, *Version 9.2.0.538062 64-bit (r2017a)*, Natick, Massachusetts, 2017.

Neves2017 [301] D. T. Neves, "EPIC: A framework to exploit parallelism in irregular codes," English, *Concurrency and Computation - Practice & Experience*, vol. 29, no. 2, Jan. 2017, ISSN: 1532-0626. DOI: `10.1002/cpe.3842`,
Abstract: To harness the performance potential of current multicore processors, a multitude of algorithms, frameworks and libraries have been developed. Nevertheless, it is still extremely difficult to take advantage of the full potential of multicore processors. Moreover, when using third-party tools and/or in the presence of asymmetric sets of tasks, this problem would only aggravate. The EPIC framework was developed to ease the exploitation of task parallelism in irregular applications that use third-party tools and/or generate asymmetric sets of tasks. It is based on a software design and implements two algorithms that, together, allow, in a seamlessly way, the efficient exploitation of coarse-grained parallelism, fine-grained parallelism, and the combination of both of these types. Thus, it becomes possible to make a better and transparent usage of the performance potential of

current multicore processors on shared-memory systems. In this paper, we present two refinements to the EPIC framework: one that refines the software design of the EPIC framework and another that refines the scheduling algorithm of the EPIC framework. Together, these refinements allow to cope with a special class of sets of tasks: sets of tasks where asymmetry is insignificant or can be neglected. Thus, these refinements broaden the applicability of the EPIC framework to a large class of irregular applications where task parallelism can be exploited. To assess the feasibility and the benefit of using this new version of the EPIC framework to exploit task parallelism, we used four real-world irregular applications three from phylogenetics and another from astrophysics and several input data sets with different characteristics. Our studies show groundbreaking results in terms of the achieved speedups and that scalability is not impaired, even when using third-party tools and/or in the presence of (a)symmetric sets of tasks.

Zhong2017 [302] W. Zhong, Y. Cao, J. Li, J. Sun, and H. Chen, "Specialization or generalization: A study on breadth-first graph traversal on GPUs," in *Proceedings of the International Conference on Progress in Informatics and Computing*, ser. PIC, Dec. 2017, pp. 294–301.

Abdelfattah2018 [303] A. Abdelfattah, A. Haidar, S. Tomov, and J. Dongarra, "Analysis and design techniques towards high-performance and energy-efficient dense linear solvers on GPUs," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–14, 2018, ISSN: 1045–9219. DOI: 10.1109/TPDS.2018.2842785,
Abstract: Graphics Processing Units (GPUs) are widely used in accelerating dense linear solvers. The matrix factorizations, which dominate the runtime for these solvers, are often designed using a hybrid scheme, where GPUs perform trailing matrix updates, while the CPUs perform the panel factorizations. Consequently, hybrid solutions require high-end CPUs and optimized CPU software in order to deliver high performance. Furthermore, they lack the energy efficiency inherent for GPUs due to the use of less energy-efficient CPUs, as well as CPU-GPU communications.

Abdelfattah2018a [304] A. Abdelfattah, A. Haidar, S. Tomov, and J. Dongarra, "Optimizing gpu kernels for irregular batch workloads: A case study for cholesky factorization," in *IEEE High Performance Extreme Computing Conference (HPEC'18)*, IEEE, Waltham, MA: IEEE, Sep. 2018,
Abstract: This paper introduces several frameworks for the design and implementation of high performance GPU kernels that target batch workloads with irregular sizes. Such workloads are ubiquitous in many scientific applications, including sparse direct solvers, astrophysics, and quantum chemistry. The paper addresses two main categories of frameworks, taking the Cholesky factorization as a case study. The first uses host-side kernel launches, and the second uses device-side launches. Within each category, different design options are introduced, with an emphasis on the advantages and the disadvantages of each approach. Our best performing design outperforms the state-of-the-art CPU implementation, scoring up to 4.7× speedup in double precision on a Pascal P100 GPU.

Abe2018 [305] K. Abe, "On convergence speed of parallel variants of bicgstab for solving linear equations," in *Methods and Applications for Modeling and Simulation of Complex Systems*, L. Li, K. Hasegawa, and S. Tanaka, Eds., Singapore: Springer Singapore, 2018, pp. 401–413, ISBN: 978-981-13-2853-4,
Abstract: A number of hybrid Bi-Conjugate Gradient (Bi-CG) methods such as the Bi-CG STABilized (BiCGSTAB) method have been developed for solving linear equations. BiCGSTAB has been most often used for efficiently solving the linear equations, but we have sometimes seen the convergence behavior with a long stagnation phase. In such cases, it is important to have Bi-CG coefficients that are as accurate as possible, and the stabilization strategy for improving the accuracy of the Bi-CG coefficients has been proposed. In present petascale high-performance computing hardware, the main bottleneck of Krylov subspace methods for efficient parallelization is the inner products which require a global reduction. The parallel variants of BiCGSTAB such as communication avoiding and pipelined BiCGSTAB reducing the number of global communication phases and hiding the communication latency have been proposed. However, the numerical stability, specifically, the convergence speed of the parallel variants of BiCGSTAB has not previously been clarified on problems with situations where the convergence is slow (strongly affected by rounding errors). In this paper, therefore, we examine the convergence speed between the standard BiCGSTAB and

the parallel variants, and the effectiveness of the stabilization strategy by numerical experiments on the problems where the convergence has a long stagnation phase.

Abubaker2018 [306] N. F. T. Abubaker, K. Akbudak, and C. Aykanat, "Spatiotemporal graph and hypergraph partitioning models for sparse matrix-vector multiplication on many-core architectures," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–1, 2018, ISSN: 1045-9219. DOI: 10.1109/TPDS.2018.2864729,
Abstract: There exist graph/hypergraph partitioning-based row/column reordering methods for encoding either spatial or temporal locality separately for sparse matrix-vector multiplication (SpMV) operations. Spatial and temporal hypergraph models in these methods are extended to encapsulate both spatial and temporal localities based on cut/uncut net categorization obtained from vertex partitioning. These extensions of spatial and temporal hypergraph models encode the spatial locality primarily and the temporal locality secondarily, and vice-versa, respectively. However, the literature lacks models that simultaneously encode both spatial and temporal localities utilizing only vertex partitioning for further improving the performance of SpMV on shared-memory architectures. In order to fill this gap, we propose a novel spatiotemporal hypergraph model that leads to a one-phase spatiotemporal reordering method which encodes both types of locality simultaneously. We also propose a framework for spatiotemporal methods which encodes both types of locality in two dependent phases and two separate phases. The validity of the proposed spatiotemporal models and methods are tested on a wide range of sparse matrices and the experiments are performed on both a 60-core Intel Xeon Phi processor and a Xeon processor. Results show the validity of the methods via almost doubling the Gflop/s performance through enhancing data locality in parallel SpMV operations.

Acer2018 [307] S. Acer, O. Selvitopi, and C. Aykanat, "Optimizing nonzero-based sparse matrix partitioning models via reducing latency," *Journal of Parallel and Distributed Computing*, 2018, ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2018.08.005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731518305860,
Abstract: For the parallelization of sparse matrix–vector multiplication (SpMV) on distributed memory systems, nonzero-based fine-grain and medium-grain partitioning models attain the lowest communication volume and computational imbalance among all partitioning models. This usually comes, however, at the expense of high message count, i.e., high latency overhead. This work addresses this shortcoming by proposing new fine-grain and medium-grain models that are able to minimize communication volume and message count in a single partitioning phase. The new models utilize message nets in order to encapsulate the minimization of total message count. We further fine-tune these models by proposing delayed addition and thresholding for message nets in order to establish a trade-off between the conflicting objectives of minimizing communication volume and message count. The experiments on an extensive dataset of nearly one thousand matrices show that the proposed models improve the total message count of the original nonzero-based models by up to 27% on the average, which is reflected on the parallel runtime of SpMV as an average reduction of 15% on 512 processors.

Aliaga2018 [308] J. I. Aliaga, E. Dufrechou, P. Ezzatti, and E. S. Quintana-Ortíz, "An efficient gpu version of the preconditioned gmres method," *The Journal of Supercomputing*, Oct. 2018, ISSN: 1573-0484. DOI: 10.1007/s11227-018-2658-1. [Online]. Available: https://doi.org/10.1007/s11227-018-2658-1,
Abstract: In a large number of scientific applications, the solution of sparse linear systems is the stage that concentrates most of the computational effort. This situation has motivated the study and development of several iterative solvers, among which preconditioned Krylov subspace methods occupy a place of privilege. In a previous effort, we developed a GPU-aware version of the GMRES method included in ILUPACK, a package of solvers distinguished by its inverse-based multilevel ILU preconditioner. In this work, we study the performance of our previous proposal and integrate several enhancements in order to mitigate its principal bottlenecks. The numerical evaluation shows that our novel proposal can reach important run-time reductions.

Anzt2018 [309] H. Anzt, E. Chow, and J. Dongarra, "Parilut – a new parallel threshold ilu factorization,"

SIAM Journal on Scientific Computing, vol. 40, no. 4, pp. C503–C519, 2018. DOI: 10.1137/16M1079506,
Abstract: We propose a parallel algorithm for computing a threshold incomplete LU (ILU) factorization. The main idea is to interleave a parallel fixed-point iteration that approximates an incomplete factorization for a given sparsity pattern with a procedure that adjusts the pattern. We describe and test a strategy for identifying nonzeros to be added and nonzeros to be removed from the sparsity pattern. The resulting pattern may be different and more effective than that of existing threshold ILU algorithms. Also in contrast to other parallel threshold ILU algorithms, much of the new algorithm has fine-grained parallelism.

Anzt2018a [310] H. Anzt, T. K. Huckle, J. Bräckle, and J. Dongarra, "Incomplete sparse approximate inverses for parallel preconditioning," *Parallel Computing*, vol. 71, pp. 1–22, 2018, ISSN: 0167-8191. DOI: https://doi.org/10.1016/j.parco.2017.10.003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016781911730176X,
Abstract: In this paper, we propose a new preconditioning method that can be seen as a generalization of block-Jacobi methods, or as a simplification of the sparse approximate inverse (SAI) preconditioners. The Incomplete Sparse Approximate Inverses (ISAI) is in particular efficient in the solution of sparse triangular linear systems of equations. Those arise, for example, in the context of incomplete factorization preconditioning. ISAI preconditioners can be generated via an algorithm providing fine-grained parallelism, which makes them attractive for hardware with a high concurrency level. In a study covering a large number of matrices, we identify the ISAI preconditioner as an attractive alternative to exact triangular solves in the context of incomplete factorization preconditioning.

Benatia2018 [311] A. Benatia, W. Ji, Y. Wang, and F. Shi, "BestSF: A sparse meta-format for optimizing SpMV on GPU," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 15, no. 3, 29:1–29:27, Sep. 2018, ISSN: 1544-3566. DOI: 10.1145/3226228. [Online]. Available: http://doi.acm.org/10.1145/3226228,
Abstract: The Sparse Matrix-Vector Multiplication (SpMV) kernel dominates the computing cost in numerous scientific applications. Many implementations based on different sparse formats were proposed to improve this kernel on the recent GPU architectures. However, it has been widely observed that there is no "best-for-all" sparse format for the SpMV kernel on GPU. Indeed, serious performance degradation of an order of magnitude can be observed without a careful selection of the sparse format to use. To address this problem, we propose in this article BestSF (Best Sparse Format), a new learning-based sparse meta-format that automatically selects the most appropriate sparse format for a given input matrix. To do so, BestSF relies on a cost-sensitive classification system trained using Weighted Support Vector Machines (WSVMs) to predict the best sparse format for each input sparse matrix. Our experimental results on two different NVIDIA GPU architectures using a large number of real-world sparse matrices show that BestSF achieved a noticeable overall performance improvement over using a single sparse format. While BestSF is trained to select the best sparse format in terms of performance (GFLOPS), our further experimental investigations revealed that using BestSF also led, in most of the test cases, to the best energy efficiency (MFLOPS/W). To prove its practical effectiveness, we also evaluate the performance and energy efficiency improvement achieved when using BestSF as a building block in a GPU-based Preconditioned Conjugate Gradient (PCG) iterative solver.

Bernaschi2018 [312] M. Bernaschi, P. D'Ambra, and D. Pasquini, "Amg based on compatible weighted matching," *Parallel Computing*, 2018,
Abstract: We describe main issues and design principles of an efficient implementation, tailored to recent generations of Nvidia Graphics Processing Units (GPUs), of an Algebraic MultiGrid (AMG) preconditioner previously proposed by one of the authors and already available in the open-source package BootCMatch: Bootstrap algebraic multigrid based on Compatible weighted Matching for standard CPU. The AMG method relies on a new approach for coarsening sparse symmetric positive definite (s.p.d.) matrices, named coarsening based on compatible weighted matching. It exploits maximum weight matching in the adjacency graph of the sparse matrix, driven by the principle of compatible relaxation, providing a suitable aggregation of unknowns

which goes beyond the limits of the usual heuristics applied in the current methods. We adopt an approximate solution of the maximum weight matching problem, based on a recently proposed parallel algorithm, referred as the Suitor algorithm, and show that it allow us to obtain good quality coarse matrices for our AMG on GPUs. We exploit inherent parallelism of modern GPUs in all the kernels involving sparse matrix computations both for the setup of the preconditioner and for its application in a Krylov solver, outperforming preconditioners available in Nvidia AmgX library. We report results about a large set of linear systems arising from discretization of scalar and vector partial differential equations (PDEs).

Booth2018 [313] J. D. Booth and G. Bolet. (Dec. 13, 2018). Javelin: A scalable implementation for sparse incomplete LU factorization. arXiv: 1812.06160v1 [cs.MS],
Abstract: In this work, we present a new scalable incomplete LU factorization framework called Javelin to be used as a preconditioner for solving sparse linear systems with iterative methods. Javelin allows for improved parallel factorization on shared-memory many-core systems, while packaging the coefficient matrix into a format that allows for high performance sparse matrix-vector multiplication and sparse triangular solves with minimal overheads. The framework achieves these goals by using a collection of traditional permutations, point-to-point thread synchronizations, tasking, and segmented prefix scans in a conventional compressed sparse row format. Using these changes, traditional fill-in and drop tolerance methods can be used, while still being able to have observed speedups of up to 42× on 68 Intel Knights Landing cores and 12× on 14 Intel Haswell cores.

Bromberger2018 [314] M. Bromberger, M. Hoffmann, and R. Rehrmann, "Do iterative solvers benefit from approximate computing? an evaluation study considering orthogonal approximation methods," in *Architecture of Computing Systems*, M. Berekovic, R. Buchty, H. Hamann, D. Koch, and T. Pionteck, Eds., ser. ARCS 2018, Cham: Springer International Publishing, 2018, pp. 297–310, ISBN: 978-3-319-77610-1,
Abstract: Employing algorithms of scientific computing often comes in hand with finding a trade-off between accuracy and performance. Novel parallel hardware and algorithms only slightly improve these issues due to the increasing size of the problems. While high accuracy is inevitable for most problems, there are parts in scientific computing that allow us to introduce approximation. Therefore, in this paper we give answers to the following questions: (1) Can we exploit different approximate computing strategies in scientific computing? (2) Is there a strategy to combine approaches? To answer these questions, we apply different approximation strategies to a widely used iterative solver for linear systems of equations. We show the advantages and the limits of each strategy and a way to configure a combination of strategies according to a given relative error. Combining orthogonal strategies as an overall concept gives us significant opportunities to increase the performance.

Buttari2018 [315] A. Buttari, "Scalability of parallel sparse direct solvers: Methods, memory and performance," IRIT, Institut de recherche en informatique de Toulouse, Habilitation à diriger des recherches, Sep. 2018. [Online]. Available: https://hal.archives-ouvertes.fr/tel-01913033,
Abstract: The fast and accurate solution of large size sparse systems of linear equations is at the heart of numerical applications from a very broad range of domains including structural mechanics, fluid dynamics, geophysics, medical imaging, chemistry. Among the most commonly used techniques, direct methods, based on the factorization of the system matrix, are generally appreciated for their numerical robustness and ease of use. These advantages, however, come at the price of a considerable operations count and memory footprint. The work presented in this thesis is concerned with improving the scalability of sparse direct solvers, intended as the ability to solve problems of larger and larger size. More precisely, our work aims at developing solvers which are scalable in performance, memory consumption and complexity. We address performance scalability, that is the ability to reduce the execution time as more computational resources are available, introducing algorithms that improve parallelism by reducing communications and synchronizations. We discuss the use of novel parallel programming paradigms and tools to achieve their implementation in an efficient and portable way on modern, heterogeneous supercomputers.

We present methods that make sparse direct solvers memory-scalable, that is, capable of taking advantage of parallelism without increasing the overall memory footprint. Finally we show how it is possible to use data sparsity to achieve an asymptotic reduction of the cost of such methods. The presented algorithms have been implemented in the freely distributed MUMPS and qr_mumps solver packages and their effectiveness assessed on real life problems from academic and industrial applications.

Carson2018 [316] E. Carson, M. Rozložník, Z. Strakoš, P. Tichý, and M. Tůma, "The numerical stability analysis of pipelined conjugate gradient methods: Historical context and methodology," *SIAM Journal on Scientific Computing*, vol. 40, no. 5, A3549–A3580, 2018. DOI: 10.1137/16M1103361,

Abstract: Algebraic solvers based on preconditioned Krylov subspace methods are among the most powerful tools for large-scale numerical computations in applied mathematics, sciences, technology, as well as in emerging applications in social sciences. As the name suggests, Krylov subspace methods can be viewed as a sequence of projections onto nested subspaces of increasing dimension. They are therefore by their nature implemented as synchronized recurrences. This is the fundamental obstacle to efficient parallel implementation. Standard approaches to overcoming this obstacle described in the literature involve reducing the number of global synchronization points and increasing parallelism in performing arithmetic operations within individual iterations. One such approach, employed by the so-called pipelined Krylov subspace methods, involves overlapping the global communication needed for computing inner products with local arithmetic computations. Inexact computations in Krylov subspace methods, due to either floating point roundoff error or intentional action motivated by savings in computing time or energy consumption, have two basic effects, namely, slowing down convergence and limiting attainable accuracy. Although the methodologies for their investigation are different, these phenomena are closely related and cannot be separated from one another. The study of mathematical properties of Krylov subspace methods, in both the cases of exact and inexact computations, is a very active area of research and many issues in the analytic theory of Krylov subspace methods remain open. Numerical stability issues have been studied since the formulation of the conjugate gradient method in the middle of the last century, with many remarkable results achieved since then. Recently, the issues of attainable accuracy and delayed convergence caused by inexact computations became of interest in relation to pipelined conjugate gradient methods and their generalizations. In this contribution we recall the related early results and developments in synchronization-reducing conjugate gradient methods, identify the main factors determining possible numerical instabilities, and present a methodology for the analysis and understanding of pipelined conjugate gradient methods. We derive an expression for the residual gap that applies to any conjugate gradient method variant that uses a particular auxiliary vector in updating the residual, including pipelined conjugate gradient methods, and show how this result can be used to perform a full-scale analysis for a particular implementation. The paper concludes with a brief perspective on Krylov subspace methods in the forthcoming exascale era.

Cartis2018 [317] C. Cartis, N. I. M. Gould, and P. L. Toint, "Sharp worst-case evaluation complexity bounds for arbitrary-order nonconvex optimization with inexpensive constraints," 2018. arXiv: arXiv:1811.01220,

Abstract: We provide sharp worst-case evaluation complexity bounds for non convex minimization problems with general inexpensive constraints, i.e. problems where the cost of evaluating/enforcing of the (possibly nonconvex or even disconnected) constraints, if any, is negligible compared to that of evaluating the objective function. These bounds unify, extend or improve all known upper and lower complexity bounds for unconstrained and convexly-constrained problems. It is shown that, given an accuracy level ϵ, a degree of highest available Lipschitz continuous derivatives p and a desired optimality order q between one and p, a conceptual regularization algorithm requires no more than $O(\epsilon^{-\frac{p+1}{p-q+1}}$ evaluations of the objective function and its derivatives to compute a suitably approximate q-th order minimizer. With an appropriate choice of the regularization, a similar result also holds if the p-th derivative is merely Holder rather than Lipschitz continuous. We provide an example that shows that the above complexity bound is sharp for unconstrained

and a wide class of constrained problems; we also give reasons for the optimality of regularization methods from a worst-case complexity point of view, within a large class of algorithms that use the same derivative information.

Cartis2018a [318] ——, (2018). Sharp worst-case evaluation complexity bounds for arbitrary-order nonconvex optimization with inexpensive constraints. arXiv: arXiv:1811.01220,
Abstract: We provide sharp worst-case evaluation complexity bounds for non convex minimization problems with general inexpensive constraints, i.e. problems where the cost of evaluating/enforcing of the (possibly nonconvex or even disconnected) constraints, if any, is negligible compared to that of evaluating the objective function. These bounds unify, extend or improve all known upper and lower complexity bounds for unconstrained and convexly-constrained problems. It is shown that, given an accuracy level ϵ, a degree of highest available Lipschitz continuous derivatives p and a desired optimality order q between one and p, a conceptual regularization algorithm requires no more than $O(\epsilon^{-\frac{p+1}{p-q+1}}$ evaluations of the objective function and its derivatives to compute a suitably approximate q-th order minimizer. With an appropriate choice of the regularization, a similar result also holds if the p-th derivative is merely Holder rather than Lipschitz continuous. We provide an example that shows that the above complexity bound is sharp for unconstrained and a wide class of constrained problems; we also give reasons for the optimality of regularization methods from a worst-case complexity point of view, within a large class of algorithms that use the same derivative information.

Cartis2018b [319] ——, *Sharp worst-case evaluation complexity bounds for arbitrary-order nonconvex optimization with inexpensive constraints*, 2018. eprint: arXiv:1811.01220,
Abstract: We provide sharp worst-case evaluation complexity bounds for non convex minimization problems with general inexpensive constraints, i.e. problems where the cost of evaluating/enforcing of the (possibly nonconvex or even disconnected) constraints, if any, is negligible compared to that of evaluating the objective function. These bounds unify, extend or improve all known upper and lower complexity bounds for unconstrained and convexly-constrained problems. It is shown that, given an accuracy level ϵ, a degree of highest available Lipschitz continuous derivatives p and a desired optimality order q between one and p, a conceptual regularization algorithm requires no more than $O(\epsilon^{-\frac{p+1}{p-q+1}}$ evaluations of the objective function and its derivatives to compute a suitably approximate q-th order minimizer. With an appropriate choice of the regularization, a similar result also holds if the p-th derivative is merely Holder rather than Lipschitz continuous. We provide an example that shows that the above complexity bound is sharp for unconstrained and a wide class of constrained problems; we also give reasons for the optimality of regularization methods from a worst-case complexity point of view, within a large class of algorithms that use the same derivative information.

Cheshmi2018 [320] K. Cheshmi, S. Kamil, M. M. Strout, and M. M. Dehnavi, "Parsy: Inspection and transformation of sparse matrix computations for parallelism," in *Proceedings of The International Conference for High Performance Computing, Networking, Storage, and Analysis*, ser. SC '18, Dallas, TX, US, 2018,
Abstract: ParSy is a framework that generates parallel code for sparse matrix computations. It uses a novel inspection strategy along with code transformations to generate parallel code for shared memory processors that is optimized for locality and load balance. Code generated by existing automatic parallelism approaches for sparse algorithms can suffer from load imbalance and excessive synchronization, resulting in performance that does not scale well on multi-core systems. We propose a novel task coarsening strategy that creates well-balanced tasks that can execute in parallel. ParSy-generated code outperforms existing highly-optimized sparse matrix codes such as the Cholesky factorization on multi-core processors with speed-ups of 2.8× and 3.1× over the MKL Pardiso and PaStiX libraries respectively.

Chow2018 [321] E. Chow, H. Anzt, J. Scott, and J. Dongarra, "Using jacobi iterations and blocking for solving sparse triangular systems in incomplete factorization preconditioning," *Journal of Parallel and Distributed Computing*, vol. 119, pp. 219–230, 2018, ISSN: 0743-7315. DOI:

https://doi.org/10.1016/j.jpdc.2018.04.017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731518303034,
Abstract: When using incomplete factorization preconditioners with an iterative method to solve large sparse linear systems, each application of the preconditioner involves solving two sparse triangular systems. These triangular systems are challenging to solve efficiently on computers with high levels of concurrency. On such computers, it has recently been proposed to use Jacobi iterations, which are highly parallel, to approximately solve the triangular systems from incomplete factorizations. The effectiveness of this approach, however, is problem-dependent: the Jacobi iterations may not always converge quickly enough for all problems. Thus, as a necessary and important step to evaluate this approach, we experimentally test the approach on a large number of realistic symmetric positive definite problems. We also show that by using block Jacobi iterations, we can extend the range of problems for which such an approach can be effective. For block Jacobi iterations, it is essential for the blocking to be cognizant of the matrix structure.

Coutinho2018 [322] D. Coutinho, S. Xavier-de-Souza, and D. Aloise. (2018). A scalable shared-memory parallel simplex for large-scale linear programming. arXiv: arXiv:1804.04737,
Abstract: We present a shared-memory parallel implementation of the Simplex tableau algorithm for dense large-scale Linear Programming (LP) problems. We present the general scheme and explain each parallelization step of the standard simplex algorithm, emphasizing important solutions for solving performance bottlenecks. We analyzed the speedup and the parallel efficiency for the proposed implementation relative to the standard Simplex algorithm using a shared-memory system with 64 processing cores. The experiments were performed for several different problems, with up to 8192 variables and constraints, in their primal and dual formulations. The results show that the performance is mostly much better when we use the formulation with more variables than inequality constraints. Also, they show that the parallelization strategies applied to avoid bottlenecks caused the implementation to scale well with the problem size and the core count up to a certain limit of problem size. Further analysis showed that this was an effect of resource limitation. Even though, our implementation was able to reach speedups in the order of 19×.

Coutinho2018a [323] ——, "A scalable shared-memory parallel simplex for large-scale linear programming," 2018. eprint: arXiv:1804.04737,
Abstract: We present a shared-memory parallel implementation of the Simplex tableau algorithm for dense large-scale Linear Programming (LP) problems. We present the general scheme and explain each parallelization step of the standard simplex algorithm, emphasizing important solutions for solving performance bottlenecks. We analyzed the speedup and the parallel efficiency for the proposed implementation relative to the standard Simplex algorithm using a shared-memory system with 64 processing cores. The experiments were performed for several different problems, with up to 8192 variables and constraints, in their primal and dual formulations. The results show that the performance is mostly much better when we use the formulation with more variables than inequality constraints. Also, they show that the parallelization strategies applied to avoid bottlenecks caused the implementation to scale well with the problem size and the core count up to a certain limit of problem size. Further analysis showed that this was an effect of resource limitation. Even though, our implementation was able to reach speedups in the order of 19×.

Dahiya2018 [324] Y. Dahiya, D. Konomis, and D. P. Woodruff, "An empirical evaluation of sketching for numerical linear algebra," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '18, London, United Kingdom: ACM, 2018, pp. 1292–1300, ISBN: 978-1-4503-5552-0. DOI: 10.1145/3219819.3220098. [Online]. Available: http://doi.acm.org/10.1145/3219819.3220098,
Abstract: Over the last ten years, tremendous speedups for problems in randomized numerical linear algebra such as low rank approximation and regression have been made possible via the technique of randomized data dimensionality reduction, also known as sketching. In theory, such algorithms have led to optimal input sparsity time algorithms for a wide array of problems. While several scattered implementations of such methods exist, the goal of this work is to provide a comprehensive comparison of such methods to alternative approaches. We investigate least squares regression, iteratively reweighted least squares, logistic regression, robust regression with Huber

and Bisquare loss functions, leverage score computation, Frobenius norm low rank approximation, and entrywise ℓ_1-low rank approximation. We give various implementation techniques to speed up several of these algorithms, and the resulting implementations demonstrate the tradeoffs of such techniques in practice.

Davis2018a [325] T. A. Davis, "Graph algorithms via SuiteSparse:GraphBLAS:triangle counting and k-truss," in *Proceedings of the IEEE High Performance Extreme Computing Conference*, ser. HPEC '18, 2018,
Abstract: SuiteSparse:GraphBLAS is a full implementation of the GraphBLAS standard, which defines a set of sparse matrix operations on an extended algebra of semirings using an almost unlimited variety of operators and types. When applied to sparse adjacency matrices, these algebraic operations are equivalent tocomputations on graphs. GraphBLAS provides a powerful and expressive framework for creating graph algorithms based on the elegant mathematics of sparse matrix operations on a semiring. To illustrate GraphBLAS, two graph algorithms are constructed in GraphBLAS and compared with efficient implementations without GraphBLAS: triangle counting and constructing the k-truss of a graph.

Davis2018 [326] T. A. Davis, W. W. Hager, S. P. Kolodziej, and S. N. Yearalan, "Algorithm XXX: Mongoose, a graph coarsening and partitioning library," *ACM Transactions on Mathematical Software*, vol. 0, no. 0, Apr. 2018,
Abstract: Partitioning graphs is a common and useful operation in many areas, from parallel computing to VLSI design to sparse matrix algorithms. In this paper, we introduce Mongoose, a multilevel hybrid graph partitioning algorithm and library. Building on previous work in multilevel partitioning frameworks and combinatoric approaches, we introduce novel stall-reducing and stall-free coarsening strategies, as well as an efficient hybrid algorithm leveraging 1) traditional combinatoric methods and 2) continuous quadratic programming formulations. We demonstrate how this new hybrid algorithm outperforms either strategy in isolation, and we also compare Mongoose to METIS and demonstrate its effectiveness on large and social networking (power law) graphs.

Dufrechou2018a [327] E. Dufrechou and P. Ezzatti, "A new gpu algorithm to compute a level set-based analysis for the parallel solution of sparse triangular systems," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, ser. IPDPS '18, May 2018, pp. 920–929. DOI: 10.1109/IPDPS.2018.00101,
Abstract: A myriad of problems in science and engineering, involve the solution of sparse triangular linear systems. They arise frequently as part of direct and iterative solvers for linear systems and eigenvalue problems, and hence can be considered as a key building block of sparse numerical linear algebra. This is why, since the early days, their parallel solution has been exhaustively studied, and efficient implementations of this kernel can be found for almost every hardware platform. In the GPU context, the most widespread implementation of this kernel is the one distributed in NVIDIA CUSPARSE library, which relies on a preprocessing stage to aggregate the unknowns of the triangular system into level sets. This determines an execution schedule for the solution of the system, where the level sets have to be processed sequentially while the unknowns that belong to one level set can be solved in parallel. One of the disadvantages of the CUSPARSE implementation is that this preprocessing stage is often extremely slow in comparison to the runtime of the solving phase. In this work, we present a parallel GPU algorithm that is able to compute the same level sets as CU S PARSE but takes significantly less runtime. Our experiments on a set of matrices from the SuiteSparse collection show acceleration factors of up to 44×. Additionally, we provide a routine capable of solving a triangular linear system on the same pass used to calculate the level sets, yielding important performance benefits.

Dufrechou2018 [328] E. Dufrechou and P. Ezzatti, "Solving sparse triangular linear systems in modern GPUs: A synchronization-free algorithm," in *Proceedings of the 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, ser. PDP, Cambridge, UK, 2018, pp. 196–203. DOI: 10.1109/PDP2018.2018.00034,

Abstract: Sparse triangular linear systems are ubiquitous in a wide range of science and engineering fields, and represent one of the most important building blocks of Sparse Numerical Lineal Algebra methods. For this reason, their parallel solution has been subject of exhaustive study, and efficient implementations of this kernel can be found for almost every hardware platform. However, the strong data dependencies that serialize a great deal of the execution and the load imbalance inherent to the triangular structure poses serious difficulties for its parallel performance, specially in the context of massively- parallel processors such as GPUs. To this day, the most widespread GPU implementation of this kernel is the one distributed in NVIDIA CUSPARSE library, which relies on a preprocessing stage to determine the parallel execution schedule. Although the solution phase is highly efficient, this strategy pays the cost of constant synchronizations with the CPU. In this work, we present a synchronization-free GPU al- gorithm to solve sparse triangular linear systems for the CSR format. The experimental evaluation shows performance improvements over CUSPARSE and a recently proposed synchronization-free method for the CSC matrix format.

Dziekonski2018 [329] A. Dziekonski and M. Mrozowski, "A GPU solver for sparse generalized eigenvalue problems with symmetric complex-valued matrices obtained using higher-order fem," *IEEE Access*, 2018. DOI: 10.1109/access.2018.2871219,

Abstract: The paper discusses a fast implementation of the stabilized locally optimal block preconditioned conjugate gradient (sLOBPCG) method, using a hierarchical multilevel preconditioner to solve non-Hermitian sparse generalized eigenvalue problems with large symmetric complex-valued matrices obtained using the higher-order finite-element method (FEM), applied to the analysis of a microwave resonator. The resonant frequencies of the low-order modes are the eigenvalues of the smallest real part of a complex symmetric (though non-Hermitian) matrix pencil. These type of pencils arise in the FEM analysis of resonant cavities loaded with a lossy material. To accelerate the computations, graphics processing units (GPU, NVIDIA Pascal P100) were used. Single and dual-GPU variants are considered and a GPU-memory-saving implementation is proposed. An efficient sliced ELLR-T sparse matrix storage format was used and operations were performed on blocks of vectors for best performance on a GPU. As a result, significant speedups (exceeding a factor of six in some computational scenarios) were achieved over the reference parallel implementation using a multicore central processing unit (CPU, Intel Xeon E5-2680 v3, twelve cores). These results indicate that the solution of generalized eigenproblems needs much more GPU memory than iterative techniques when solving a sparse system of equations, and also requires a second GPU to store some data structures in order to reduce the footprint, even for a moderately large systems.

Ekstrom2018 [330] S.-E. Ekström and C. Garoni, "A matrix-less and parallel interpolation–extrapolation algorithm for computing the eigenvalues of preconditioned banded symmetric toeplitz matrices," *Numerical Algorithms*, 2018. DOI: 10.1007/s11075-018-0508-0,

Abstract: In the past few years, Bogoya, Bottcher, Grudsky, and Maximenko obtained the precise asymptotic expansion for the eigenvalues of a Toeplitz matrix $T_n(f)$, under suitable assumptions on the generating function f, as the matrix size n goes to infinity. On the basis of several numerical experiments, it was conjectured by Serra-Capizzano that a completely analogous expansion also holds for the eigenvalues of the preconditioned Toeplitz matrix $T_n(u)^{-1}T_n(v)$, provided $f = v/u$ is monotone and further conditions on u and v are satisfied. Based on this expansion, we here propose and analyze an interpolation–extrapolation algorithm for computing the eigenvalues of $T_n(u)^{-1}T_n(v)$. The algorithm is suited for parallel implementation and it may be called "matrix-less" as it does not need to store the entries of the matrix. We illustrate the performance of the algorithm through numerical experiments and we also present its generalization to the case where $f = v/u$ is non-monotone.

Franceschini2018 [331] A. Franceschini, M. Ferronato, C. Janna, and V. A. Magri, "Recent advancements in preconditioning techniques for large size linear systems suited for high performance computing," in *Seminari Padovani di Analisi Numerica 2018*, ser. SPAN2018, vol. 11, 2018, pp. 11–22,

Abstract: The numerical simulations of real-world engineering problems create models with several millions or even billions of degrees of freedom. Most of these simulations are centered on the solution of systems of non-linear equations, that, once linearized, become a sequence of linear systems, whose solution is often the most time-demanding task. Thus, in order to increase the capability of

modeling larger cases, it is of paramount importance to exploit the resources of High Performance Computing architectures. In this framework, the development of new algorithms to accelerate the solution of linear systems for many-core architectures is a really active research field. Our main focus is algebraic preconditioning and, among the various options, we elect to develop approximate inverses for symmetric and positive definite (SPD) linear systems [22], both as stand-alone preconditioner or smoother for AMG techniques. This choice is mainly supported by the almost perfect parallelism that intrinsically characterizes these algorithms. As basic kernel, the Factorized Sparse Approximate Inverse (FSAI) developed in its adaptive form by Janna and Ferronato [18] is selected. Recent developments are i) a robust multilevel approach for SPD problems based on FSAI preconditioning, which eliminates the chance of algorithmic breakdowns independently of the preconditioner sparsity [14] and ii) a novel AMG approach featuring the adaptive FSAI method as a flexible smoother as well as new approaches to adaptively compute the prolongation operator. In this latter work, a new technique to build the prolongation is also presented.

Franchetti2018 [332] F. Franchetti, J. M. F. Moura, D. A. Padua, and J. Dongarra, "From high-level specification to high-performance code," *Proceedings of the IEEE*, vol. 106, pp. 1875–1878, 11 2018,
Abstract: Computer architectures and systems are becoming ever more powerful but increasingly more complex. With the end of frequency scaling (about 2004) and the era of multicores/manycores/accelerators, it is exceedingly hard to extract the promised performance, in particular, at a reasonable energy budget. Only highly trained and educated experts can hope to conquer this barrier that, if not appropriately dealt with, can translate into multiple orders of magnitude of underutilization of computer systems when programmed by less specialized programmers or domain scientists. To overcome this challenge, the last ten years have seen a flurry of activity to automate the design and generation of highly efficient implementations for these multicore/ manycore architectures, and to translate high level descriptions of programs into high performance and power efficiency.

Fukaya2018 [333] T. Fukaya, R. Kannan, Y. Nakatsukasa, Y. Yamamoto, and Y. Yanagisawa. (2018). Shifted choleskyqr for computing the qr factorization of ill-conditioned matrices. arXiv: `arXiv:1809.11085`,
Abstract: The Cholesky QR algorithm is an efficient communication-minimizing algorithm for computing the QR factorization of a tall-skinny matrix. Unfortunately it has the inherent numerical instability and breakdown when the matrix is ill-conditioned. A recent work establishes that the instability can be cured by repeating the algorithm twice (called CholeskyQR2). However, the applicability of CholeskyQR2 is still limited by the requirement that the Cholesky factorization of the Gram matrix runs to completion, which means it does not always work for matrices X with $\kappa_2(X) \geq u^{-\frac{1}{2}}$ where u is the unit roundoff. In this work we extend the applicability to $\kappa_2(X) = \mathcal{O}(u^{-1})$ by introducing a shift to the computed Gram matrix so as to guarantee the Cholesky factorization $R^T R = A^T A + sI$ succeeds numerically. We show that the computed AR^{-1} has reduced condition number $\leq u^{-\frac{1}{2*}}$, for which CholeskyQR2 safely computes the QR factorization, yielding a computed Q of orthogonality $\mid Q^T Q - I \mid_2$ and residual $\mid A - QR \mid_F / \mid A \mid_F$ both $\mathcal{O}(u)$. Thus we obtain the required QR factorization by essentially running Cholesky QR thrice. We extensively analyze the resulting algorithm shiftedCholeskyQR to reveal its excellent numerical stability. shiftedCholeskyQR is also highly parallelizable, and applicable and effective also when working in an oblique inner product space. We illustrate our findings through experiments, in which we achieve significant (up to $\times 40$) speedup over alternative methods.

Fukaya2018a [334] ——, *Shifted choleskyqr for computing the qr factorization of ill-conditioned matrices*, 2018. eprint: `arXiv:1809.11085`,
Abstract: The Cholesky QR algorithm is an efficient communication-minimizing algorithm for computing the QR factorization of a tall-skinny matrix. Unfortunately it has the inherent numerical instability and breakdown when the matrix is ill-conditioned. A recent work establishes that the instability can be cured by repeating the algorithm twice (called CholeskyQR2). However, the applicability of CholeskyQR2 is still limited by the requirement that the Cholesky factorization of the Gram matrix runs to completion, which means it does not always work for matrices X with $\kappa_2(X) \geq u^{-\frac{1}{2}}$ where u is the unit roundoff. In this work we extend the applicability to $\kappa_2(X) = \mathcal{O}(u^{-1})$ by introducing a shift to the computed Gram matrix so as to guarantee the Cholesky

factorization $R^TR = A^TA + sI$ succeeds numerically. We show that the computed AR^{-1} has reduced condition number $\leq u^{-\frac{1}{2*}}$, for which CholeskyQR2 safely computes the QR factorization, yielding a computed Q of orthogonality $\mid Q^TQ - I \mid_2$ and residual $\mid A - QR \mid_F / \mid A \mid_F$ both $\mathcal{O}(u)$. Thus we obtain the required QR factorization by essentially running Cholesky QR thrice. We extensively analyze the resulting algorithm shiftedCholeskyQR to reveal its excellent numerical stability. shiftedCholeskyQR is also highly parallelizable, and applicable and effective also when working in an oblique inner product space. We illustrate our findings through experiments, in which we achieve significant (up to $\times 40$) speedup over alternative methods.

Goddard2018 [335] A. Goddard and A. Wathen, "A note on parallel preconditioning for all-at-once evolutionary PDEs," *Electronic Transactions on Numerical Analysis*, vol. XX, 2018,
Abstract: McDonald, Pestana and Wathen (SIAM J. Sci. Comput. 40 (2), pp. A2012–A1033, 2018) present a method for preconditioning of time-dependent PDEs via approximation by a nearby time-periodic problem, that is, they employ circulant-related matrices as preconditioners for the non-symmetric block Toeplitz matrices which arise from an all-at-once formulation. They suggest that such an approach might be efficiently implemented in parallel. In this short article, we present parallel numerical results for their preconditioner which exhibit strong scaling. We also extend their preconditioner via a Neumann series approach, which also allows for efficient parallel execution. Our simple implementation (in C++ and MPI) is available at the Git repository PARALAAOMPI.

Haidar2018 [336] A. Haidar, S. Tomov, J. Dongarra, and N. J. Higham, "Harnessing a GPU's tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers," in *Proceedings of NVIDIA's GPU Technology Conference*, ser. GTC '18, 2018,
Abstract: Recent in-hardware GPU acceleration of half precision arithmetic (FP16) – motivated by various machine learning (ML) and artificial intelligence (AI) applications – has reinvigorated a great interest in the mixed-precision iterative refinement technique. The technique is based on use of low precision arithmetic to accelerate the general HPC problem of solving Ax = b, where A is a large dense matrix, and the solution is needed in FP64 accuracy. While being a well known technique, its successful modification, software development, and adjustment to match architecture specifics, is challenging. For current manycore GPUs the challenges range from efficient parallelization to scaling, and using the FP16 arithmetic. Here, we address these challenges by showing how to algorithmically modify, develop high-performance implementations, and in general, how to use the FP16 arithmetic to significantly accelerate, as well as make more energy efficient, FP64-precision Ax = b solvers. One can reproduce our results as the developments will be made available through the MAGMA library. We quantify in practice the performance, and limitations of the approach stressing on the use of the Volta V100 Tensor Cores that provide additional FP16 performance boost.

Hong2018 [337] C. Hong, A. Sukumaran-Rajam, B. Bandyopadhyay, J. Kim, S. E. Kurt, I. Nisa, S. Sabhlok, Ü. V. Çatalyürek, S. Parthasarathy, and P. Sadayappan, "Efficient sparse-matrix multi-vector product on GPUs," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '18, Tempe, Arizona: ACM, 2018, pp. 66–79, ISBN: 978-1-4503-5785-2. DOI: 10.1145/3208040.3208062. [Online]. Available: http://doi.acm.org/10.1145/3208040.3208062,
Abstract: Sparse Matrix-Vector (SpMV) and Sparse Matrix-Multivector (SpMM) products are key kernels for computational science and data science. While GPUs offer significantly higher peak performance and memory bandwidth than multicore CPUs, achieving high performance on sparse computations on GPUs is very challenging. A tremendous amount of recent research has focused on various GPU implementations of the SpMV kernel. But the multi-vector SpMM kernel has received much less attention. In this paper, we present an in-depth analysis to contrast SpMV and SpMM, and develop a new sparse-matrix representation and computation approach suited to achieving high data-movement efficiency and effective GPU parallelization of SpMM. Experimental evaluation using the entire SuiteSparse matrix suite demonstrates significant performance improvement over existing SpMM implementations from vendor libraries.

Jacquelin2018 [338] M. Jacquelin, E. G. Ng, and B. W. Peyton, "Fast and effective reordering of columns within supernodes using partition refinement," in *Proceedings of the Seventh SIAM Workshop on*

Combinatorial Scientific Computing. 2018, pp. 76–86. DOI: 10.1137/1.9781611975215.8. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611975215.8,
Abstract: In this paper, we consider the problem of computing a triangular factorization of a sparse symmetric matrix using Gaussian elimination. We assume that the sparse matrix has been permuted using a fill-reducing ordering algorithm. When the matrix is symmetric positive definite, the sparsity structure of the triangular factor can be determined once the fill-reducing ordering has been computed. Thus, an efficient numerical factorization scheme can be designed so that only the nonzero entries are stored and operated on. On modern architectures, the positions of the nonzero entries in the triangular factor play a big role in determining the efficiency. It is desirable to have dense blocks in the factor so that the computation can be cast in terms of level-3 BLAS as much as possible. On architectures with GPUs, for example, it is also desirable for these dense blocks to be as large as possible in order to reduce the times to transfer data between the main CPU and the GPUs. We address the problem of locally refining the ordering so that the number of dense blocks is reduced and the sizes of these dense blocks are increased in the triangular factor.

Jaiganesh2018 [339] J. Jaiganesh and M. Burtscher, "A high-performance connected components implementation for gpus," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '18, Tempe, Arizona: ACM, 2018, pp. 92–104, ISBN: 978-1-4503-5785-2. DOI: 10.1145/3208040.3208041,
Abstract: Computing connected components is an important graph algorithm that is used, for example, in medicine, image processing, and biochemistry. This paper presents a fast connected-components implementation for GPUs called ECL-CC. It builds upon the best features of prior algorithms and augments them with GPU-specific optimizations. For example, it incorporates a parallelism-friendly version of pointer jumping to speed up union-find operations and uses several compute kernels to exploit the multiple levels of hardware parallelism. The resulting CUDA code is asynchronous and lock free, employs load balancing, visits each edge exactly once, and only processes edges in one direction. It is 1.8 times faster on average than the fastest prior GPU implementation running on a Titan X and faster on most of the eighteen real-world and synthetic graphs we tested.

Aliaga2018a [340] A. C. José I. Aliaga María Barreda, "Energy-aware strategies for task-parallel sparse linear system solvers," *Concurrency and Computation Practice and Experience*, 2018. DOI: 10.1002/cpe.4633,
Abstract: We present some energy-aware strategies to improve the energy efficiency of a task-parallel preconditioned Conjugate Gradient (PCG) iterative solver on a Haswell-EP Intel Xeon. These techniques leverage the power-saving states of the processor, promoting the hardware into a more energy-efficient C-state and modifying the CPU frequency (P-states of the processors) of some operations of the PCG. We demonstrate that the application of these strategies during the main operations of the iterative solver can reduce its energy consumption considerably.

Jun2018 [341] S. Jun, A. Wright, S. Zhang, S. Xu, and Arvind, "Grafboost: Using accelerated flash storage for external graph analytics," in *ACM/IEEE 45th Annual International Symposium on Computer Architecture*, Jun. 2018, pp. 411–424. DOI: 10.1109/ISCA.2018.00042,
Abstract: We describe GraFBoost, a flash-based architecture with hardware acceleration for external analytics of multi-terabyte graphs. We compare the performance of GraFBoost with 1 GB of DRAM against various state-of-the-art graph analytics software including FlashGraph, running on a 32-thread Xeon server with 128 GB of DRAM. We demonstrate that despite the relatively small amount of DRAM, GraFBoost achieves high performance with very large graphs no other system can handle, and rivals the performance of the fastest software platforms on sizes of graphs that existing platforms can handle. Unlike in-memory and semi-external systems, GraFBoost uses a constant amount of memory for all problems, and its performance decreases very slowly as graph sizes increase, allowing GraFBoost to scale to much larger problems than possible with existing systems while using much less resources on a single-node system. The key component of GraFBoost is the sort-reduce accelerator, which implements a novel method to sequentialize fine-grained random accesses to flash storage. The sort-reduce accelerator logs random update requests and then uses hardware-accelerated external sorting with interleaved reduction functions. GraFBoost also

stores newly updated vertex values generated in each superstep of the algorithm lazily with the old vertex values to further reduce I/O traffic. We evaluate the performance of GraFBoost for PageRank, breadth-first search and betweenness centrality on our FPGA-based prototype (Xilinx VC707 with 1 GB DRAM and 1 TB flash) and compare it to other graph processing systems including a pure software implementation of GrapFBoost.

Kaya2018 [342] O. Kaya, R. Kannan, and G. Ballard, "Partitioning and communication strategies for sparse non-negative matrix factorization," INRIA Bordeaux Sud-Ouest, Tech. Rep., 2018,
Abstract: Non-negative matrix factorization (NMF), the problem of finding two non-negative low-rank factors whose product approximates an input matrix, is a useful tool for many data mining and scientific applications such as topic modeling in text mining and blind source separation in microscopy. In this paper, we focus on scaling algorithms for NMF to very large sparse datasets and massively parallel machines by employing effective algorithms, communication patterns, and partitioning schemes that leverage the sparsity of the input matrix. In the case of machine learning workflow, the computations after SpMM must deal with dense matrices, as Sparse-Dense matrix multiplication will result in a dense matrix. Hence, the partitioning strategy considering only SpMM will result in a huge imbalance in the overall workflow especially on computations after SpMM and in this specific case of NMF on non-negative least squares computations. Towards this, we consider two previous works developed for related problems, one that uses a fine-grained partitioning strategy using a point-to-point communication pattern and on that uses a checkerboard partitioning strategy using a collective-based communication pattern. We show that a combination of the previous approaches balances the demands of the various computations within NMF algorithms and achieves high efficiency and scalability. From the experiments, we could see that our proposed algorithm communicates atleast 4x less than the collective and achieves upto 100× speed up over the baseline FAUN on real world datasets. Our algorithm was experimented in two different super computing platforms and we could scale up to 32000 processors on Bluegene/Q.

Kirmani2018 [343] S. Kirmani, H. Sun, and P. Raghavan, "A scalability and sensitivity study of parallel geometric algorithms for graph partitioning," 2018. DOI: 10.13140/RG.2.2.32020.96644,
Abstract: Graph partitioning arises in many computational simulation workloads, including those that involve finite difference or finite element methods, where partitioning enables efficient parallel processing of the entire simulation. We focus on parallel geometric algorithms for partitioning large graphs whose vertices are associated with coordinates in two-or three-dimensional space on multi-core processors. Compared with other types of partitioning algorithms, geometric schemes generally show better scalability on a large number of processors or cores. This paper studies the scalability and sensitivity of two parallel algorithms, namely, recursive coordinate bisection (denoted by pRCB) and geometric mesh partitioning (denoted by pGMP), in terms of their robustness to several key factors that affect the partition quality, including coordinate perturbation, approximate embedding, mesh quality and graph planarity. Our results indicate that the quality of a partition as measured by the size of the edge separator (or cutsize) remains consistently better for pGMP compared to pRCB. On average for our test suite, relative to pRCB, pGMP yields 25% smaller cutsizes on the original embedding, and across all perturbations cutsizes that are smaller by at least 8% and by as much as 50%. Not surprisingly, higher quality cuts are obtained at the expense of longer execution times; on a single core, pGMP has an average execution time that is almost 10 times slower than that of pRCB, but it scales better and catches up at 32-cores to be slower by less than 20%. With the current trends in core counts that continue to increase per chip, these results suggest that pGMP presents an attractive solution if a modest number of cores can be deployed to reduce execution times while providing high quality partitions.

Knigge2018 [344] T. E. Knigge and R. H. Bisseling. (2018). An improved exact algorithm and an np-completeness proof for sparse matrix bipartitioning. arXiv: arXiv:1811.02043,
Abstract: We formulate the sparse matrix bipartitioning problem of minimizing the communication volume in parallel sparse matrix-vector multiplication. We prove its NP-completeness in the perfectly balanced case, where both parts of the partitioned matrix must have an equal number of nonzeros, by reduction from the graph bisection problem. We present an improved exact branch-and-bound algorithm which finds the minimum communication volume for a given maxi-

mum allowed imbalance. The algorithm is based on a maximum-flow bound and a pack- ing bound, which extend previous matching and packing bounds. We implemented the algorithm in a new program called MP (Matrix Partitioner), which solved 839 matrices from the SuiteSparse collection to optimality, each within 24 hours of CPU-time. Furthermore, MP solved the difficult problem of the matrix cage6 in about 3 days. The new program is about 13.8 times faster than the previous program MondriaanOpt.

Knigge2018a [345] ——, *An improved exact algorithm and an np-completeness proof for sparse matrix bipartitioning*, 2018. eprint: arXiv:1811.02043,
Abstract: We formulate the sparse matrix bipartitioning problem of minimizing the communication volume in parallel sparse matrix-vector multiplication. We prove its NP-completeness in the perfectly balanced case, where both parts of the partitioned matrix must have an equal number of nonzeros, by reduction from the graph bisection problem. We present an improved exact branch-and-bound algorithm which finds the minimum communication volume for a given maximum allowed imbalance. The algorithm is based on a maximum-flow bound and a pack- ing bound, which extend previous matching and packing bounds. We implemented the algorithm in a new program called MP (Matrix Partitioner), which solved 839 matrices from the SuiteSparse collection to optimality, each within 24 hours of CPU-time. Furthermore, MP solved the difficult problem of the matrix cage6 in about 3 days. The new program is about 13.8 times faster than the previous program MondriaanOpt.

kurzak2018 [346] J. Kurzak, M. Gates, I. Yamazaki, A. Charara, A. YarKhan, J. Finney, G. Ragghianti, P. Luszczek, and J. Dongarra, "SLATE working note 8: Linear systems performance report," Innovative Computing Laboratory, University of Tennessee, Tech. Rep. ICL-UT-XX-XX, Sep. 2018, revision 09-2018,
Abstract: Software for Linear Algebra Targeting Exascale (SLATE) is being developed as part of the Exascale Computing Project (ECP), which is a collaborative effort between two US Department of Energy (DOE) organizations, the Office of Science and the National Nuclear Security Administration (NNSA). The purpose of SLATE is to serve as a replacement for ScaLAPACK for the upcoming pre-exascale and exascale DOE machines. SLATE will accomplish this objective by leveraging recent progress in parallel programming models and by strongly focusing on supporting hardware accelerators. This report focuses on the set of SLATE routines that solve linear systems of equations. Specifically, initial performance numbers are reported, alongside ScaLAPACK performance numbers, on the SummitDev machine at the Oak Ridge Leadership Computing Facility (OLCF). More details about the design of the SLATE software infrastructure can be found in the report by Kurzak et al.

Li2018 [347] R. Li, Y. Xi, L. Erlandson, and Y. Saad. (2018). The eigenvalues slicing library (evsl): Algorithms, implementation, and software. arXiv: arXiv:1802.05215,
Abstract: This paper describes a software package called EVSL (for EigenValues Slicing Library) for solving large sparse real symmetric standard and generalized eigenvalue problems. As its name indicates, the package exploits spectrum slicing, a strategy that consists of dividing the spectrum into a number of subintervals and extracting eigenpairs from each subinterval independently. In order to enable such a strategy, the methods implemented in EVSL rely on a quick calculation of the spectral density of a given matrix, or a matrix pair. What distinguishes EVSL from other currently available packages is that EVSL relies entirely on filtering techniques. Polynomial and rational filtering are both implemented and are coupled with Krylov subspace methods and the subspace iteration algorithm. On the implementation side, the package offers interfaces for various scenarios including matrix-free modes, whereby the user can supply his/her own functions to perform matrix-vector operations or to solve sparse linear systems. The paper describes the algorithms in EVSL, provides details on their implementations, and discusses performance issues for the various methods.

Li2018a [348] ——, "The eigenvalues slicing library (evsl): Algorithms, implementation, and software," 2018. eprint: arXiv:1802.05215,

Abstract: This paper describes a software package called EVSL (for EigenValues Slicing Library) for solving large sparse real symmetric standard and generalized eigenvalue problems. As its name indicates, the package exploits spectrum slicing, a strategy that consists of dividing the spectrum into a number of subintervals and extracting eigenpairs from each subinterval independently. In order to enable such a strategy, the methods implemented in EVSL rely on a quick calculation of the spectral density of a given matrix, or a matrix pair. What distinguishes EVSL from other currently available packages is that EVSL relies entirely on filtering techniques. Polynomial and rational filtering are both implemented and are coupled with Krylov subspace methods and the subspace iteration algorithm. On the implementation side, the package offers interfaces for various scenarios including matrix-free modes, whereby the user can supply his/her own functions to perform matrix-vector operations or to solve sparse linear systems. The paper describes the algorithms in EVSL, provides details on their implementations, and discusses performance issues for the various methods.

Loncaric2018 [349] C. Loncaric, M. D. Ernst, and E. Torlak, "Generalized data structure synthesis," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE'18, Gothenburg, Sweden: ACM, 2018, pp. 958–968, ISBN: 978-1-4503-5638-1. DOI: 10.1145/3180155.3180211. [Online]. Available: http://doi.acm.org/10.1145/3180155.3180211,
Abstract: Data structure synthesis is the task of generating data structure implementations from high-level specifications. Recent work in this area has shown potential to save programmer time and reduce the risk of defects. Existing techniques focus on data structures for manipulating subsets of a single collection, but real-world programs often track multiple related collections and aggregate properties such as sums, counts, minimums, and maximums.
This paper shows how to synthesize data structures that track subsets and aggregations of multiple related collections. Our technique decomposes the synthesis task into alternating steps of query synthesis and incrementalization. The query synthesis step implements pure operations over the data structure state by leveraging existing enumerative synthesis techniques, specialized to the data structures domain. The incrementalization step implements imperative state modifications by re-framing them as fresh queries that determine what to change, coupled with a small amount of code to apply the change. As an added benefit of this approach over previous work, the synthesized data structure is optimized for not only the queries in the specification but also the required update operations. We have evaluated our approach in four large case studies, demonstrating that these extensions are broadly applicable.

Miyata2018 [350] T. Miyata, "On correction-based iterative methods for eigenvalue problems," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E101.A, no. 10, pp. 1668–1675, 2018. DOI: 10.1587/transfun.E101.A.1668,
Abstract: The Jacobi-Davidson method and the Riccati method for eigenvalue problems are studied. In the methods, one has to solve a nonlinear equation called the correction equation per iteration, and the difference between the methods comes from how to solve the equation. In the Jacobi-Davidson/Riccati method the correction equation is solved with/without linearization. In the literature, avoiding the linearization is known as an improvement to get a better solution of the equation and bring the faster convergence. In fact, the Riccati method showed superior convergence behavior for some problems. Nevertheless the advantage of the Riccati method is still unclear, because the correction equation is solved not exactly but with low accuracy. In this paper, we analyzed the approximate solution of the correction equation and clarified the point that the Riccati method is specialized for computing particular solutions of eigenvalue problems. The result suggests that the two methods should be selectively used depending on target solutions. Our analysis was verified by numerical experiments.

Mohammad2018 [351] H. Mohammad, M. Y. Waziri, and S. A. Santos, "A brief survey of methods for solving nonlinear least-squares problems," *Numerical Algebra, Control & Optimization*, vol. 9, no. 2155-3289_2019_1_1, p. 1, 2018, ISSN: 2155-3289. DOI: 10.3934/naco.2019001. [Online]. Available: http://aimsciences.org//article/id/7e25fb2d-b50c-46dd-9be5-23deee2b4242,

Abstract: In this paper, we present a brief survey of methods for solving nonlinear least-squares problems. We pay specific attention to methods that take into account the special structure of the problems. Most of the methods discussed belong to the quasi-Newton family (i.e. the structured quasi-Newton methods (SQN)). Our survey comprises some of the traditional and modern developed methods for nonlinear least-squares problems. At the end, we suggest a few topics for further research.

Mohammadi2018 [352] M. S. Mohammadi, K. Cheshmi, G. Gopalakrishnan, M. W. Hall, M. M. Dehnavi, A. Venkat, T. Yuki, and M. M. Strout. (2018). Sparse matrix code dependence analysis simplification at compile time. arXiv: arXiv:1807.10852,
Abstract: Analyzing array-based computations to determine data dependences is useful for many applications including automatic parallelization, race detection, computation and communication overlap, verification, and shape analysis. For sparse matrix codes, array data dependence analysis is made more difficult by the use of index arrays that make it possible to store only the nonzero entries of the matrix (e.g., in A[B[i]], B is an index array). Here, dependence analysis is often stymied by such indirect array accesses due to the values of the index array not being available at compile time. Consequently, many dependences cannot be proven unsatisfiable or determined until runtime. Nonetheless, index arrays in sparse matrix codes often have properties such as monotonicity of index array elements that can be exploited to reduce the amount of runtime analysis needed. In this paper, we contribute a formulation of array data dependence analysis that includes encoding index array properties as universally quantified constraints. This makes it possible to leverage existing SMT solvers to determine whether such dependences are unsatisfiable and significantly reduces the number of dependences that require runtime analysis in a set of eight sparse matrix kernels. Another contribution is an algorithm for simplifying the remaining satisfiable data dependences by discovering equalities and/or subset relationships. These simplifications are essential to make a runtime-inspection-based approach feasible.

Mohammadi2018a [353] ——, "Sparse matrix code dependence analysis simplification at compile time," *CoRR*, vol. abs/1807.10852, 2018. arXiv: 1807.10852. [Online]. Available: http://arxiv.org/abs/1807.10852,
Abstract: Analyzing array-based computations to determine data dependences is useful for many applications including automatic parallelization, race detection, computation and communication overlap, verification, and shape analysis. For sparse matrix codes, array data dependence analysis is made more difficult by the use of index arrays that make it possible to store only the nonzero entries of the matrix (e.g., in A[B[i]], B is an index array). Here, dependence analysis is often stymied by such indirect array accesses due to the values of the index array not being available at compile time. Consequently, many dependences cannot be proven unsatisfiable or determined until runtime. Nonetheless, index arrays in sparse matrix codes often have properties such as monotonicity of index array elements that can be exploited to reduce the amount of runtime analysis needed. In this paper, we contribute a formulation of array data dependence analysis that includes encoding index array properties as universally quantified constraints. This makes it possible to leverage existing SMT solvers to determine whether such dependences are unsatisfiable and significantly reduces the number of dependences that require runtime analysis in a set of eight sparse matrix kernels. Another contribution is an algorithm for simplifying the remaining satisfiable data dependences by discovering equalities and/or subset relationships. These simplifications are essential to make a runtime-inspection-based approach feasible.

Nagasaka2018 [354] Y. Nagasaka, S. Matsuoka, A. Azad, and A. Buluç, "High-performance sparse matrix-matrix products on intel knl and multicore architectures," in *Proceedings of the 47th International Conference on Parallel Processing (Workshops)*, ser. ICPPW '18, 2018. arXiv: arXiv:1804.01698,
Abstract: Sparse matrix-matrix multiplication (SpGEMM) is a computational primitive that is widely used in areas ranging from traditional numerical applications to recent big data analysis and machine learning. Although many SpGEMM algorithms have been proposed, hardware specific optimizations for multi- and many-core processors are lacking and a detailed analysis of their performance under various use cases and matrices is not available. We firstly identify and

mitigate multiple bottlenecks with memory management and thread scheduling on Intel Xeon Phi (Knights Landing or KNL). Specifically targeting multi- and many-core processors, we develop a hash-table-based algorithm and optimize a heap-based shared-memory SpGEMM algorithm. We examine their performance together with other publicly available codes. Different from the literature, our evaluation also includes use cases that are representative of real graph algorithms, such as multi-source breadth-first search or triangle counting. Our hash-table and heap-based algorithms are showing significant speedups from libraries in the majority of the cases while different algorithms dominate the other scenarios with different matrix size, sparsity, compression factor and operation type. We wrap up in-depth evaluation results and make a recipe to give the best SpGEMM algorithm for target scenario. A critical finding is that hash-table-based SpGEMM gets a significant performance boost if the nonzeros are not required to be sorted within each row of the output matrix.

Neumann2018 [355] C. Neumann and O. Stein, "Generating feasible points for mixed-integer convex optimization problems by inner parallel cuts," *Optimization Online*, 2018, Preprint ID 2018-11-6947,
Abstract: In this article we introduce an inner parallel cutting plane method (IPCP) to compute good feasible points for mixed-integer convex optimization problems. The method iteratively generates polyhedral outer approximations of an enlarged inner parallel set (EIPS) of the continuously relaxed feasible set. This EIPS possesses the crucial property that any rounding of any of its elements is feasible for the original problem. The outer approximations are refined in each iteration by using modified Kelley cutting planes, which are defined via rounded optimal points of linear opti- mization problems (LPs).
We show that the method either computes a feasible point or certifies that the EIPS is empty. Moreover, we provide bounds on the objective value of the generated feasible point. As there exist consistent problems which possess an empty EIPS, the IPCP is not guaranteed to find a feasible point for the latter. Yet, the crucial advantage of the method lies in the complexity of each iteration: While other approaches need to solve a mixed-integer linear optimization problem, the IPCP only needs to solve an LP, which can be carried out efficiently. Our computational study indicates that the IPCP is able to quickly find feasible points for many practical applications. It further demonstrates that the objective values of the computed feasible points are generally of good quality and sometimes not easily obtainable by other methods.

Rong2018 [356] H. Rong. (2018). Expressing sparse matrix computations for productive performance on spatial architectures. arXiv: arXiv:1810.07517,
Abstract: This paper addresses spatial programming of sparse matrix computations for productive performance. The challenge is how to express an irregular computation and its optimizations in a regular way. A sparse matrix has (non-zero) values and a structure. In this paper, we propose to classify the implementations of a computation on a sparse matrix into two categories: (1) structure-driven, or top-down, approach, which traverses the structure with given row and column indices and locates the corresponding values, and (2) values-driven, or bottom-up, approach, which loads and processes the values in parallel streams, and decodes the structure for the values' corresponding row and column indices. On a spatial architecture like FPGAs, the values-driven approach is the norm. We show how to express a sparse matrix computation and its optimizations for a values-driven implementation. A compiler automatically synthesizes a code to decode the structure. In this way, programmers focus on optimizing the processing of the values, using familiar optimizations for dense matrices, while leaving the complex, irregular structure traversal to an automatic compiler. We also attempt to regularize the optimizations of the reduction for a dynamic number of values, which is common in a sparse matrix computation.

Rong2018a [357] ——, *Expressing sparse matrix computations for productive performance on spatial architectures*, 2018. eprint: arXiv:1810.07517,
Abstract: This paper addresses spatial programming of sparse matrix computations for productive performance. The challenge is how to express an irregular computation and its optimizations in a regular way. A sparse matrix has (non-zero) values and a structure. In this paper, we propose to classify the implementations of a computation on a sparse matrix into two categories: (1) structure-driven, or top-down, approach, which traverses the structure with given row and column indices and

locates the corresponding values, and (2) values-driven, or bottom-up, approach, which loads and processes the values in parallel streams, and decodes the structure for the values' corresponding row and column indices. On a spatial architecture like FPGAs, the values-driven approach is the norm. We show how to express a sparse matrix computation and its optimizations for a values-driven implementation. A compiler automatically synthesizes a code to decode the structure. In this way, programmers focus on optimizing the processing of the values, using familiar optimizations for dense matrices, while leaving the complex, irregular structure traversal to an automatic compiler. We also attempt to regularize the optimizations of the reduction for a dynamic number of values, which is common in a sparse matrix computation.

Schlag2018 [358] S. Schlag, C. Schulz, D. Seemaier, and D. Strash. (2018). Scalable edge partitioning. eprint: arXiv:1808.06411,
Abstract: Edge-centric distributed computations have appeared as a recent technique to improve the shortcomings of think-like-a-vertex algorithms on large scale-free networks. In order to increase parallelism on this model, edge partitioning - partitioning edges into roughly equally sized blocks - has emerged as an alternative to traditional (node-based) graph partitioning. In this work, we give a distributed memory parallel algorithm to compute high-quality edge partitions in a scalable way. Our algorithm scales to networks with billions of edges, and runs efficiently on thousands of PEs. Our technique is based on a fast parallelization of split graph construction and a use of advanced node partitioning algorithms. Our extensive experiments show that our algorithm has high quality on large real-world networks and large hyperbolic random graphs, which have a power law degree distribution and are therefore specifically targeted by edge partitioning.

Cayrols2018 [359] F. L. Sébastien Cayrols Iain Duff, "Parallelization of the solve phase in a task-based cholesky solver using a sequential task flow model," Science & Technology Facilities Council, UK, Technical Report, 2018,
Abstract: We describe the parallelization of the solve phase in the sparse Cholesky solver SpLLT [Duff, Hogg, and Lopez. Numerical Algebra, Control and Optimization. Volume 8, 235-237, 2018] when using a sequential task flow (STF) model. In the context of direct methods, the solution of a sparse linear system is achieved through three main phases: the analyse, the factorization and the solve phases. In the last two phases which involve numerical computation, the factorization corresponds to the most computationally costly phase, and it is therefore crucial to parallelize this phase in order to reduce the time-to-solution on modern architectures. As a consequence, the solve phase is often not as optimized as the factorization in state-of-the-art solvers and opportunities for parallelism are often not exploited in this phase. However, in some applications, the time spent in the solve phase is comparable or even greater than the time for the factorization and the user could dramatically benefit from a faster solve routine. This is the case, for example, for a CG solver using a block Jacobi preconditioner. The diagonal blocks are factorized once only but their factors are used to solve subsystems at each CG iteration. In this study we design and implement a parallel version of a task-based solve routine for an OpenMP version of the SpLLT solver. We show that we can obtain good scalability on a multicore architecture enabling a dramatic reduction of the overall time-to-solution in some applications.

Simpson2018 [360] T. Simpson, D. Dimosthenis Pasadakis ad Kourounis, K. Fujita, T. Yamaguchi, I. Tsuyoshi, and O. Schenk, "Balanced graph partition refinement using the graph p-laplacian," in *Proceedings of the ACM Platform for Advanced Scientific Computing Conference*, ser. PASC '18, Jun. 2018. DOI: 10.1145/3218176.3218232,
Abstract: A continuous formulation of the optimal 2-way graph partitioning based on the p-norm minimization of the graph Laplacian Rayleigh quotient is presented, which provides a sharp approximation to the balanced graph partitioning problem, the optimality of which is known to be NP-hard. The minimization is initialized from a cut provided by a state-of-the-art multilevel recursive bisection algorithm, and then a continuation approach reduces the p-norm from a 2-norm towards a 1-norm, employing for each value of p a feasibility-preserving steepest-descent method that converges on the p-Laplacian eigenvector. A filter favors iterates advancing towards minimum edgecut and partition load imbalance. The complexity of the suggested approach is linear in graph edges. The simplicity of the steepest-descent algorithm renders the overall approach highly scalable

and efficient in parallel distributed architectures. Parallel implementation of recursive bisection on multi-core CPUs and GPUs are presented for large-scale graphs with up to 1.9 billion tetrahedra. The suggested approach exhibits improvements of up to 52.8% over METIS for graphs originating from triangular Delaunay meshes, 34.7% over METIS and 21.9% over KaHIP for power network graphs, 40.8% over METIS and 20.6% over KaHIP for sparse matrix graphs, and finally 93.2% over METIS for graphs emerging from social networks.

Stoltzfus2018 [361] L. Stoltzfus, M. Emani, P.-H. Lin, and C. Liao, "Data placement optimization in gpu memory hierarchy using predictive modeling," in *Proceedings of the Workshop on Memory Centric High Performance Computing*, ser. MCHPC'18, Dallas, TX, USA: ACM, 2018, pp. 45–49, ISBN: 978-1-4503-6113-2. DOI: 10.1145/3286475.3286482. [Online]. Available: http://doi.acm.org/10.1145/3286475.3286482,

Abstract: Modern supercomputers often use Graphic Processing Units (or GPUs) to meet the ever-growing demands for high performance computing. GPUs typically have a complex memory architecture with various types of memories and caches, such as global memory, shared memory, constant memory, and texture memory. The placement of data on these memories has a tremendous impact on the performance of the HPC applications and identifying the optimal placement location is non-trivial.

In this paper, we propose a machine learning-based approach to build a classifier to determine the best class of GPU memory that will minimize GPU kernel execution time. This approach utilizes a set of performance counters obtained from profiling runs along with hardware features to generate the trained model. We evaluate our approach on several generations of NVIDIA GPUs, including Kepler, Maxwell, Pascal, and Volta on a set of benchmarks. The results show that the trained model achieves prediction accuracy over 90% and given a global version, the classifier can accurately determine which data placement variant would yield the best performance.

Su2018 [362] X. Su, X. Liao, H. Jiang, C. Yang, and J. Xue, "SCP: Shared cache partitioning for high-performance GEMM," *ACM Transactions on Architecture and Code Optimization*, TACO, vol. 15, no. 4, 43:1–43:21, Oct. 2018, ISSN: 1544-3566. DOI: 10.1145/3274654,

Abstract: GEneral Matrix Multiply (GEMM) is the most fundamental computational kernel routine in the BLAS library. To achieve high performance, in-memory data must be prefetched into fast on-chip caches before they are used. Two techniques, software prefetching and data packing, have been used to effectively exploit the capability of on-chip least recent used (LRU) caches, which are popular in traditional high-performance processors used in high-end servers and supercomputers. However, the market has recently witnessed a new diversity in processor design, resulting in high-performance processors equipped with shared caches with non-LRU replacement policies. This poses a challenge to the development of high-performance GEMM in a multithreaded context. As several threads try to load data into a shared cache simultaneously, interthread cache conflicts will increase significantly. We present a Shared Cache Partitioning (SCP) method to eliminate interthread cache conflicts in the GEMM routines, by partitioning a shared cache into physically disjoint sets and assigning different sets to different threads. We have implemented SCP in the OpenBLAS library and evaluated it on Phytium 2000+, a 64-core AArch64 processor with private LRU L1 caches and shared pseudo-random L2 caches (per four-core cluster). Our evaluation shows that SCP has effectively reduced the conflict misses in both L1 and L2 caches in a highly optimized GEMM implementation, resulting in an improvement of its performance by 2.75% to 6.91%.

Sun2018 [363] X. Sun, K. Wei, L. Lai, S. Tsai, and C. Wu, "Optimizing sparse matrix-vector multiplication on GPUs via index compression," in *Proceedings of the 3rd IEEE Advanced Information Technology, Electronic and Automation Control Conference*, ser. IAEAC, Oct. 2018, pp. 598–602. DOI: 10.1109/IAEAC.2018.8577693,

Abstract: Sparse matrix-vector multiplication (SpMV) as one of the most significant scientific kernels has been widely used in many scientific disciplines. In practical applications, large-scale spare matrices are usually used for calculation. During these years, Graphic Processing Unit (GPU) has become a powerful platform for high-performance computing, and optimizing$S$pMV on GPU based systems for efficient performance is the principal interest in many researches. In this paper, we proposed a new method to optimize SpMV on GPUs via index compression. Our index compression

method can reduce the index value of the access space. The memory space for recording each column index is significantly reduced from two bytes to one byte, which outperforms the previous work on access performance. The main contributions we make are as follows: (1) Only one byte for each column index is required, which can significantly reduce the working set of the column index and further improve the cache hit ration. (2) Our method can be applied to any kind of matrices, while the previous work can only apply to subset of the matrices. Computational experiments on problems according to the previous work reveal that the best performance improvement ration for ours is up to about 1.5.

Tan2018 [364] G. Tan, J. Liu, and J. Li, "Design and implementation of adaptive spmv library for multicore and many-core architecture," *ACM Trans. Math. Softw.*, vol. 44, no. 4, 46:1–46:25, Aug. 2018, ISSN: 0098-3500. DOI: 10.1145/3218823. [Online]. Available: http://doi.acm.org/10.1145/3218823,

Abstract: Sparse matrix vector multiplication (SpMV) is an important computational kernel in traditional high-performance computing and emerging data-intensive applications. Previous SpMV libraries are optimized by either application-specific or architecture-specific approaches but present difficulties for use in real applications. In this work, we develop an auto-tuning system (SMATER) to bridge the gap between specific optimizations and general-purpose use. SMATER provides programmers a unified interface based on the compressed sparse row (CSR) sparse matrix format by implicitly choosing the best format and fastest implementation for any input sparse matrix during runtime. SMATER leverages a machine-learning model and retargetable back-end library to quickly predict the optimal combination. Performance parameters are extracted from 2,386 matrices in the SuiteSparse matrix collection. The experiments show that SMATER achieves good performance (up to 10 times that of the Intel Math Kernel Library (MKL) on Intel E5-2680 v3) while being portable on state-of-the-art x86 multicore processors, NVIDIA GPUs, and Intel Xeon Phi accelerators. Compared with the Intel MKL library, SMATER runs faster by more than 2.5 times on average. We further demonstrate its adaptivity in an algebraic multigrid solver from the Hypre library and report greater than 20% performance improvement.

Tavernier2018 [365] J. Tavernier, J. Simm, K. Meerbergen, and Y. Moreau. (2018). Multilevel preconditioning for ridge regression. arXiv: arXiv:1806.05826,

Abstract: Solving linear systems is often the computational bottleneck in real-life problems. Iterative solvers are the only option due to the complexity of direct algorithms or because the system matrix is not explicitly known. Here, we develop a multilevel preconditioner for regularized least squares linear systems involving a feature or data matrix. Variants of this linear system may appear in machine learning applications, such as ridge regression, logistic regression, support vector machines and matrix factorization with side information. We use clustering algorithms to create coarser levels that preserve the principal components of the covariance or Gram matrix. These coarser levels approximate the dominant eigenvectors and are used to build a multilevel preconditioner accelerating the Conjugate Gradient method. We observed speed-ups for artificial and real-life data. For a specific data set, we achieved speed-up up to a factor 100.

Tavernier2018a [366] ——, "Multilevel preconditioning for ridge regression," 2018. eprint: arXiv:1806.05826,

Abstract: Solving linear systems is often the computational bottleneck in real-life problems. Iterative solvers are the only option due to the complexity of direct algorithms or because the system matrix is not explicitly known. Here, we develop a multilevel preconditioner for regularized least squares linear systems involving a feature or data matrix. Variants of this linear system may appear in machine learning applications, such as ridge regression, logistic regression, support vector machines and matrix factorization with side information. We use clustering algorithms to create coarser levels that preserve the principal components of the covariance or Gram matrix. These coarser levels approximate the dominant eigenvectors and are used to build a multilevel preconditioner accelerating the Conjugate Gradient method. We observed speed-ups for artificial and real-life data. For a specific data set, we achieved speed-up up to a factor 100.

Wang2018 [367] X. Wang, W. Liu, W. Xue, and L. Wu, "Swsptrsv: A fast sparse triangular solve with sparse level tile layout on sunway architectures," in *Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '18, Vienna, Aus-

tria: ACM, 2018, pp. 338–353, ISBN: 978-1-4503-4982-6. DOI: 10.1145/3178487.3178513. [Online]. Available: http://doi.acm.org/10.1145/3178487.3178513,
Abstract: Sparse triangular solve (SpTRSV) is one of the most important kernels in many real-world applications. Currently, much research on parallel SpTRSV focuses on level-set construction for reducing the number of inter-level synchronizations. However, the out-of-control data reuse and high cost for global memory or shared cache access in inter-level synchronization have been largely neglected in existing work.
In this paper, we propose a novel data layout called Sparse Level Tile to make all data reuse under control, and design a Producer-Consumer pairing method to make any inter-level synchronization only happen in very fast register communication. We implement our data layout and algorithms on an SW26010 many-core processor, which is the main building-block of the current world fastest supercomputer Sunway Taihulight. The experimental results of testing all 2057 square matrices from the Florida Matrix Collection show that our method achieves an average speedup of 6.9 and the best speedup of 38.5 over parallel level-set method. Our method also outperforms the latest methods on a KNC many-core processor in 1856 matrices and the latest methods on a K80 GPU in 1672 matrices, respectively.

Wang2018a [368] X. Wang, P. Xu, W. Xue, Y. Ao, C. Yang, H. Fu, L. Gan, G. Yang, and W. Zheng, "A fast sparse triangular solver for structured-grid problems on sunway many-core processor sw26010," in *Proceedings of the 47th International Conference on Parallel Processing*, ser. ICPP 2018, Eugene, OR, USA: ACM, 2018, 53:1–53:11, ISBN: 978-1-4503-6510-9. DOI: 10.1145/3225058.3225071. [Online]. Available: http://doi.acm.org/10.1145/3225058.3225071,
Abstract: The sparse triangular solver (SpTRSV) is one of the most essential kernels in many scientific and engineering applications. Efficiently parallelizing the SpTRSV on modern many-core architectures is considerably difficult due to inherent dependency of computation and discontinuous memory accesses. Achieving high performance of SpTRSV is even more challenging for SW26010, the new-generation customized heterogeneous many-core processor equipped in the top-rank Sunway TaihuLight supercomputer. Owing to regular sparse pattern, structured-grid triangular problems show much different computing characteristics with general ones as well as new opportunities to algorithm design on many-core architectures, which ever lacks attention. In this work, we focus on how to design and implement fast SpTRSV for structured-grid problems on SW26010. A generalized algorithm framework of parallel SpTRSV is proposed for best utilization of the features and flexibilities of SW26010 many-core architecture according to the fine-grained Producer-Consumer model. Moreover, a novel parallel structured-grid SpTRSV is presented by using direct data transfers across registers of the computing elements of SW26010. Experiments on four typical structured-grid triangular problems with different problem sizes demonstrate that our SpTRSV can achieve an average momory bandwidth utilization of 79.7% according to the stream benchmark, which leads to a speedup of 17.7 over serial version on SW26010. Furthermore, experiments with real world sparse linear problems show that our proposed SpTRSV can achieve superior preconditioning performance over the Intel Xeon E5-2670 v3 CPU and Intel Xeon Phi 7210 KNL over DDR4 memory.

Yang2018 [369] C. Yang, "Linear algebra is the right way to think about graphs," in *Proceedings of the 2018 ACM/IEEE Supercomputing Conference*, ser. SC'18, 2018,
Abstract: Graph algorithms are challenging to implement on new accelerators such as GPUs. To address this problem, GraphBLAS is an innovative on-going effort by the graph analytics community to formulate graph algorithms as sparse linear algebra, so that they can be expressed in a performant, succinct and in a backend-agnostic manner. Initial research efforts in implementing GraphBLAS on GPUs for graph processing and analytics have been promising, but challenges such as feature-incompleteness and poor performance still exist compared to their vertex-centric ("think like a vertex") graph framework counterparts. For our thesis, we propose a multi-language graph framework aiming to simplify the development of graph algorithms, which 1) provides a multi-language GraphBLAS interface for the end-users to express, develop, and refine graph algorithms more succinctly than existing distributed graph frameworks; 2) abstracts away from the end-users

performance tuning decisions; 3) utilizes the advantages of existing low-level GPU computing primitives to maintain high performance.

Yang2018b [370] C. Yang, A. Buluc, and J. D. Owens. (2018). Design principles for sparse matrix multiplication on the gpu. arXiv: arXiv:1803.08601,
Abstract: We implement two novel algorithms for sparse-matrix dense-matrix multiplication (SpMM) on the GPU. Our algorithms expect the sparse input in the popular compressed-sparse-row (CSR) format and thus do not require expensive format conversion. While previous SpMM work concentrates on thread-level parallelism, we additionally focus on latency hiding with instruction-level parallelism and load-balancing. We show, both theoretically and experimentally, that the proposed SpMM is a better fit for the GPU than previous approaches. We identify a key memory access pattern that allows efficient access into both input and output matrices that is crucial to getting excellent performance on SpMM. By combining these two ingredients – (i) merge-based load-balancing and (ii) row-major coalesced memory access – we demonstrate a 3.6x peak speedup and a 23.5% geomean speedup over state-of-the-art SpMM implementations on real-world datasets.

Yang2018c [371] ——, "Design principles for sparse matrix multiplication on the gpu," Mar. 2018,
Abstract: We implement two novel algorithms for sparse-matrix dense-matrix multiplication (SpMM) on the GPU. Our algorithms expect the sparse input in the popular compressed-sparse-row (CSR) format and thus do not require expensive format conversion. While previous SpMM work concentrates on thread-level parallelism, we additionally focus on latency hiding with instruction-level parallelism and load-balancing. We show, both theoretically and experimentally, that the proposed SpMM is a better fit for the GPU than previous approaches. We identify a key memory access pattern that allows efficient access into both input and output matrices that is crucial to getting excellent performance on SpMM. By combining these two ingredients – (i) merge-based load-balancing and (ii) row-major coalesced memory access – we demonstrate a 3.6x peak speedup and a 23.5% geomean speedup over state-of-the-art SpMM implementations on real-world datasets.

Yang2018a [372] C. Yang, A. Buluç, and J. D. Owens. (2018). Implementing push-pull efficiently in graphblas. arXiv: arXiv:1804.03327,
Abstract: We factor Beamer's push-pull, also known as direction-optimized breadth-first-search (DOBFS) into 3 separable optimizations, and analyze them for generalizability, asymptotic speedup, and contribution to overall speedup. We demonstrate that masking is critical for high performance and can be generalized to all graph algorithms where the sparsity pattern of the output is known a priori. We show that these graph algorithm optimizations, which together constitute DOBFS, can be neatly and separably described using linear algebra and can be expressed in the GraphBLAS linear-algebra-based framework. We provide experimental evidence that with these optimizations, a DOBFS expressed in a linear-algebra-based graph framework attains competitive performance with state-of-the-art graph frameworks on the GPU and on a multi-threaded CPU, achieving 101 GTEPS on a Scale 22 RMAT graph.

Yang2018d [373] C. Yang, A. Buluç, and J. D. Owens, "Implementing push-pull efficiently in graphblas," *CoRR*, vol. abs/1804.03327, 2018. arXiv: 1804.03327. [Online]. Available: http://arxiv.org/abs/1804.03327,
Abstract: We factor Beamer's push-pull, also known as direction-optimized breadth-first-search (DOBFS) into 3 separable optimizations, and analyze them for generalizability, asymptotic speedup, and contribution to overall speedup. We demonstrate that masking is critical for high performance and can be generalized to all graph algorithms where the sparsity pattern of the output is known a priori. We show that these graph algorithm optimizations, which together constitute DOBFS, can be neatly and separably described using linear algebra and can be expressed in the GraphBLAS linear-algebra-based framework. We provide experimental evidence that with these optimizations, a DOBFS expressed in a linear-algebra-based graph framework attains competitive performance with state-of-the-art graph frameworks on the GPU and on a multi-threaded CPU, achieving 101 GTEPS on a Scale 22 RMAT graph.

Zhang2018 [374] Y. Zhang, M. Yang, R. Baghdadi, S. Kamil, J. Shun, and S. Amarasinghe, "Graphit: A high-

performance graph dsl," *Proceedings of the ACM Conference on Programming Languages*, OOPSLA'18, vol. 2, 121:1–121:30, Oct. 2018, ISSN: 2475-1421. DOI: 10.1145/3276491,
Abstract: The performance bottlenecks of graph applications depend not only on the algorithm and the underlying hardware, but also on the size and structure of the input graph. As a result, programmers must try different combinations of a large set of techniques, which make tradeoffs among locality, work-efficiency, and parallelism, to develop the best implementation for a specific algorithm and type of graph. Existing graph frameworks and domain specific languages (DSLs) lack flexibility, supporting only a limited set of optimizations.
This paper introduces GraphIt, a new DSL for graph computations that generates fast implementations for algorithms with different performance characteristics running on graphs with different sizes and structures. GraphIt separates what is computed (algorithm) from how it is computed (schedule). Programmers specify the algorithm using an algorithm language, and performance optimizations are specified using a separate scheduling language. The algorithm language simplifies expressing the algorithms, while exposing opportunities for optimizations. We formulate graph optimizations, including edge traversal direction, data layout, parallelization, cache, NUMA, and kernel fusion optimizations, as tradeoffs among locality, parallelism, and work-efficiency. The scheduling language enables programmers to easily search through this complicated tradeoff space by composing together a large set of edge traversal, vertex data layout, and program structure optimizations. The separation of algorithm and schedule also enables us to build an autotuner on top of GraphIt to automatically find high-performance schedules. The compiler uses a new scheduling representation, the graph iteration space, to model, compose, and ensure the validity of the large number of optimizations. We evaluate GraphIt's performance with seven algorithms on graphs with different structures and sizes. GraphIt outperforms the next fastest of six state-of-the-art shared-memory frameworks (Ligra, Green-Marl, GraphMat, Galois, Gemini, and Grazelle) on 24 out of 32 experiments by up to 4.8×, and is never more than 43% slower than the fastest framework on the other experiments. GraphIt also reduces the lines of code by up to an order of magnitude compared to the next fastest framework.

Hassan2019 [375] A. Abdullahi Hassan, V. Cardellini, P. D'Ambra, D. di Serafino, and S. Filippone, "Efficient algebraic multigrid preconditioners on clusters of GPUs," *Parallel Processing Letters*, vol. 29, no. 01, p. 1 950 001, 2019. DOI: 10.1142/S0129626419500014,
Abstract: Many scientific applications require the solution of large and sparse linear systems of equations using Krylov subspace methods; in this case, the choice of an effective preconditioner may be crucial for the convergence of the Krylov solver. Algebraic MultiGrid (AMG) methods are widely used as preconditioners, because of their optimal computational cost and their algorithmic scalability. The wide availability of GPUs, now found in many of the fastest supercomputers, poses the problem of implementing efficiently these methods on high-throughput processors. In this work we focus on the application phase of AMG preconditioners, and in particular on the choice and implementation of smoothers and coarsest-level solvers capable of exploiting the computational power of clusters of GPUs. We consider block-Jacobi smoothers using sparse approximate inverses in the solve phase associated with the local blocks. The choice of approximate inverses instead of sparse matrix factorizations is driven by the large amount of parallelism exposed by the matrix-vector product as compared to the solution of large triangular systems on GPUs. The selected smoothers and solvers are implemented within the AMG preconditioning framework provided by the MLD2P4 library, using suitable sparse matrix data structures from the PSBLAS library. Their behaviour is illustrated in terms of execution speed and scalability, on a test case concerning groundwater modelling, provided by the Jülich Supercomputing Center within the Horizon 2020 Project EoCoE.

Aliaga2019 [376] J. I. Aliaga, E. Dufrechou, P. Ezzatti, and E. S. Quintana-Ortí, "Accelerating the task/data-parallel version of ilupack's bicg in multi-cpu/gpu configurations," *Parallel Computing*, 2019, ISSN: 0167-8191. DOI: 10.1016/j.parco.2019.02.005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167819118301777,
Abstract: ILUPACK is a valuable tool for the solution of sparse linear systems via iterative Krylov subspace-based methods. Its relevance for the solution of real problems has motivated several

efforts to enhance its performance on parallel machines. In this work we focus on exploiting the task-level parallelism derived from the structure of the BiCG method, in addition to the data-level parallelism of the internal matrix computations, with the goal of boosting the performance of a GPU (graphics processing unit) implementation of this solver. First, we revisit the use of dual-GPU systems to execute independent stages of the BiCG concurrently on both accelerators, while leveraging the extra memory space to improve the data access patterns. In addition, we extend our ideas to compute the BiCG method efficiently in multicore platforms with a single GPU. In this line, we study the possibilities offered by hybrid CPU-GPU computations, as well as a novel synchronization-free sparse triangular linear solver. The experimental results with the new solvers show important acceleration factors with respect to the previous data-parallel CPU and GPU versions.

Anzt2019a [377] H. Anzt, Y.-C. Chen, T. Cojean, J. Dongarra, G. Flegar, P. Nayak, E. S. Quintana-Ortí, Y. M. Tsai, and W. Wang, "Towards continuous benchmarking: An automated performance evaluation framework for high performance software," in *Proceedings of the Platform for Advanced Scientific Computing Conference*, ser. PASC '19, Zurich, Switzerland: ACM, 2019, pp. 1–11, ISBN: 978-1-4503-6770-7. DOI: 10.1145/3324989.3325719,

Abstract: We present an automated performance evaluation framework that enables an automated workflow for testing and performance evaluation of software libraries. Integrating this component into an ecosystem enables sustainable software development, as a community effort, via a web application for interactively evaluating the performance of individual software components. The performance evaluation tool is based exclusively on web technologies, which removes the burden of downloading performance data or installing additional software. We employ this framework for the Ginkgo software ecosystem, but the framework can be used with essentially any software project, including the comparison between different software libraries. The Continuous Integration (CI) framework of Ginkgo is also extended to automatically run a benchmark suite on predetermined HPC systems, store the state of the machine and the environment along with the compiled binaries, and collect results in a publicly accessible performance data repository based on Git. The Ginkgo performance explorer (GPE) can be used to retrieve the performance data from the repository, and visualizes it in a web browser. GPE also implements an interface that allows users to write scripts, archived in a Git repository, to extract particular data, compute particular metrics, and visualize them in many different formats (as specified by the script). The combination of these approaches creates a workflow which enables performance reproducibility and software sustainability of scientific software. In this paper, we present example scripts that extract and visualize performance data for Ginkgo's SpMV kernels that allow users to identify the optimal kernel for specific problem characteristics.

Anzt2019 [378] H. Anzt, G. Flegar, T. Grützmacher, and E. S. Quintana-Ortí, "Toward a modular precision ecosystem for high-performance computing," *The International Journal of High Performance Computing Applications*, 2019. DOI: 10.1177/1094342019846547,

Abstract: With the memory bandwidth of current computer architectures being significantly slower than the (floating point) arithmetic performance, many scientific computations only leverage a fraction of the computational power in today's high-performance architectures. At the same time, memory operations are the primary energy consumer of modern architectures, heavily impacting the resource cost of large-scale applications and the battery life of mobile devices. This article tackles this mismatch between floating point arithmetic throughput and memory bandwidth by advocating a disruptive paradigm change with respect to how data are stored and processed in scientific applications. Concretely, the goal is to radically decouple the data storage format from the processing format and, ultimately, design a "modular precision ecosystem" that allows for more flexibility in terms of customized data access. For memory-bounded scientific applications, dynamically adapting the memory precision to the numerical requirements allows for attractive resource savings. In this article, we demonstrate the potential of employing a modular precision ecosystem for the block-Jacobi preconditioner and the PageRank algorithm – two applications that are popular in the communities and at the same characteristic representatives for the field of numerical linear algebra and data analytics, respectively.

Augustine2019 [379] T. Augustine, J. Sarma, L.-N. Pouchet, and G. Rodríguez, "Generating piecewise-regular code from irregular structures," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI 2019, Phoenix, AZ, USA: ACM, 2019, pp. 625–639, ISBN: 978-1-4503-6712-7. DOI: 10.1145/3314221.3314615. [Online]. Available: http://doi.acm.org/10.1145/3314221.3314615,
Abstract: Irregular data structures, as exemplified with sparse matrices, have proved to be essential in modern computing. Numerous sparse formats have been investigated to improve the overall performance of Sparse Matrix-Vector multiply (SpMV). But in this work we propose instead to take a fundamentally different approach: to automatically build sets of regular sub-computations by mining for regular sub-regions in the irregular data structure. Our approach leads to code that is specialized to the sparsity structure of the input matrix, but which does not need anymore any indirection array, thereby improving SIMD vectorizability. We particularly focus on small sparse structures (below 10M nonzeros), and demonstrate substantial performance improvements and compaction capabilities compared to a classical CSR implementation and Intel MKL IE's SpMV implementation, evaluating on 200+ different matrices from the SuiteSparse repository.

Balaji2019 [380] V. Balaji and B. Lucia, "Combining data duplication and graph reordering to accelerate parallel graph processing," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, Phoenix, AX, USA, 2019,
Abstract: Performance of single-machine, shared memory graph processing is affected by expensive atomic updates and poor cache locality. Data duplication, a popular approach to eliminate atomic updates by creating thread-local copies of shared data, incurs extreme memory overheads due to the large sizes of typical input graphs. Even memory-efficient duplication strategies that exploit the power-law structure common to many graphs (by duplicating only the highly-connected "hub" vertices) suffer from overheads for having to dynamically identify the hub vertices. Degree Sorting, a popular graph reordering technique that re-assigns hub vertices consecutive IDs in a bid to improve spatial locality, is effective for single-threaded graph applications but suffers from increased false sharing in parallel executions.
The main insight of this work is that the combination of data duplication and Degree Sorting eliminates the overheads of each optimization. Degree Sorting improves the efficiency of data duplication by assigning hub vertices consecutive IDs which enables easy identification of the hub vertices. Additionally, duplicating the hub vertex data eliminates false sharing in Degree Sorting since each thread updates its local copy of the hub vertex data. We evaluate this mutually-enabling combination of power-law-specific data duplication and Degree Sorting in a system called RADAR. RADAR improves performance by eliminating atomic updates for hub vertices and improving the cache locality of graph applications, providing speedups of up to 166x (1.88x on average) across different graph applications and input graphs.

Barratt2019 [381] S. Barratt and S. Boyd. (2019). Least squares auto-tuning. arXiv: arXiv:1904.05460.

Bernaschi2019 [382] M. Bernaschi, M. Carrozzo, A. Franceschini, and C. Janna, "A dynamic pattern factored sparse approximate inverse preconditioner on graphics processing units," *SIAM Journal on Scientific Computing*, vol. 41, no. 3, pp. C139–C160, 2019. DOI: 10.1137/18M1197461,
Abstract: One of the most time-consuming tasks in the procedures for the numerical study of PDEs is the solution to linear systems of equations. To that purpose, iterative solvers are viewed as a promising alternative to direct methods on high-performance computers since, in theory, they are almost perfectly parallelizable. Their main drawback is the need of finding a suitable preconditioner to accelerate convergence. The factorized sparse approximate inverse (FSAI), mainly in its adaptive form, has proven to be an effective parallel preconditioner for several problems. In the present work, we report about two novel ideas to dynamically compute, on graphics processing units (GPUs), the FSAI sparsity pattern, which is the main task in its setup. The first approach, borrowed from the CPU implementation, uses a global array as a nonzero indicator, whereas the second one relies on a merge-sort procedure of multiple arrays. We will show that the second approach requires significantly less memory and overcomes issues related to the limited global memory available on GPUs. Numerical tests prove that the GPU implementation of FSAI allows for an average speed-up of 7.5 over a parallel CPU implementation. Moreover, we will show that the preconditioner

computation is still feasible using single precision arithmetic with a further 20% reduction of the setup cost. Finally, the strong scalability of the overall approach in shown in a multi-GPU setting.

Blass2019 [383] T. Blaß and M. Philippsen, "Which graph representation to select for static graph-algorithms on a cuda-capable gpu," in *Proceedings of the 12th Workshop on General Purpose Processing Using GPUs*, ser. GPGPU '19, Providence, RI, USA: ACM, 2019, pp. 22–31, ISBN: 978-1-4503-6255-9. DOI: 10.1145/3300053.3319416. [Online]. Available: http://doi.acm.org/10.1145/3300053.3319416,

Abstract: GPUs seem to be ideal for algorithms that work in parallel. A number of ways to represent graphs in GPU memory are known. But so far there are no guidelines to select the representation that is likely to result in the best performance.

This a comprehensive study investigates for CUDA-capable GPUs how different graph representations influence the performance of highly optimized graph processing algorithms that traverse the graphs without modifying them. We evaluate three different graph exchange formats and how efficiently they can be imported into eight graph data structures. We use ten state-of-the-art benchmarks that employ different traversals pattern. We evaluate them on 19 input graphs with different characteristics. The measurements show that there is not a single best data structure; the runtime performance can vary up to a factor of 2 between two representations.

The main contribution is a set of rules that helps in picking the best-performing graph representation for a given situation.

Cools2019 [384] S. Cools, J. Cornelis, P. Ghysels, and W. Vanroose, "Improving strong scaling of the conjugate gradient method for solving large linear systems using global reduction pipelining," in *Proceedings of the 2019 EuroMPI conference*, ser. EuroMPI'19, 2019. arXiv: arXiv:1905.06850,

Abstract: This paper presents performance results comparing MPI-based implementations of the popular Conjugate Gradient (CG) method and several of its communication hiding (or "pipelined") variants. Pipelined CG methods are designed to efficiently solve SPD linear systems on massively parallel distributed memory hardware, and typically display significantly improved strong scaling compared to classic CG. This increase in parallel performance is achieved by overlapping the global reduction phase (MPI_Iallreduce) required to compute the inner products in each iteration by (chiefly local) computational work such as the matrix-vector product as well as other global communication. This work includes a brief introduction to the deep pipelined CG method for readers that may be unfamiliar with the specifics of the method. A brief overview of implementation details provides the practical tools required for implementation of the algorithm. Subsequently, easily reproducible strong scaling results on the US Department of Energy (DoE) NERSC machine "Cori" (Phase I – Haswell nodes) on up to 1024 nodes with 16 MPI ranks per node are presented using an implementation of $p(l)$-CG that is available in the open source PETSc library. Observations on the staggering and overlap of the asynchronous, non-blocking global communication phases with communication and computational kernels are drawn from the experiments.

Demirci2019 [385] G. V. Demirci and C. Aykanat, "Scaling sparse matrix-matrix multiplication in the accumulo database," *Distributed and Parallel Databases*, Jan. 2019, ISSN: 1573-7578. DOI: 10.1007/s10619-019-07257-y,

Abstract: We propose and implement a sparse matrix-matrix multiplication (SpGEMM) algorithm running on top of Accumulo's iterator framework which enables high performance distributed parallelism. The proposed algorithm provides write-locality while ingesting the output matrix back to database via utilizing row-by-row parallel SpGEMM. The proposed solution also alleviates scanning of input matrices multiple times by making use of Accumulo's batch scanning capability which is used for accessing multiple ranges of key-value pairs in parallel. Even though the use of batch-scanning introduces some latency overheads, these overheads are alleviated by the proposed solution and by using node-level parallelism structures. We also propose a matrix partitioning scheme which reduces the total communication volume and provides a balance of workload among servers. The results of extensive experiments performed on both real-world and synthetic sparse matrices show that the proposed algorithm scales significantly better than the outer-product par-

allel SpGEMM algorithm available in the Graphulo library. By applying the proposed matrix partitioning, the performance of the proposed algorithm is further improved considerably.

Ernst2019 [386] J. T. Dominik Ernst Georg Hager and G. Wellein, "Performance engineering for a tall & skinny matrix multiplication kernel on gpus," 2019. arXiv: arXiv:1905.03136,
Abstract: General matrix-matrix multiplications (GEMM) in vendor-supplied BLAS libraries are best optimized for square matrices but often show bad performance for tall and skinny matrices, which are much taller than wide. Nvidia's current CUBLAS implementation delivers only a fraction of the potential performance (as given by the roofline model) in this case. We describe the challenges and key properties of an implementation that can achieve perfect performance. We further evaluate different approaches of parallelization and thread distribution, and devise a flexible, configurable mapping scheme. A code generation approach enables a simultaneously flexible and specialized implementation with autotuning. This results in perfect performance for a large range of matrix sizes in the domain of interest, and at least 2/3 of maximum performance for the rest on an Nvidia Volta GPGPU.

Elgohary2019 [387] A. Elgohary, M. Boehm, P. J. Haas, F. R. Reiss, and B. Reinwald, "Compressed linear algebra for declarative large-scale machine learning," *Communications of the ACM*, vol. 62, no. 5, pp. 83–91, Apr. 2019, ISSN: 0001-0782. DOI: 10.1145/3318221. [Online]. Available: http://doi.acm.org/10.1145/3318221,
Abstract: Large-scale Machine Learning (ML) algorithms are often iterative, using repeated read-only data access and I/O-bound matrix-vector multiplications. Hence, it is crucial for performance to fit the data into single-node or distributed main memory to enable fast matrix-vector operations. General-purpose compression struggles to achieve both good compression ratios and fast decompression for block-wise uncompressed operations. Therefore, we introduce Compressed Linear Algebra (CLA) for lossless matrix compression. CLA encodes matrices with lightweight, value-based compression techniques and executes linear algebra operations directly on the compressed representations. We contribute effective column compression schemes, cache-conscious operations, and an efficient sampling-based compression algorithm. Our experiments show good compression ratios and operations performance close to the uncompressed case, which enables fitting larger datasets into available memory. We thereby obtain significant end-to-end performance improvements.

Agullo2019 [388] L. P. Emmanuel Agullo Luc Giraud, "Robust preconditioners via generalized eigenproblems for hybrid sparse linear solvers," *SIAM Journal on Matrix Analysis and Applications*, 2019. [Online]. Available: https://hal.inria.fr/hal-02074474/document,
Abstract: The solution of large sparse linear systems is one of the most time consuming kernels in many numerical simulations. The domain decomposition community has developed many efficient and robust methods in the last decades. While many of these solvers fall into the abstract Schwarz (aS) framework, their robustness has originally been demonstrated on a case-by-case basis. In this paper, we propose a bound for the condition number of all deflated aS methods provided that the coarse grid consists of the assembly of local components that contain the kernel of some local operators. We show that classical results from the literature on particular instances of aS methods can be retrieved from this bound. We then show that such a coarse grid correction can be explicitly obtained algebraically via generalized eigenproblems, leading to a condition number independent of the number of domains. This result can be readily applied to retrieve or improve the bounds previously obtained via generalized eigenproblems in the particular cases of Neumann-Neumann (NN), Additive Schwarz (AS) and optimized Robin but also generalizes them when applied with approximate local solvers. Interestingly, the proposed methodology turns out to be a comparison of the considered particular aS method with generalized versions of both NN and AS for tackling the lower and upper part of the spectrum, respectively. We furthermore show that the application of the considered grid corrections in an additive fashion is robust in the AS case although it is not robust for aS methods in general. In particular, the proposed framework allows for ensuring the robustness of the AS method applied on the Schur complement (AS/S), either with deflation or additively, and with the freedom of relying on an approximate local Schur complement. Numerical experiments illustrate these statements.

Fuchs2019 [389] A. Fuchs and D. Wentzlaff, "The accelerator wall: Limits of chip specialization," in *Proceedings of the 25th IEEE International Symposium on High-Performance Computer Architecture*, ser. HPCA'18, 2019,
Abstract: Specializing chips using hardware accelerators has become the prime means to alleviate the gap between the growing computational demands and the stagnating transistor budgets caused by the slowdown of CMOS scaling. Much of the benefits of chip specialization stems from optimizing a computational problem within a given chip's transistor budget. Unfortunately, the stagnation of the number of transistors available on a chip will limit the accelerator design optimization space, leading to diminishing specialization returns, ultimately hitting an accelerator wall.
In this work, we tackle the question of what are the limits of future accelerators and chip specialization? We do this by characterizing how current accelerators depend on CMOS scaling, based on a physical modeling tool that we constructed using datasheets of thousands of chips. We identify key concepts used in chip specialization, and explore case studies to understand how specialization has progressed over time in different applications and chip platforms (e.g., GPUs, FPGAs, ASICs). Utilizing these insights, we build a model which projects forward to see what future gains can and cannot be enabled from chip specialization. A quantitative analysis of specialization returns and technological boundaries is critical to help researchers understand the limits of accelerators and develop methods to surmount them.

Yuan2019 [390] W.-S. Z. Ganzhao Yuan Li Shen, "A block decomposition algorithm for sparse optimization," 2019. arXiv: arXiv:1905.11031,
Abstract: Sparse optimization is a central problem in machine learning and computer vision. However, this problem is inherently NP-hard and thus difficult to solve in general. Combinatorial search methods find the global optimal solution but are confined to small-sized problems, while coordinate descent methods are efficient but often suffer from poor local minima. This paper considers a new block decomposition algorithm that combines the effectiveness of combinatorial search methods and the efficiency of coordinate descent methods. Specifically, we consider a random strategy or/and a greedy strategy to select a subset of coordinates as the working set, and then perform a global combinatorial search over the working set based on the original objective function. We show that our method finds stronger stationary points than Amir Beck et al.'s coordinate-wise optimization method. In addition, we establish the global convergence and convergence rate of our block decomposition algorithm. Our experiments on solving sparse regularized and sparsity constrained least squares optimization problems demonstrate that our method achieves state-ofthe-art performance in terms of accuracy.

Georgieva2019 [391] I. Georgieva, S. Harizanov, and C. Hofreither, "Iterative low-rank approximation solvers for the extension method for fractional diffusion," Johann Radon Institute for Computational and Applied Mathematics (RICAM), Tech. Rep. RICAM-Report 2019-14, 2019,
Abstract: We consider the numerical method for fractional diffusion problems which is based on an extension to a mixed boundary value problem for a local operator in a higher dimensional space. We observe that, when this problem is discretized using tensor product spaces as is commonly done, the solution can be very well approximated by low-rank tensors. This motivates us to apply iterative low-rank approximation algorithms in order to efficiently solve this extended problem. In particular, we employ a recently proposed greedy Tucker approximation method as well as a more classical greedy rank one update method. Throughout, all objects of interest are kept in suitable low-rank approximations, which dramatically reduces the required amount of memory compared to the full formulation of the extended problem.
Our approach can be used for general, non-structured space discretizations. If the space discretization itself has tensor product structure, we can further decompose the problem in order to deal with even lower dimensional objects. We also note that the approach can be directly applied to higher-order discretizations both in space and the extended variable.
In several numerical examples, we demonstrate the convergence behaviour of the proposed methods. In particular, the Tucker approximation approach requires only a few iterations in order to reach the discretization error in all tested settings.

Harvey2019 [392] D. Harvey and J. van der Hoeven, "Integer multiplication in time $o(nlogn)$," HAL archives,

Tech. Rep. hal-02070778, 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02070778,
Abstract: We present an algorithm that computes the product of two n-bitintegers in $O(nlogn)$ bit operations.

Hoemmen2019 [393] "Historical lessons for c++ linear algebra library standardization," ISO C++ standards meeting (Kona), Tech. Rep. P1417R0, 2019. [Online]. Available: http://www.open-std.org/JTC1/SC22/WG21/docs/papers/2019/p1417r0.pdf.

Hong2019 [394] C. Hong, A. Sukumaran-Rajam, I. Nisa, K. Singh, and P. Sadayappan, "Adaptive sparse tiling for sparse matrix multiplication," in *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '19, Washington, District of Columbia: ACM, 2019, pp. 300–314, ISBN: 978-1-4503-6225-2. DOI: 10.1145/3293883.3295712,
Abstract: Tiling is a key technique for data locality optimization and is widely used in high-performance implementations of dense matrix-matrix multiplication for multicore/manycore CPUs and GPUs. However, the irregular and matrix-dependent data access pattern of sparse matrix multiplication makes it challenging to use tiling to enhance data reuse. In this paper, we devise an adaptive tiling strategy and apply it to enhance the performance of two primitives: SpMM (product of sparse matrix and dense matrix) and SDDMM (sampled dense-dense matrix multiplication). In contrast to studies that have resorted to non-standard sparse-matrix representations to enhance performance, we use the standard Compressed Sparse Row (CSR) representation, within which intra-row reordering is performed to enable adaptive tiling. Experimental evaluation using an extensive set of matrices from the Sparse Suite collection demonstrates significant performance improvement over currently available state-of-the-art alternatives.

Huckle2019 [395] T. K. Huckle. (2019). Accelerated jacobi iterations for bidiagonal and sparse triangular matrices, [Online]. Available: https://www5.in.tum.de/persons/huckle/it_triang.pdf,
Abstract: In many applications a sparse linear system of equations $Ax = b$ has to be solved. For applying iterative solvers like preconditioned conjugate gradient (pcg) or GMRES, effective preconditioners are necessary, e.g. Jacobi, Gauss-Seidel, or incomplete LU factorization (ILU). Often, effective preconditioners are given via sparse triangular matrices L, that have to be solved in every iteration step. Recent work by Edmond Chow introduced an easy to parallelize fixed-point iteration for computing approximations to (I)LU factorizations. Therefore, the aching handicap in parallel solution methods for sparse matrices is the solving of sparse triangular systems, e.g. bidiagonal matrices. In a parallel environment direct solvers can take only restricted advantage of parallelism. Therefore, in this paper we develop a fast iterative solution method for sparse triangular matrices. In contrast to direct solvers for triangular matrices L like graph-based methods, sparse factorization methods, or Sherman-Morrison-Woodbury, here we want to consider stationary Jacobi iterations. In its original form the Jacobi iteration for ill-conditioned matrices can lead to very slow convergence. Therefore, we introduce different acceleration tools like preconditioning (block Jacobi and Incomplete Sparse Approximate Inverse ISAI), and a recursive acceleration of the Jacobi method. Here the Neumann series is replaced by the Euler expansion (see [4, 19, 8]). This is derived by a recursive computation of the Neumann series using powers of the initial Jacobi iteration matrix. The goal is to shift the major part of the operations from cheap but numerous iteration steps to better parallelizable cheap and sparse matrix-matrix products reducing the number of necessary iterations considerably, e.g. to less than $\log_2(n)$ for an $n \times n$ matrix.

Duff2019 [396] J. H. Iain Duff and F. Lopez, "A new sparse symmetric indefinite solver using a posteriori threshold pivoting," Science & Technology Facilities Council, UK, Technical Report RAL-TR-2018-008, 2019,
Abstract: The factorization of sparse symmetric indefinite systems is particularly challenging since pivoting is required to maintain stability of the factorization. Pivoting techniques generally offer limited parallelism and are associated with significant data movement hindering the scalability of these methods. Variants of the Threshold Partial Pivoting (TPP) algorithm for example have been often used because of its numerical robustness but standard implementations exhibit poor parallel performance. On the other hand, some methods trade stability for performance on parallel

architectures such as the Supernode Bunch-Kaufman (SBK) used in the PARDISO solver. In this case, however, the factors obtained might not be used to accurately compute the solution of the system. For this reason we have designed a task-based LDL^T factorization algorithm based on a new pivoting strategy called A Posteriori Threshold Pivoting (APTP) that is much more suitable for modern multicore architectures and has the same numerical robustness as the TPP strategy. We implemented our algorithm in a new version of the SPRAL Sparse Symmetric Indefinite Direct Solver (SSIDS) which initially supported GPU-only factorization. We have used OpenMP 4 task features to implement a multifrontal algorithm with dense factorizations using the novel APTP, and we show that it performs favourably compared to the state-of-the-art solvers HSL_MA86, HSL_MA97 and PARDISO both in terms of performance on a multicore machine and in terms of numerical robustness. Finally we show that this new solver is able to make use of GPU devices for accelerating the factorization on heterogeneous architectures.

Idreos2019 [397] S. Idreos, N. Dayan, W. Qin, M. Akmanalp, S. Hilgard, A. Ross, J. Lennon, V. Jain, H. Gupta, D. Li, and Z. Zhu, "Design continuums and the path toward self-designing key-value stores that know and learn," in *Biennial Conference on Innovative Data Systems Research*, 2019,
Abstract: We introduce the concept of design continuums for the data layout of key-value stores. A design continuum unifies major distinct data structure designs under the same model. The critical insight and potential long-term impact is that such unifying models 1) render what we consider up to now as fundamentally different data structures to be seen as views" of the very same overall design space, and 2) allow seeing" new data structure designs with performance properties that are not feasible by existing designs. The core intuition behind the construction of design continuums is that all data structures arise from the very same set of fundamental design principles, i.e., a small set of data layout design concepts out of which we can synthesize any design that exists in the literature as well as new ones. We show how to construct, evaluate, and expand, design continuums and we also present the first continuum that unifies major data structure designs, i.e., B+Tree, BeTree, LSM-tree, and LSH-Table.
The practical benefit of a design continuum is that it creates a fast inference engine for the design of data structures. For example, we can near instantly predict how a specific design change in the underlying storage of a data system would affect performance, or reversely what would be the optimal data structure (from a given set of designs) given workload characteristics and a memory budget. In turn, these properties allow us to envision a new class of self-designing key-value stores with a substantially improved ability to adapt to workload and hardware changes by transitioning between drastically different data structure designs to assume a diverse set of performance properties at will.

Ioannidis2019 [398] E. Ioannidis, N. Cheimarios, A. Spyropoulos, and A. Boudouvis, "On the performance of various parallel GMRES implementations on CPU and GPU clusters," 2019. arXiv: arXiv:1906.04051v1.

Jagode2019 [399] H. Jagode, A. Danalis, H. Anzt, and J. Dongarra, "Papi software-defined events for in-depth performance analysis," *The International Journal of High Performance Computing Applications*, 2019. DOI: 10.1177/1094342019846287,
Abstract: The methodology and standardization layer provided by the Performance Application Programming Interface (PAPI) has played a vital role in application profiling for almost two decades. It has enabled sophisticated performance analysis tool designers and performance-conscious scientists to gain insights into their applications by simply instrumenting their code using a handful of PAPI functions that "just work" across different hardware components. In the past, PAPI development had focused primarily on hardware-specific performance metrics. However, the rapidly increasing complexity of software infrastructure poses new measurement and analysis challenges for the developers of large-scale applications. In particular, acquiring information regarding the behavior of libraries and runtimes—used by scientific applications—requires low-level binary instrumentation, or APIs specific to each library and runtime. No uniform API for monitoring events that originate from inside the software stack has emerged. In this article, we present our efforts to extend PAPI's role so that it becomes the de facto standard for exposing performance-critical

events, which we refer to as software-defined events (SDEs), from different software layers. Upgrading PAPI with SDEs enables monitoring of both types of performance events—hardware- and software-related events—in a uniform way, through the same consistent PAPI. The goal of this article is threefold. First, we motivate the need for SDEs and describe our design decisions regarding the functionality we offer through PAPI's new SDE interface. Second, we illustrate how SDEs can be utilized by different software packages, specifically, by showcasing their use in the numerical linear algebra library MAGMA-Sparse, the tensor algebra library TAMM that is part of the NWChem suite, and the compiler-based performance analysis tool Byfl. Third, we provide a performance analysis of the overhead that results from monitoring SDEs and discuss the trade-offs between overhead and functionality.

Kawaguchi2019 [400] K. Kawaguchi and L. Pack Kaelbling. (Apr. 2019). Every local minimum is a global minimum of an induced model. arXiv: arXiv:1904.03673 [stat.ML],
Abstract: For non-convex optimization in machine learning, this paper proves that every local minimum achieves the global optimality of the perturbable gradient basis model at any differentiable point. As a result, non-convex machine learning is theoretically as supported as convex machine learning with a hand-crafted basis in terms of the loss at differentiable local minima, except in the case when a preference is given to the hand-crafted basis over the perturbable gradient basis. The proofs of these results are derived under mild assumptions. Accordingly, the proven results are directly applicable to many machine learning models, including practical deep neural networks, without any modification of practical methods. Furthermore, as special cases of our general results, this paper improves or complements several state-of-the-art theoretical results in the literature with a simple and unified proof technique.

Kong2019 [401] F. Kong, "Parallel memory-efficient all-at-once algorithms for the sparse matrix triple products in multigrid methods," *The International Journal of High Performance Computing Applications*, 2019. arXiv: arXiv:1905.08423.

Lagraviere2019 [402] J. Lagravière, J. Langguth, M. Prugger, L. Einkemmer, P. H. Ha, and X. Cai, "Performance optimization and modeling of fine-grained irregular communication in UPC," *Scientific Programming*, vol. 2019, 2019. DOI: 10.1155/2019/6825728,
Abstract: .e Unified Parallel C (UPC) programming language offers parallelism via logically partitioned shared memory, which typically spans physically disjoint memory subsystems. One convenient feature of UPC is its ability to automatically execute betweenthread data movement, such that the entire content of a shared data array appears to be freely accessible by all the threads. .e programmer friendliness, however, can come at the cost of substantial performance penalties. .is is especially true when indirectly indexing the elements of a shared array, for which the induced between-thread data communication can be irregular and have a fine-grained pattern. In this paper, we study performance enhancement strategies specifically targeting such finegrained irregular communication in UPC. Starting from explicit thread privatization, continuing with block-wise communication, and arriving at message condensing and consolidation, we obtained considerable performance improvement of UPC programs that originally require fine-grained irregular communication. Besides the performance enhancement strategies, the main contribution of the present paper is to propose performance models for the different scenarios, in the form of quantifiable formulas that hinge on the actual volumes of various data movements plus a small number of easily obtainable hardware characteristic parameters. .ese performance models help to verify the enhancements obtained, while also providing insightful predictions of similar parallel implementations, not limited to UPC, that also involve between-thread or between-process irregular communication. As a further validation, we also apply our performance modeling methodology and hardware characteristic parameters to an existing UPC code for solving a 2D heat equation on a uniform mesh.

Lei2019 [403] D. Lei, M. Du, H. Chen, Z. Li, and Y. Wu, "Distributed parallel sparse multinomial logistic regression," *IEEE Access*, vol. 7, pp. 55 496–55 508, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2913280,
Abstract: Sparse Multinomial Logistic Regression (SMLR) is widely used in the field of image classification, multi-class object recognition, and so on, because it has the function of embedding feature selection during classification. However, it cannot meet the time and memory requirements

for processing large-scale data. We have reinvestigated the classification accuracy and running efficiency of the algorithm for solving SMLR problems using the Alternating Direction Method of Multipliers (ADMM), which is called fast SMLR (FSMLR) algorithm in this paper. By reformulating the optimization problem of FSMLR, we transform the serial convex optimization problem to the distributed convex optimization problem, i.e., global consensus problem and sharing problem. Based on the distributed optimization problem, we propose two distribute parallel SMLR algorithms, sample partitioning-based distributed SMLR (SP-SMLR), and feature partitioning-based distributed SMLR (FP-SMLR), for a large-scale sample and large-scale feature datasets in big data scenario, respectively. The experimental results show that the FSMLR algorithm has higher accuracy than the original SMLR algorithm. The big data experiments show that our distributed parallel SMLR algorithms can scale for massive samples and large-scale features, with high precision. In a word, our proposed serial and distribute SMLR algorithms outperform the state-of-the-art algorithms.

Li2019 [404] Y. Li, P. Xie, X. Chen, J. Liu, B. Yang, S. Li, C. Gong, X. Gan, and H. Xu, "Vbsf: A new storage format for simd sparse matrix–vector multiplication on modern processors," *The Journal of Supercomputing*, Apr. 2019, ISSN: 1573-0484. DOI: 10.1007/s11227-019-02835-4. [Online]. Available: https://doi.org/10.1007/s11227-019-02835-4,
Abstract: Sparse matrix–vector multiplication (SpMV) is one of the most indispensable kernels of solving problems in numerous applications, but its performance of SpMV is limited by the need for frequent memory access. Modern processors exploit data-level parallelism to improve the performance using single-instruction multiple data (SIMD). In order to take full advantage of SIMD acceleration technology, a new storage format called Variable Blocked-σ-SIMD Format (VBSF) is proposed in this paper to change the irregular nature of traditional matrix storage formats. This format combines the adjacent nonzero elements into variable size blocks to ensure that SpMV can be computed with SIMD vector units. We compare the VBSF-based SpMV with traditional storage formats using 15 matrices as a benchmark suite on three computing platforms (FT2000, Intel Xeon E5 and Intel Silver) with different SIMD length. For the matrices in the benchmark suite, the VBSF obtains great performance improvement on three platforms, respectively, and it proves to have better storage efficiency compared with other storage formats.

Liu2019 [405] H. Liu, Y. Tian, H. Zong, Q. Ma, M. Y. Wang, and L. Zhang, "Fully parallel level set method for large-scale structural topology optimization," *Computers & Structures*, vol. 221, pp. 13–27, 2019, ISSN: 0045-7949. DOI: 10.1016/j.compstruc.2019.05.010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045794918316511,
Abstract: To realize large-scale or high-resolution structural topology optimization design, a fully parallel parameterized level set method with compactly supported radial basis functions (CSRBFs) is developed based on both the uniform and non-uniform structured meshes. In this work, the whole computation process is parallelized, including mesh generation, sensitivity analysis, calculation and assembly of the element stiffness matrices, solving of the structural state equation, parameterization and updating of the level set function, and output of the computational results during the optimization iterations. In addition, some typical numerical examples, in which the calculation scale is up to 7 million 8-node hexahedral elements, are carried out for verifying the effectiveness of the proposed method. Finally, the computing time is also analyzed in detail. It is found that: (1) In the optimized structures, the thin sheet-like components gradually replace the truss-like ones when refining the mesh, (2) the parameterization process of the level set function will become fast as long as the non-uniformity of mesh is not very high and the supported radius of CSRBF is small enough, and (3) more than 80% of the total computing time is always consumed for solving the structural state equation during the finite element analysis (FEA).

Nisa2019 [406] "Load-balanced sparse mttkrp on gpus," in *Proceedings of the 2019 International Parallel and Distributed Processing Symposium*, ser. IPDPS'19, 2019,
Abstract: Sparse matricized tensor times Khatri-Rao product (MTTKRP) is one of the most computationally expensive kernels in sparse tensor computations. This work focuses on optimizing the MTTKRP operation on GPUs, addressing both performance and storage requirements. We begin by identifying the performance bottlenecks in directly extending the state-ofthe-art CSF (com-

pressed sparse fiber) format from CPUs to GPUs. A significant challenge with GPUs compared to multicore CPUs is that of utilizing the much greater degree of parallelism in a load-balanced fashion for irregular computations like sparse MTTKRP. To address this issue, we develop a new storage-efficient representation for tensors that enables highperformance, load-balanced execution of MTTKRP on GPUs. A GPU implementation of sparse MTTKRP using the new sparse tensor representation is shown to outperform all currently known parallel sparse CPU and GPU MTTKRP implementations.

Ma2019 [407] S. Ma, Z. Liu, S. Chen, L. Huang, Y. Guo, Z. Wang, and M. Zhang, "Coordinated dma: Improving the dram access efficiency for matrix multiplication," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–1, 2019, ISSN: 1045-9219. DOI: 10.1109/TPDS.2019.2906891,

Abstract: High performance implementation of matrix multiplication is essential for scientific computing. The memory access procedure is quite possible to be the bottleneck of matrix multiplication. The widely used GotoBLAS GEMM implementation divides the integral matrix into several partitions to be assigned to different cores for parallelization. Traditionally, each core deploys a DMA transfer to access its own partition in the DRAM memory. However, deploying an independent DMA transfer for each core cannot efficiently exploit the inter-core locality. Also, multiple concurrent DMA transfers interfere with each other, further reducing the DRAM access efficiency. We observe that the same row of neighboring partitions is in the same DRAM page, which means that there is significant locality inherent in the address layout. We propose the coordinated DMA to efficiently exploit the locality. It invokes one transfer to serve all cores and moves data in a row-major manner to improve the DRAM access efficiency. Compared with a baseline design, the coordinated DMA improves the bandwidth by 84.8% and reduces DRAM energy consumption by 43.1% for micro-benchmarks. It achieves higher performance for the GEMM and Linpack benchmark. With much less hardware costs, the coordinated DMA significantly outperforms an out-of-order memory controller.

Mendoza2019 [408] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, M. Urban, M. Burkart, M. Dippel, M. Lindauer, and F. Hutter, "Towards automatically-tuned deep neural networks," in *Automated Machine Learning: Methods, Systems, Challenges*, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Cham: Springer International Publishing, 2019, pp. 135–149, ISBN: 978-3-030-05318-5. DOI: 10.1007/978-3-030-05318-5_7,

Abstract: Recent advances in AutoML have led to automated tools that can compete with machine learning experts on supervised learning tasks. In this work, we present two versions of Auto-Net, which provide automatically-tuned deep neural networks without any human intervention. The first version, Auto-Net 1.0, builds upon ideas from the competition-winning system Auto-sklearn by using the Bayesian Optimization method SMAC and uses Lasagne as the underlying deep learning (DL) library. The more recent Auto-Net 2.0 builds upon a recent combination of Bayesian Optimization and HyperBand, called BOHB, and uses PyTorch as DL library. To the best of our knowledge, Auto-Net 1.0 was the first automatically-tuned neural network to win competition datasets against human experts (as part of the first AutoML challenge). Further empirical results show that ensembling Auto-Net 1.0 with Auto-sklearn can perform better than either approach alone, and that Auto-Net 2.0 can perform better yet.

Meng2019 [409] K. Meng, J. Li, G. Tan, and N. Sun, "A pattern based algorithmic autotuner for graph processing on gpus," in *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '19, Washington, District of Columbia: ACM, 2019, pp. 201–213, ISBN: 978-1-4503-6225-2. DOI: 10.1145/3293883.3295716,

Abstract: This paper proposes Gswitch, a pattern-based algorithmic auto-tuning system that dynamically switches between optimization variants with negligible overhead. Its novelty lies in a small set of algorithmic patterns that allow for the configurable assembly of variants of the algorithm. The fast transition of Gswitch is based on a machine learning model trained using 644 real graphs. Moreover, Gswitch provides a simple programming interface that conceals low-level tuning details from the user. We evaluate Gswitch on typical graph algorithms (BFS, CC, PR, SSSP, and BC) using Nvidia Kepler and Pascal GPUs. The results show that Gswitch runs up to

10*times* faster than the best configuration of the state-of-the-art programmable GPU-based graph processing libraries on 10 representative graphs. Gswitch outperforms Gunrock on 92.4% cases of 644 graphs which is the largest dataset evaluation reported to date.

Mohammadi2019 [410] M. S. Mohammadi, T. Yuki, K. Cheshmi, E. C. Davis, M. Hall, M. M. Dehnavi, P. Nandy, C. Olschanowsky, A. Venkat, and M. M. Strout, "Sparse computation data dependence simplification for efficient compiler-generated inspectors," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI 2019, Phoenix, AZ, USA: ACM, 2019, pp. 594–609, ISBN: 978-1-4503-6712-7. DOI: 10.1145/3314221.3314646,

Abstract: This paper presents a combined compile-time and runtime loop-carried dependence analysis of sparse matrix codes and evaluates its performance in the context of wavefront parallellism. Sparse computations incorporate indirect memory accesses such as x[col[j]] whose memory locations cannot be determined until runtime. The key contributions of this paper are two compile-time techniques for significantly reducing the overhead of runtime dependence testing: (1) identifying new equality constraints that result in more efficient runtime inspectors, and (2) identifying subset relations between dependence constraints such that one dependence test subsumes another one that is therefore eliminated. New equality constraints discovery is enabled by taking advantage of domain-specific knowledge about index arrays, such as col[j]. These simplifications lead to automatically-generated inspectors that make it practical to parallelize such computations. We analyze our simplification methods for a collection of seven sparse computations. The evaluation shows our methods reduce the complexity of the runtime inspectors significantly. Experimental results for a collection of five large matrices show parallel speedups ranging from 2x to more than 8x running on a 8-core CPU.

Montagne2019 [411] E. Montagne and R. Surós, "Systolic sparse matrix vector multiply in the age of tpus and accelerators," in *2019 Spring Simulation Conference (SpringSim)*, Apr. 2019, pp. 1–10. DOI: 10.23919/SpringSim.2019.8732860,

Abstract: Tensor Processing Units has brought back systolic arrays as a computational alternative to high performance computing. Recently Google presented a Tensor Processing Unit for handling matrix multiplication using systolic arrays. This unit is designed for dense matrices only. As they stated, sparse architectural support was omitted momentarily but they will focus on sparsity in future designs. We propose a systolic array to compute the Sparse Matrix Vector product in $T2(n) \approx \lceil \frac{nnz}{2} \rceil + 2n + 2$ using 2 n+2 processing elements. The systolic array we propose also use accumulators to collect the partial results of the resulting vector and supports adapting tiling.

Muro2019 [412] R. Muro, A. Fujii, and T. Tanaka, "Acceleration of symmetric sparse matrix-vector product using improved hierarchical diagonal blocking format," in *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, ser. HPC Asia 2019, Guangzhou, China: ACM, 2019, pp. 63–70, ISBN: 978-1-4503-6632-8. DOI: 10.1145/3293320.3293332,

Abstract: In the previous study, Guy et al. proposed sparse matrix-vector product (SpMV) acceleration using the Hierarchical Diagonal Blocking (HDB) format that recursively repeated partitioning, reordering, and blocking on symmetric sparse matrix. The HDB format stores sparse matrix hierarchically using tree structure. Each node of tree structure of HDB format store small sparse matrices using CSR format.

In this present study, we examined two problems with the HDB format and provided a solution for each problem.

First, SpMV using the HDB format has a partial dependent relationship among hierarchies. The problem with the HDB format is that the parallelism of computation decreases as the hierarchy of nodes gets closer to the root. Thus, we propose cutting of dependency using work vectors to solve this problem.

Second, each node of the conventional HDB format is stored in Compressed Sparse Row (CSR) format. Block compressed Sparse Row (BSR) format often becomes faster than CSR format in SpMV performance. Thus, we evaluated the effectiveness of our proposed method with work vectors also for BSR-HDB format.

In addition, we compare the performance in the general format (CSR format, BSR format) using the Intel Math Kernel Library (MKL), the conventional HDB format, and the expanded HDB format by using 22 types of sparse matrix that from various field. The results showed that the SpMV performance was highest in the HDB format that we expanded in 19 types of sparse matrix, which was 1.99 times faster than the CSR format.

Nie2019 [413] Q. Nie and S. Malik, "SpFlow: Memory-driven data flow optimization for sparse matrix-matrix multiplication," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, ser. ISCAS'19, May 2019, pp. 1–5. DOI: 10.1109/ISCAS.2019.8702111,
Abstract: To improve the performance of sparse matrix-matrix multiplication (SpMM) running on a specialized architecture, orchestrating a data flow that maximizes data reuse in local memory is critical but challenging due to the irregular non-zero element locations and the wide range of sparsity. In this work, we proposed SpFlow, a memory-driven data flow optimization framework for SpMM. SpFlow can realize 54X fewer DRAM accesses and 97X fewer SRAM accesses on average than a GPU running the cuSPARSE kernel. And in comparison with a state-of-the-art accelerator, the performance can be improved by 3X, and SRAM accesses reduced by 5X on average.

Muhammad2019 [414] M. Osama, M. Truong, C. Yang, A. Buluç, and J. D. Owens, "Graph coloring on the gpu," UC Davis: College of Engineering, Tech. Rep., 2019. [Online]. Available: https://escholarship.org/uc/item/6kp4p18t,
Abstract: We design and implement parallel graph coloring algorithms on the GPU using two different abstractions—one datacentric (Gunrock), the other linear-algebra-based (GraphBLAS). We analyze the impact of variations of a baseline independent-set algorithm on quality and runtime. We study how optimizations such as hashing, avoiding atomics, and a max-min heuristic affect performance. Our Gunrock graph coloring implementation has a peak 2× speed-up, a geomean speed-up of 1.3× and produces 1.6× more colors over previous hardwired state-of-theart implementations on real-world datasets. Our GraphBLAS implementation of Luby's algorithm produces 1.9× fewer colors than the previous state-of-the-art parallel implementation at the cost of 3× extra runtime, and 1.014× fewer colors than a greedy, sequential algorithm with a geomean speed-up of 2.6×.

Das2019 [415] "Parallel and communication avoiding least angle regression," 2019. arXiv: arXiv:1905.11340,
Abstract: We are interested in parallelizing the Least Angle Regression (LARS) algorithm for fitting linear regression models to high-dimensional data. We consider two parallel and communication avoiding versions of the basic LARS algorithm. The two algorithms apply to data that have different layout patterns (one is appropriate for row-partitioned data, and the other is appropriate for column-partitioned data), and they have different asymptotic costs and practical performance. The first is bLARS, a block version of LARS algorithm, where we update b columns at each iteration. Assuming that the data are row-partitioned, bLARS reduces the number of arithmetic operations, latency, and bandwidth by a factor of b. The second is Tournament-bLARS (T-bLARS), a tournament version of LARS, in which case processors compete, by running several LARS computations in parallel, to choose b new columns to be added into the solution. Assuming that the data are column-partitioned, T-bLARS reduces latency by a factor of b. Similarly to LARS, our proposed methods generate a sequence of linear models. We present extensive numerical experiments that illustrate speed-ups up to 25× compared to LARS.

Rais2019 [416] H. M. Rais, S. A. Abed, and J. Watada, "Computational comparison of major proposed methods for graph partitioning problem," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 23, no. 1, pp. 5–17, 2019. DOI: 10.20965/jaciii.2019.p0005,
Abstract: k-way graph partitioning is an NP-complete problem, which is applied to various tasks such as route planning, image segmentation, community detection, and high-performance computing. The approximate methods constitute a useful solution for these types of problems. Thus, many research studies have focused on developing meta-heuristic algorithms to tackle the graph partitioning problem. Local search is one of the earliest methods that has been applied efficiently to this type of problem. Recent studies have explored various types of local search methods and have

improved them such that they can be used with the partitioning process. Moreover, local search methods are widely integrated with population-based approaches, to provide the best diversification and intensification for the problem space. This study emphasizes the local search approaches, as well as their combination with other graph partitioning approaches. At present, none of the surveys in the literature has focused on this class of state of the art approaches in much detail. In this study, the vital parts of these approaches including neighborhood structure, acceptance criterion, and the ways of combining them with other approaches, are highlighted. Additionally, we provide an experimental comparison that shows the variance in the performance of the reviewed methods. Hence, this study clarifies these methods to show their advantages and limitations for the targeted problem, and thus can aid in the direction of research flow towards the area of graph partitioning.

Reguly2019 [417] I. Z. Reguly, G. R. Mudalige, M. B. Giles, and S. Maheswaran, "Improving resilience of scientific software through a domain-specific approach," *Journal of Parallel and Distributed Computing*, 2019, ISSN: 0743-7315. DOI: 10.1016/j.jpdc.2019.01.015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731519300917,
Abstract: In this paper we present research on improving the resilience of the execution of scientific software, an increasingly important concern in High Performance Computing (HPC). We build on an existing high-level abstraction framework, the Oxford Parallel library for Structured meshes (OPS), developed for the solution of multi-block structured mesh-based applications, and implement an algorithm in the library to carry out checkpointing automatically, without the intervention of the user. The target applications are a hydrodynamics benchmark application from the Mantevo Suite, CloverLeaf 3D, the sparse linear solver proxy application TeaLeaf, and the OpenSBLI compressible Navier–Stokes direct numerical simulation (DNS) solver. We present (1) the basic algorithm that OPS relies on to determine the optimal checkpoint in terms of size and location, (2) improvements that supply additional information to improve the decision, (3) techniques that reduce the cost of writing the checkpoints to non-volatile storage, (4) a performance analysis of the developed techniques on a single workstation and on several supercomputers, including ORNL's Titan. Our results demonstrate the utility of the high-level abstractions approach in automating the checkpointing process and show that performance is comparable to, or better than the reference in all cases.

Sao2019 [418] P. Sao, X. S. Li, and R. Vuduc, "A communication-avoiding 3d algorithm for sparse lu factorization on heterogeneous systems," *Journal of Parallel and Distributed Computing*, 2019, ISSN: 0743-7315. DOI: 10.1016/j.jpdc.2019.03.004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731518305197,
Abstract: We propose a new algorithm to improve the strong scalability of right-looking sparse LU factorization on distributed memory systems. Our 3D algorithm for sparse LU uses a three-dimensional MPI process grid, exploits elimination tree parallelism, and trades off increased memory for reduced per-process communication. We also analyze the asymptotic improvements for planar graphs (e.g., those arising from 2D grid or mesh discretizations) and certain non-planar graphs (specifically for 3D grids and meshes). For a planar graph with n vertices, our algorithm reduces communication volume asymptotically in n by a factor of Ologn and latency by a factor of Ologn. For non-planar cases, our algorithm can reduce the per-process communication volume by 3× and latency by On13 times. In all cases, the memory needed to achieve these gains is a constant factor. We implemented our algorithm by extending the 2D data structure used in SuperLU_DIST. Our new 3D code achieves empirical speedups up to 27× for planar graphs and up to 3.3× for non-planar graphs over the baseline 2D SuperLU_DIST when run on 24,000 cores of a Cray XC30. We extend the 3D algorithm for heterogeneous architectures by adding the Highly Asynchronous Lazy Offload (Halo) algorithm for co-processor offload [44]. On 4096 nodes of a Cray XK7 with 32,768 CPU cores and 4096 Nvidia K20x GPUs, the 3D algorithm achieves empirical speedups up to 24× for planar graphs and 3.5× for non-planar graphs over the baseline 2D SuperLU_DIST with co-processor acceleration.

Scott2019 [419] J. Scott and M. Tuma, "Sparse stretching for solving sparse-dense linear least-squares problems," *SIAM Journal on Scientific Computing*, 2019, ISSN: 1095-7197,

Abstract: Large-scale linear least-squares problems arise in a wide range of practical applications. In some cases, the system matrix contains a small number of dense rows. These make the problem significantly harder to solve because their presence limits the direct applicability of sparse matrix techniques. In particular, the normal matrix is (close to) dense, so that forming it is impractical. One way to help overcome the dense row problem is to employ matrix stretching. Stretching is a sparse matrix technique that improves sparsity by making the least-squares problem larger. We show that standard stretching can still result in the normal matrix for the stretched problem having an unacceptably large amount of fill. This motivates us to propose a new sparse stretching strategy that performs the stretching so as to limit the fill in the normal matrix and its Cholesky factor. Numerical examples from real problems are used to illustrate the potential gains.

Winter2019 [420] M. Winter, D. Mlakar, R. Zayer, H.-P. Seidel, and M. Steinberger, "Adaptive sparse matrix-matrix multiplication on the gpu," in *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '19, Washington, District of Columbia: ACM, 2019, pp. 68–81, ISBN: 978-1-4503-6225-2. DOI: 10.1145/3293883.3295701,
Abstract: In the ongoing efforts targeting the vectorization of linear algebra primitives, sparse matrix-matrix multiplication (SpGEMM) has received considerably less attention than sparse Matrix-Vector multiplication (SpMV). While both are equally important, this disparity can be attributed mainly to the additional formidable challenges raised by SpGEMM.
In this paper, we present a dynamic approach for addressing SpGEMM on the GPU. Our approach works directly on the standard compressed sparse rows (CSR) data format. In comparison to previous SpGEMM implementations, our approach guarantees a homogeneous, load-balanced access pattern to the first input matrix and improves memory access to the second input matrix. It adaptively re-purposes GPU threads during execution and maximizes the time efficient on-chip scratchpad memory can be used. Adhering to a completely deterministic scheduling pattern guarantees bit-stable results during repetitive execution, a property missing from other approaches. Evaluation on an extensive sparse matrix benchmark suggests our approach being the fastest SpGEMM implementation for highly sparse matrices (80% of the set). When bit-stable results are sought, our approach is the fastest across the entire test set.

Wu2019 [421] R. Wu, "Dynamic scheduling strategy for block parallel cholesky factorization based on activity on edge network," *IEEE Access*, vol. 7, pp. 66 317–66 324, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2917714,
Abstract: The efficient development of system software and design applications in parallel architecture is a notable challenge considering various aspects, such as load balancing, memory spaces, communication, and synchronization. This paper presents a block parallel Cholesky factorization algorithm for a multicore system, which is developed based on activity on edge network. First, the basic block computing tasks and their dependencies are taken as vertices and edges, respectively, and a directed acyclic graph corresponding to the specific block parallel Cholesky factorization is generated. Next, each edge of the directed acyclic graph is assigned to a weight equal to the processing time of the initial vertex of the edge, and the directed acyclic graph becomes an activity on edge network with only one starting and one ending vertex. Finally, a queuing algorithm is designed for the basic block computing tasks according to the edge activity on edge network, and a dynamic scheduling strategy is developed for block parallel Cholesky factorization. The results of the experiments concerning the parallel execution time of the algorithm in multicore systems with different configurations demonstrate that the proposed algorithm has notable advantages compared with the traditional static scheduling algorithm, and it exhibits satisfactory load balancing, parallelism, and scalability capacities.

Xiao2019 [422] G. Xiao, K. Li, Y. Chen, W. He, A. Zomaya, and T. Li, "CASpMV: A customized and accelerative spmv framework for the sunway taihulight," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–1, 2019, ISSN: 1045-9219. DOI: 10.1109/TPDS.2019.2907537,
Abstract: The Sunway TaihuLight, equipped with 10 million cores, is currently the world's third fastest supercomputer. SpMV is one of core algorithms in many high-performance computing applications. This paper implements a fine-grained design for generic parallel SpMV based on the special Sunway architecture and finds three main performance limitations, i.e., storage limitation,

load imbalance, and huge overhead of irregular memory accesses. To address these problems, this paper introduces a customized and accelerative framework for SpMV (CASpMV) on the Sunway. The CASpMV customizes an auto-tuning four-way partition scheme for SpMV based on the proposed statistical model, which describes the sparse matrix structure characteristics, to make it better fit in with the computing architecture and memory hierarchy of the Sunway. Moreover, the CASpMV provides an accelerative method and customized optimizations to avoid irregular memory accesses and further improve its performance on the Sunway. Our CASpMV achieves a performance improvement that ranges from 588.05% to 2118.62% over the generic parallel SpMV on a CG (which corresponds to an MPI process) of the Sunway on average and has good scalability on multiple CGs. The performance comparisons of the CASpMV with state-of-the-art methods on the Sunway indicate that the sparsity and irregularity of data structures have less impact on CASpMV.

Xie2019 [423] Z. Xie, G. Tan, W. Liu, and N. Sun, "IA-SpGEMM: An input-aware auto-tuning framework for parallel sparse matrix-matrix multiplication," in *Proceedings of the 33rd ACM Conference on Supercomputing*, ser. ICS '19, Phoenix, AZ, USA, 2019. [Online]. Available: https://folk.idi.ntnu.no/weifengl/papers/spgemm_xie_ics19.pdf,
Abstract: Sparse matrix-matrix multiplication (SpGEMM) is a sparse kernel that is used in a number of scientific applications. Although several SpGEMM algorithms have been proposed, almost all of them are restricted to the compressed sparse row (CSR) format, and the possible performance gain from exploiting other formats has not been well studied. The particular format and algorithm that yield the best performance for SpGEMM also remain undetermined.
In this work, we conduct a prospective study on format-specific parallel SpGEMM algorithms, and analyze their pros and cons. We then propose IA-SpGEMM, an input-aware auto-tuning Framework for SpGEMM, that provides a unified programming interface in the CSR format and automatically determines the best format and algorithm for arbitrary sparse matrices. For this purpose, we set-up an algorithm set and design a deep learning model called MatNet that is trained by over 2,700 matrices from the SuiteSparse Matrix Collection to quickly and accurately predict the best solution by using sparse features and density representations. We evaluate our framework on CPUs and a GPU, and the results show that IA-SpGEMM is on average 3.27× and 13.17× faster than MKL on an Intel and an AMD platform, respectively, and is 2.23× faster than cuSPARSE on an NVIDIA GPU.

Xu2019 [424] Z. Xu, X. Chen, J. Shen, Y. Zhang, C. Chen, and C. Yang, "GARDENIA: A graph processing benchmark suite for next-generation accelerators," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 15, no. 1, 9:1–9:13, Jan. 2019, ISSN: 1550-4832. DOI: 10.1145/3283450,
Abstract: This article presents the Graph Algorithm Repository for Designing Next-generation Accelerators (GARDENIA), a benchmark suite for studying irregular graph algorithms on massively parallel accelerators. Applications with limited control and data irregularity are the main focus of existing generic benchmarks for accelerators, while available graph processing benchmarks do not apply state-of-the-art algorithms and/or optimization techniques. GARDENIA includes emerging graph processing workloads from graph analytics, sparse linear algebra, and machine-learning domains, which mimic massively multithreaded commercial programs running on modern large-scale datacenters. Our characterization shows that GARDENIA exhibits irregular microarchitectural behavior, which is quite different from structured workloads and straightforward-implemented graph benchmarks.

Yamamoto2019 [425] Y. Yamamoto, "High-performance algorithms for numerical linear algebra," in *The Art of High Performance Computing for Computational Science, Vol. 1: Techniques of Speedup and Parallelization for General Purposes*, M. Geshi, Ed. Singapore: Springer Singapore, 2019, pp. 113–136, ISBN: 978-981-13-6194-4. DOI: 10.1007/978-981-13-6194-4_7,
Abstract: Matrix computations lie at the heart of many scientific computations. While sophisticated algorithms have been established for various numerical linear algebra problems such as the solution of linear simultaneous equationsLinear simultaneous equation and eigenvalue problemsEigenvalue problem, they require considerable modification with the advent of exaFLOPS- scale

supercomputers, which are expected to have a huge number of computing cores, deep memory hierarchyMemory hierarchy, and increased probability of hardware errors. In this chapter, we discuss projected hardware characteristics of exaFLOPS machines and summarize the challenges to be faced by numerical linear algebra algorithms in the near future. Based on these preparations, we present a brief survey of recent research efforts in the field of numerical linear algebra targeted at meeting these challenges.

Yu2019 [426] T. Yu and M. Liu, "A memory efficient clique enumeration method for sparse graphs with a parallel implementation," *Parallel Computing*, 2019, ISSN: 0167-8191. DOI: https://doi.org/10.1016/j.parco.2019.05.005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167819118301297,

Abstract: Maximal clique enumeration (MCE) is a widely studied problem that plays a crucial role in structure mining of undirected graphs. The increasing scale of real-world graphs has brought the challenges of high memory cost and high CPU workload to the problem. In this paper, we propose a memory efficient method named CMC-bit for MCE on sparse graphs. It reduces the memory cost via minimizing the candidate cliques and representing them by the data structure bitset. It generates an appropriate order for the vertex set according to two optimized principles to reduce the CPU cost. We further design a partition-based CMC-bit algorithm with a one-side extending strategy to solve the memory-limited problem. We parallelize the CMC-bit algorithm based on MapReduce with a range-based partition strategy to make an optimal trade-off between the shuffling workload of graph decomposition and load balance in the Reduce phase. We conduct extensive experiments on 30 real-world datasets. The results demonstrate that both the CMC-bit algorithm and its parallel implementation significantly outperform the respective state-of-the-art algorithms in speed. We also show that the parallel CMC-bit algorithm achieves good performance on the scalability with respect to both the reducer number and the CPU number.

Zhang2019 [427] K. Zhang, J. Liu, J. Zhang, and J. Wang, "Greedy orthogonal pivoting algorithm for non-negative matrix factorization," in *Proceedings of the 36th International Conference on Machine Learning*, ser. PMLR'19, 2019. [Online]. Available: http://proceedings.mlr.press/v97/zhang19r/zhang19r.pdf,

Abstract: Non-negative matrix factorization is a powerful tool for learning useful representations in the data and has been widely applied in many problems such as data mining and signal processing. Orthogonal NMF, which can further improve the locality of decomposition, has drawn considerable interest in clustering problems. However, imposing simultaneous non-negative and orthogonal structure can be difficult, and so existing algorithms can only solve it approximately. To address this challenge, we propose an innovative procedure called Greedy Orthogonal Pivoting Algorithm (GOPA). The GOPA method fully exploits the sparsity of non-negative orthogonal solutions to break the global problem into a series of local optimizations, in which an adaptive subset of coordinates are updated in a greedy, closed-form manner. The biggest advantage of GOPA is that it promotes exact orthogonality and provides solid empirical evidence that stronger orthogonality does contribute favorably to better clustering performance. On the other hand, we have designed randomized and batch-mode version of GOPA, which can further reduce the computational cost and improve accuracy, making it suitable for large data.

Zheng2019 [428] H. Zheng, S. Vong, and L. Liu, "A direct preconditioned modulus-based iteration method for solving nonlinear complementarity problems of h-matrices," *Applied Mathematics and Computation*, vol. 353, pp. 396–405, 2019, ISSN: 0096-3003. DOI: 10.1016/j.amc.2019.02.015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0096300319301134,

Abstract: In this paper, we establish a direct preconditioned modulus-based iteration method for solving a class of nonlinear complementarity problems with the system matrix being an H-matrix. The convergence theorems of the proposed method are given, which generalize and improve the existing ones. Numerical examples show that the proposed method is efficient.

Zhou2019 [429] G. Zhou, Y. Feng, R. Bo, and T. Zhang, "GPU-accelerated sparse matrices parallel inversion algorithm for large-scale power systems," *International Journal of Electrical Power & Energy Systems*, vol. 111, pp. 34–43, 2019, ISSN: 0142-0615. DOI: https://doi.org/10.

1016/j.ijepes.2019.03.074. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142061518325109,
Abstract: State-of-the-art Graphics Processing Unit (GPU) has superior performances on float-pointing calculation and memory bandwidth, and therefore has great potential in many computationally intensive power system applications, one of which is the inversion of large-scale sparse matrix. It is a fundamental component for many power system analyses which requires to solve massive number of forward and backward substitution (F&B) subtasks and seems to be a good GPU-accelerated candidate application. By means of solving multiple F&B subtasks concurrently and a serial of performance tunings in compliance with GPU's architectures, we successfully develop a batch F&B algorithm on GPUs, which not only extracts the intra-level and intra-level parallelisms inside single F&B subtask but also explores a more regular parallelism among massive F&B subtasks, called inter-task parallelism. Case study on a 9241-dimension case shows that the proposed batch F&B solver consumes 2.92 μs per forward substitution (FS) subtask when the batch size is equal to 3072, achieving 65 times speedup relative to KLU library. And on the basis the complete design process of GPU-based inversion algorithm is proposed. By offloading the tremendous computational burden to GPU, the inversion of 9241-dimension case consumes only 97 ms, which can achieve 8.1 times speedup relative to the 12-core CPU inversion solver based on KLU library. The proposed batch F&B solver is practically very promising in many other power system applications requiring solving massive F&B subtasks, such as probabilistic power flow analysis.

Burkhardt2019 [430] P. Burkhardt, "Optimal algebraic breadth-first search for sparse graphs," arXiv: arXiv:1906.03113v1,
Abstract: There has been a rise in the popularity of algebraic methods for graph algorithms given the development of the GraphBLAS library and other sparse matrix methods. These are useful in practice because many graph algorithms are amenable to sparse matrix multiplication. An exemplar for these approaches is Breadth-First Search (BFS). The algebraic BFS algorithm is simply a recursion of matrix-vector multiplications with the $n \times n$ adjacency matrix. Despite many redundant operations over nonzeros that ultimately lead to suboptimal performance, the algebraic BFS is appealing for practical implementations because it is simple and embarrassingly parallel. By using highly tuned matrix libraries it can be faster in practice than the theoretically optimal combinatorial algorithm. Therefore an optimal algebraic BFS should be of keen interest especially if it is easily integrated with existing matrix methods.
Current methods, notably in the GraphBLAS, use a Sparse Matrix Sparse Vector (SpMSpV) multiplication in which the input vector is kept in a sparse representation in each step of the BFS. But simply applying SpMSpV in BFS does not lead to optimal runtime. Each nonzero in the vector must be masked in subsequent steps. This has been an area of recent recent in GraphBLAS and other libraries. While in theory these masking methods are asymptotically optimal on sparse graphs, many add work that leads to suboptimal runtime. We give a new optimal, algebraic BFS for sparse graphs that is also a constant factor faster than theoretically optimal SpMSpV methods. We show how to eliminate redundant operations so an element in the adjacency matrix is operated upon no more than once, thus taking $O(m)$ operations for a graph with $O(m)$ edges.
Our method multiplies progressively smaller submatrices of the adjacency matrix at each step. The matrix remains unchanged, rather we are masking the rows and columns in the matrix that corresponds to previously visited vertices. The input vector in each step is also effectively masked so it is a sparse vector. Thus our method multiplies a sparse submatrix by a sparse vector in decreasing size each step. Our sequential algebraic BFS algorithm takes $O(m)$ algebraic operations on a sparse graph as opposed to $O(mn)$ operations of other sparse matrix approaches. Our analysis closes a gap in the literature.

Gould2019 [431] T. R. Nicholas I. M. Gould and J. A. Scott, "Convergence and evaluation-complexity analysis of a regularized tensor-newton method for solving nonlinear least-squares problems," *Computational Optimization and Applications*, DOI: 10.1007/s10589-019-00064-2,
Abstract: Given a twice-continuously differentiable vector-valued function $r(x)$, a local minimizer of $\|r(x)\|_2$ is sought. We propose and analyse tensor-Newton methods, in which $r(x)$ is replaced locally by its second-order Taylor approximation. Convergence is controlled by regularization of various or-

ders. We establish global convergence to a first-order critical point of $\|r(x)\|_2$, and provide function evaluation bounds that agree with the best-known bounds for methods using second derivatives. Numerical experiments comparing tensor-Newton methods with regularized Gauss–Newton and Newton methods demonstrate the practical performance of the newly proposed method.

Villa2009 [432] O. Villa, D. Chavarría-Miranda, V. Gurumoorthi, A. Marquez, and S. Krishamoorthy, "Effects of floating-point non-associativity on numerical computations on massively multi-threaded systems," *Cray User Group*,

Abstract: Floating-point operations, as defined in the IEEE-754 standard, are not associative. The ordering of large numbers of operations (such as summa-tions) that deal with operands of substantially dif-ferent magnitudes can significantly affect the final result. On massively multi-threaded systems, the non-deterministic nature of how machine floating-point operations are interleaved, combined with the fact that intermediate values have to be rounded or truncated to fit in the available precision leads to non-deterministic numerical error propagation. We have investigated on a Cray XMT system the effect of non-deterministic error propagation by observing the convergence rate of a conjugate gradient calcula-tion used as part of a Power State Estimation (PSE) application. As a possible mitigation strategy, we have explored quadruple precision accumulation, as well as a deterministic parallel tree scheme. The tree based approach has consistently outperformed the quadruple precision approach due to an improved convergence rate. As a consequence, we motivate the need for compile time mechanisms that enable enforcement of parallel deterministic operations on the Cray XMT.