DOE Award/Contract Number: DE-SC0005136

Recipient/Contractor Organization: University of California, Berkeley with MIT as subcontractor.

STI Product Type: Final Technical Report

STI Product Title: Software Synthesis for High Productivity Exascale Computing

STI Product Date: 09/01/2010

Author(s): Bodik, Rastislav, Demmel, James, Solar-Lezama, Armando

Report/Product Number: DOE-UCB-0005136

Sponsoring DOE Program Office: USDOE

Description/Abstract: The length should be no more than 5,000 characters. To fill this field, you can cut and paste from any word processing file.

Over the three years of our project, we accomplished three key milestones:

We demonstrated how ideas from generative programming and software synthesis can help support the development of bulk-synchronous distributed memory kernels. These ideas are realized in a new language called MSL, a C-like language that combines synthesis features with high level notations for array manipulation and bulk-synchronous parallelism to simplify the semantic analysis required for synthesis. We also demonstrated that these high level notations map easily to low level C code and show that the performance of this generated code matches that of handwritten Fortran.

Second, we introduced the idea of solver-aided domain-specific languages (SDSLs), which are an emerging class of computer-aided programming systems. SDSLs ease the construction of programs by automating tasks such as verification, debugging, synthesis, and non-deterministic execution. SDSLs are implemented by translating the DSL program into logical constraints. Next, we developed a symbolic virtual machine called Rosette, which simplifies the construction of such SDSLs and their compilers. We have used Rosette to build SynthCL, a subset of OpenCL that supports synthesis.

Third, we developed novel numeric algorithms that move as little data as possible, either between levels of a memory hierarchy or between parallel processors over a network. We achieved progress in three aspects of this problem. First we determined lower bounds on communication. Second, we compared these lower bounds to widely used versions of these algorithms, and noted that these widely used algorithms usually communicate asymptotically more than is necessary. Third, we identified or invented new algorithms for most linear algebra problems that do attain these lower bounds, and demonstrated large speed-ups in theory and practice.

Recipient/Contractor Point of Contact: Rastislav Bodik

Summary of Publications:

"MSL: a Synthesis Enabled Language for Distributed Implementations"
Zhilei Xu, Shoaib Kamil, Armando Solar-Lezama
International Conference for High Performance Computing, Networking, Storage and Analysis, 2014.

Abstract:
This paper demonstrates how ideas from generative programming and software synthesis can help support the development of bulk-synchronous distributed memory kernels. These ideas are realized in a new language called MSL, a C-like language that combines synthesis features with high level notations for array manipulation and bulk-synchronous parallelism to simplify the semantic analysis required for synthesis. The paper shows that by leveraging these high level notations, it is possible to scale the synthesis and automated bug-finding technologies that underlie MSL to realistic computational kernels. Specifically, we demonstrate the methodology through case studies implementing non-trivial distributed kernels—both regular and irregular—from the NAS parallel benchmarks. We show that our approach can automatically infer many challenging details from these benchmarks and can enable high level implementation ideas to be reused between similar kernels. We also demonstrate that these high level notations map easily to low level C code and show that the performance of this generated code matches that of handwritten Fortran.

"Chlorophyll: Synthesis-Aided Compiler for Low-Power Spatial Architectures"
Phitchaya Mangpo Phothilimthana, Tikhon Jelves, Rohin Shah, Nishant Totla, Sarah Chasins, and Rastislav Bodik
ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 2014

Abstract:
We developed Chlorophyll, a synthesis-aided programming model and compiler for the GreenArrays GA144, an extremely minimalist low-power spatial architecture that requires partitioning the program into fragments of no more than 256 instructions and 64 words of data. This processor is 100-times more energy efficient than its competitors, but currently can only be programmed using a low-level stack-based language.
The Chlorophyll programming model allows programmers to provide human insight by specifying partial partitioning of data and computation. The Chlorophyll compiler relies on synthesis, sidestepping the need to develop classical optimizations, which may be challenging given the unusual architecture. To scale synthesis to real problems, we decompose the compilation into smaller synthesis subproblems—partitioning, layout, and code generation. We show that the synthesized programs are no more than 65% slower than highly optimized expert-written programs and are faster than programs produced by a heuristic, non-synthesizing version of our compiler.

"A Lightweight Symbolic Virtual Machine for Solver-Aided Host Languages"
Emina Torlak and Rastislav Bodik
ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 2014.
Abstract:

Solver-aided domain-specific languages (SDSLs) are an emerging class of computer-aided programming systems. They ease the construction of programs by using satisfiability solvers to automate tasks such as verification, debugging, synthesis, and non-deterministic execution. But reducing programming tasks to satisfiability problems involves translating programs to logical constraints, which is an engineering challenge even for domain-specific languages.
We have previously shown that translation to constraints can be avoided if SDSLs are implemented by (traditional) embedding into a host language that is itself solver-aided. This paper describes how to implement a symbolic virtual machine (SVM) for such a host language. Our symbolic virtual machine is lightweight because it compiles to constraints only a small subset of the host's constructs, while allowing SDSL designers to use the entire language, including constructs for DSL embedding. This lightweight compilation employs a novel symbolic execution technique with two key properties: it produces compact encodings, and it enables concrete evaluation to strip away host constructs that are outside the subset compilable to constraints. Our symbolic virtual machine architecture is at the heart of ROSETTE, a solver-aided language that is host to several new SDSLs.

"Growing Solver-Aided Languages with Rosette"
Emina Torlak and Rastislav Bodik
Symposium on New Ideas in Programming and Reflections on Software (Onward!), 2013.

Abstract:
SAT and SMT solvers have automated a spectrum of programming tasks, including program synthesis, code checking, bug localization, program repair, and programming with oracles. In principle, we obtain all these benefits by translating the program (once) to a constraint system understood by the solver. In practice, however, compiling a language to logical formulas is a tricky process, complicated by having to map the solution back to the program level and extend the language with new solver-aided constructs, such as symbolic holes used in synthesis.
This paper introduces ROSETTE, a framework for designing solver-aided languages. ROSETTE is realized as a solveraided language embedded in Racket, from which it inherits extensive support for meta-programming. Our framework frees designers from having to compile their languages to constraints: new languages, and their solver-aided constructs, are defined by shallow (library-based) or deep (interpreterbased) embedding in ROSETTE itself.
We describe three case studies, by ourselves and others, of using ROSETTE to implement languages and synthesizers for web scraping, spatial programming, and superoptimization of bitvector programs.

"Parallel Schedule Synthesis for Attribute Grammars"
Leo A. Meyerovich, Matthew E. Torok, Eric Atkinson, Rastislav Bodik
18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), 2013.

Abstract:
We examine how to synthesize a parallel schedule of structured traversals over trees. In our system, programs are declaratively specified as attribute grammars. Our synthesizer automatically, correctly, and quickly schedules the attribute grammar as a composition of parallel tree traversals. Our downstream compiler optimizes for GPUs and multicore CPUs.

We provide support for designing efficient schedules. First, we introduce a declarative language of schedules where programmers may constrain any part of the schedule and the synthesizer will complete and autotune the rest. Furthermore, the synthesizer answers debugging queries about how schedules may be completed.
We evaluate our approach with two case studies. First, we created the first parallel schedule for a large fragment of CSS and report a 3X multicore speedup. Second, we created an interactive GPU-accelerated animation of over 100,000 nodes.

"Accuracy of the s-step Lanczos method for the symmetric eigenproblem"
Erin Carson and James Demmel
UC Berkeley EECS Tech report UCB/EECS-2014-165, 2014

Abstract.
The s-step Lanczos method is an attractive alternative to the classical Lanczos method as it enables an O(s) reduction in data movement over a xed number of iterations. This can significantly improve performance on modern computers. In order for s-step methods to be widely adopted, it is important to better understand their error properties. Although the s-step Lanczos method is equivalent to the classical Lanczos method in exact arithmetic, empirical observations demonstrate that it can behave quite differently in finite precision.
In this paper, we demonstrate that bounds on accuracy for the nite precision Lanczos method given by Paige [Lin. Alg. Appl. , 34:235{258, 1980] can be extended to the s-step Lanczos case assuming a bound on the condition numbers of the computed s-step bases. Our results confirm theoretically what is well-known empirically: the conditioning of the Krylov bases plays a large role in determining nite precision behavior. In particular, if one can guarantee that the basis condition number is not too large throughout the iterations, the accuracy and convergence of eigenvalues in the s-step Lanczos method should be similar to those of classical Lanczos. This indicates that, under certain restrictions, the s-step Lanczos method can be made suitable for use in many practical cases.

"Parallel Reproducible Summation"
James Demmel and Hong Diep Nguyen
IEEE Transactions on Computers, August 2014

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 n*2^(-28)*macheps*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. 1 reduction of an array of 6 double

precision floating point numbers per sum, and hence is well suited for massively parallel environments.

"Contention Bounds for Combinations of Computation Graphs and Network Topologies"
Grey Ballard, James Demmel, Andrew Gearhart, Benjamin Lipshitz, Oded Schwartz, and Sivan Toledo
UC Berkeley EECS Tech Report UCB/EECS-2014-147, Aug 2014

Abstract:
Network topologies can have significant effect on the costs of algorithms due to inter-processor communication. Parallel algorithms that ignore network topology can suffer from contention along network links. However, for particular combinations of computations and network topologies, costly network contention may inevitably become a bottleneck, even for optimally designed algorithms. We obtain a novel contention lower bound that is a function of the network and the computation graph parameters. To this end, we compare the communication bandwidth needs of subsets of processors and the available network capacity (as opposed to per-processor analysis in most previous studies). Applying this analysis we improve communication cost lower bounds for several combinations of fundamental computations on common network topologies.

"A massively parallel tensor contraction framework for coupled-cluster computations"
Edgar Solomonik, Devin Matthews, Jeff Hammond, John Stanton, James Demmel
Journal of Parallel and Distributed Computing, v. 74, i. 12, Dec 2014
UC Berkeley EECS Tech Report UCB/EECS-2014-143, Aug 2014

Abstract:
Precise calculation of molecular electronic wavefunctions by methods such as coupled-cluster requires the computation of tensor contractions, the cost of which has polynomial computational scaling with respect to the system and basis set sizes. Each contraction may be executed via matrix multiplication on a properly ordered and structured tensor. However, data transpositions are often needed to reorder the tensors for each contraction. Writing and optimizing distributed-memory kernels for each transposition and contraction is tedious since the number of contractions scales combinatorially with the number of tensor indices. We present a distributed-memory numerical library (Cyclops Tensor Framework (CTF)) that automatically manages tensor blocking and redistribution to perform any user-specified contractions. CTF serves as the distributed memory contraction engine in Aquarius, a new program designed for high-accuracy and massively-parallel quantum chemical computations. Aquarius implements a range of coupled-cluster and related methods such as CCSD and CCSDT by writing the equations on top of a C++ templated domain-specific language. This DSL calls CTF directly to manage the data and perform the contractions. Our CCSD and CCSDT implementations achieve high parallel scalability on the BlueGene/Q and Cray XC30 supercomputer architectures showing that accurate electronic structure calculations can be effectively carried out on top of general distributed memory tensor primitives.

"Communication lower bounds and optimal algorithms for numerical linear algebra"
Grey Ballard, Erin Carson, James Demmel, Mark Hoemmen, Nicholas Knight, Oded Schwartz
Acta Numerica, v 23, May 2014, pages 1-155 (invited)

Abstract:
The traditional metric for the efficiency of a numerical algorithm has been the number of arithmetic operations it performs. Technological trends have long been reducing the time to perform an arithmetic operation, so it is no longer the bottleneck in many algorithms; rather, communication, or moving data, is the bottleneck. This motivates us to seek algorithms that move as little data as possible, either between levels of a memory hierarchy or between parallel processors over a network. In this paper we summarize recent progress in three aspects of this problem. First we describe lower bounds on communication. Some of these generalize known lower bounds for dense classical (O(n^3)) matrix multiplication to all direct methods of linear algebra, to sequential and parallel algorithms, and to dense and sparse matrices. We also present lower bounds for Strassen-like algorithms, and for iterative methods, in particular Krylov subspace methods applied to sparse matrices. Second, we compare these lower bounds to widely used versions of these algorithms, and note that these widely used algorithms usually communicate asymptotically more than is necessary. Third, we identify or invent new algorithms for most linear algebra problems that do attain these lower bounds, and demonstrate large speed-ups in theory and practice.

"s-step Krylov Subspace Methods as Bottom Solvers for Geometric Multigrid"
Sam Williams, Mike Lijewski, Ann Almgren, Brian Van Straalen, Erin Carson, Nicholas Knight, James Demmel
IEEE 28th International Parallel & Distibuted Processing Symposium (IPDPS'14), May 2014

Abstract:
Geometric multigrid solvers within adaptive mesh refinement (AMR) applications often reach a point where further coarsening of the grid becomes impractical as individual subdomain sizes approach unity. At this point the most common solution is to use a bottom solver, such a BiCGStab, to reduce the residual by a fixed factor at the coarsest level. Each iteration of BiCGStab requires multiple global reductions (MPI collectives). As the number of BiCGStab iterations required for convergence grows with problem size, and the time for each collective operation increases with machine scale, bottom solves in large-scale applications can constitute a significant fraction of the overall multigrid solve time. In this paper, we implement, evaluate, and optimize a communication-avoiding s-step formulation of BiCGStab (CABiCGStab for short) as a high-performance, distributed-memory bottom solver for geometric multigrid solvers. This is the first time s-step Krylov subspace methods have been leveraged to improve multigrid bottom solver performance. We use a synthetic benchmark for detailed analysis and integrate the best implementation into BoxLib in order to evaluate the benefit of a s-step Krylov subspace method on the multigrid solves found in the applications LMC and Nyx on up to 32,768 cores on the Cray XE6 at NERSC. Overall, we see bottom solver improvements of up to 4.2x on synthetic problems and up to 2.7× in real applications. This results in as much as a 1.5× improvement in solver performance in real application.

"Error analysis of the s-step Lanczos method in finite precision"
Erin Carson, James Demmel
UC Berkeley EECS Tech Report UCB/EECS-2014-55, May 6, 2014

Abstract:

The s-step Lanczos method is an attractive alternative to the classical Lanczos method as it enables an O(s) reduction in data movement over a fixed number of iterations. This can significantly improve performance on modern computers. In order for s-step methods to be widely adopted, it is important to better understand their error properties. Although the s-step Lanczos method is equivalent to the classical Lanczos method in exact arithmetic, empirical observations demonstrate that it can behave quite differently in finite precision. In the s-step Lanczos method, the computed Lanczos vectors can lose orthogonality at a much quicker rate than the classical method, a property which seems to worsen with increasing s.
In this paper, we present, for the first time, a complete rounding error analysis of the s-step Lanczos method. Our methodology is analogous to Paige's rounding error analysis for the classical Lanczos method [IMA J. Appl. Math., 18(3):341-349, 1976]. Our analysis gives upper bounds on the loss of normality of and orthogonality between the computed Lanczos vectors, as well as a recurrence for the loss of orthogonality. The derived bounds are very similar to those of Paige for classical Lanczos, but with the addition of an amplification term which depends on the condition number of the Krylov bases computed every s-steps. Our results confirm theoretically what is well-known empirically: the conditioning of the Krylov bases plays a large role in determining finite precision behavior.

"Tradeoffs between synchronization, communication and work in parallel linear algebra computations"
Edgar Solomonik, Erin Carson, Nicholas Knight, James Demmel
UC Berkeley EECS Tech Report UCB/EECS-2014-8, Jan 25, 2014

Abstract:
This paper derives tradeoffs between three basic costs of a parallel algorithm: synchronization, data movement, and computational cost. Our theoretical model counts the amount of work and data movement as a maximum of any execution path during the parallel computation. By considering this metric, rather than the total communication volume over the whole machine, we obtain new insight into the characteristics of parallel schedules for algorithms with non-trivial dependency structures. The tradeoffs we derive are lower bounds on the execution time of the algorithm which are independent of the number of processors, but dependent on the problem size. Therefore, these tradeoffs provide lower bounds on the parallel execution time of any algorithm computed by a system composed of any number of homogeneous components each with associated computational, communication, and synchronization payloads. We first state our results for general graphs, based on expansion parameters, then we apply the theorem to a number of specific algorithms in numerical linear algebra, namely triangular substitution, Gaussian elimination, and Krylov subspace methods. Our lower bound for LU factorization demonstrates the optimality of Tiskin's LU algorithm [24] answering an open question posed in his paper, as well as of the 2.5D LU [20] algorithm which has analogous costs. We treat the computations in a general manner by noting that the computations share a similar dependency hypergraph structure and analyzing the communication requirements of lattice hypergraph structures.

"Reconstructing Householder Vectors from Tall-Skinny QR"
Grey Ballard, James Demmel, Laura Grigori, Mathias Jacquelin, Hong Diep Nguyen, Edgar Solomonik
UC Berkeley EECS Tech Report UCB/EECS-2013-175, Oct 26 2013

Abstract:
The Tall-Skinny QR (TSQR) algorithm is more communication efficient than the standard Householder algorithm for QR decomposition of matrices with many more rows than columns. However, TSQR produces a different representation of the orthogonal factor and therefore requires more software development to support the new representation. Further, implicitly applying the orthogonal factor to the trailing matrix in the context of factoring a square matrix is more complicated and costly than with the Householder representation.
We show how to perform TSQR and then reconstruct the Householder vector representation with the same asymptotic communication efficiency and little extra computational cost. We demonstrate the high performance and numerical stability of this algorithm both theoretically and empirically. The new Householder reconstruction algorithm allows us to design more efficient parallel QR algorithms, with significantly lower latency cost compared to Householder QR and lower bandwidth and latency costs compared with Communication-Avoiding QR (CAQR) algorithm. As a result, our final parallel QR algorithm outperforms ScaLAPACK and Elemental implementations of Householder QR and our implementation of CAQR on the Hopper Cray XE6 NERSC system.

"Communication Costs of Strassen's Matrix Multiplication"
Grey Ballard, James Demmel, Olga Holtz, Oded Schwartz
Research Highlight (invited), Communications of the ACM, Feb 2014 (v. 57, n. 2)

Abstract:
Algorithms have historically been evaluated in terms of the number of arithmetic operations they performed. This analysis is no longer sufficient for predicting running times on today's machines. Moving data through memory hierarchies and among processors requires much more time (and energy) than performing computations. Hardware trends suggest that the relative costs of this communication will only increase. Proving lower bounds on the communication of algorithms and finding algorithms that attain these bounds are therefore fundamental goals. We show that the communication cost of an algorithm is closely related to the graph expansion properties of its corresponding computation graph.
Matrix multiplication is one of the most fundamental problems in scientific computing and in parallel computing. Applying expansion analysis to Strassen's and other fast matrix multiplication algorithms, we obtain the first lower bounds on their communication costs. These bounds show that the current sequential algorithms are optimal but that previous parallel algorithms communicate more than necessary. Our new parallelization of Strassen's algorithm is communication-optimal and outperforms all previous matrix multiplication algorithms.

"Communication Efficient Gaussian Elimination with Partial Pivoting using a Shape Morphing Data Layout"
Grey Ballard, James Demmel, Benjamin Lipshitz, Oded Schwartz, Sivan Toledo
25th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'13)

Abstract:
High performance for numerical linear algebra often comes at the expense of stability. Computing the LU decomposition of a matrix via Gaussian Elimination can be organized so that the computation involves regular and efficient data access. However, maintaining numerical stability via partial pivoting involves row interchanges that lead to inefficient data access patterns. To optimize communication efficiency throughout the memory hierarchy we confront two seemingly

contradictory requirements: partial pivoting is efficient with column-major layout, whereas a block-recursive layout is optimal for the rest of the computation. We resolve this by introducing a shape morphing procedure that dynamically matches the layout to the computation throughout the algorithm, and show that Gaussian Elimination with partial pivoting can be performed in a communication efficient and cache-oblivious way. Our technique extends to QR decomposition, where computing Householder vectors prefers a different data layout than the rest of the computation.

"Communication Optimal Parallel Multiplication of Sparse Random Matrices"
Grey Ballard, Aydin Buluc, James Demmel, Laura Grigori, Benjamin Lipshitz,
Oded Schwartz, Sivan Toledo
25th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'13)

Abstract:
Parallel algorithms for sparse matrix-matrix multiplication typically spend most of their time on inter-processor communication rather than on computation, and hardware trends predict the relative cost of communication will only increase. Thus, sparse matrix multiplication algorithms must minimize communication costs in order to scale to large processor counts.
In this paper, we consider multiplying sparse matrices corresponding to Erdos-R_enyi random graphs on distributed-memory parallel machines. We prove a new lower bound on the expected communication cost for a wide class of algorithms. Our analysis of existing algorithms shows that, while some are optimal for a limited range of matrix density and number of processors, none is optimal in general. We obtain two new parallel algorithms and prove that they match the expected communication cost lower bound, and hence they are optimal.

"Implementing a Blocked Aasen's Algorithm with a Dynamic Scheduler on Multicore Architectures"
Grey Ballard, Dulcenia Becker, James Demmel, Jack Dongarra, Alex Druinsky,
Inon Peled, Oded Schwartz, Sivan Toledo, Ichitaro Yamazaki
IEEE International Parallel & Distributed Processing Symposium 2013 (IPDPS'13)

Abstract:
Factorization of a dense symmetric indefinite matrix is a key computational kernel in many scientific and engineering simulations. However, there is no scalable factorization algorithm that takes advantage of the symmetry and guarantees numerical stability through pivoting at the same time. This is because such an algorithm exhibits many of the fundamental challenges in parallel programming like irregular data accesses and irregular task dependencies. In this paper, we address these challenges in a tiled implementation of a blocked Aasen's algorithm using a dynamic scheduler. To fully exploit the limited parallelism in this left-looking algorithm, we study several performance enhancing techniques; e.g., parallel reduction to update a panel, tall-skinny LU factorization algorithms to factorize the panel, and a parallel implementation of symmetric pivoting. Our performance results on up to 48 AMD Opteron processors demonstrate that our implementation obtains speedups of up to 2.8 over MKL, while losing only one or two digits in the computed residual norms.

"Perfect Strong Scaling Using no Addition Energy"
James Demmel, Andrew Gearhart, Benjamin Lipshitz, Oded Schwartz
IEEE International Parallel & Distributed Processing Symposium 2013 (IPDPS'13)

Abstract:
Energy efficiency of computing devices has become a dominant area of research interest in recent years. Most previous work has focused on architectural techniques to improve power and energy efficiency; only a few consider saving energy at the algorithmic level. We prove that a region of perfect strong scaling in energy exists for matrix multiplication (classical and Strassen) and the direct n-body problem via the use of algorithms that use all available memory to replicate data. This means that we can increase the number of processors by some factor and decrease the runtime (both computation and communication) by the same factor, without changing the total energy use.

"Communication-Optimal Parallel Recursive Rectangular Matrix Multiplication"
James Demmel, David Eliahu, Armando Fox, Shoaib Kamil, Benjamin Lipshitz,
Oded Schwartz, Omer Spillinger
IEEE International Parallel & Distributed Processing Symposium 2013 (IPDPS'13)

Abstract:
Communication-optimal algorithms are known for square matrix multiplication. Here, we obtain the first communication-optimal algorithm for all dimensions of rectangular matrices. Combining the dimension-splitting technique of Frigo, Leiserson, Prokop and Ramachandran (1999) with the recursive BFS/DFS approach of Ballard, Demmel, Holtz, Lipshitz and Schwartz (2012) allows for a communication-optimal as well as cache- and network-oblivious algorithm. Moreover, the implementation is simple: approximately 50 lines of code for the shared-memory version. Since the new algorithm minimizes communication across the network, between NUMA domains, and between levels of cache, it performs well in practice on both shared- and distributed-memory machines. We show significant speedups over existing parallel linear algebra libraries both on a 32-core shared-memory machine and on a distributed-memory supercomputer.

"Avoiding Communication in Dense Linear Algebra"
Grey Ballard
PhD Dissertation, Computer Science Division, University of California, Berkeley
Aug 16, 2013

Abstract:
Dense linear algebra computations are essential to nearly every problem in scientific computing and to countless other elds. Most matrix computations enjoy a high computational intensity (i.e., ratio of computation to data), and therefore the algorithms for the computations have a potential for high efficiency. However, performance for many linear algebra algorithms is limited by the cost of moving data between processors on a parallel computer or throughout the memory hierarchy of a single processor, which we will refer to generally as communication. Technological trends indicate that algorithmic performance will become even more limited by communication in the future. In this thesis, we consider the fundamental computations within dense linear algebra and address the following question: can we significantly improve the current algorithms for these computations, in terms of the communication they require and their performance in practice? To answer the question,

we analyze algorithms on sequential and parallel architectural models that are simple enough to determine coarse communication costs but accurate enough to predict performance of implementations on real hardware. For most of the computations, we prove lower bounds on the communication that any algorithm must perform. If an algorithm exists with communication costs that match the lower bounds (at least in an asymptotic sense), we call the algorithm communication optimal. In many cases, the most commonly used algorithms are not communication optimal, and we can develop new algorithms that require less data movement and attain the communication lower bounds.
In this thesis, we develop both new communication lower bounds and new algorithms, tightening (and in many cases closing) the gap between best known lower bound and best known algorithm (or upper bound). We consider both sequential and parallel algorithms, and we asses both classical and fast algorithms (e.g., Strassen's matrix multiplication algorithm).
In particular, the central contributions of this thesis are
_1) proving new communication lower bounds for nearly all classical direct linear algebra computations (dense or sparse), including factorizations for solving linear systems, least squares problems, and eigenvalue and singular value problems,
_2) proving new communication lower bounds for Strassen's and other fast matrix multiplication algorithms,
_3) proving new parallel communication lower bounds for classical and fast computations that set limits on an algorithm's ability to perfectly strong scale,
_4) summarizing the state-of-the-art in communication efficiency for both sequential and parallel algorithms for the computations to which the lower bounds apply,
_5) developing a new communication-optimal algorithm for computing a symmetric-indefinite factorization (observing speedups of up to 2.8x compared to alternative shared- memory parallel algorithms),
_6) developing new, more communication-efficient algorithms for reducing a symmetric band matrix to tridiagonal form via orthogonal transformations (observing speedups of 2-6x compared to alternative sequential and parallel algorithms), and
_7) developing a new communication-optimal parallelization of Strassen's matrix multiplication algorithm (observing speedups of up to 2.84x
_compared to alternative distributed-memory parallel algorithms).

“Communication Lower Bounds and Optimal Algorithm for Programs that Reference Arrays”
Michael Christ, James Demmel, Nicholas Knight, Thomas Scanlon, Katherine Yelick

Abstract:
Communication, i.e., moving data, between levels of a memory hierarchy or between parallel processors on a network, can greatly dominate the cost of computation, so algorithms that minimize communication can run much faster (and use less energy) than algorithms that do not. Motivated by this, attainable communication lower bounds were established for a variety of algorithms including matrix computations. The lower bound approach used before Theta(n^3) _matrix multiplication, and later for many other linear algebra algorithms, depended on a geometric result by Loomis and Whitney: this result bounded the volume of a 3D set (representing multiply-adds done in the inner loop of the algorithm) using the product of the areas of certain 2D projections of this set (representing the matrix entries available locally, i.e., without communication). Using a recent generalization of Loomis' and Whitney's result, we generalize this lower bound approach to a much larger class of algorithms, that may have arbitrary numbers of loops and arrays with arbitrary dimensions, as long as the index expressions are affine combinations of loop variables. In other

words, the algorithm can do arbitrary operations on any number of variables like A(i1, i2, i2 - 2*i1, 3-4*i3 + 7*i4,…). Moreover, the result applies to recursive programs, irregular iteration spaces, sparse matrices, and other data structures as long as the computation can be logically mapped to loops and indexed data structure accesses.
We also discuss when optimal algorithms exist that attain the lower bounds; this leads to new asymptotically faster algorithms for several problems.

"Exploiting Data Sparsity in Parallel Matrix Powers Computations"
Nicholas Knight, Erin Carson, James Demmel
Parallel Processing and Applied Mathematics,
Eds: R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Waniewski, eds.,
Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2014, pp.15-25

Abstract:
The increasingly high relative cost of moving data on modern parallel machines has caused a paradigm shift in the design of high-performance algorithms: to achieve efficiency, one must focus on strategies which minimize data movement, rather than minimize arithmetic operations. We call this a communication-avoiding approach to algorithm design.
In this work, we derive a new parallel communication-avoiding matrix powers algorithm for matrices of the form A = D + U*S*U^H, where D is sparse and U*S*V^H has low rank but may be dense. Matrices of this form arise in many practical applications, including power-law graph analysis, circuit simulation, and algorithms involving hierarchical (H) matrices, such as multigrid methods, fast multipole methods, numerical partial differential equation solvers, and preconditioned iterative methods. If A has this form, our algorithm enables a communication-avoiding approach. We demonstrate that, with respect to the cost of computing k sparse matrix-vector multiplications, our algorithm asymptotically reduces the parallel latency by a factor of O(k) for small additional bandwidth and computation costs. Using problems from real-world applications, our performance model predicts that this reduction in communication allows for up to 24x speedups on petascale machines.

"Communication avoiding rank revealing QR factorization with column pivoting"
James Demmel, Laura Grigori, Ming Gu, Hua Xiang,
UC Berkeley EECS Tech Report UCB/EECS-2013-46, May 2013,
to appear in SIAM J. Matrix Anal. Appl.

Abstract:
In this paper we introduce CARRQR, a communication avoiding rank revealing QR factorization with tournament pivoting. We show that CARRQR reveals the numerical rank of a matrix in an analogous way to QR factorization with column pivoting (QRCP). Although the upper bound of a quantity involved in the characterization of a rank revealing factorization is worse for CARRQR than for QRCP, our numerical experiments on a set of challenging matrices show that this upper bound is very pessimistic, and CARRQR is an effective tool in revealing the rank in practical problems. Our main motivation for introducing CARRQR is that it minimizes data transfer, modulo poly-logarithmic factors, on both sequential and parallel machines, while previous factorizations as QRCP are communication sub-optimal and require asymptotically more communication than CARRQR. Hence CARRQR is expected to have a better performance on current and future computers, where communication is a major bottleneck that highly impacts the performance of an algorithm.

"Cyclops Tensor Framework: reducing communication and eliminating load imbalance in massively parallel contractions"
Edgar Solomonik, Devin Matthews, Jeff Hammond, James Demmel
UC Berkeley Tech Report UCB/EECS-2013-11, Feb 13, 2013
IEEE International Parallel & Distributed Processing Symposium 2013 (IPDPS'13)

Abstract:
Cyclops (cyclic-operations) Tensor Framework (CTF) is a distributed library for tensor contractions. CTF aims to scale high-dimensional tensor contractions such as those required in the Coupled Cluster (CC) electronic structure method to massively-parallel supercomputers. The framework preserves tensor structure by subdividing tensors cyclically, producing a regular parallel decomposition. An internal virtualization layer provides completely general mapping support while maintaining ideal load balance. The mapping framework decides on the best mapping for each tensor contraction at run-time via explicit calculations of memory usage and communication volume. CTF employs a general redistribution kernel, which transposes tensors of any dimension between arbitrary distributed layouts, yet touches each piece of data only once. Sequential symmetric contractions are reduced to matrix multiplication calls via tensor index transpositions and partial unpacking. The user-level interface elegantly expresses arbitrary-dimensional generalized tensor contractions in the form of a domain specific language.
We demonstrate performance of CC with single and double excitations on 8192 nodes of Blue Gene/Q and show that CTF outperforms NWChem on Cray XE6 supercomputers for benchmarked systems.

"Minimizing communication in all-pairs shortest paths"
Edgar Solomonik, Aydin Buluc, James Demmel
UC Berkeley EECS Tech Report UCB/EECS-2013-10, Feb 13 2013
IEEE International Parallel & Distributed Processing Symposium 2013 (IPDPS'13)

Abstract:
We consider distributed memory algorithms for the all-pairs shortest paths (APSP) problem. Scaling the APSP problem to high concurrencies requires both minimizing inter-processor communication as well as maximizing temporal data locality. The 2.5D APSP algorithm, which is based on the divide-and-conquer paradigm, satisfies both of these requirements: it can utilize any extra available memory to perform asymptotically less communication, and it is rich in semiring matrix multiplications, which have high temporal locality. We start by introducing a block-cyclic 2D (minimal memory) APSP algorithm. With a careful choice of block-size, this algorithm achieves known communication lower-bounds for latency and bandwidth. We extend this 2D block-cyclic algorithm to a 2.5D algorithm, which can use c extra copies of data to reduce the bandwidth cost by a factor of sqrt(c), compared to its 2D counterpart. However, the 2.5D algorithm increases the latency cost by sqrt(c). We provide a tighter lower bound on latency, which dictates that the latency overhead is necessary to reduce bandwidth along the critical path of execution. Our implementation achieves impressive performance and scaling to 24,576 cores of a Cray XE6 supercomputer by utilizing well-tuned intra-node kernels within the distributed memory algorithm.

"Autotuning sparse Matrix-Vector Multiplication for Multicore"
Jong-Ho Byun, Richard Lin, Katherine Yelick, James Demmel
UC Berkeley EECS Tech Report UCB/EECS-2012-2015, Nov 28, 2012

Abstract:
Sparse matrix-vector multiplication (SpMV) is an important kernel in scientific and engineering computing. Straightforward parallel implementations of SpMV often perform poorly, and with the increasing variety of architectural features in multicore processors, it is getting more difficult to determine the sparse matrix data structure and corresponding SpMV implementation that optimize performance. In this paper we present pOSKI, an autotuning system for SpMV that automatically searches over a large set of possible data structures and implementations to optimize SpMV performance on multicore platforms. pOSKI explores a design space that depends on both the nonzero pattern of the sparse matrix, typically not known until run-time, and the architecture, which is explored off-line as much as possible, in order to reduce tuning time. We demonstrate significant performance improvements compared to previous serial and parallel implementations, and compare performance to upper bounds based on architectural models.

Software to include:

pOSKI: Parallel Optimized Sparse Kernel Interface
Website: bebop.cs.berkeley.edu/poski

The parallel Optimized Sparse Kernel Interface (pOSKI) Library is a collection of kernels that provides automatically tuned (autotuned) high performance computational kernels for sparse matrices, such as Sparse-Matrix-Vector-Multiplication (SpMV). pOSKI targets both uniprocessor and multicore machines. pOSKI builds on prior work on the OSKI library, which provided autotuned kernels for SpMV and other kernels on cache-based superscalar uniprocessors. The purpose of both pOSKI and OSKI is to make it easy for developers of solver libraries, and of scientific and engineering applications, to more easily attain high performance in commonly used sparse matrix operations, via autotuning.
Sketch
https://bitbucket.org/gatoatigrado/sketch-frontend/wiki/Home

SKETCH is a software synthesis tool that allows for rapid development of highly tuned bug-free algorithm implementations. To do this, the programmer develops a sketch, or partial implementation, and a separate specification of the desired functionality. The synthesizer then completes the sketch to behave like the specification. The correctness of the synthesized implementation is guaranteed by the compiler.

Rosette: a solver-aided host language
Web site: https://github.com/emina/rosette

Rosette is a new *solver-aided programming language* that extends Racket with facilities for program synthesis, verification, debugging, and angelic execution. These facilities are based on satisfiability solvers. To find out if a program is buggy, for example, Rosette's *symbolic virtual machine* (SVM) compiles it to logical constraints and uses an off-the-shelf solver to search for an input that triggers a failure. Thanks to the virtualized access to the underlying solver, and the powerful metaprogramming features inherited from Racket, Rosette makes it easy to develop new domain-specific solver-aided languages (SDSLs) by simply writing an interpreter or a library.