Final Report on DOE Grant No. DE-SC0010200

Title: EXTREME-SCALE ALGORITHMS & SOFTWARE RESILIENCE (EASIR) ARCHITECTURE-AWARE ALGORITHMS FOR SCALABLE PERFORMANCE AND RESILIENCE ON HETEROGENEOUS ARCHITECTURES

Grant Name: THE REGENTS OF THE UNIVERSITY OF CALIFORNIA

PERIOD OF PERFORMANCE: 06/15/13 THRU 06/14/17

PRINCIPAL INVESTIGATOR: DR. JAMES DEMMEL

Progress/Accomplishments

Before reporting on new work, we note that some previously reported work has now appeared in journals [1,2,3,4].

Paper [5] explores communication-avoiding iterative solvers in multigrid. 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 as 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 2.7x in real applications. This results in as much as a 1.5x improvement in solver performance in real applications.

Paper [6] extends the concept of communication-avoiding algorithms to write avoiding algorithms. This is motivated by the fact that previous work does not distinguish between the costs of reads and writes, even though writes can be much more expensive than reads in some current and emerging technologies. The first example is nonvolatile memory, such as Flash and Phase Change Memory. Second, in cloud computing frameworks like MapReduce, Hadoop, and Spark, intermediate results may be written to disk for reliability purposes, whereas read-only data may be kept in DRAM. Third, in a shared memory environment, writes may result in more coherency traffic over a shared bus than reads.

This motivates us to first ask whether there are lower bounds on the number of writes that certain algorithms must perform, and when these bounds are

such algorithms “write-avoiding (WA)", to distinguish them from “communication-avoiding (CA)" algorithms, which only minimize the sum of reads and writes. We identify a number of cases in linear algebra and direct N-body methods where known CA algorithms are also WA (some are and some aren't). We also identify classes of algorithms, including Strassen's matrix multiplication, Cooley-Tukey FFT, and cache oblivious algorithms for classical linear algebra, where a WA algorithm cannot exist: the number of writes is unavoidably high, within a constant factor of the total number of reads and writes. We explore the interaction of WA algorithms with cache replacement policies and argue that
the Least Recently Used (LRU) policy works well with the WA algorithms in this paper. We provide empirical hardware counter measurements from Intel's Nehalem-EX microarchitecture to validate our theory. In the parallel case, for classical linear algebra, we show that it is impossible to attain lower bounds both on interprocessor communication and on writes to local memory, but either one is attainable by itself. Finally, we discuss WA algorithms for sparse iterative linear algebra. We show that, for sequential communication-avoiding Krylov subspace methods, which can perform s iterations of the conventional algorithm for the communication cost of 1 classical iteration, it is possible to reduce the number of writes by a factor of Theta(s) by interleaving a matrix powers computation with orthogonalization operations in a blockwise fashion.

Report [7], in submission for publication in ACM Trans. Math. Software, addresses the reproducibility of floating point summation. We define reproducibility to mean getting bitwise identical results from multiple runs of the same program, perhaps with different hardware resources or other changes that should ideally not change the answer. Many users depend on reproducibility for debugging or correctness. However, dynamic scheduling of parallel computing resources, combined with nonassociativity of floating point addition, makes attaining reproducibility a challenge even for simple operations like summing a vector of numbers, or more complicated operations like the Basic Linear Algebra Subprograms (BLAS). We describe an algorithm that computes a reproducible sum of floating point numbers, independent of the order of summation. The algorithm depends only on a subset of the IEEE Floating Point Standard 754-2008. It is communication-optimal, in the sense that it does just one pass over the data in the sequential case, or one reduction operation in the parallel case, requiring an “accumulator" represented by just 6 floating point words (more can be used if higher precision is desired). The arithmetic cost with a 6-word accumulator is 7n floating point additions to sum n words, and (in IEEE double precision) the final error bound can be up to 10^8 times smaller than the error bound for conventional summation. We describe the basic summation algorithm, the software infrastructure used to build reproducible BLAS (ReproBLAS), and performance results. For example, when computing the dot product of 4096 double precision floating point numbers, we get a 4x slow-down compared to Intel R Math Kernel Library (MKL) running on an Intel R Core i7-2600 CPU operating at 3.4 GHz and 256 KB L2 Cache.

This work is having a significant impact on emerging standards: There is a new proposed floating point instruction in the latest draft of the IEEE 754-2018 floating point standard, which if approved, would accelerate both the reproducible summation algorithm and other well-known double-double precision algorithms. It has also been proposed for inclusion in the next BLAS standard [8], which is also under current discussion.

Report [9] answers a hard open question in the design of communication avoiding algorithms: Can the lower bounds we previously established for arbitrary algorithms, that can be expressed as perfectly nested loops accessing arrays, whose subscripts can be arbitrary affine functions of the loop indices (eg A(I), B(I+J, K), C(K, I+2*J+3*K-7, …) etc) always be attained? Surprisingly the answer is always yes, i.e. there exists an optimal tiling (and an algorithm to determine it) for any such algorithm. I consider this result a breakthrough, since it covers such a wide class of algorithms. But it leaves several important open problems, including how to deal with loop-carried dependencies, which may prevent the use of the optimal tiling, and what to do when some loop bounds are too small for the optimal tile to fit in the iteration space.

Finally, in some related work [10,11], we have applied our ideas on communication-avoiding algorithms to machine learning. We consider the problem of how to design and implement communication-efficient versions of parallel support vector machines, a widely used classifier in statistical machine learning, for distributed memory clusters and supercomputers. The main computational bottleneck is the training phase, in which a statistical model is built from an input data set. Prior to our study, the parallel isoefficiency of a state-of the-art implementation scaled as W= Omega(P^3), where W is the problem size and P the number of processors; this scaling is worse than even an one-dimensional block row dense matrix vector multiplication, which has W = Omega(P^2). This study considers a series of algorithmic refinements, leading ultimately to a Communication-Avoiding SVM (CA-SVM) method that improves the isoefficiency to nearly W = Omega(P).

We evaluate these methods on 96 to 1536 processors, and show average speedups of 3x to 16x (7x on average) over Dis-SMO, and a 95% weak-scaling efficiency on size real-world datasets, with only modest losses in overall classification accuracy.

[1] “Reconstructing Householder Vectors from Tall-Skinny QR,"
G. Ballard, J. Demmel, L. Grigori, M. Jacquelin, H. D. Nguyen,
J. Par. Distr. Comp., Vol. 85, Is. C, Nov 2015, pp 3-31,
doi:10.1016/j.jpdc.2015.06.003; previously appeared in IPDPS’14

[2] "Tradeoffs between synchronization, communication and work
in parallel linear algebra computations,"
E. Solomonik, J. Demmel, E. Carson, N. Knight,
ACM Trans. Par. Comp. (invited), Vol. 3, Is. 1, June 2016;
previously appeared in SPAA'14

[3] "Accuracy of the s-step Lanczos method for the symmetric eigenproblem,"
J. Demmel, E. Carson, SIAM J. Mat. Anal. Appl., v. 36, n. 2, pp 793-819, 2015,
previously appeared as UC Berkeley Tech Report UCB/EECS-2014-165

[4] "Communication-Avoiding Symmetric-Indefinite Factorization,"
G. Ballard, D. Becker, J. Demmel, J. Dongarra, A. Druinsky, I. Peled,
O. Schwartz, S. Toledo, and I. Yamazaki, SIMAX v. 35, i. 4, 2014,
previously appeared in IPDPS'13 (Best Paper Award in Algorithms Track)

[5] "s-step Krylov Subspace Methods as Bottom Solvers for Geometric Multigrid,"
A. Almgren, E. Carson, J. Demmel, N. Knight, M. Lijewski, B. Van Straalen,
S. Williams, IPDPS'14

[6] "Write-Avoiding Algorithms,"
J. Demmel E. Carson, L. Grigori, N. Knight, P. Koanatakool, O. Schwartz,
H. V. Simhadri, IPDPS'16

[7] "Efficient Reproducible Floating Point Summation and BLAS,"
P. Ahrens, J. Demmel, H.-D. Nguyen,
UC Berkeley Tech Report UCB/EECS-2016-121, 18 June, 2016, 106 pages

[8] "A proposal for a next-generation BLAS",
J. Demmel, G. Henry, P. Tang, X. Li, J. Riedy, M. Gates,
https://docs.google.com/document/d/1DY4ImZT1coqri2382GusXgBTTTVdBDvtD5I14QHp9OE/edit#heading=h.jtgipeoidy9

[9] "Parallelepipeds obtaining HBL lower bounds", A. Rusciano, J. Demmel,
https://arxiv.org/pdf/1611.05944.pdf, Nov 21, 2016

[10] "Design and Implementation of a Communication-Optimal Classifier for
Distributed Kernel Support Vector Machines,"
Y. You, J. Demmel, K. Czechowski, L. Song, R. Vuduc,
IEEE Trans. Par. Dist. Systems, Vol. 28, No. 4, April 2017

[11] "CA-SVM: Communication-Avoiding Support Vectors Machines on Distributed
Systems", Y. You, J. Demmel, K. Czechowski, L. Song, R. Vuduc, IPDPS'15