Communication-optimal iterative methods

J. Demmel\textsuperscript{1}, M. Hoemmen\textsuperscript{1}, M. Mohiyuddin\textsuperscript{1} and K. Yelick\textsuperscript{1,2}

\textsuperscript{1} Computer Science Division, University of California Berkeley, Berkeley, CA 94720-1776
\textsuperscript{2} Lawrence Berkeley National Laboratory, One Cyclotron Road, MS 50B-4230, Berkeley, CA 94720

E-mail: \{mhoemmen, demmel, marghoob, yelick\}@eecs.berkeley.edu

Abstract. Data movement, both within the memory system of a single processor node and between multiple nodes in a system, limits the performance of many Krylov subspace methods that solve sparse linear systems and eigenvalue problems. Here, \(s\) iterations of algorithms such as CG, GMRES, Lanczos, and Arnoldi perform \(s\) sparse matrix-vector multiplications and \(\Omega(s)\) vector reductions, resulting in a growth of \(\Omega(s)\) in both single-node and network communication. By reorganizing the sparse matrix kernel to compute a set of matrix-vector products at once and reorganizing the rest of the algorithm accordingly, we can perform \(s\) iterations by sending \(O(\log P)\) messages instead of \(\Omega(s \cdot \log P)\) messages on a parallel machine, and reading the on-node components of the matrix \(A\) from DRAM to cache just once on a single node instead of \(s\) times. This reduces communication to the minimum possible. We discuss both algorithms and an implementation of GMRES on a single node of an 8-core Intel Clovertown. Our implementations achieve significant speedups over the conventional algorithms.

1. Introduction

The costs of arithmetic and communication continue to decrease, where “communication” means moving data, either between fast and slow memory (e.g., cache and DRAM) in the sequential case, or between processors over a network in the parallel case. However, arithmetic is getting faster much more quickly than communication, so that the primary challenge is developing algorithms to avoid communication. This is especially true in sparse matrix computations, where conventional algorithms do only a few arithmetic operations per datum, so that communication costs frequently dominate already.

Many Krylov subspace methods (KSMs) for solving \(Ax = b\) and \(Ax = \lambda x\) begin with a starting vector \(v_1\), create a sequence of vectors \(v_1, \ldots, v_{s+1}\) where each \(v_i\) is gotten by multiplying some linear combination of \(v_0, \ldots, v_{i-1}\) by \(A\), and then chooses the “best” (in some sense) approximate solution to \(Ax = b\) resp. \(Ax = \lambda x\) using the Krylov subspace span \(\{v_1, \ldots, v_{s+1}\}\). This means that the cost in arithmetic and communication of GMRES (or any KSM) grows like \(\Omega(s)\) (in fact GMRES does \(\Omega(s^2)\) floating-point operations). Preconditioning only adds an additional step without changing the performance characteristics of the above two.

In [1], we show several new iterative methods that reduce the aforementioned communication cost from \(\Omega(s)\) to \(\Theta(1)\), which is optimal. In the parallel case, we decrease the latency cost (total number of messages) to \(\Theta(1)\) from \(\Omega(s)\), and in the sequential case, we decrease the bandwidth cost (total numbers of words moved) from \(\Omega(s)\) (and sometimes \(\Theta(s^2)\)) to \(\Theta(1)\). The new methods are mathematically equivalent to existing methods and converge in about the same number of iterations in practice. They include both sparse linear system solvers (with or
without preconditioning) and sparse eigenvalue solvers. They are based on algorithms presented in [2, 3, 4, 5] and [6, 7]. Unlike previous authors, we have managed to reduce communication in all steps of the algorithms. Our other contributions include sparse eigenvalue solvers, new ways to improve numerical stability, entirely new algorithms that reduce the need to restart frequently, and the ability to add preconditioning to the methods while still avoiding communication. For details, see [1].

In [8], we implemented one of our iterative methods, CA-GMRES. It is mathematically equivalent to standard GMRES (the Generalized Minimal Residual Method [9]), a widely used iterative solver for sparse systems of equations \(Ax = b\), but communicates a factor of \(\Omega(s^2)\) less.

These new Krylov subspace methods and implementations required several algorithmic innovations. First, we developed a new computational kernel, the \textit{matrix powers kernel}. It computes

\[ W = [p_0(A) \cdot b, p_1(A) \cdot b, \ldots, p_k(A) \cdot b], \]

where \(p_i(A) = \prod_{j=1}^i (A - \lambda_j \cdot I)\) is a degree-\(i\) polynomial in the sparse matrix \(A\), and the \(\lambda_i\) are constants chosen as described in [8] and [1]. Using ideas previously introduced in [10, 11], we describe how to avoid all but \(\Theta(1)\) communication while computing \(W\). In [8], our matrix powers implementation is hierarchical: it uses the parallel algorithm on the top level (for multiple cores), and a sequential algorithm at the lower level (for off-chip data movement). Thus, the matrix is thread-blocked first for the parallel algorithm and then cache-blocked within each thread. Performance data on an 8-core 2.33 GHz Intel Clovertown shows speedups for computing \(W\) of up to \(2.7 \times\) over \(s\) calls to the best optimized algorithm just for a single sparse matrix-vector multiplication \(A \cdot x\) [8]. Earlier work in [11, 10] discusses parallel algorithms using distributed memory, as well as out-of-core sequential algorithms, which also achieved significant modeled and benchmarked speedups.

The columns of \(W\) span the same Krylov subspace as the vectors originally computed by GMRES, but GMRES must be reformulated to compute the best linear combination of these new basis vectors. Part of this reformulation is described in [8], where a communication-avoiding QR factorization of \(W\) is discussed; this was introduced in [12]. Performance data shows a speedup of \(8 \times\) over LAPACK’s QR factorization in Intel’s MKL 10.0.1.014 library, for a \(10,000,000 \times 10\) matrix on an 8-core Intel Clovertown machine [8], as well as significant performance gains on multiple-node clusters [12]. TSQR also does significantly better on both shared-memory and distributed-memory platforms than the Modified Gram-Schmidt (MGS) algorithm used in many conventional iterative methods (such as Arnoldi and GMRES) [12].

Given the performance of the above two kernels, we predict significant speedups in our iterative solvers. We are nearly finished completing benchmark implementations. Future work includes developing and implementing preconditioners that work well with our iterative methods.

### 2. Matrix powers kernel performance results

In [8], we presented the performance results of our matrix powers kernel on an 8-core Intel Clovertown, which runs at 2.3 GHz, and has 2 MB of L2 cache per core (4 MB L2 cache shared by 2 cores). We compared it against applying SpMV \(s\) times, where we used a highly optimized SpMV implementation [13]. For speedup calculation, the time taken for the matrix powers kernel is normalized by dividing by the number of steps \(s\) and compared with the time taken for a single SpMV. Thus, the speedup is defined as: \(\frac{\text{time(matrix powers kernel)}}{\text{time(SpMV)}}\). In addition to matrices whose graphs are meshes, which are expected to perform well [11], we selected sparse matrices from a variety of real applications [14]. Figure 1 shows the performance of the matrix powers kernel for different matrices.
3. TSQR performance results

The TSQR factorization used in these performance results is discussed in [8]. It is a hybrid of the sequential and parallel TSQR algorithms described in [12], and requires only a constant number of synchronization steps, rather than $\Omega(s)$ reductions for $s$ columns as either the textbook Householder QR algorithm or the blocked Householder QR implementations in LAPACK and ScaLAPACK require. Figure 2 shows the performance results on a single 8-core Intel Clovertown node. We presented modeled and benchmarked performance results of TSQR in [12].

References

[1] Demmel J W and Hoemmen M in preparation Communication-optimal Krylov subspace methods Tech. rep. University of California Berkeley, Department of Electrical Engineering and Computer Science
[2] Walker H F 1988 SIAM Journal on Scientific and Statistical Computing 9 152–163
[3] de Sturler E 1991 A parallel variant of GMRES(m) Proceedings of the 13th IMACS World Congress on Computation and Applied Mathematics ed Miller J J H and Vichnevetsky R (Dublin, Ireland: Criterion Press)
[4] Joubert W D and Carey G F 1992 International Journal of Computer Mathematics 44 243–267
[5] Erhel J 1995 Electronic Transactions on Numerical Analysis 3 160–176
[6] Chronopoulos A T and Gear C W 1989 J. Comput. Appl. Math. 25 153–168 ISSN 0377-0427
[7] Toledo S 1995 Quantitative performance modeling of scientific computations and creating locality in numerical algorithms Ph.D. thesis Massachusetts Institute of Technology
[8] Mohiyuddin M, Hoemmen M, Demmel J and Yelick K 2009 Minimizing communication in sparse matrix solvers (Submitted to Supercomputing 2009)
[9] Saad Y and Schultz M H 1986 SIAM J. Sci. Stat. Comput. 7
[10] Demmel J, Hoemmen M, Mohiyuddin M and Yelick K 2008 Avoiding Communication in Sparse Matrix Computations Proceedings of the IEEE International Parallel and Distributed Processing Symposium
[11] Demmel J, Hoemmen M, Mohiyuddin M and Yelick K 2007 Avoiding communication in computing Krylov subspaces Tech. Rep. UCB/EECS-2007-123 EECS Department, University of California, Berkeley URL \url{http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-123.html}
[12] Demmel J, Grigori L, Hoemmen M and Langou J 2008 Communication-optimal parallel and sequential QR
Figure 2: Performance of TSQR and LAPACK QR for $m \times 10$ matrices, where $m$ varies from $10^4$ to $10^7$. 

[13] Williams S, Oliker L, Vuduc R, Shalf J, Yelick K and Demmel J 2007 Optimization of Sparse Matrix-Vector Multiplication on Emerging Multicore Platforms Proc. of Supercomputing

[14] Im E J, Yelick K and Vuduc R 2004 Int. J. HPCA