

Minimizing Communication in Linear Algebra

Grey Ballard James Demmel Olga Holtz Oded Schwartz

May 20, 2009

1 Introduction

Algorithms have two kinds of costs: arithmetic and communication, by which we mean either moving data between levels of a memory hierarchy (in the sequential case) or over a network connecting processors (in the parallel case). There are two costs associated with communication: *bandwidth* (proportional to the total number of words of data moved) and *latency* (proportional to the number of messages in which these words are packed and sent). For example, we may model the cost of sending m words in a single message as $\alpha + \beta m$, where α is the latency (measured in seconds) and β is the reciprocal bandwidth (measured in seconds per word). Depending on the technology, either latency or bandwidth costs may be larger, often dominating the cost of arithmetic. So it is of interest to have algorithms minimizing both communication costs.

In this paper we prove a general lower bound on the amount of data moved (i.e., bandwidth) by a general class of algorithms, including most dense and sparse linear algebra algorithms, as well as some graph theoretic algorithms. Our model is the result of Hong and Kung [HK81] which says that to multiply two dense n -by- n matrices on a machine with a large slow memory (in which the matrices initially reside) and a small fast memory of size M (too small to store the matrices, but arithmetic may only be done on data in fast memory), $\Omega(n^3/\sqrt{M})$ words of data must be moved between fast and slow memory. This lower bound is attained by a variety of “blocked” algorithms. This lower bound may also be expressed as $\Omega(\#\text{arithmetic_operations} / \sqrt{M})$ ¹.

This result was proven differently by Irony, Toledo and Tiskin [ITT04] and generalized to the parallel case, where P processors multiply two n -by- n matrices. In the “memory-scalable” case, where each processor stores the minimal $O(n^2/P)$ words of data, they obtain the lower bound $\Omega(\#\text{arithmetic_operations_per_processor} / \sqrt{\text{memory_per_processor}}) = \Omega(\frac{n^3/P}{\sqrt{n^2/P}}) = \Omega(\frac{n^2}{\sqrt{P}})$, which is attained by Cannon’s algorithm [Can69] [Dem96, Lecture 11]. The paper [ITT04] also considers the “3D” case, which does less communication by replicating the matrices and so using $O(P^{1/3})$ times as much memory as the minimal possible.

Here we begin with the proof in [ITT04], which starts with the sum $C_{ij} = \sum_k A_{ik} \cdot B_{kj}$, and uses a geometric argument on the lattice of indices (i, j, k) to bound the number of updates $C_{ij} + A_{ik} \cdot B_{kj}$ that can be performed when a subset of matrix entries are in fast memory. This proof generalizes in a number of ways: in particular it does not depend on the matrices being dense, or the output being distinct from the input. These observations let us state and prove a general Theorem (main theorem) that a lower bound on the number of words moved into or out of a fast or local memory of size M is $\Omega(\#\text{arithmetic_operations} / \sqrt{M})$. This applies to both the sequential case (where M is a fast memory) and the parallel case; in the parallel case further assumptions

¹The sequential communication model used here is sometimes called the *two-level I/O model* or *disk access machine (DAM)* model (see [AV88], [BBF⁺07], [CR06]). Our model follows that of [HK81] and [ITT04] in that it assumes the block-transfer size is one word of data ($B = 1$ in the common notation).

about whether the algorithm is memory balanced (to estimate the effective M) are needed to get a lower bound on the overall algorithm.

We obtain a simple lower bound on latency (just the lower bound on bandwidth divided by the largest possible message size, namely the memory size M). Both bandwidth and latency lower bounds apply straightforwardly to a nested memory hierarchy with more than two layers, bounding from below the communication between any adjacent layers in the hierarchy [Sav95, BDHS09].

We present simple corollaries applying the main theorem to conventional (non-Strassen-like) implementations of matrix multiplication and other BLAS operations [BLA, BDD⁺02, BDD⁺01] (dense or sparse), LU factorization, Cholesky factorization and “ LDL^T ” factorization. These factorizations may also be dense or sparse, with any kind of pivoting, and be exact or “incomplete”, e.g., ILU [Saa96] (some of these results can be also obtained, just for dense matrices, by suitable reductions from [HK81] or [ITT04], and we point these out).

We consider lower bounds for algorithms that apply orthogonal transformations to the left and/or right of matrices. This class includes the QR factorization, the standard algorithms for eigenvalues and eigenvectors, and the singular value decomposition (SVD). For reasons explained there, the counting techniques of [HK81] and [ITT04] do not apply, so we need a different but related lower bound argument.

We demonstrate how to extend our lower bounds to more general computations where we compose a sequence of simpler linear algebra operations (like matrix multiplication, LU decomposition, etc.), so the outputs of one operation may be inputs to later ones. If these intermediate results do not need to be saved in slow memory, or if some inputs are given by formulas (like $A(i, j) = 1/(i+j)$) and so do not need to be fetched from memory, or if the final output is just a scalar (the norm or determinant of a matrix), then it is natural to ask whether there is a better algorithm than just using optimized versions of each operation in the sequence. We give examples where this simple approach is optimal, and when it is not. We also exploit the natural correspondence between matrices and graphs to derive communication lower bounds for certain graph algorithms, like All-Pairs-Shortest-Path.

Finally, we discuss attainability of these lower bounds, and open problems. Briefly, in the dense case all the lower bounds are attainable (in the parallel case, this is modulo $\text{polylog } P$ factors, and assuming the minimal $O(n^2/P)$ storage per processor). The optimal algorithms for square matrix multiplication are well known, as mentioned above. Optimal algorithms for dense LU, Cholesky, QR, eigenvalue problems and the SVD are more recent, and not part of standard libraries like LAPACK [ABB⁺92] and ScaLAPACK [BCC⁺97]. Only in the case of Cholesky do we know of a sequential algorithm that both minimizes bandwidth and latency across arbitrary levels of memory hierarchy. No optimal algorithm is known for architecture mixing parallelism and multiple memory hierarchies, i.e., most real architectures. Optimal “3D” algorithms for anything other than matrix-multiplication, and optimal sparse algorithms for anything are unknown. For highly rectangular dense matrices (e.g., matrix-vector multiplication), or for sufficiently sparse matrices, our new lower bound is sometimes lower than the trivial lower bound ($\#inputs + \#outputs$), and so it is not always attainable.

A full version of this paper is available on the arXiv:
Minimizing Communication in Linear Algebra
Grey Ballard, James Demmel, Olga Holtz, Oded Schwartz
<http://arxiv.org/abs/0905.2485>

References

- [ABB⁺92] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK's user's guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992. Also available from <http://www.netlib.org/lapack/>.
- [AV88] A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. *Commun. ACM*, 31(9):1116–1127, 1988.
- [BBF⁺07] M. A. Bender, G. S. Brodal, R. Fagerberg, R. Jacob, and E. Vicari. Optimal sparse matrix dense vector multiplication in the I/O-model. In *SPAA '07: Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*, pages 61–70, New York, NY, USA, 2007. ACM.
- [BCC⁺97] L. S. Blackford, J. Choi, A. Cleary, E. Dazevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide*. SIAM, Philadelphia, PA, USA, May 1997. Also available from <http://www.netlib.org/scalapack/>.
- [BDD⁺01] 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, R. C. Whaley, Z. Maany, F. Krough, G. Corliss, C. Hu, B. Keafott, W. Walster, and J. Wolff v. Gudenberg. Basic Linear Algebra Subprograms Technical (BLAST) Forum Standard. *Intern. J. High Performance Comput.*, 15(3-4), 2001.
- [BDD⁺02] 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 Subroutines (BLAS). *ACM Trans. Math. Soft.*, 28(2), June 2002.
- [BDHS09] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Communication-optimal Parallel and Sequential Cholesky Decomposition, 2009. Submitted. Also available in EECS Tech Report and from the arXiv: <http://arxiv.org/abs/0902.2537>.
- [BLA] BLAS - Basic Linear Algebra Subroutines. www.netlib.org/blas.
- [Can69] L. Cannon. *A cellular computer to implement the Kalman filter algorithm*. PhD thesis, Montana State University, Bozeman, MN, 1969.
- [CR06] R. A. Chowdhury and V. Ramachandran. Cache-oblivious dynamic programming. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 591–600, New York, NY, USA, 2006. ACM.
- [Dem96] J. Demmel. CS 267 Course Notes: Applications of Parallel Processing. Computer Science Division, University of California, 1996. <http://www.cs.berkeley.edu/~demmel/cs267>.
- [HK81] J. W. Hong and H. T. Kung. I/O complexity: The red-blue pebble game. In *STOC '81: Proceedings of the thirteenth annual ACM symposium on Theory of computing*, pages 326–333, New York, NY, USA, 1981. ACM.
- [ITT04] D. Irony, S. Toledo, and A. Tiskin. Communication lower bounds for distributed-memory matrix multiplication. *J. Parallel Distrib. Comput.*, 64(9):1017–1026, 2004.

- [Saa96] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Co., Boston, 1996.
- [Sav95] J. E. Savage. Extending the Hong-Kung model to memory hierarchies. In *COCOON*, pages 270–281, 1995.

Grey Ballard

ballard@eecs.berkeley.edu

Computer Science Division, University of California, Berkeley, CA 94720. Research supported by Microsoft and Intel funding (Award #20080469) and by matching funding by U.C. Discovery (Award #DIG07-10227).

James Demmel

demmel@cs.berkeley.edu

Mathematics Department and Computer Science Division, University of California, Berkeley, CA 94720.

Olga Holtz

oholtz@eecs.berkeley.edu

Departments of Mathematics, University of California, Berkeley and Technische Universität Berlin. O. Holtz acknowledges support of the Sofja Kovalevskaja programm of Alexander von Humboldt Foundation.

Oded Schwartz

odedsc@math.tu-berlin.de

Departments of Mathematics, Technische Universität Berlin, 10623 Berlin, Germany.