Sequential Communication Bounds for Fast Linear Algebra

Grey Ballard
James Demmel
Olga Holtz
Oded Schwartz

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2012-36
http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-36.html

March 30, 2012

Acknowledgement

This work is supported by Microsoft (Award #024263) and Intel (Award #024894) funding and by matching funding by U.C. Discovery (Award #DIG07-10227); additional support from Par Lab affiliates National Instruments, NEC, Nokia, NVIDIA, and Samsung. Research supported by U.S. Department of Energy grants under Grant Numbers DE-SC0003959, DE-SC0004938, and DE-FC02-06-ER25786, as well as Lawrence Berkeley National Laboratory Contract DE-AC02-05CH11231. Research supported by the Sofja Kovalevskaja programme of Alexander von Humboldt Foundation and by the National Science Foundation under agreement DMS-0635607. Research supported by ERC Starting Grant Number 239985.

Sequential Communication Bounds for Fast Linear Algebra*

Grey Ballard, James Demmel, Olga Holtz, Oded Schwartz

Abstract

In this note we obtain communication cost lower and upper bounds on the algorithms for LU and QR given in (Demmel, Dumitriu, and Holtz 2007). The algorithms there use fast, stable matrix multiplication as a subroutine and are shown to be as stable and as computationally efficient as the matrix multiplication subroutine. We show here that they are also as communication-efficient (in the sequential, two-level memory model) as the matrix multiplication algorithm. The analysis for LU and QR extends to all the algorithms in (Demmel, Dumitriu, and Holtz 2007). Further, we prove that in the case of using Strassen-like matrix multiplication, these algorithms are communication optimal.

*This work is supported by Microsoft (Award #024263) and Intel (Award #024894) funding and by matching funding by U.C. Discovery (Award #DIG07-10227); additional support from Par Lab affiliates National Instruments, NEC, Nokia, NVIDIA, and Samsung. Research supported by U.S. Department of Energy grants under Grant Numbers DE-SC0003959, DE-SC0004938, and DE-FC02-06-ER25786, as well as Lawrence Berkeley National Laboratory Contract DE-AC02-05CH11231. Research supported by the Sofja Kovalevskaja programme of Alexander von Humboldt Foundation and by the National Science Foundation under agreement DMS-0635607. Research supported by ERC Starting Grant Number 239985.

1 Introduction

The running time of a numerical computation depends both on the number of floating point operations (*flops*) and on the amount of data it moves, which we call *communication*. In order to determine the communication cost of a sequential algorithm, we use a simple machine model which consists of a fast memory of size M words (where computation takes place) connected to a slow memory with unbounded size. We assume the data involved in the computation is too large to fit entirely in the fast memory, and we measure the communication in terms of floating point numbers (*words*) moved between the two memory levels. We will refer to the number of words moved by an algorithm its *bandwidth cost*.

The naïve approach of multiplying two $n \times n$ matrices using three nested loops performs $2n^3$ flops and moves $O(n^3)$ words of data. The $2n^3$ flops can be re-ordered into a blocked or recursive algorithm which moves $O\left(\frac{n^3}{\sqrt{M}}\right)$ words, a significant improvement over the naïve approach. Hong and Kung [12] proved that this communication reduction is asymptotically optimal, that performing $2n^3$ flops requires moving at least $\Omega\left(\frac{n^3}{\sqrt{M}}\right)$ words.

Strassen [13] showed that matrix multiplication can be performed with $O(n^{\log_2 7})$ flops, asymptotically fewer than the classical $O(n^3)$ approach, and many further improvements on the exponent have been made. By reducing the number of flops performed, it is also possible to reduce the communication below the bound proved by Hong and Kung. In [4], we prove a communication lower bound for fast matrix multiplication algorithms: executing a matrix multiplication algorithm with $O(n^{\omega_0})$ flops requires moving $\Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\omega_0} M\right)$ words.[1] This lower bound is attained by the natural recursive algorithm.

Demmel, Dumitriu, and Holtz [6] showed that nearly all of the fundamental algorithms in dense linear algebra can be executed with asymptotically the same number of flops as matrix multiplication. To obtain practical numerical algorithms, one must also consider stability and communication issues. Although the stability properties of fast matrix multiplication are slightly weaker than those of classical matrix multiplication, the authors show in [7] that fast matrix multiplication is stable. Further, in [6] they show that fast linear algebra can be made stable at the expense of only a polylogarithmic (*i.e.*, polynomial in $\log n$) factor increase in cost. That is, to maintain stability, one can use polylogarithmically more bits to represent each floating point number and to compute each flop. While this increases the time to perform one flop or move one word, it does not change the number of flops computed or words moved by the algorithm.

The main contribution of this note is the extension of both upper and lower communication bounds for the algorithms presented in [6]. Stability and computational complexity were the main concerns in [6]. Here, we show that the communication cost of the algorithms also matches that of the matrix multiplication subroutines they employ and that the recursive implementations yield a communication-optimal ordering of the computation. We summarize the previous results and those shown here in Table 1.

[1] Under certain technical assumptions. See Section 2 for details.

Algorithm	Flops	Words	Lower Bound
Classical matrix multiplication	$O(n^3)$	$O\left(\frac{n^3}{\sqrt{M}} + n^2\right)$	$\Omega\left(\frac{n^3}{\sqrt{M}} + n^2\right)$
Classical linear algebra			
Fast matrix multiplication	$O(n^{\omega_0})$	$O\left(\frac{n^{\omega_0}}{M^{\omega_0/2-1}} + n^2\right)$	$\Omega\left(\frac{n^{\omega_0}}{M^{\omega_0/2-1}} + n^2\right)$
Fast linear algebra		$O\left(\frac{n^{\omega_0}}{M^{\omega_0/2-1}} + n^2 \log n\right)$	

Table 1: Summary of cost comparisons between classical and fast algorithms. The lower bounds given correspond to the number of words moved.

2 Lower Bounds

In this section we obtain communication lower bounds for the algorithms in [6] using the results from [4]. The applicability of these bounds are similar to those for classical algorithms [5] and for Strassen-like matrix multiplication algorithms [4]: they apply to any re-ordering of the computation (which respects the dependencies) and any use of commutativity and associativity of addition. The lower bounds here depend directly on the choice of fast matrix multiplication algorithm used as a subroutine, and they apply only to "Strassen-like" algorithms which are defined below.

Note that for all dense matrix algorithms there exists a trivial lower bound of $\Omega(n^2)$ words since every matrix element must be accessed at least once. We include this term in the statements of the lower bounds in Table 1.

2.1 Strassen-like Matrix Multiplication

Communication lower bounds for Strassen's and Strassen-like matrix multiplication algorithms are proved in [4] using analysis of the computation directed acyclic graph *(CDAG)*. We repeat the definition of Strassen-like and the main result from [4] here, as our results for Strassen-like linear algebra will be direct extensions.

Definition 2.1. *A* Strassen-like matrix multiplication algorithm *is a recursive algorithm for multiplying square matrices which is constructed from a base case of multiplying $n_0 \times n_0$ matrices using $m_0 < n_0^3$ scalar multiplications, resulting in an algorithm for multiplying $n \times n$ matrices requiring $O(n^{\omega_0})$ flops where $\omega_0 = \log_{n_0} m_0$. In order to be* Strassen-like*, the base case decoding graph, which gives the dependencies between the m_0 scalar multiplication results and the n_0^2 entries of the output matrix, must be connected.*[2]

Theorem 2.2 ([4])**.** *The number of words moved by a Strassen-like matrix multiplication algorithm with $O(n^{\omega_0})$ flops on a machine with fast memory of size M, assuming no inter-*

[2]See [4] for further discussion.

mediate value is computed twice, is

$$B(n) = \Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\omega_0} \cdot M\right).$$

2.2 Linear Algebra with Strassen-like Matrix Multiplication

The following lower bound applies to all the algorithms in [6] assuming the fast matrix multiplication subroutine is Strassen-like. In particular, along with the results of Sections 4 and 5, it implies that the LU and QR algorithms are communication-optimal.

Corollary 2.3. *Suppose an algorithm has a CDAG containing as a subgraph the CDAG of a Strassen-like matrix multiplication algorithm with input size $\Theta(n)$ which performs $\Theta(n^{\omega_0})$ flops. Then, assuming that no intermediate value is computed twice, the number of words moved during the computation on a machine with fast memory of size M is*

$$B(n) = \Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\omega_0} \cdot M\right).$$

Proof. The proof of Theorem 2.2 is based on an analysis of the CDAG of a Strassen-like matrix multiplication algorithm. If the CDAG of a computation includes as a subgraph a CDAG which corresponds to $\Theta(n) \times \Theta(n)$ Strassen-like matrix multiplication, then the analysis yields the same communication lower bound for that subset of the computation and therefore the entire computation. □

Note that there may be many different CDAGs which correspond to computing an LU decomposition using a Strassen-like matrix multiplication as a subroutine. For example, the algorithms of [6] split the matrix into equal-sized left and right halves, but another algorithm may split the matrix into a tall-skinny panel and a larger trailing matrix. Corollary 2.3 applies to all such algorithms that contain a sufficiently large subgraph corresponding to a Strassen-like matrix multiplication.

This result implies that given the CDAG that a recursive algorithm of [6] produces, no reordering of the computation can improve the communication costs by more than a constant factor compared to the depth-first ordering given by the recursive algorithm. The result does not apply to algorithms which restructure the CDAG beyond the freedom allowed by commutativity and associativity of addition.

Unlike the case of classical matrix multiplication and other $O(n^3)$ algorithms, the freedom to exploit commutativity and associativity within fast matrix multiplication and fast linear algebra is quite limited and can affect communication costs only by a constant factor. In the case of classical algorithms, exploiting commutativity and associativity is an important means of blocking algorithms to improve communication costs and can alter the bandwidth cost by up to a factor of $O(\sqrt{M})$. In the case of fast algorithms, the freedom to order the computation allows much greater flexibility than just exploiting commutativity and associativity: a pessimal ordering results in $O(n^{\omega_0})$ bandwidth cost while the optimal ordering decreases that cost by a factor of $O(M^{\omega_0/2-1})$. In this sense, the applicability of the lower bounds for fast linear algebra are as general as those for fast matrix multiplication.

3 Rectangular Recursive $O(n^3)$ Algorithms

In this section we review the communication cost upper bounds for rectangular recursive algorithms for Cholesky, LU, and QR which use classical matrix multiplication as a subroutine. We provide the detailed analysis here because it does not appear elsewhere in the literature in full generality, and because the analysis for fast linear algebra (*i.e.*, using fast matrix multiplication as a subroutine) is very similar. We will assume the use of classical matrix multiplication and triangular solve (with multiple right hand sides) subroutines which attain their communication lower bounds. For upper bound analysis of recursive versions of these algorithms, see [3].

The bandwidth cost recurrence for rectangular recursive algorithms for Cholesky [3], LU [11, 14], and QR [8, 9] on $m \times n$ matrices (where $m \geq n$) is given by

$$B(m,n) = \begin{cases} B\left(m, \frac{n}{2}\right) + B\left(m - \frac{n}{2}, \frac{n}{2}\right) + \Theta\left(\frac{mn^2}{\sqrt{M}} + mn\right) & \text{if } n > 1 \text{ and } mn > M \\ \Theta(mn) & \text{if } n = 1 \text{ or } mn \leq M. \end{cases} \quad (1)$$

The base case occurs either when the recursion stops in order to factor one column ($n = 1$) or the problem fits entirely into fast memory ($mn < M$). The fact that either base case can occur is different than in many other communication cost recurrences and is the main reason the analysis is more complicated. In either base case, the bandwidth cost is only a constant factor more than the cost of reading the matrix once.

An equivalent recurrence appears in [8] (see equation (3.5)) for the case of QR, but the cost of the update of the right half of the matrix is written as $O\left(\frac{mn^2}{\sqrt{M}}\right)$ (in the notation here), which ignores the mn term. The mn term dominates $\frac{mn}{\sqrt{M}}$ when $n < \sqrt{M}$; note that this case may occur simultaneously with $mn > M$ for sufficiently large m. In this case, the problem is too large to fit into fast memory but the number of columns is too small to attain $\Theta(\sqrt{M})$ re-use within matrix multiplication. Thus, the mn term cannot be ignored.

A simplified recurrence for LU appears in [14], where the base case $mn < M$ is not considered. This implies the recurrence provided in [14] is accurate only in certain cases, but the recurrence provides a valid upper bound in all cases (only upper bounds are considered in [14]). Tighter analysis, including consideration of lower bounds, is provided in [3] for Cholesky, and the two cases $m \leq M$ and $m > M$ are treated separately. However, in the case $m > M$, the error in the statement of the recurrence is propagated from [14]. The recurrence assumes that if $m > M$, then no subproblem will fit in fast memory throughout the recursion; however, the number of rows in one of the subproblems is reduced, and therefore both base cases are possible. The recurrence is valid only if $m - n > M$. That is, the smallest number of rows among all subproblems is still greater than the size of the fast memory.

Here we show both upper and lower bounds[3] for equation (1), thus showing that each term is inherent to the algorithm and not due to lax analysis. Since the cost of factoring

[3]Note that this is a lower bound for the recurrence corresponding to these particular rectangular recursive algorithms, not for a class of algorithms for computing these decompositions. Lower bounds for a class of algorithms is the subject of [4, 5] and Section 2.

the left half of the matrix is greater than the cost of factoring the right half, we can upper bound the right hand side of equation (1) (as done in [8]) with

$$B(m,n) \leq \begin{cases} 2B\left(m, \frac{n}{2}\right) + O\left(\frac{mn^2}{\sqrt{M}} + mn\right) & \text{if } n > 1 \text{ and } mn > M \\ O(mn) & \text{if } n = 1 \text{ or } mn \leq M. \end{cases}$$

Thus,

$$\begin{aligned} B(m,n) &\leq O\left(2^t \cdot m\frac{n}{2^t}\right) + \sum_{i=0}^{t-1} 2^i \cdot O\left(\frac{m\left(\frac{n}{2^i}\right)^2}{\sqrt{M}} + m\frac{n}{2^i}\right) \\ &= O\left(\frac{mn^2}{\sqrt{M}} + mnt\right) \end{aligned}$$

where t is the height of the recursion tree. In the case $m - n > M$, the base case $mn < M$ never occurs because the number of rows in any subproblem is always too large to fit one column in fast memory. Thus, the only relevant base case is $n = 1$, and so $t = \log n$. In the case $m < M$, the base case $n = 1$ never occurs because each subproblem will fit in fast memory before the number of columns is reduced to 1. Thus, the only relevant base case is $mn < M$, and so $t = \log \frac{mn}{M}$.

Similarly, we can lower bound the right hand side of equation (1) with

$$B(m,n) \geq \begin{cases} 2B\left(m - \frac{n}{2}, \frac{n}{2}\right) + \Omega\left(\frac{mn^2}{\sqrt{M}} + mn\right) & \text{if } n > 1 \text{ and } mn > M \\ \Omega(mn) & \text{if } n = 1 \text{ or } mn \leq M. \end{cases}$$

Thus, since the number of rows in any subproblem is at least $m - n$, we have

$$\begin{aligned} B(m,n) &\geq \Omega\left(2^t \cdot m\frac{n}{2^t}\right) + \sum_{i=0}^{t-1} 2^i \cdot \Omega\left(\frac{\left(m - \sum_{j=1}^{i} \frac{n}{2^j}\right)\left(\frac{n}{2^i}\right)^2}{\sqrt{M}} + \left(m - \sum_{j=1}^{i} \frac{n}{2^j}\right)\left(\frac{n}{2^i}\right)\right) \\ &\geq \Omega(mn) + \sum_{i=0}^{t-1} 2^i \cdot \Omega\left(\frac{(m-n)\left(\frac{n}{2^i}\right)^2}{\sqrt{M}} + (m-n)\left(\frac{n}{2^i}\right)\right) \\ &= \Omega\left(\frac{(m-n)n^2}{\sqrt{M}} + (m-n)nt\right) \end{aligned}$$

where $t = \log n$ in the case $m - n > M$, or $t = \log \frac{mn}{M}$ in the case $m < M$.

Thus, for $m \geq cn$ for some constant $c > 1$ and $m - n > M$, we have

$$B(m,n) = \Theta\left(\frac{mn^2}{\sqrt{M}} + mn \log n\right),$$

and for $m \geq cn$ for some constant $c > 1$ and $m \leq M$, we have

$$B(m,n) = \Theta\left(\frac{mn^2}{\sqrt{M}} + mn \log \frac{mn}{M}\right).$$

Note the $mn \log n$ and $mn \log \frac{mn}{M}$ terms: they arise because of the extra mn term in the bandwidth cost of matrix multiplication (and triangular solve, possibly) at each internal node of the recursion tree. This is why the solutions given here differ from [8]. Also, as argued in [14], pivoting in the case of LU adds another $O(m^2 \log n)$ term to the total bandwidth cost.

While the condition $m \geq cn$ with $c > 1$ does not hold for square matrices, we may focus attention on factoring the left half of the matrix in which case the number of rows is twice the number of columns and obtain both upper and lower bounds. Thus, for $n > 2M$, we have $B(n) = \Theta\left(\frac{n^3}{\sqrt{M}} + n^2 \log n\right)$, and for $n < M$, we have $B(n) = \Theta\left(\frac{n^3}{\sqrt{M}} + n^2 \log \frac{n^2}{M}\right)$.

For each of these recurrences, the base cases together cost only $\Theta(n^2)$ words (the size of the input/output). Since the cost of the internal nodes dominates, using a faster matrix multiplication subroutine (that also communicates less) improves the total communication costs of these algorithms. However, using a faster matrix multiplication cannot reduce the extra $n^2 \log n$ or $n^2 \log \frac{n^2}{M}$ term that accumulates from the $\Theta(n^2)$ bandwidth cost required for each recursive level of internal nodes.

4 LU with Fast Matrix Multiplication

The LU decomposition algorithm of [6] includes computing and multiplying by explicit inverses of diagonal blocks of L (which are triangular). Thus, we first establish the bandwidth costs of fast matrix multiplication and triangular matrix inversion using fast matrix multiplication.

Fast Matrix Multiplication The communication cost of fast matrix multiplication implemented with the natural recursive algorithm is given by

$$B_{MM}(n) = \Theta\left(\frac{n^{\omega_0}}{M^{\omega_0/2-1}} + n^2\right)$$

(see equation (1) in [4]) where ω_0 is the exponent corresponding to the computational cost.

Triangular Matrix Inversion with Fast Matrix Multiplication. Following Section 3.1 of [6], the recursive algorithm for triangular matrix inversion[4] involves two recursive calls and two matrix multiplications (which ignore triangular sparsity). Again, the cost recurrence given for arithmetic also applies to the bandwidth cost. The bandwidth cost is

$$B_{TRTRI}(n) = \begin{cases} 2B_{TRTRI}\left(\frac{n}{2}\right) + \Theta\left(\frac{(n/2)^{\omega_0}}{M^{\omega_0/2-1}} + (n/2)^2\right) & \text{if } n^2 > 2M \\ \Theta(n^2) & \text{if } n^2 \leq 2M. \end{cases}$$

The solution to this recurrence is

$$B_{TRTRI}(n) = \Theta\left(\frac{n^{\omega_0}}{M^{\omega_0/2-1}} + n^2\right).$$

[4]We use the LAPACK acronym TRTRI to refer to triangular matrix inversion.

Note that this algorithm is communication optimal.

In order to re-use the cost function for LU decomposition given in Section 4.2 of [6], we compute the bandwidth cost of LU without pivoting first and then add in the costs of pivoting. With a slight change of notation from [6] (interchanging m and n), we obtain a similar recursive bound on the bandwidth cost:

$$B_{LU}(m,n) \leq 2B_{LU}\left(m,\frac{n}{2}\right) + 4\frac{m}{n}B_{MM}\left(\frac{n}{2}\right) + B_{TRI}\left(\frac{n}{2}\right) + O\left(m\frac{n}{2}\right).$$

Note that the base cases of the recurrence are different than in [6] and match those in Section 3. Since the bandwidth cost of triangular inversion matches that of matrix multiplication, we obtain

$$B_{LU}(m,n) = \begin{cases} 2B_{LU}\left(m,\frac{n}{2}\right) + O\left(\frac{m(n/2)^{\omega_0-1}}{M^{\omega_0/2-1}} + m(n/2)\right) & \text{if } n > 1 \text{ and } mn > M \\ \Theta(mn) & \text{if } n = 1 \text{ or } mn \leq M. \end{cases}$$

Since the pivoting is done the same way as in the classical $O(n^3)$ recursive algorithm, the bandwidth cost due to pivoting is $O(m^2 \log n)$ as shown in [14]. Following the same analysis as in Section 3 and including the cost of pivoting, we have that the bandwidth cost of LU decomposition with partial pivoting is

$$B_{LU}(m,n) = \Theta\left(\frac{mn^{\omega_0-1}}{M^{\omega_0/2-1}} + m^2 \log n\right).$$

Note that since $m \geq n$, $m^2 \log n$ dominates $mn \log n$ and $mn \log \frac{mn}{M}$ (for $m < M$).

Thus, for a square matrix, the bandwidth cost is

$$B_{LU}(n) = \Theta\left(\frac{n^{\omega_0}}{M^{\omega_0/2-1}} + n^2 \log n\right)$$

and by Corollary 2.3 the algorithm is communication-optimal for $M = O\left(\frac{n^2}{\log n}\right)$.

5 QR with Fast Matrix Multiplication

Section 4.1 of [6] presents a recursive QR decomposition algorithm based on that of Elmroth and Gustavson [9] which exploits fast matrix multiplication and has the same asymptotic computational cost as the matrix multiplication subroutine. The recursive cost function, given in [6] to represent arithmetic cost, also represents the bandwidth cost. Bounding it in the same way (again with a slight change in notation and different base cases), we obtain

$$B_{QR}(m,n) \leq 2B_{QR}\left(m,\frac{n}{2}\right) + 8\frac{m}{n}B_{MM}\left(\frac{n}{2}\right) + O\left(m\frac{n}{2}\right).$$

Thus, we have

$$B_{QR}(m,n) = \begin{cases} 2B_{QR}\left(m,\frac{n}{2}\right) + O\left(\frac{m(n/2)^{\omega_0-1}}{M^{\omega_0/2-1}} + m(n/2)\right) & \text{if } n > 1 \text{ and } mn > M \\ \Theta(mn) & \text{if } n = 1 \text{ or } mn \leq M. \end{cases}$$

Following the analysis from Section 3, in the case $m > 2M$, the depth of the recursion tree is $\log n$, and therefore

$$B_{QR}(m,n) = \Theta\left(\frac{mn^{\omega_0-1}}{M^{\omega_0/2-1}} + mn\log n\right). \tag{2}$$

For smaller values of m, when one or more columns of the matrix fit in fast memory, the depth of the recursion tree is $\log \frac{mn}{M}$, and the bandwidth cost becomes

$$B_{QR}(m,n) = \Theta\left(\frac{mn^{\omega_0-1}}{M^{\omega_0/2-1}} + mn\log \frac{mn}{M}\right).$$

Note that the cost is dominated by the internal nodes of the recursion tree (the leaves contribute $\Theta(mn)$ words total). Also note that $\log \frac{mn}{M} < \log n$ when $m < M$, which implies that equation (2) is a valid upper bound in all cases.

Thus, for a square matrix, the bandwidth cost is

$$B_{QR}(n) = \Theta\left(\frac{n^{\omega_0}}{M^{\omega_0/2-1}} + n^2\log n\right)$$

and by Corollary 2.3 the algorithm is communication-optimal for $M = O\left(\frac{n^2}{\log n}\right)$.

6 Discussion and Extensions

In this note we provide the communication cost analysis for recursive LU and QR algorithms which use fast matrix multiplication subroutines as presented in [6], and we show that these algorithms are communication optimal. Several more algorithms are given in [6], including randomized rank-revealing URV decomposition, eigenvalue and singular value decompositions, solving the Sylvester equation, and computing eigenvectors from Schur form. Corollary 2.3 applies to each of these algorithms, and the communication costs of these algorithms attain the lower bounds up to additive $n^2 \log n$ terms, though we omit the proofs here.

There are several extensions to this work. First, all of the results here are for the sequential, two-level memory model. The lower bound here can be easily applied to every pair of adjacent levels in the sequential, hierarchical memory model discussed in [5]; the recursive structure of these algorithms leads to cache-obliviousness [10] and optimality in this hierarchical model.

The lower bound can also be extended to the parallel, distributed-memory model as is done in [4] and [1]. However, it is unclear if there exist communication optimal parallel algorithms for LU and QR in the parallel case. An optimal algorithm for Strassen's matrix multiplication was obtained only recently [2]. It seems likely that the parallelization approach employed there can be used for other linear algebra algorithms.

Another useful communication metric is latency cost. The *latency cost* of an algorithm is the number of messages communicated between fast and slow memory, where a message

consists of many words stored contiguously in slow memory which are communicated as a unit. The bandwidth cost lower bound easily translates to a latency cost lower bound. In order to consider the latency cost of an algorithm, we must define the data layout used for the input and output matrices. As argued in [3], rectangular recursive algorithms which minimize bandwidth cost do not necessarily also minimize latency cost.

References

[1] G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, and O. Schwartz. Brief announcement: Strong scaling of matrix multiplication algorithms and memory-independent communication lower bounds, Mar. 2012. Submitted to SPAA, available as EECS Tech. Report No. UCB/EECS-2012-31.

[2] G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, and O. Schwartz. Communication-optimal parallel algorithm for Strassen's matrix multiplication, Mar. 2012. Submitted to SPAA, available as EECS Tech. Report No. UCB/EECS-2012-32.

[3] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Communication-optimal parallel and sequential Cholesky decomposition. *SIAM Journal on Scientific Computing*, 32(6):3495–3523, 2010.

[4] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Graph expansion and communication costs of fast matrix multiplication: regular submission. In *Proceedings of the 23rd ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '11, pages 1–12, New York, NY, USA, 2011. ACM.

[5] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Minimizing communication in numerical linear algebra. *SIAM Journal on Matrix Analysis and Applications*, 32(3):866–901, 2011.

[6] J. Demmel, I. Dumitriu, and O. Holtz. Fast linear algebra is stable. *Numerische Mathematik*, 108(1):59–91, Oct. 2007.

[7] J. Demmel, I. Dumitriu, O. Holtz, and R. Kleinberg. Fast matrix multiplication is stable. *Numerische Mathematik*, 106(2):199–224, Mar. 2007.

[8] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou. Communication-optimal parallel and sequential QR and LU factorizations. *SIAM Journal on Scientific Computing*, 34(1):A206–A239, 2012.

[9] E. Elmroth and F. Gustavson. New serial and parallel recursive QR factorization algorithms for SMP systems. In B. K. et al., editor, *Applied Parallel Computing. Large Scale Scientific and Industrial Problems.*, volume 1541 of *Lecture Notes in Computer Science*, pages 120–128. Springer, 1998.

[10] M. Frigo, C. Leiserson, H. Prokop, and S. Ramachandran. Cache-oblivious algorithms. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 285, Washington, DC, USA, 1999. IEEE Computer Society.

[11] F. Gustavson. Recursion leads to automatic variable blocking for dense linear-algebra algorithms. *IBM J. Res. Dev.*, 41(6):737–756, 1997.

[12] 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.

[13] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969. 10.1007/BF02165411.

[14] S. Toledo. Locality of reference in LU decomposition with partial pivoting. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1065–1081, 1997.