Tight Memory-Independent Parallel Matrix Multiplication Communication Lower Bounds

Hussam Al Daas, Grey Ballard, Laura Grigori, Suraj Kumar, Kathryn Rouse

Tight Memory-Independent Parallel Matrix Multiplication Communication Lower Bounds

HUSSAM AL DAAS, Rutherford Appleton Laboratory, UK
GREY BALLARD, Wake Forest University, USA
LAURA GRIGORI, Inria Paris, France
SURAJ KUMAR, Inria Paris, France
KATHRYN ROUSE, Inmar Intelligence, USA

Communication lower bounds have long been established for matrix multiplication algorithms. However, most methods of asymptotic analysis have either ignored the constant factors or not obtained the tightest possible values. Recent work has demonstrated that more careful analysis improves the best known constants for some classical matrix multiplication lower bounds and helps to identify more efficient algorithms that match the leading-order terms in the lower bounds exactly and improve practical performance. The main result of this work is the establishment of memory-independent communication lower bounds with tight constants for parallel matrix multiplication. Our constants improve on previous work in each of three cases that depend on the relative sizes of the aspect ratios of the matrices.

1 INTRODUCTION

The cost of communication relative to computation continues to grow, so the time complexity of an algorithm must account for both the computation it performs and the data that it communicates. Communication lower bounds for computations set targets for efficient algorithms and spur algorithmic development. Matrix multiplication is one of the most fundamental computations, and its I/O complexity on sequential machines and parallel communication costs have been well studied over decades [2, 11, 13, 14, 21].

The earliest results established asymptotic lower bounds, ignoring constant factors and lower order terms. Because of the ubiquity of matrix multiplication in high performance computations, more recent attempts have tightened the analysis to obtain the best constant factors [17, 20, 22]. These improvements in the lower bound also helped identify the best performing sequential and parallel algorithms that can be further tuned for high performance in settings where even small constant factors make significant differences. We review these advances and other related work in § 2.

The main result of this paper is the establishment of tight constants for memory-independent communication lower bounds for parallel classical matrix multiplication. In the context of a distributed-memory parallel machine model (see § 3.1), these bounds apply even when the local memory is infinite, and they are the tightest bounds in many cases when the memory is limited. Demmel et al. [11] prove asymptotic bounds for general rectangular matrix multiplication and show that three different bounds are asymptotically tight in separate cases that depend on the relative sizes of matrix dimensions and the number of processors. Our main result, Theorem 1 in § 4, reproduces those asymptotic bounds and improves the constants in all three cases. Further, in § 5, we analyze a known algorithm to show that it attains the lower bounds exactly, proving that the constants we obtain are tight. We present a comparison to previous work in Tab. 1 and discuss it in detail in § 6.

We believe one of the main features of our lower bound result is the simplicity of the proof technique, which makes a unified argument that applies to all three cases. The key idea is to cast the lower bound as the solution to a

Authors' addresses: Hussam Al Daas, Rutherford Appleton Laboratory, Didcot, Oxfordshire, UK, hussam.al-daas@stfc.ac.uk; Grey Ballard, Wake Forest University, Winston-Salem, NC, USA, ballard@wfu.edu; Laura Grigori, Inria Paris, Paris, France, laura.grigori@inria.fr; Suraj Kumar, Inria Paris, Paris, France, suraj.kumar@inria.fr; Kathryn Rouse, Inmar Intelligence, Winston-Salem, NC, USA, kathryn.rouse@inmar.com.

constrained optimization problem (see Lemma 5) whose objective function is the sum of variables that correspond to the amount of data of each matrix required by a single processor's computation. The constraints include the well-known Loomis-Whitney inequality [18] as well as new lower bounds on individual array access (see Lemma 4) that are necessary to establish separate results for the three cases. All of the complexity of the three cases, including establishing the thresholds between cases and the leading terms in each case, are confined to a single optimization problem. We use fundamental results from convex optimization (reviewed in § 3.2) to solve the problem analytically. This unified argument is elegant, it improves previous results to obtain tight constants, and it can be applied more generally to other computations that have iteration spaces with uneven dimensions.

2 RELATED WORK

2.1 Memory-Dependent Bounds for Matrix Multiplication

The first communication lower bound for matrix multiplication was established by Hong and Kung [13], who obtain the result using computation directed acyclic graph (CDAG) analysis that multiplication of square $n \times n$ matrices on a machine with cache size M requires $\Omega(n^3/\sqrt{M})$ words of communication. Irony, Toledo, and Tiskin [14] reproduce the result using a geometric proof based on the Loomis-Whitney inequality (Lemma 1), show it applies to general rectangular matrices (replacing n^3 with $n_1 n_2 n_3$), and obtain an explicit constant of $(1/2)^{3/2} \approx .35$. They also observe that the result is easily extended to the distributed memory parallel computing model (as described in § 3.1) under mild assumptions by dividing the bound by the number of processors P. We refer to such bounds as "memory-dependent," following [3], where the cache size M is interpreted as the size of each processor's local memory. Later, Dongarra et al. [12] tightened the constant for sequential and parallel memory-dependent bounds to $(3/2)^{3/2} \approx 1.84$. More recently, Smith et al. [22] prove the constant of 2 and show that it is obtained by an optimal sequential algorithm and is therefore tight. Both of these results are proved using the Loomis-Whitney inequality. Kwasniewski et al. [17] use CDAG analysis to obtain the same constant of 2 and show that it is tight in the parallel case (when memory is limited) by providing an optimal algorithm.

2.2 Bounds for Other Computations

Hong and Kung's proof technique can be applied to a more general set of computations, including the FFT [13]. Ballard et al. [4] use the proof technique of [14] to generalize the lower bound (with the same explicit constant) to other linear algebra computations such as LU, Cholesky, and QR factorizations. The constants of the lower bounds for these and other computations are tightened by Olivry et al. [20], including reproducing the constant of 2 for matrix multiplication. Kwasniewski et al. [16] also obtain tighter constants for LU and Cholesky factorizations using CDAG analysis. Christ et al. [10] show that a generalization of the Loomis-Whitney inequality can be used to prove communication lower bounds for a much wider set of computations, but the asymptotic bounds do not include explicit constants. This approach is applied to a tensor computation by Ballard and Rouse [5, 6].

2.3 Memory-Independent Bounds for Parallel Matrix Multiplication

This section describes the related work that focuses on the topic of this paper. Aggarwal, Chandra, and Snir [2] extend Hong and Kung's result for matrix multiplication to the LPRAM parallel model, which closely resembles the model we consider with the exception that there exists a global shared memory where the inputs initially reside and where the output must be stored at the end of the computation. Communication bounds in the related BSP parallel

	$1 \le P \le \frac{m}{n}$	$\frac{m}{n} \le P \le \frac{mn}{k^2}$	$\frac{mn}{k^2} \le P$
Leading term	nk	$\left(\frac{mnk^2}{P}\right)^{1/2}$	$\left(\frac{mnk}{P}\right)^{2/3}$
[2]	-	-	$\left(\frac{1}{2}\right)^{2/3} \approx .63$
[14]	-	-	$\frac{1}{2} = .5$
[11]	$\frac{16}{25} = .64$	$\left(\frac{2}{3}\right)^{1/2} \approx .82$	1
Theorem 1	1	2	3

Table 1. Summary of explicit constants of leading term of parallel memory-independent rectangular matrix multiplication communication lower bounds for multiplication dimensions $m \ge n \ge k$ and P processors

model are also memory independent, and Scquizzato and Silvestri [21] establish the same asymptotic lower bounds for matrix multiplication in that model. In addition to proving bounds for sequential matrix multiplication and the associated memory-dependent bound for parallel matrix multiplication, Irony, Toledo, and Tiskin [14] prove also that a parallel algorithm must communicate $\Omega(n^2/P^{2/3})$ words, and they provide explicit constants in their analysis. Note that the size of the local memory M does not appear in this bound. Ballard et al. [3] reproduce this result for classical matrix multiplication as well as prove similar results for fast matrix multiplication. They distinguish between memory-dependent bounds (results described in § 2.1) and memory-independent bounds for parallel algorithms, and they show the two bounds relate and affect strong scaling behavior. In particular, when $M \gg n^2/P^{2/3}$ (or equivalently $P \gg n^3/M^{3/2}$), the memory-dependent bound is unattainable because the memory-independent bound is larger. Demmel et al. [11] extend the memory-independent results to the rectangular case (multiplying matrices of dimensions $n_1 \times n_2$ and $n_2 \times n_3$), showing that three different bounds apply that depend on the relative sizes of the three dimensions and the number of processors, and their proof provides explicit constants. For one of the cases, and for a restricted class of parallelizations, Kwasniewski et al. [17] prove a tighter constant and show that it is attainable by an optimal algorithm.

We summarize the constants obtained by these previous works and compare them to our results in Tab. 1. Further details of the comparison are given in § 6.1. Following Theorem 1, the table assumes $m = \max\{n_1, n_2, n_3\}$, $n = \text{median}\{n_1, n_2, n_3\}$, and $k = \min\{n_1, n_2, n_3\}$.

2.4 Communication-Optimal Parallel Matrix Multiplication Algorithms

Both theoretical and practical algorithms that attain the communication lower bounds have been proposed for various computation models and implemented on many different types of parallel systems. The idea of "3D algorithms" for matrix multiplication was developed soon after communication lower bounds were established; see [1, 2, 7, 15] for a few examples. These algorithms partition the 3D iteration space of matrix multiplication in each of the three dimensions and assign subblocks across a 3D logical grid of processors. McColl and Tiskin [19] and Demmel et al. [11] present recursive algorithms that effectively achieve similar 3D logical processor grid for square and general rectangular problems, respectively. High-performance implementations of these algorithms on today's supercomputers demonstrate that these algorithms are indeed practical and outperform standard library implementations [17, 23].

3 PRELIMINARIES

3.1 Parallel Computation Model

We consider the α-β-γ parallel machine model [9, 24]. In this model, each of P processors has its own local memory of size M and can compute only with data in its local memory. The processors can communicate data to and from other processors via messages that are sent over a fully connected network (i.e., each pair of processors has a dedicated link so that there is no contention on the network). Further, we assume the links are bidirectional so that a pair of processors can exchange data with no contention. Each processor can send and receive at most one message at the same time. The cost of communication is a function of two parameters α and β, where α is the per-message latency cost and β is the per-word bandwidth cost. A message of w words sent from one processor to another costs $\alpha + \beta w$. The parameter γ is the cost to perform a single arithmetic operation. For dense matrix multiplication when sufficiently large local memory is available, bandwidth cost nearly always dominates latency cost. Hence, we focus on the bandwidth cost in this work. In our model, the communication cost of an algorithm is counted along the critical path of the algorithm so that if two pairs of processors are communicating messages simultaneously, the communication cost is that of the largest message. In this work, we focus on memory-independent analysis, so the local memory size M can be assumed to be infinite. We consider limited-memory scenarios in § 6.2.

3.2 Fundamental Results

In this section we collect the fundamental existing results we use to prove our main result, Theorem 1. The first lemma is a geometric inequality that has been used before in establishing communication lower bounds for matrix multiplication [4, 11, 14]. We use it to relate the computation performed by a processor to the data it must access.

Lemma 1 (Loomis-Whitney [18]). *Let V be a finite set of lattice points in $\mathbb{R}^3$, i.e., points (i, j, k) with integer coordinates. Let $\phi_i(V)$ be the projection of V in the i-direction, i.e., all points (j, k) such that there exists an i so that $(i, j, k) \in V$. Define $\phi_j(V)$ and $\phi_k(V)$ similarly. Then*

$$|V| \leq |\phi_i(V)| \cdot |\phi_j(V)| \cdot |\phi_k(V)|,$$

where $|\cdot|$ denotes the cardinality of a set.

The next set of definitions and lemmas allow us to solve the key constrained optimization problem (Lemma 5) analytically. We first remind the reader of the definitions of differentiable convex and quasiconvex functions and of the Karush-Kuhn-Tucker (KKT) conditions. Here and throughout, we use boldface to indicate vectors and matrices and subscripts to index them, so that x_i is the ith element of $\boldsymbol{x}$, for example.

Definition 1 ([8, eq. (3.2)]). *A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is* convex *if its domain is a convex set and for all $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{dom}\ f$,*

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle.$$

Definition 2 ([8, eq. (3.20)]). *A differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ is* quasiconvex *if its domain is a convex set and for all $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{dom}\ g$,*

$$g(\boldsymbol{y}) \leq g(\boldsymbol{x}) \text{ implies that } \langle \nabla g(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle \leq 0.$$

Definition 3 ([8, eq. (5.49)]). *Consider an optimization problem of the form*

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0} \tag{1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ and $\boldsymbol{g} : \mathbb{R}^d \to \mathbb{R}^c$ are both differentiable. Define the dual variables $\boldsymbol{\mu} \in \mathbb{R}^c$, and let $\boldsymbol{J_g}$ be the Jacobian of $\boldsymbol{g}$. The Karush-Kuhn-Tucker (KKT) *conditions of $(\boldsymbol{x}, \boldsymbol{\mu})$ are as follows:*

- Primal feasibility*: $\boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0}$;*
- Dual feasibility*: $\boldsymbol{\mu} \geq 0$;*
- Stationarity*: $\nabla f(\boldsymbol{x}) + \boldsymbol{\mu} \cdot \boldsymbol{J_g}(\boldsymbol{x}) = \boldsymbol{0}$;*
- Complementary slackness*: $\mu_i g_i(\boldsymbol{x}) = 0$ for all $i \in \{1, \ldots, c\}$.*

The next two results establish that our key optimization problem in Lemma 5 can be solved analytically using the KKT conditions. While the results are not novel, we provide proofs for completeness.

Lemma 2 ([6, Lemma 2.2]). *The function $g_0(\boldsymbol{x}) = L - x_1 x_2 x_3$, for some constant L, is quasiconvex in the positive octant.*

Proof. Let $\boldsymbol{x}, \boldsymbol{y}$ be points in the positive octant with $g_0(\boldsymbol{y}) \leq g_0(\boldsymbol{x})$. Then $y_1 y_2 y_3 \geq x_1 x_2 x_3$. Applying the inequality of arithmetic and geometric means (AM-GM inequality) to the values y_1/x_1, y_2/x_2, y_3/x_3 (which are all positive), we have

$$\frac{1}{3}\left(\frac{y_1}{x_1} + \frac{y_2}{x_2} + \frac{y_3}{x_3}\right) \geq \left(\frac{y_1 y_2 y_3}{x_1 x_2 x_3}\right)^{1/3} \geq 1. \tag{2}$$

Then $\nabla g_0(\boldsymbol{x}) = \begin{bmatrix} -x_2 x_3 & -x_1 x_3 & -x_1 x_2 \end{bmatrix}$, and

$$\begin{aligned} \langle \nabla g_0(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle &= 3x_1 x_2 x_3 - y_1 x_2 x_3 - x_1 y_2 x_3 - x_1 x_2 y_3 \\ &= 3x_1 x_2 x_3 \left(1 - \frac{1}{3}\left(\frac{y_1}{x_1} + \frac{y_2}{x_2} + \frac{y_3}{x_3}\right)\right) \\ &\leq 0, \end{aligned}$$

where the last inequality follows from eq. (2). Then by Def. 2, g_0 is quasiconvex on the positive octant. □

Lemma 3. *Consider an optimization problem of the form given in eq.* (1)*. If f is a convex function and each g_i is a quasiconvex function, then the KKT conditions are sufficient for optimality.*

Proof. Suppose $\boldsymbol{x}^*$ and $\boldsymbol{\mu}^*$ satisfy the KKT conditions given in Def. 3. If $\boldsymbol{\mu}^* = \boldsymbol{0}$, then by stationarity, $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$. Then the convexity of f (Def. 1) implies

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*) + \langle \nabla f(\boldsymbol{x}^*), \boldsymbol{x} - \boldsymbol{x}^* \rangle = f(\boldsymbol{x}^*)$$

for all $\boldsymbol{x} \in \mathbf{dom}\ f$, which implies that $\boldsymbol{x}^*$ is a global optimum.

Now suppose $\boldsymbol{\mu}^* \neq \boldsymbol{0}$, then without loss of generality (and by dual feasibility) there exists $m \leq c$ such that $\mu_i^* > 0$ for $1 \leq i \leq m$ and $\mu_i^* = 0$ for $m < i \leq c$. Complementary slackness implies that $g_i(\boldsymbol{x}^*) = 0$ for $1 \leq i \leq m$. Consider any (primal) feasible $\boldsymbol{x} \in \mathbf{dom}\ f$. Then $g_i(\boldsymbol{x}) \leq 0$ for all i, and thus $g_i(\boldsymbol{x}) \leq g_i(\boldsymbol{x}^*)$ for $1 \leq i \leq m$. By quasiconvexity of g_i (Def. 2), this implies

$$\langle \nabla g_i(\boldsymbol{x}^*), \boldsymbol{x} - \boldsymbol{x}^* \rangle \leq 0.$$

Stationarity implies that $\nabla f(\boldsymbol{x}^*) = -\sum_{i=1}^m \mu_i^* \nabla g_i(\boldsymbol{x}^*)$, and thus

$$\langle \nabla f(\boldsymbol{x}^*), \boldsymbol{x} - \boldsymbol{x}^* \rangle = -\sum_{i=1}^m \mu_i^* \langle \nabla g_i(\boldsymbol{x}^*), \boldsymbol{x} - \boldsymbol{x}^* \rangle \geq 0.$$

By convexity of f (Def. 1), we therefore have

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*) + \langle \nabla f(\boldsymbol{x}^*), \boldsymbol{x} - \boldsymbol{x}^* \rangle \geq f(\boldsymbol{x}^*),$$

and thus $\boldsymbol{x}^*$ is a global optimum. □

4 MAIN LOWER BOUND RESULT

4.1 Lower Bounds on Individual Array Access

The following lemma establishes lower bounds on the number of elements of each individual matrix a processor must access based on the number of computations a given element is involved with. This result is used to establish a set of constraints in the key optimization problem used in the proof of Theorem 1.

Lemma 4. *Given a parallel matrix multiplication algorithm that multiplies an $n_1 \times n_2$ matrix $\mathbf{A}$ by an $n_2 \times n_3$ matrix $\mathbf{B}$ using P processors, any processor that performs at least $1/P$th of the scalar multiplications must access at least n_1n_2/P elements of $\mathbf{A}$ and at least n_2n_3/P elements of $\mathbf{B}$ and also compute contributions to at least n_1n_3/P elements of $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$.*

Proof. The total number of scalar multiplications that must be computed is $n_1n_2n_3$. Consider a processor that computes at least $1/P$th of these computations. Each element of $\mathbf{A}$ is involved in n_3 multiplications. If the processor accesses fewer than n_1n_2/P elements of $\mathbf{A}$, then it would perform fewer than $n_3 \cdot n_1n_2/P$ scalar multiplications, which is a contradiction. Likewise, each element of $\mathbf{B}$ is involved in n_1 multiplications. If the processor accesses fewer than n_2n_3/P elements of $\mathbf{B}$, then it would perform fewer than $n_1 \cdot n_2n_3/P$ scalar multiplications, which is a contradiction. Finally, each element of $\mathbf{C}$ is the sum of n_2 scalar multiplications. If the processor computes contributions to fewer than n_1n_3/P elements of $\mathbf{C}$, then it would perform fewer than $n_2 \cdot n_1n_3/P$ scalar multiplications, which is again a contradiction. □

4.2 Key Optimization Problem

The following lemma is the crux of the proof of our main result (Theorem 1). We state the optimization problem abstractly here, but it may be useful to have the following intuition: the variable vector $\boldsymbol{x}$ represents the sizes of the projections of the computation assigned to a single processor onto the three matrices, where x_1 corresponds to the smallest matrix and x_3 corresponds to the largest matrix. In order to design a communication-efficient algorithm, we wish to minimize the sum of the sizes of these projections subject to the constraints of matrix multiplication (and the processor performing $1/P$th of the computation), as specified by the Loomis-Whitney inequality (Lemma 1) and Lemma 4. A more rigorous argument that any parallel matrix multiplication algorithm is subject to these constraints is given in Theorem 1.

We are able to solve this optimization problem analytically using properties of convex optimization (Lemma 3). The three cases of the solution correspond to how many of the individual variable constraints are tight. When none of them is tight, we can minimize the sum of variables subject to the bound on their product by setting them all equal to each other (Case 3). However, when the individual variable constraints make this solution infeasible, those become active and the free variables must be adjusted (Cases 1 and 2).

Lemma 5. *Consider the following optimization problem:*

$$\min_{\boldsymbol{x} \in \mathbb{R}^3} x_1 + x_2 + x_3$$

such that

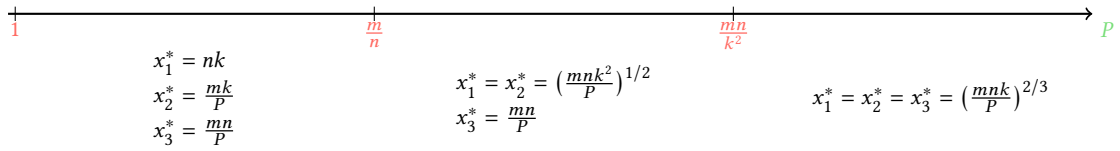
$$\left(\frac{mnk}{P}\right)^2 \leq x_1x_2x_3$$

$$\frac{nk}{P} \leq x_1$$
$$\frac{mk}{P} \leq x_2$$
$$\frac{mn}{P} \leq x_3,$$

where $m \geq n \geq k \geq 1$ and $P \geq 1$. The optimal solution $\boldsymbol{x}^*$ depends on the relative values of the constraints, yielding three cases:

(1) if $P \leq \frac{m}{n}$, then $x_1^* = nk$, $x_2^* = \frac{mk}{P}$, $x_3^* = \frac{mn}{P}$;
(2) if $\frac{m}{n} \leq P \leq \frac{mn}{k^2}$, then $x_1^* = x_2^* = \left(\frac{mnk^2}{P}\right)^{1/2}$, $x_3^* = \frac{mn}{P}$;
(3) if $\frac{mn}{k^2} \leq P$, then $x_1^* = x_2^* = x_3^* = \left(\frac{mnk}{P}\right)^{\frac{2}{3}}$.

This can be visualized as follows:



Proof. By Lemma 3, we can establish the optimality of the solution for each case by verifying that there exist dual variables such that the KKT conditions specified in Def. 3 are satisfied. This optimization problem fits the assumptions of Lemma 3 because the objective function and all but the first constraint are affine functions, which are convex and quasiconvex, and the first constraint is quasiconvex on the positive octant (which contains the intersection of the affine constraints) by Lemma 2.

To match standard notation (and that of Lemma 3), we let

$$f(\boldsymbol{x}) = x_1 + x_2 + x_3$$

and

$$\boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} (mnk/P)^2 - x_1x_2x_3 \\ nk/P - x_1 \\ mk/P - x_2 \\ mn/P - x_3 \end{bmatrix}.$$

Thus the gradient of the objective function is $\nabla f(\boldsymbol{x}) = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ and the Jacobian of the constraint function is

$$\boldsymbol{J_g}(\boldsymbol{x}) = \begin{bmatrix} -x_2x_3 & -x_1x_3 & -x_1x_2 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}.$$

Case 1 ($P \leq \frac{n}{m}$). We let

$$\boldsymbol{x}^* = \begin{bmatrix} nk & \frac{mk}{P} & \frac{mn}{P} \end{bmatrix}$$

and

$$\boldsymbol{\mu}^* = \begin{bmatrix} \frac{P^2}{m^2nk} & 0 & 1 - \frac{Pn}{m} & 1 - \frac{Pk}{m} \end{bmatrix}$$

and verify the KKT conditions. Primal feasibility is immediate, and dual feasibility follows from $P \leq \frac{m}{n} \leq \frac{m}{k}$, the condition of this case and by the assumption $n \geq k$. Stationarity follows from direct verification that

$$\boldsymbol{\mu}^* \cdot \boldsymbol{J_g}(\boldsymbol{x}^*) = \begin{bmatrix} -1 & -1 & -1 \end{bmatrix}.$$

Complementary slackness is satisfied because the only nonzero dual variables are μ_1^*, μ_3^*, and μ_4^*, and the 1st, 3rd, and 4th constraints are tight.

Case 2 ($\frac{m}{n} \leq P \leq \frac{mn}{k^2}$). We let

$$\boldsymbol{x}^* = \begin{bmatrix} \left(\frac{mnk^2}{P}\right)^{1/2} & \left(\frac{mnk^2}{P}\right)^{1/2} & \frac{mn}{P} \end{bmatrix}$$

and

$$\boldsymbol{\mu}^* = \begin{bmatrix} \left(\frac{P}{mnk^{2/3}}\right)^{3/2} & 0 & 0 & 1 - \left(\frac{Pk^2}{mn}\right)^{1/2} \end{bmatrix}$$

and verify the KKT conditions. The primal feasibility of $x_1 = x_2$ is satisfied because

$$\frac{nk}{P} \leq \frac{mk}{P} \leq \left(\frac{mnk^2}{P}\right)^{1/2}$$

where the first inequality follows from the assumption $m \geq n$ and the second inequality follows from $m/n \leq P$ (one condition of this case). The other constraints are clearly satisfied. Dual feasibility requires that $1-(Pk^2/mn)^{1/2} \geq 0$, which is satisfied because $P \leq mn/k^2$ (the other condition of this case). Stationarity can be directly verified. Complementary slackness is satisfied because the 1st and 4th constraints are both tight for $\boldsymbol{x}^*$, corresponding to the only nonzeros in $\boldsymbol{\mu}^*$.

Case 3 ($\frac{mn}{k^2} \leq P$). We let

$$\boldsymbol{x}^* = \begin{bmatrix} \left(\frac{mnk}{P}\right)^{2/3} & \left(\frac{mnk}{P}\right)^{2/3} & \left(\frac{mnk}{P}\right)^{2/3} \end{bmatrix}$$

and

$$\boldsymbol{\mu}^* = \begin{bmatrix} \left(\frac{P}{mnk}\right)^{4/3} & 0 & 0 & 0 \end{bmatrix}$$

and verify the KKT conditions. We first consider the primal feasibility conditions. We have

$$\frac{nk}{P} \leq \frac{mk}{P} \leq \frac{mn}{P} \leq \left(\frac{mnk}{P}\right)^{2/3},$$

where the first two inequalities are implied by the assumption $m \geq n \geq k$ and the last follows from $\frac{mn}{k^2} \leq P$, the condition of this case. Dual feasibility is immediate, and stationarity is directly verified. Complementary slackness is satisfied because the 1st constraint is tight for $\boldsymbol{x}^*$ and μ_1^* is the only nonzero.

Note that the optimal solutions coincide at boundary points between cases so that the values are continuous as P varies. □

4.3 Communication Lower Bound

We now state our main result, memory-independent communication lower bounds for general matrix multiplication with tight constants. After the general result, we also present a corollary for square matrix multiplication. The tightness of the constants in the lower bound is proved in § 5.

Theorem 1. *Consider a classical matrix multiplication computation involving matrices of size $n_1 \times n_2$ and $n_2 \times n_3$. Let $m = \max\{n_1, n_2, n_3\}$, $n = \text{median}\{n_1, n_2, n_3\}$, and $k = \min\{n_1, n_2, n_3\}$, so that $m \geq n \geq k$. Any parallel algorithm*

using P processors that starts with one copy of the two input matrices and ends with one copy of the output matrix and load balances either the computation or the data must communicate at least

$$D - \frac{mn + mk + nk}{P} \text{ words of data,}$$

where

$$D = \begin{cases} \frac{mn+mk}{P} + nk & \text{if} \quad 1 \le P \le \frac{m}{n} \\ 2\left(\frac{mnk^2}{P}\right)^{1/2} + \frac{mn}{P} & \text{if} \quad \frac{m}{n} \le P \le \frac{mn}{k^2} \\ 3\left(\frac{mnk}{P}\right)^{2/3} & \text{if} \quad \frac{mn}{k^2} \le P. \end{cases}$$

Proof. To establish the lower bound, we focus on a single processor. If the algorithm load balances the computation, then every processor performs mnk/P scalar multiplications, and there exists some processor whose input data at the start of the algorithm plus output data at the end of the algorithm must be at most $(mn + mk + nk)/P$ words of data (otherwise the algorithm would either start with more than one copy of the input matrices or end with more than one copy of the output matrix). If the algorithm load balances the data, then every processor starts and end with a total of $(mn + mk + nk)/P$ words, and some processor must perform at least mnk/P scalar multiplications (otherwise fewer than mnk multiplications are performed). In either case, there exists a processor that performs at least mnk/P multiplications and has access to at most $(mn + mk + nk)/P$ data.

Let F be the set of multiplications assigned to this processor, so that $|F| \ge mnk/P$. Each element of F can be indexed by three indices (i_1, i_2, i_3) and corresponds to the multiplication of $\mathbf{A}(i_1, i_2)$ with $\mathbf{B}(i_2, i_3)$, which contributes to the result $\mathbf{C}(i_1, i_3)$. Let $\phi_{\mathbf{A}}(F)$ be the projection of the set F onto the matrix $\mathbf{A}$, so that $\phi_{\mathbf{A}}(F)$ are the entries of $\mathbf{A}$ required for the processor to perform the scalar multiplications in F. Here, elements of $\phi_{\mathbf{A}}(F)$ can be indexed by two indices: $\phi_{\mathbf{A}}(F) = \{(i_1, i_2) : \exists\ i_3 \text{ s.t. } (i_1, i_2, i_3) \in F\}$. Define $\phi_{\mathbf{B}}(F)$ and $\phi_{\mathbf{C}}(F)$ similarly. The processor must access all of the elements in $\phi_{\mathbf{A}}(F)$, $\phi_{\mathbf{B}}(F)$, and $\phi_{\mathbf{C}}(F)$ in order to perform all the scalar multiplications in F. Because the processor starts and ends with at most $(mn + mk + nk)/P$ data, the communication performed by the processor is at least

$$|\phi_{\mathbf{A}}(F)| + |\phi_{\mathbf{B}}(F)| + |\phi_{\mathbf{C}}(F)| - \frac{mn + mk + nk}{P},$$

which is a lower bound on the communication along the critical path of the algorithm.

In order to lower bound $|\phi_{\mathbf{A}}(F)| + |\phi_{\mathbf{B}}(F)| + |\phi_{\mathbf{C}}(F)|$, we form a constrained minimization problem with this expression as the objective function and constraints derived from Lemmas 1 and 4. The Loomis-Whitney inequality (Lemma 1) implies that

$$|\phi_{\mathbf{A}}(F)| \cdot |\phi_{\mathbf{B}}(F)| \cdot |\phi_{\mathbf{C}}(F)| \ge |F| \ge \frac{n_1 n_2 n_3}{P} = \frac{mnk}{P},$$

and the lower bound on the projections from Lemma 4 means

$$|\phi_{\mathbf{A}}(F)| \ge \frac{n_1 n_2}{P}, \quad |\phi_{\mathbf{B}}(F)| \ge \frac{n_2 n_3}{P}, \quad |\phi_{\mathbf{C}}(F)| \ge \frac{n_1 n_3}{P}.$$

For any algorithm, the processor's projections must satisfy these constraints, so the sum of their sizes must be at least the minimum value of optimization problem. Then by Lemma 5 (and assigning the projections to x_1, x_2, x_3 appropriately based on the relative sizes of n_1, n_2, n_3), the result follows. □

We also state the result for square matrix multiplication, which is a direct application of Theorem 1 with $n_1 = n_2 = n_3$.

Corollary 2. *Consider a classical matrix multiplication computation involving two matrices of size $n \times n$. Any parallel algorithm using P processors that starts with one copy of the input data and ends with one copy of the output data and load*

balances either the computation or the data must communicate at least

$$3\frac{n^2}{P^{2/3}} - 3\frac{n^2}{P} \text{ words of data.}$$

5 OPTIMAL PARALLEL ALGORITHM

In this section we present an optimal parallel algorithm (Alg. 1) to show that the lower bound (including the constants) is tight. The idea is to organize the processors into a 3D processor grid and assign the computation of the matrix multiplication (a 3D iteration space) to processors according to their location in the grid. The algorithm is not new, but we present it here in full detail for completeness and to provide the complete analysis, which has not appeared before. In particular, Alg. 1 is nearly identical to the one proposed by Aggarwal et al. [1], though they use the LPRAM model and analyze only the case where P is large. In the LPRAM model, for example, processors can read concurrently from a global shared memory, while in the α-β-γ model, the data is distributed across local memories and is shared via collectives like All-Gathers. Demmel et al. [11] present and analyze their recursive algorithm to show its asymptotic optimality in all three cases, but they do not track constants. See § 2.4 for more discussion of previous work on optimal parallel algorithms.

Consider the multiplication of an $n_1 \times n_2$ matrix $\mathbf{A}$ with an $n_2 \times n_3$ matrix $\mathbf{B}$, and let $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$. Algorithm 1 organizes the P processors into a 3-dimensional $p_1 \times p_2 \times p_3$ logical processor grid with $p_1 p_2 p_3 = P$. Note that one or two of the processor grid dimensions may be equal to 1, which simplifies to a 2- or 1-dimensional grid. A processor coordinate is represented as (p_1', p_2', p_3'), where $1 \le p_k' \le p_k$, for $k = 1, 2, 3$.

The basic idea of the algorithm is to perform two collective operations, All-Gathers, so that each processor receives the input data it needs to perform all of its computation (in an All-Gather, all the processors involved end up with the union of the input data that starts on each processor). The result of each local computation must be summed with all other contributions to the same output matrix entries from other processors, and the summations are performed via a Reduce-Scatter collective operation (in a Reduce-Scatter, the sum of the input data from all processors is computed so that it ends up evenly distributed across processors).

Algorithm 1 Comm-Optimal Parallel Matrix Multiplication

Input: $\mathbf{A}$ is $n_1 \times n_2$, $\mathbf{B}$ is $n_2 \times n_3$, $p_1 \times p_2 \times p_3$ logical processor grid
Output: $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$ is $n_1 \times n_3$

1: (p_1', p_2', p_3') is my processor ID
2: // Gather input matrix data
3: $\mathbf{A}_{p_1' p_2'}$ = All-Gather($\mathbf{A}_{p_1' p_2' p_3'}$, $(p_1', p_2', :)$)
4: $\mathbf{B}_{p_2' p_3'}$ = All-Gather($\mathbf{B}_{p_1' p_2' p_3'}$, $(:, p_2', p_3')$)
5: // Perform local computation
6: $\mathbf{D}_{p_1' p_2' p_3'} = \mathbf{A}_{p_1' p_2'} \cdot \mathbf{B}_{p_2' p_3'}$
7: // Sum results to compute $\mathbf{C}_{p_1' p_3'}$
8: $\mathbf{C}_{p_1' p_2' p_3'}$ = Reduce-Scatter($\mathbf{D}_{p_1' p_2' p_3'}$, $(p_1', :, p_3')$)

Algorithm 1 imposes requirements on the initial distribution of the input matrices and the final distribution of the output. These conditions do not always hold in practice, but to show that the lower bound (which makes no assumption on data distribution except that only 1 copy of the input exists at the start of the computation) is tight, we allow the algorithm to specify its distributions. For simplicity of explanation, we assume that p_1, p_2, p_3 evenly divide n_1, n_2, n_3,

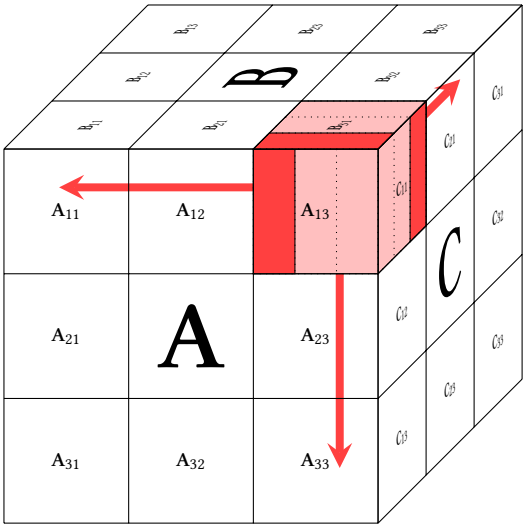


Fig. 1. Visualization of Alg. 1 with a $3 \times 3 \times 3$ processor grid. The 3D iteration space is mapped onto the processor grid, and the matrices are mapped to the faces of the grid. The dark highlighting corresponds to the input data initially owned and the output data finally owned by processor (1, 3, 1), and the light highlighting signifies the data of other processors it uses to perform the local computation. The arrows show the sets of processors involved in the three collective operations involving processor (1, 3, 1).

respectively. We use the notation $\mathbf{A}_{p_1'p_2'}$ to denote the submatrix of $\mathbf{A}$ such that

$$\mathbf{A}_{p_1'p_2'} = \mathbf{A}\left((p_1'-1)\cdot\frac{n_1}{p_1}+1 : p_1'\cdot\frac{n_1}{p_1}, (p_2'-1)\cdot\frac{n_2}{p_2}+1 : p_2'\cdot\frac{n_2}{p_2}\right),$$

and we define $\mathbf{B}_{p_2'p_3'}$ and $\mathbf{C}_{p_1'p_3'}$ similarly. The algorithm assumes that at the start of the computation, $\mathbf{A}_{p_1'p_2'}$ is distributed evenly across processors $(p_1', p_2', :)$ and $\mathbf{B}_{p_2'p_3'}$ is distributed evenly across processors $(:, p_2', p_3')$. We define $\mathbf{A}_{p_1'p_2'p_3'}$ and $\mathbf{B}_{p_1'p_2'p_3'}$ as the elements of the input matrices that processor (p_1', p_2', p_3') initially owns. At the end of the algorithm, $\mathbf{C}_{p_1'p_3'}$ is distributed evenly across processors $(p_1', :, p_3')$, and we let $\mathbf{C}_{p_1'p_2'p_3'}$ be the elements owned by processor (p_1', p_2', p_3').

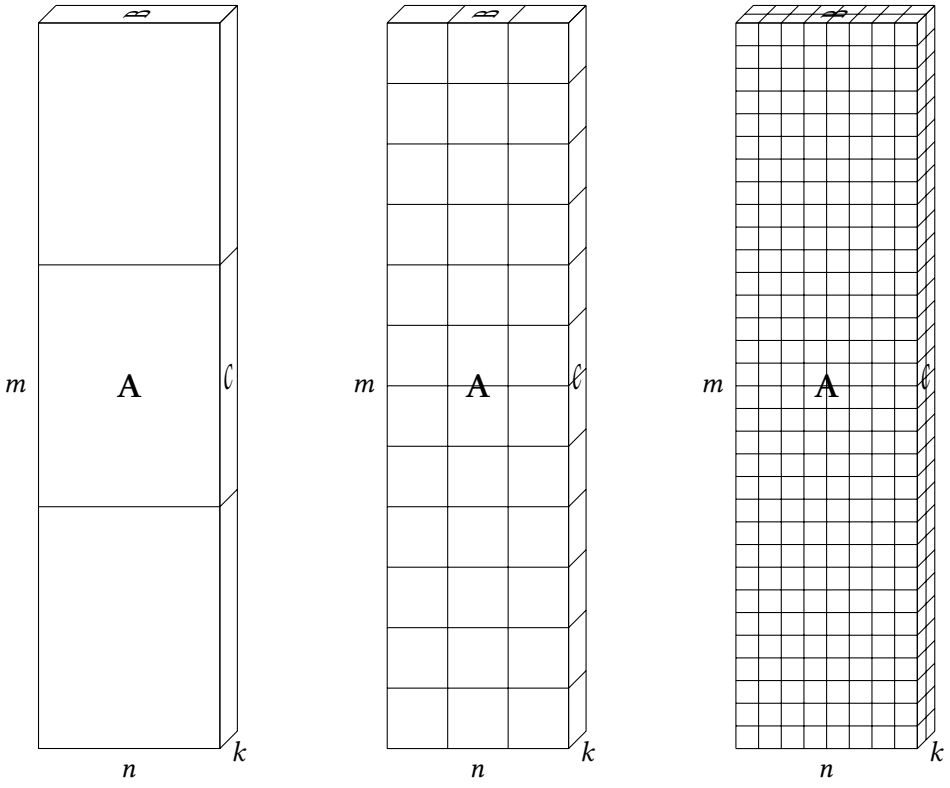
Figure 1 presents a visualization of Alg. 1. In this example, we have $n_1 = n_2 = n_3$, and 27 processors are arranged in a $3 \times 3 \times 3$ grid. We highlight the data and communication of a particular processor with ID $(1, 3, 1)$. The dark highlighting corresponds to the input data initially owned by the processor ($\mathbf{A}_{131}$ and $\mathbf{B}_{131}$) as well as the output data owned by the processor at the end of the computation ($\mathbf{C}_{131}$). The figure shows each of these submatrices as block columns of the submatrices $\mathbf{A}_{13}$, $\mathbf{B}_{31}$, and $\mathbf{C}_{11}$, but any even distribution of these across the same set of processors suffices. The light highlighting of the submatrices $\mathbf{A}_{13}$, $\mathbf{B}_{31}$, and $\mathbf{C}_{11}$ corresponds to the data of other processors involved in the local computation on processor $(1, 3, 1)$, and their size corresponds to the communication cost. The three collectives that involve processor $(1, 3, 1)$ occur across three different fibers in the processor grid, as depicted by the arrows in the figure.

5.1 Cost Analysis

Now we analyze computation and communication costs of the algorithm. Each processor performs $\frac{n_1}{p_1}\cdot\frac{n_2}{p_2}\cdot\frac{n_3}{p_3} = \frac{n_1n_2n_3}{P}$ local computations in Line 6. Communication occurs only in the All-Gather and Reduce-Scatter collectives in Lines 3, 4, and 8. Each processor is involved in two All-Gathers involving input matrices and one Reduce-Scatter involving

the output matrix. Lines 3, 4 specify simultaneous All-Gathers across sets of p_3, p_1 processors, respectively, and Line 8 specifies simultaneous Reduce-Scatters across sets of p_2 processors. Note that if $p_k = 1$ for any $k = 1, 2, 3$, then the corresponding collective can be ignored as no communication occurs. The difference between Alg. 1 and [1, Algorithm 1] is the Reduce-Scatter collective, which replaces the All-to-All collective and has smaller latency cost.

We assume that bandwidth-optimal algorithms, such as bidirectional exchange or recursive doubling/halving, are used for the All-Gather and Reduce-Scatter collectives. The optimal communication cost of these collectives on p processors is $(1 - \frac{1}{p})w$, where w is the words of data in each processor after All-Gather or before Reduce-Scatter collective [9, 24]. Each processor also performs $(1 - \frac{1}{p})w$ computations for the Reduce-Scatter collective.



(a) 1D case: $P = 3$ with grid $3 \times 1 \times 1$ (b) 2D case: $P = 36$ with grid $12 \times 3 \times 1$ (c) 3D case: $P = 512$ with grid $32 \times 8 \times 2$

Fig. 2. Example parallelizations of iteration space of multiplication of a 9600×2400 matrix $\mathbf{A}$ with a 2400×600 matrix $\mathbf{B}$

Hence the communication costs of Lines 3, 4 in Algorithm 1 are $(1 - \frac{1}{p_3})\frac{n_1 n_2}{p_1 p_2}$ and $(1 - \frac{1}{p_1})\frac{n_2 n_3}{p_2 p_3}$, respectively, to accomplish All-Gather operations, and the communication cost of performing the Reduce-Scatter operation in Line 8 is $(1 - \frac{1}{p_2})\frac{n_1 n_3}{p_1 p_3}$. Note that if $p_k = 1$ for any $k = 1, 2, 3$, then the cost of the corresponding collective reduces to 0. Thus the overall cost of Algorithm 1 is

$$\frac{n_1 n_2}{p_1 p_2} + \frac{n_2 n_3}{p_2 p_3} + \frac{n_1 n_3}{p_1 p_3} - \left(\frac{n_1 n_2 + n_2 n_3 + n_1 n_3}{P}\right). \tag{3}$$

Due to Reduce-Scatter operation, each processor also performs $(1 - \frac{1}{p_2})\frac{n_1 n_3}{p_1 p_3}$ computations, which is dominated by the $\frac{n_1 n_2 n_3}{P}$ computations of Line 6.

5.2 Optimal Processor Grid Selection

The communication cost of Algorithm 1, given by eq. (3), depends on the processor grid dimensions. Here we discuss how to select the processor grid dimensions such that the lower bound on communication given in Theorem 1 is attained by Alg. 1 given the matrices dimensions n_1, n_2 and n_3 and the number of processors P. As before, we let m, n, k represent the maximum, median, and minimum values of the three dimensions. Letting p_1, p_2, p_3 be the grid dimensions, we similarly define p, q, r to be the processor grid dimensions corresponding to matrix dimensions m, n, k, respectively. Because the order of processor grid dimensions will be chosen to be consistent with the matrix dimensions, we will have $p \geq q \geq r$. To demonstrate the tightness of the lower bound, the analysis below assumes that the processor grid dimensions divide the matrices dimensions.

Following Theorem 1, depending on the relative sizes of the aspect ratios among matrix dimensions and the number of processors, we encounter three cases that correspond to 3D, 2D, and 1D processor grids. That is, when $p_i = 1$ for one value of i, then the processor grid is effectively 2D, and when $p_i = 1$ for two values of i, the processor grid is effectively 1D. In the following we show how to obtain the grid dimensions and show that Algorithm 1 attains the communication lower bound given in Theorem 1 in each case.

First, suppose $P \leq \frac{m}{n}$. In this case, we set $r = q = 1$, and set $p = P$ to obtain a 1D grid. From eq. (3), Algorithm 1 has communication cost

$$\frac{mn + mk}{P} + nk - \frac{mn + mk + nk}{P} = \left(1 - \frac{1}{P}\right) nk,$$

which matches the 1st case of Theorem 1.

Now suppose $\frac{m}{n} < P \leq \frac{mn}{k^2}$. We set $r = 1$, and set p and q such that $\frac{m}{p} = \frac{n}{q}$, yielding $p = \left(\frac{P}{mn}\right)^{1/2} m$ and $q = \left(\frac{P}{mn}\right)^{1/2} n$. Note that the assumption $\frac{m}{n} < P$ is required so that $q > 1$, and $p > 1$ also follows. Our analysis also assumes that p and q are integers, which is sufficient to show that the lower bound is tight in general as there are an infinite number of dimensions for which the assumption holds. In this case, we have a 2D processor grid, and Algorithm 1 has communication cost

$$\frac{mn}{pq} + \frac{mk}{p} + \frac{nk}{q} - \frac{mn + mk + nk}{pq} = 2\left(\frac{mnk^2}{P}\right)^{1/2} - \frac{mk + nk}{P},$$

matching the 2nd case of Theorem 1.

Finally, suppose $\frac{mn}{k^2} < P$. As suggested in [1], we set the grid dimensions such that $\frac{m}{p} = \frac{n}{q} = \frac{k}{r}$. That is, $r = \left(\frac{P}{mnk}\right)^{1/3} k$, $q = \left(\frac{P}{mnk}\right)^{1/3} n$, and $p = \left(\frac{P}{mnk}\right)^{1/3} m$. Note that the assumption $\frac{mn}{k^2} < P$ is required so that $r > 1$ (which also implies $q > 1$ and $p > 1$). This assumption was implicit in the analysis of [1]. Again, we assume that p, q, r are integers. Thus, we have a 3D processor grid and a communication cost of

$$3\left(\frac{mnk}{P}\right)^{2/3} - \frac{mn + mk + nk}{P},$$

which matches the 3rd case of Theorem 1.

Comparing the obtained communication cost in each case with the lower bound obtained in Theorem 1 we conclude that Algorithm 1 is optimal given the grid dimensions are selected as above.

5.3 Optimal Processor Grid Examples

Figure 2 illustrates each of the three cases for a fixed set of matrix dimensions. Here we consider multiplying a 9600×2400 matrix $\mathbf{A}$ with a 2400×600 $\mathbf{B}$ so that $\mathbf{C}$ is 9600×600, so in our notation with $m \geq n \geq k$, $\mathbf{A}$ is $m \times n$, $\mathbf{B}$ is $n \times k$, and $\mathbf{C}$ is $m \times k$. The 3D $m \times n \times k$ iteration space is visualized with faces corresponding to correctly oriented matrices. In this example, we consider $P \in \{3, 36, 512\}$.

With 3 processors, we fall into the 1st case, as $P \leq \frac{m}{n} = 4$, and the optimal processor grid is $3 \times 1 \times 1$, which is 1D as shown in Fig. 2a. Note that the computation assigned to each processor is not a cube in this case, that is, $\frac{m}{p} \neq \frac{n}{q} \neq \frac{k}{r}$. The only data that must be communicated are entries of $\mathbf{B}$, though all processors need all of $\mathbf{B}$.

When $P = 36$, we fall into the 2nd case, and the optimal processor grid is 2D: $12 \times 3 \times 1$, as shown in Fig. 2b. Here we see that the iteration space assigned to each processor is $800 \times 800 \times 600$, so $\frac{m}{p} = \frac{n}{q} \neq \frac{k}{r}$. In this case, entries of $\mathbf{B}$ and $\mathbf{C}$ must be communicated, but each entry of $\mathbf{A}$ is required by only one processor.

Finally, for $P = 512$, we satisfy $P > \frac{mn}{k^2} = 64$ and fall into the 3rd case. The optimal processor grid is shown in Fig. 2c to be $32 \times 8 \times 2$, and we see that the local computation of each processor is a cube: $\frac{m}{p} = \frac{n}{q} = \frac{k}{r}$. For a 3D grid, entries of all 3 matrices are communicated.

6 CONCLUSION

Theorem 1 establishes memory-independent communication lower bounds for parallel matrix multiplication. By casting the lower bound on accessed data as the solution to a constrained optimization problem, we are able to obtain a result with explicit constants spanning over three scenarios that depend on the relative sizes of the matrix aspect ratios and the number of processors. Algorithm 1 demonstrates that the constants established in Theorem 1 are tight, as the algorithm is general enough to be applied in each of the three scenarios by tuning the processor grid. As we discuss below, our lower bound proof technique tightens the constants proved in earlier work, and we believe it can be generalized to improve known communication lower bounds for other computations.

6.1 Comparison to Existing Results

We now provide full details of the constants presented in Tab. 1, and compare the previous results with the constants of Theorem 1. The first row of the table gives the constant from the proof by Aggarwal, Chandra, and Snir [2, Theorem 2.3]. While the result is stated asymptotically, an explicit constant is given in a key lemma ([2, Lemma 2.2]) used in the proof, from which we can derive the constant for the main result.

The second row of the table corresponds to the work of Irony, Toledo, and Tiskin [14], who establish the first parallel bounds for matrix multiplication. Their memory-independent bound is stated for square matrices with a parametrized prefactor corresponding to the amount of local memory available [14, Theorem 5.1]. If we generalize it straightforwardly to rectangular dimensions and minimize the prefactor over any amount of local memory, then we obtain a bound of at least $1/2 \cdot (mnk/P)^{2/3}$, which is asymptotically tight for $mn/k^2 \leq P$. They do not provide any tighter results for $P < mn/k^2$.

The third row of the table corresponds to the results of Demmel et al. [11]. This work was the first to establish bounds for small values of P and identify the three cases of asymptotic expressions. Theorem 1 obtains the same cases and leading order terms (up to constant factors) [11, Table I], and we present the explicit constant factors for leading order terms derived in [11, Section II.B]. We note that the boundaries between cases differ by a constant in that paper, which we do not reflect in Tab. 1. Compared to these results, Theorem 1 establishes a tighter constant in all three cases.

We note that Kwasniewski et al. claim a combined result of memory-dependent and memory-independent parallel bounds [17, Theorem 2]. The memory-independent term has a constant that matches the 3rd case of Theorem 1. However, the proof includes a restrictive assumption on parallelization strategies, requiring that each processor is assigned a set of domains that are subblocks of the iteration space with dimensions $a \times a \times b$ for some a, b, and therefore does not apply to all parallel algorithms.

6.2 Limited-Memory Scenarios

The local memory required by Alg. 1 matches the amount of communication performed plus the data already owned by the processor, which is given by the positive terms in eq. (3) and matches the value of D in Theorem 1 with the optimal processor grid. Note that the local memory M must be large enough to store the inputs and output matrices, so $M \geq (mn + mk + nk)/P$. When 1D or 2D processor grids are used, the local memory required is no more than a constant more than the minimum required to store the problem. Further, Alg. 1 can be adapted to reduce the temporary memory required to a negligible amount at the expense of higher latency cost but without affecting the bandwidth cost, and thus the algorithmic approach can be used even in extremely limited memory scenarios. In the case of 3D processor grids, however, the temporary memory used by Alg. 1 asymptotically dominates the minimum required, and thus the algorithm cannot be applied in limited-memory scenarios. Reducing the memory footprint in this case necessarily increases the bandwidth cost. Algorithms that smoothly trade off memory for communication savings in these limited memory scenarios are well studied [3, 17, 19, 23].

From the lower bound point of view, while Theorem 1 is always valid, it may not be the tightest bound in limited-memory scenarios. The memory-dependent bound with leading term $2mnk/(P\sqrt{M})$ (see [17, 20, 22] and discussion in § 2.1) can be larger. In particular, this occurs when $mn/k^2 < P \leq 8/27 \cdot mnk/M^{3/2}$, and the memory-dependent bound dominates the memory-independent bound of $3(mnk/P)^{2/3}$ in that case. This scenario implies that $M < 4/9 \cdot (mnk/P)^{2/3}$, in which case the temporary space required by Alg. 1 exceeds the available memory. Thus, the tightness of Theorem 1 for $mn/k^2 < P$ requires an assumption of sufficient memory.

When $P \leq mn/k^2$, the memory-independent bounds in the first two cases of Theorem 1 are always tight, with no assumption on local memory size. That is, the memory-dependent bound never dominates the memory-independent bound. Consider the 2nd case, so that $m/n \leq P \leq mn/k^2$ and the memory-independent bound is $2(mnk^2/P)^{1/2}$. Because the local memory must be large enough to store the largest matrix as well as the other two matrices, we have $M > mn/P$. This implies $2mnk/(P\sqrt{M}) < 2(mnk^2/P)^{1/2}$, so the memory-independent bound dominates.

Suppose further that $P \leq \frac{m}{n}$. In this case, the leading-order term of the memory-independent bound is nk. This 1st-case bound dominates the 2nd-case bound, which dominates the memory-dependent bound from the argument above. Comparison of the full bounds of the 1st and 2nd cases simplifies to $2(mnk^2/P)^{1/2} \leq mk/P + nk$, which holds by the arithmetic-geometric mean inequality.

6.3 Extensions

The proof technique we use to obtain Theorem 1 is more generally applicable. The basic approach of defining a constrained optimization problem to minimize the sum amount of data accessed subject to constraints on that data that depend on the nature of the computation has been used before for matrix multiplication [22] and for tensor computations [5, 6]. The key to the results presented in this work is the imposition of lower bound constraints on the data accessed in each individual array given by Lemma 4. These lower bounds become active when the aspect ratios of the matrices are large relative to the number of processors and allow for tighter lower bounds in those cases. The

argument given in Lemma 4 is not specific to matrix multiplication, it depends only on the number of operations a given word of data is involved in, so it can be applied to many other computations that have iteration spaces with uneven dimensions. We believe this will yield new or tighter parallel communication bounds in several cases.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under Grant No. CCF-1942892 and OAC-2106920. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement No. 810367).

REFERENCES

[1] R. C. Agarwal, S. M. Balle, F. G. Gustavson, M. Joshi, and P. Palkar. 1995. A three-dimensional approach to parallel matrix multiplication. *IBM Journal of Research and Development* 39, 5 (1995), 575–582. https://doi.org/10.1147/rd.395.0575

[2] A. Aggarwal, A. K. Chandra, and M. Snir. 1990. Communication complexity of PRAMs. *Theor. Comp. Sci.* 71, 1 (1990), 3–28. https://doi.org/10.1016/0304-3975(90)90188-N

[3] G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, and O. Schwartz. 2012. Brief announcement: strong scaling of matrix multiplication algorithms and memory-independent communication lower bounds. In *Proceedings of the 24th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '12)*. ACM, New York, NY, USA, 77–79. https://doi.org/10.1145/2312005.2312021

[4] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. 2012. Graph expansion and communication costs of fast matrix multiplication. *J. ACM* 59, 6, Article 32 (2012), 23 pages. https://doi.org/10.1145/2395116.2395121

[5] G. Ballard, N. Knight, and K. Rouse. 2018. Communication Lower Bounds for Matricized Tensor Times Khatri-Rao Product. In *IPDPS*. 557–567. https://doi.org/10.1109/IPDPS.2018.00065

[6] G. Ballard and K. Rouse. 2020. General Memory-Independent Lower Bound for MTTKRP. In *SIAM PP*. 1–11. https://doi.org/10.1137/1.9781611976137.1

[7] J. Berntsen. 1989. Communication efficient matrix multiplication on hypercubes. *Parallel Comput.* 12, 3 (1989), 335–342. https://doi.org/10.1016/0167-8191(89)90091-4

[8] S. Boyd and L. Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press. https://web.stanford.edu/~boyd/cvxbook/

[9] E. Chan, M. Heimlich, A. Purkayastha, and R. van de Geijn. 2007. Collective communication: theory, practice, and experience. *Concurrency and Computation: Practice and Experience* 19, 13 (2007), 1749–1783. https://doi.org/10.1002/cpe.1206

[10] M. Christ, J. Demmel, N. Knight, T. Scanlon, and K. Yelick. 2013. *Communication Lower Bounds and Optimal Algorithms for Programs That Reference Arrays - Part 1*. Technical Report UCB/EECS-2013-61. EECS Department, University of California, Berkeley. http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-61.html

[11] J. Demmel, D. Eliahu, A. Fox, S. Kamil, B. Lipshitz, O. Schwartz, and O. Spillinger. 2013. Communication-Optimal Parallel Recursive Rectangular Matrix Multiplication. In *IPDPS*. 261–272. https://doi.org/10.1109/IPDPS.2013.80

[12] Jack Dongarra, Jean-François Pineau, Yves Robert, Zhiao Shi, and Frédéric Vivien. 2008. Revisiting Matrix Product on Master-Worker Platforms. *International Journal of Foundations of Computer Science* 19, 06 (2008), 1317–1336. https://doi.org/10.1142/S0129054108006303

[13] J. W. Hong and H. T. Kung. 1981. I/O complexity: The red-blue pebble game. In *STOC*. ACM, 326–333. https://doi.org/10.1145/800076.802486

[14] D. Irony, S. Toledo, and A. Tiskin. 2004. Communication lower bounds for distributed-memory matrix multiplication. *J. Par. and Dist. Comp.* 64, 9 (2004), 1017–1026. https://doi.org/10.1016/j.jpdc.2004.03.021

[15] S. Lennart Johnsson. 1993. Minimizing the communication time for matrix multiplication on multiprocessors. *Parallel Comput.* 19, 11 (1993), 1235 – 1257. https://doi.org/10.1016/0167-8191(93)90029-K

[16] G. Kwasniewski, M. Kabic, T. Ben-Nun, A. N. Ziogas, J. E. Saethre, A. Gaillard, T. Schneider, M. Besta, A. Kozhevnikov, J. VandeVondele, and T. Hoefler. 2021. On the Parallel I/O Optimality of Linear Algebra Kernels: Near-Optimal Matrix Factorizations. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '21)*. Association for Computing Machinery, New York, NY, USA, Article 70, 15 pages. https://doi.org/10.1145/3458817.3476167

[17] G. Kwasniewski, M. Kabić, M. Besta, J. VandeVondele, R. Solcà, and T. Hoefler. 2019. Red-Blue Pebbling Revisited: Near Optimal Parallel Matrix-Matrix Multiplication. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '19)*. Association for Computing Machinery, New York, NY, USA, Article 24, 22 pages. https://doi.org/10.1145/3295500.3356181

[18] L. H. Loomis and H. Whitney. 1949. An inequality related to the isoperimetric inequality. *Bull. Amer. Math. Soc.* 55, 10 (1949), 961 – 962. https://doi.org/10.1090/S0002-9904-1949-09320-5

[19] W. McColl and A. Tiskin. 1999. Memory-Efficient Matrix Multiplication in the BSP Model. *Algorithmica* 24, 3-4 (1999), 287–297. https://doi.org/10.1007/PL00008264

[20] A. Olivry, J. Langou, L.-N. Pouchet, P. Sadayappan, and F. Rastello. 2020. Automated Derivation of Parametric Data Movement Lower Bounds for Affine Programs. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2020)*. ACM, New

York, NY, USA, 808–822. https://doi.org/10.1145/3385412.3385989

[21] M. Scquizzato and F. Silvestri. 2014. Communication Lower Bounds for Distributed-Memory Computations. In *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, Vol. 25. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 627–638. https://doi.org/10.4230/LIPIcs.STACS.2014.627

[22] T. M. Smith, B. Lowery, J. Langou, and R. A. van de Geijn. 2019. *A Tight I/O Lower Bound for Matrix Multiplication.* Technical Report. arXiv. https://doi.org/10.48550/arXiv.1702.02017

[23] E. Solomonik and J. Demmel. 2011. Communication-Optimal Parallel 2.5D Matrix Multiplication and LU Factorization Algorithms. In *Euro-Par 2011 Parallel Processing*, Emmanuel Jeannot, Raymond Namyst, and Jean Roman (Eds.). Lecture Notes in Computer Science, Vol. 6853. Springer Berlin Heidelberg, 90–109. https://doi.org/10.1007/978-3-642-23397-5_10

[24] R. Thakur, R. Rabenseifner, and W. Gropp. 2005. Optimization of Collective Communication Operations in MPICH. *Intl. J. High Perf. Comp. App.* 19, 1 (2005), 49–66. https://doi.org/10.1177/1094342005051521