Contention Bounds for Combinations of Computation Graphs and Network Topologies

Grey Ballard
James Demmel
Andrew Gearhart
Benjamin Lipshitz
Oded Schwartz
Sivan Toledo

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-147
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-147.html

August 8, 2014

Copyright © 2014, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Acknowledgement

Research funded by DARPA Award HR0011-12-2-0016, the Center for Future Architecture Research, a member of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA, and ASPIRE Lab industrial sponsors and affiliates Intel, Google, Nokia, NVIDIA, Oracle, MathWorks and Samsung. Also funded by U.S. DOE Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program DE-SC0004938, DE-SC0005136, DE-SC0003959,DE-SC0008700, DE-SC0008699, DE-SC0010200 and DOE AC02-05CH11231. Also supported by grants 1878/14 and 1045/09 from the Israel Science Foundation, 2010231 from the US-Israel Bi-National Science Foundation, grant 3-10891 from the Ministry of Science and

Technology, Israel and in part by the Sandia National Laboratories Truman Fellowship.

Contention Bounds for Combinations of Computation Graphs and Network Topologies

Grey Ballard
Sandia National Laboratories
gmballa@sandia.gov

James Demmel
UC Berkeley
demmel@cs.berkeley.edu

Andrew Gearhart
UC Berkeley
agearh@cs.berkeley.edu

Benjamin Lipshitz*
UC Berkeley
lipshitz@cs.berkeley.edu

Oded Schwartz
UC Berkeley
odedsc@cs.berkeley.edu

Sivan Toledo
Tel-Aviv University
stoledo@tau.ac.il

ABSTRACT

Network topologies can have significant effect on the costs of algorithms due to inter-processor communication. Parallel algorithms that ignore network topology can suffer from contention along network links. However, for particular combinations of computations and network topologies, costly network contention may inevitably become a bottleneck, even for optimally designed algorithms. We obtain a novel contention lower bound that is a function of the network and the computation graph parameters. To this end, we compare the communication bandwidth needs of subsets of processors and the available network capacity (as opposed to per-processor analysis in most previous studies). Applying this analysis we improve communication cost lower bounds for several combinations of fundamental computations on common network topologies.

Categories and Subject Descriptors

F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*Computations on matrices*

General Terms

Algorithms, Design, Performance.

Keywords

Network topology, Communication-avoiding algorithms, Strong scaling, Communication costs.

*Current affiliation: Google Inc.

1. INTRODUCTION

Good connectivity of the inter processor network is necessary for fast execution of parallel algorithms. Insufficient graph-expansion of the network provably slows down specific parallel algorithms that are communication intensive. While parallel algorithms that ignore network topology can suffer from contention along network links, for particular combinations of computations and network topologies, costly network contention may be inevitable, even for optimally designed algorithms. In this paper we obtain novel lower bounds on such *contention cost*, and point to cases where this cost is a performance bottleneck.

We use a variant of the distributed-memory communication model (cf, [16, 19, 11]), where the bandwidth-cost of an algorithm is proportional to the number of words communicated by one processor (we omit the latency cost / message count discussion from this work). As in the distributed-memory communication model we have P processors and a local memory of size M for each processor. However, here, we do not assume all-to-all connectivity, but rather some network graph G_{Net} with P vertices. In this work we assume all edges (network links) have the same bandwidth, and the nodes of the network are both processors and routers (i.e. a direct network, where no node is solely a router). We ignore processor injection rates in this model.

Most previous communication cost lower bounds for parallel algorithms utilize *per-processor* analysis. That is, the lower bounds establish that some processor must communicate a given amount of data. These include classical matrix multiply, direct and iterative linear algebra algorithms, FFT, Strassen and Strassen-like fast algorithms, graph related algorithms, N-body, sorting, and others (cf. [3, 25, 23, 32, 26, 11, 9, 15, 22, 8, 28, 35, 21, 34]).

By considering the network graphs, we introduce communication lower bounds for certain computations and networks that are tighter than what was previously known. We bound from below the number of words communicated between a subset of processors and the rest of the processors for a given parallel algorithm (defined by a computation graph and work assignment to the processors), and divide it by the number of words that the network is capable of communicating simultaneously between that subset of processors and the rest of the graph. This relates to the *contention cost* of the algorithm, which we specify in Definition 2.2. Applying the main theorem we improve (i.e., increase) communication cost lower bounds for several combinations of fundamental

.

computations on common network topologies. Note that we inherit any assumptions made in the original per-processor lower bounds, e.g. no recomputation. These contention bounds may suggest directions for hardware/network design tailored for heavily used computation kernels and may assist when scheduling users' applications on (a subset of) a supercomputer.

2. CONTENTION LOWER BOUND

In this section we state our main result, which translates per-processor bandwidth cost lower bounds to contention cost lower bounds. The following definitions differentiate these costs.

DEFINITION 2.1. *Let a parallel algorithm be run on a parallel distributed-memory machine with P processors. The* per-processor bandwidth cost W_{proc} *is the maximum over processors $1 \leq p \leq P$ of the number of words sent or received by processor p.*

Observe that for W_{proc} we can plug in two types of per-processor lower bounds: memory-independent $W_{\text{proc}}(P, N)$ (cf. [9]) and memory-dependent $W_{\text{proc}}(P, M, N)$ (cf. [26, 11, 12, 10, 21]) where N is the input and output data size.

DEFINITION 2.2. *Let a parallel algorithm be run on a parallel distributed-memory machine with network graph $G_{Net} = (V, \hat{E})$ where V and $\hat{E}$ are the set of nodes and network links in G_{Net}, respectively. The* contention cost W_{link} *is the maximum over edges $e \in \hat{E}$ of the number of words communicated along e during the execution of the algorithm.*

In order to prove our result, we will use graph expansion analysis. Recall that the small set expansion $h_s(G)$ of a d-regular graph $G = (V, \hat{E})$ is the minimum normalized number of edges leaving a set of vertices of size at most s. Formally, for $s \leq |V|/2$, we have

$$h_s(G) = \min_{S \subseteq V, |S| \leq s} \frac{|E(S, V \setminus S)|}{|E(S)|}$$

where $E(S)$ is the set of edges that have at least one endpoint in vertex subset S and $E(S, \bar{S})$ is the set of edges with only one endpoint in S. The cardinality of a set S is represented by $|S|$. In the case of d-regular graphs, $|E(S)| \leq d|S|$.

THEOREM 2.3. *Consider a distributed-memory machine with P processors, each with local memory of size M, and an inter-processor network graph G_{Net}. Given a computation with input and output data size N, and lower bound on the memory-dependent per-processor bandwidth cost $W_{proc}(P, M, N)$, for all algorithms that distribute the workload so that every processor performs $\Omega(1/P)$ of the computation, and distributing the input and output data such that every processor stores $O(1/P)$ of the data, the memory-dependent contention cost $W_{link}(P, M, N)$ is bounded below by*

$$W_{link}(P, M, N) \geq \max_{t \in T} \frac{W_{proc}(P/t, M \cdot t, N)}{d \cdot t \cdot h_t(G_{Net})}$$

where

$$T = \{t : 1 \leq t \leq P/2, \exists S \subseteq V \text{ s.t. } |S| = t \text{ and } h_t(G) = |E(S, V \setminus S)|/|E(S)|\}.$$

PROOF. Consider a partitioning of the P processors into P/t subsets of size $t \in T$ (w.l.o.g., P is divisible by t), where at least one of the subsets s_t is connected to the rest of the network graph with at most $d \cdot t \cdot h_t(G_{Net})$ edges.[1] The existence of such a set s_t is guaranteed by the definition of $h_s(G_{Net})$ and T. Then s_t has a total of $M \cdot t$ local memory. By the workload distribution assumption, the processors in s_t perform a fraction $\Omega(t/P)$ of the flops, and by the data distribution assumption, s_t has local access to fraction $O(t/P)$ of the input/output. Hence we can emulate this computation by a parallel machine with P/t processors, each with $M \cdot t$ local memory (see Figure 1), and apply the corresponding per-processor lower bound deducing that the processors in s_t require at least $W_{\text{proc}}(P/t, M \cdot t, N)$ words to be sent/received to the processors outside s_t throughout the running of the algorithm. At most $d \cdot t \cdot h_t(G_{Net})$ edges connect s_t to the rest of the graph. Hence at least one edge communicates at least $\frac{W_{\text{proc}}(P/t, M \cdot t, N)}{d \cdot t \cdot h_t(G_{Net})}$ words. Since t is a free parameter, we can pick it to maximize $W_{\text{link}}(P, M, N)$, and the theorem follows. □

Note that the memory-independent contention lower bound, $W_{\text{link}} = W_{\text{link}}(P, N)$, follows.

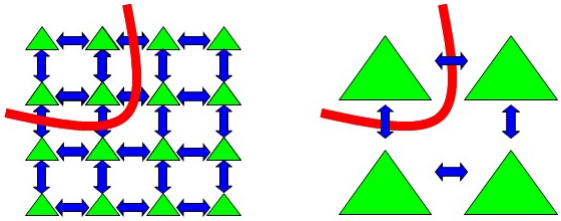

Figure 1: Computation of t = 4 processors on a 16-processor machine can be emulated as the computation of one processor on a 4-processor machine.

3. PRELIMINARIES

3.1 Per-Processor Lower Bounds

Before deriving bounds on link contention, we review the per-processor communication bounds for several classes of algorithms.

Classical Linear Algebra.

Most classical direct linear algebra computations can be specified by three nested loops, and for dense $n \times n$ matrices, the number of flops performed is $\Theta(n^3)$.[2] Informally, such computations, which include matrix multiplication, Cholesky and LU decompositions, and many others, can be defined by

$$C_{ij} = f_{ij}(\{g_{ijk}(A_{ik}, B_{kj})\}_{1 \leq k \leq n}) \text{ for } 1 \leq i, j \leq n \quad (1)$$

where f and g are sets of functions particular to the computation. For example, in the case of classical matrix multiplication, f_{ij} is a summation and g_{ijk} is a scalar multiplication for all i, j, k. For a more formal definition, see [7, Definition

[1] Note that s_t is connected to the rest of the network graph with exactly $d \cdot t \cdot h_t(G_{Net})$ edges only when $|E(S)| = d|S|$.

[2] For matrix computations, we denote the size of the input/output to be $N = \Theta(n^2)$.

4.1]. For such computations, we have the following lower bound:

THEOREM 3.1 ([11],[26]). *Consider an algorithm performing a computation of the form given by equation* (1) *on P processors, each with local memory of size M, and assume one copy of the input data is initially distributed across processors and the computation is load balanced. Then the number of words some processor must communicate is at least*

$$W_{proc}(P, M, N) = \Omega\left(\frac{n^3}{PM^{1/2}}\right).$$

Note that the local memory size M appears in the denominator of the expression above, which is why we refer to it as the memory-dependent bound. Additionally, such computations also inherit a memory-independent lower bound:

THEOREM 3.2 ([9]). *Consider an algorithm performing a computation of the form given by equation* (1) *on P processors, and assume just one copy of the input data is initially distributed across processors and the computation is load balanced. Then the number of words some processor must communicate is at least*

$$W_{proc}(P, N) = \Omega\left(\frac{n^2}{P^{2/3}}\right).$$

Strassen-like Matrix Multiplication.

Similar lower bounds exist for Strassen's matrix multiplication and similar algorithms, though the proof techniques differ substantially. Informally, we use the term "Strassen-like" to refer to algorithms that recursively multiply matrices according to a base-case computation. For square algorithms, this corresponds to multiplying $n_0 \times n_0$ matrices with m_0 scalar multiplications, where n_0 and m_0 are constants. Using recursion, this results in a square matrix multiplication flop count of $\Theta(n^{\omega_0})$ where $\omega_0 = \log_{n_0} m_0$. Note that additional technical assumptions are required for the communication lower bounds to apply and that Strassen-like algorithms may have a rectangular base case; see [12, Section 5.1] for more details. The memory-dependent communication lower bound for Strassen-like algorithms is:

THEOREM 3.3 ([12, COROLLARY 1.5]). *Consider a Strassen-like matrix multiplication algorithm that requires $\Theta(n^{\omega_0})$ total flops. Suppose a parallel algorithm performs the computation using P processors (each with local memory of size M), load balances the flops, and performs no redundant computation. Then the number of words some processor must communicate is at least*

$$W_{proc}(P, M, N) = \Omega\left(\frac{n^{\omega_0}}{PM^{\omega_0/2-1}}\right).$$

Additionally, such computations also inherit a memory-independent lower bound:

THEOREM 3.4. *Suppose a parallel algorithm performs a Strassen-like matrix multiplication algorithm requiring $\Theta(n^{\omega_0})$ flops, load balances the computation across P processors, and performs no redundant computation. Then under some technical assumptions (see [12]) the number of words some processor must communicate is at least*

$$W_{proc}(P, N) = \Omega\left(\frac{n^2}{P^{2/\omega_0}}\right).$$

PROOF. Identical to Theorem 2.1 in [9], with ω_0 replacing $log_2 7$. □

Programs Referencing Arrays.

The model defined in Equation (1) encompasses most direct linear algebra computations, but lower bounds can be obtained for a more general set of computations. In particular, Christ et al. [21] consider programs of the following form:

$$\begin{aligned} &\text{for all } \mathcal{I} \in \mathcal{Z} \subseteq \mathbb{Z}^d, \text{ in some order,} \\ &\quad \text{inner\_loop}(\mathcal{I}, (A_1, \ldots, A_m), (\phi_1, \ldots, \phi_m)) \end{aligned} \tag{2}$$

where $\mathbb{Z}^d$ is the d-dimensional space of integers and inner_loop() represents a computation involving arrays $A_1, ..., A_m$ of dimensions $d_1, ..., d_m$ that are referenced by the corresponding subscripts $\phi_1(\mathcal{I}), ..., \phi_m(\mathcal{I})$ where ϕ_i are affine maps $\phi_j : \mathbb{Z}^d \to \mathbb{Z}^{d_j}$ for iteration $\mathcal{I} = (i_1, ..., i_d)$. For example, matrix-matrix multiplication has $(A_1, A_2, A_3) = (A, B, C)$, $\phi_1(\mathcal{I}) = \phi_1(i_1, i_2, i_3) = (i_1, i_3)$, $\phi_2(\mathcal{I}) = \phi_2(i_1, i_2, i_3) = (i_3, i_2)$, $\phi_3(\mathcal{I}) = \phi_3(i_1, i_2, i_3) = (i_1, i_2)$ and the function inner_loop() is defined as $A_3(\phi_3(\mathcal{I})) = A_3(\phi_3(\mathcal{I})) + A_1(\phi_1(\mathcal{I})) * A_2(\phi_2(\mathcal{I}))$.

Because the work inside the loop is currently defined as a general function, the space of potential executions of inner_loop() must be restricted in a manageable manner, or to "legal parallel executions" as defined in [21]. To express the lower bounds, we define a set of linear constraints on a vector of unknown scalars $(s_1, ..., s_m)$

$$\text{rank}(H) \leq \sum_{j=1}^{m} s_j \text{rank}(\phi_j(H)), \tag{3}$$

for all subgroups H of $\mathbb{Z}^d$, where rank(H) is the cardinality of any maximal subset of Abelian group H that is linearly independent.[3] For such computations we have the following lower bound:

THEOREM 3.5 ([21]). *Consider an algorithm performing a computation of the form given by equation* (2) *on P processors, each with local memory of size M, and assume the input data is initially evenly distributed across processors. Then for any legal parallel execution and sufficiently large $|\mathcal{Z}|/P$, the number of words some processor must communicate is at least*

$$W_{proc}(P, M, N) = \Omega\left(\frac{|\mathcal{Z}|}{PM^{s_{HBL}-1}}\right),$$

where s_{HBL} is the minimum value of $\sum_{i=1}^{m} s_i$ subject to (3)*, assuming that this linear program is feasible (see [21]).*

We restate the memory-independent bound from [21] for such computations (note that the formal proof has not yet appeared). For legal parallel executions of computations of the form (2) on P processors, some processor must move

$$W_{\text{proc}}(P, N) = \Omega\left(\left(\frac{|\mathcal{Z}|}{P}\right)^{1/s_{\text{HBL}}} - \frac{N}{P}\right) \tag{4}$$

words where N is the sum of the sizes of arrays $\{A_i\}$ (assumed to be evenly distributed across processors) and s_{HBL} is defined as in Theorem 3.5. In most cases, the negative

[3]The rank of an Abelian group is analogous to the concept of the dimension of a vector space.

term in the expression is asymptotically dominated and can be ignored.

Note that Theorem 3.5 generalizes Theorem 3.1. For example, matrix multiplication satisfies both forms (1) and (2), where in the latter case $|\mathcal{Z}| = n^3$ and $s_{\text{HBL}} = 3/2$.

Theorem 3.5 also applies to, for example, N-body computations where all pairs of interactions are computed. In the this case, $|\mathcal{Z}| = \Theta(N^2)$ and $s_{\text{HBL}} = 2$, yielding lower bounds of $W_{\text{proc}}(P, M, N) = \Omega(N^2/(PM))$ and $W_{\text{proc}}(P, N) = \Omega(N/P^{1/2})$. We also note that Theorem 3.5 applies to N-body computations that use a distance cutoff to reduce the number of neighbor interations, i.e. $|\mathcal{Z}| \ll N^2$.

FFT/Sorting.

We are unaware of any memory-dependent lower bound per-processor bound for the FFT, although a sequential lower bound was proven by Hong and Kung [25]. A parallel memory-independent per-processor bound has been proven in the LPRAM [4] and the BSP models of computation [15]. The LPRAM model lower bound implies asymptotically the same lower bound for our distributed parallel model:

THEOREM 3.6 ([4]). *Given an algorithm that computes an n-input FFT digraph a LPRAM model of computation with P processors, and no recomputation is allowed, then the I/O complexity of the algorithm is*

$$W_{proc}(P, N) = \Omega\left(\frac{nlog(n)}{Plog(n/P)}\right).$$

3.2 Small Set Expansion of Various Networks

We next demonstrate our bounds on several classes of algorithms on a particular pair of networks: D-dimensional tori and meshes.

Toroidal networks are common topologies amongst supercomputers, with IBM's Blue Gene/L [2] and Blue Gene/P [1] machines possessing 3D tori. In Blue Gene/Q, IBM used a 5-dimensional torus [20] and the K computer in Japan utilizes a 6-dimensional network topology [6]. Intel Xeon Phi coprocessors rely on a ring-based (a 1-dimensional torus) on-chip communication network between cores [27]. In this paragraph, we derive a tight bound on the network small set expansion for this class of networks.

The D-dimensional torus or mesh graph G_{Net} has degree at most $d = O(D)$ and the small set expansion shown below. We treat D here as a constant. For a fixed dimension D the bounds are tight, up to a constant factor. For a tighter analysis of these graphs, see [17].

LEMMA 3.7. *Let G be a D-dimensional torus or mesh, with k^D vertices. Then asymptotically in s,*

$$h_s(G) = \Theta\left(s^{-1/D}\right).$$

PROOF. For an upper bound on $h_s(G)$ consider a subset $S \in V(G)$ which is a D-dimensional submesh of length $s^{1/D}$ in each dimension. The number of neighbors of this submesh on each of its $2D$ faces is $O(s^{\frac{D-1}{D}})$. Thus $|E(S, V \setminus S)| = 2D \cdot O(s^{\frac{D-1}{D}})$. The number of vertices of S is s. The degree of each vertex is $O(D)$. Hence $h_s(G) \leq 2D \cdot O(s^{(D-1)/D})/(O(D)s) = O(s^{-1/D})$.

For a lower bound on $h_s(G)$ we use the Loomis-Whitney inequality [30]. Consider a set $S \subseteq V(G)$ of size $s \leq V(G)/2$. Let $A_1, A_2, ..., A_D$ be the projections of S onto the $(D-1)$-dimensional coordinate hyperplanes; let $a_1, ..., a_D$ be their corresponding sizes. Then by the Loomis-Whitney inequality we have $s^{D-1} \leq \prod_{1 \leq i \leq D} a_i$. Letting $m = \text{argmax}_i\{a_i\}$, we have $s^{1-1/D} \leq a_m$. Consider the "pencil" of vertices that corresponds to a point in A_m: if there exists a vertex in the pencil that is not in S, then the pencil contributes at least one edge to the cut $E(S, V \setminus S)$. We say such a pencil is *partially full*. We later show that there are at least $(1 - 1/2^{1/D})a_m$ partially-full pencils. Thus they contribute a total of at least $(1 - 1/2^{1/D})a_m \geq (1 - 1/2^{1/D})s^{1-1/D}$ edges to the cut. Hence $h_s(G) \geq (1 - 1/2^{1/D})/(2D \cdot s^{1/D}) = \Omega(s^{-1/D})$. To see that the number of partially-full pencils is indeed at least $(1 - 1/2^{1/D})a_m$, assume for the sake of contradiction that more than $a_m/2^{1/D}$ pencils are full (i.e. have all their vertices in S). This implies that $s > ka_m/2^{1/D} \geq ks^{1-1/D}/2^{1/D}$, thus $s > k^D/2 = |V|/2$, which is a contradiction since $s \leq |V|/2$. □

4. APPLICATIONS

4.1 Deriving the Contention Lower Bounds

In this section, we derive contention lower bounds by plugging the memory-dependent and memory-independent per-processor lower bounds [26, 12, 9, 21] into Theorem 2.3 and using the properties of D-dimensional tori. Table 1 summarizes these results. In the algebra that follows, we assume the network topology to be a D-dimensional torus or mesh.

Direct Linear Algebra, Strassen, Strassen-like, $O(n^2)$ n-body algorithms.

We apply Theorem 2.3 to the relevant per-processor bounds given in Section 3.1. Let F denote the number of work operations (e.g. flops or loop iterations) of the different computations. The per-processor memory-dependent bound is thus:

$$W_{\text{proc}}(P, M, N) = \Omega\left(\frac{F}{PM^{\alpha-1}}\right) \tag{5}$$

where $\alpha = 3/2$ for direct dense linear algebra, $\alpha = \omega_0/2$ for Strassen-like matrix multiplication, $\alpha = 2$ for the $O(n^2)$ n-body problem. We next apply Theorem 2.3 to (5). By Lemma 3.7, for a D-dimensional torus, the denominators of the contention bounds in Theorem 2.3 and Expression (**??**) are $2D \cdot t \cdot \Theta(t^{-1/D})$. Thus, the memory-dependent contention bound is:

$$W_{\text{link}}(P, M, N) = \max_{t \in T} \Omega\left(\frac{F}{PM^{\alpha-1}} \cdot t^{1-\alpha+1/D}\right) \tag{6}$$

Note that $t^{1-\alpha+1/D}$ is monotonic (in the given range), but that the exponent can be positive, negative or zero. If the exponent of t is negative or zero, then the expression is maximized at $t = 1$, reproducing the per-processor bound (up to a constant factor). If the exponent is positive, namely $D \leq D_1 = 1/(\alpha - 1)$, then the expression is maximized at

$t = P/2$, and we obtain a new and tighter bound [4]:

$$W_{\text{link}}(P, M, N) = \Omega\left(\frac{F}{P^{\alpha-1/D}M^{\alpha-1}}\right). \quad (7)$$

The per-processor memory-independent bound is

$$W_{\text{proc}}(P, N) = \Omega\left(\frac{N}{P^{1/\alpha}}\right) \quad (8)$$

We next apply Theorem 2.3 to (8) and obtain:

$$W_{\text{link}}(P, N) = \max_{t\in T} \Omega\left(\frac{N}{P^{1/\alpha}} \cdot t^{1/\alpha-1+1/D}\right) \quad (9)$$

Again, $t^{1/\alpha-1+1/D}$ is monotonic and may be positive, negative or zero. If the exponent of t is negative or zero, then the expression is maximized at $t = 1$, reproducing the per-processor bound (up to a constant factor). If the exponent is positive, namely $D \leq D_2 = \alpha/(\alpha-1)$, then the expression is maximized at $t = P/2$, and we obtain a new and tighter bound:

$$W_{\text{link}}(P, N) = \Omega\left(\frac{N}{P^{1-1/D}}\right). \quad (10)$$

Table 1 presents the communication lower bounds for each of the computations described in Sections 3.1 on D-dimensional tori with the respective values of F and α.

Programs that Reference Arrays.

Note that if we assume that $F = O(N^\alpha)$ in the memory-independent lower bound for programs that reference arrays with $\alpha = s_{\text{HBL}}$, we arrive at the form of this bound used for the derivation of the direct linear algebra, Strassen, Strassen-like and $O(n^2)$ n-body contention bounds. In general, this does not have to be the case for the set of programs defined by Expression 2 above.

According to Theorem 3.5, the memory-dependent per-processor bandwidth lower bound for programs defined by Expression 2 is

$$W_{\text{proc}}(P, M, N) = \Omega\left(\frac{|\mathcal{Z}|}{PM^{s_{\text{HBL}}-1}}\right).$$

Similar to the derivation for the previous problems (albeit with $\alpha = s_{\text{HBL}}$), the bound becomes

$$W_{\text{link}}(P, M, N) = \max_{1\leq t\leq P/2} \Omega\left(\frac{|\mathcal{Z}|}{PM^{s_{\text{HBL}}-1}} \cdot t^{1-s_{\text{HBL}}+1/D}\right)$$

which is maximized at either $t = 1$ (the per-processor bound), or $t = P/2$ (see Footnote 4). So, we obtain

$$W_{\text{link}}(P, M, N) = \Omega\left(\frac{|\mathcal{Z}|}{P^{s_{\text{HBL}}-1/D}M^{s_{\text{HBL}}-1}}\right)$$

as a memory-dependent lower bound on contention. In a similar manner, we can derive a memory-independent contention lower bound. From Equation (4), the memory-independent per-processor bound is

$$W_{\text{proc}}(P, N) = \Omega\left(\left(\frac{|\mathcal{Z}|}{P}\right)^{1/s_{\text{HBL}}}\right)$$

assuming we drop the N/P term from the bound. At $t = P/2$ (as again we observe that the contention bound is maximized at either $t = 1$ or $t = P/2$), we derive the memory-independent lower bound on contention

$$W_{\text{link}}(P, N) = \Omega\left(\frac{|\mathcal{Z}|^{1/s_{\text{HBL}}}}{P^{1-1/D}}\right).$$

FFT/Sorting.

As with the previous algorithms, we apply Theorem 2.3 to the relevant per-processor bound given in Section 3.1.

The per-processor memory-independent bound is thus

$$W_{\text{proc}}(P, N) = \Omega\left(\frac{nlog(n)}{Plog(n/P)}\right). \quad (11)$$

We next apply this bound to Theorem 2.3 and obtain:

$$W_{\text{link}}(P, N) = \max_{1\leq t\leq P/2} \Omega\left(\frac{nlog(n)}{Plog(nt/P)t^{-1/D}}\right) \quad (12)$$

$$= \frac{nlog(n)}{P} \max_{1\leq t\leq P/2} \Omega\left(\frac{t^{1/D}}{log(nt/P)}\right).$$

Again, when $t = 1$ we obtain the original per-processor bound. Equation 12 has a stationary point at $t = PC^D/n$ (where C is the base of the logarithm), but via consideration of the second derivative wrt to t, it can be shown that this point is a minima for all relevant values of n,P and D. Thus, we can derive a memory-independent contention bound by setting $t = P/2$ (see Footnote 4):

$$W_{\text{link}}(P, N) = \Omega\left(\frac{N}{P^{1-1/D}}\right) \quad (13)$$

as $N = O(n)$.

4.2 Analysis and Interpretation

Which bound dominates?.

Our first observation is that, for these computations, the memory-independent contention bound dominates the memory-dependent contention bound for many algorithms. In the cases of direct linear algebra, Strassen and Strassen-like, and the $O(n^2)$ n-body problem we prove this by contradiction: if the memory-dependent contention bound dominates, then the problem is too large to be distributed across all the processors' local memories. Thus, if

$$\frac{F}{P^{\alpha-1/D}M^{\alpha-1}} > \frac{N}{P^{1-1/D}}$$

then, as $F = \theta(N^\alpha)$, we have

$$N^{\alpha-1} > P^{\alpha-1}M^{\alpha-1}$$

which is a contradiction as we assumed that $N \leq PM$. For programs that reference arrays, the proof requires a bit more of the theoretical apparatus from [21] and is proven in Appendix B. We note that in practice the value of constants

[4] Note that there may not be a subset of the vertices of G_{Net} that attains the small set expansion $h_t(G_{Net})$ of size *exactly* $P/2$. However, the small set expansion of tori and meshes is attained for small sets of size P/c for some constant $c \geq 2$ (e.g. consider a sub-tori), hence the following contention analysis holds up to a constant factor.

		Memory Dependent	Memory Indepedent
Direct Linear Algebra	W_{proc}	$\Omega\left(\frac{n^3}{PM^{1/2}}\right)$	$\Omega\left(\frac{n^2}{P^{2/3}}\right)$
	W_{link}	$\Omega\left(\frac{n^3}{P^{3/2-1/D}M^{1/2}}\right)$	$\Omega\left(\frac{n^2}{P^{1-1/D}}\right)$
Strassen and Strassen -like	W_{proc}	$\Omega\left(\frac{n^{\omega_0}}{PM^{\omega_0/2-1}}\right)$	$\Omega\left(\frac{n^2}{P^{2/\omega_0}}\right)$
	W_{link}	$\Omega\left(\frac{n^{\omega_0}}{P^{\omega_0/2-1/D}M^{\omega_0/2-1}}\right)$	$\Omega\left(\frac{n^2}{P^{1-1/D}}\right)$
$O(n^2)$ n-body	W_{proc}	$\Omega\left(\frac{n^2}{PM}\right)$	$\Omega\left(\frac{N}{P^{1/2}}\right)$
	W_{link}	$\Omega\left(\frac{n^2}{P^{2-1/D}M}\right)$	$\Omega\left(\frac{N}{P^{1-1/D}}\right)$
FFT/Sorting	W_{proc}	?	$\Omega\left(\frac{nlog(n)}{Plog(n/P)}\right)$
	W_{link}	?	$\Omega\left(\frac{N}{P^{1-1/D}}\right)$
Programs Referencing Arrays	W_{proc}	$\Omega\left(\frac{\lvert\mathcal{Z}\rvert}{PM^{s_{\text{HBL}}-1}}\right)$	$\Omega\left(\left(\frac{\lvert\mathcal{Z}\rvert}{P}\right)^{1/s_{\text{HBL}}}\right)$
	W_{link}	$\Omega\left(\frac{\lvert\mathcal{Z}\rvert}{P^{s_{\text{HBL}}-1/D}M^{s_{\text{HBL}}-1}}\right)$	$\Omega\left(\frac{\lvert\mathcal{Z}\rvert^{1/s_{\text{HBL}}}}{P^{1-1/D}}\right)$

Table 1: Per-processor bounds (W_{proc}) ([26, 11, 9, 12, 15]) vs. the new contention bounds (W_{link}) on a D-dimensional torus for classical linear algebra, fast matrix multiplication, $O(n^2)$ n-body, Fast Fourier Transform (FFT) and a general set of programs that reference arrays.

may result in the memory-dependent contention bound being dominant, despite the asymptotic result.

For direct linear algebra, Strassen, Strassen-like and $O(n^2)$ n-body algorithms, Figure 2 illustrates the relationships between the four types of bounds for a fixed computation, fixed problem size N, and fixed local memory size M, varying the number of processors P and the torus dimension D. See Appendix A for the derivation of the expressions used in Figure 2.

Depending on the dimension of the torus and number of processors, the tightest bound may be one of the previously known per-processor bounds or the memory-independent contention bound. We first consider subdividing the vertical axis of Figure 2, which corresponds to the torus dimension D. Intuitively speaking, the smaller D is, the more likely contention will dominate communication costs. For a given algorithm, we let $\mathrm{D} = \lfloor 1/(\alpha-1) \rfloor = \lfloor D_1 \rfloor$ is the maximum torus dimension such that the communication cost is dominated by contention for all input and machine parameters. Similarly, we let $D = \lceil \alpha/(\alpha-1) \rceil = \lceil D_2 \rceil$ be the minimum torus dimension so that the communication cost is not dominated by the contention (at least not by the bound proved here). Note that for a combination of an algorithm and a D-dimensional torus such that $D_1 < D < D_2$, either the per-processor memory-dependent bound or the memory-independent contention bound may dominate. See Table 2 for values of D_1 and D_2 for various matrix multiplication algorithms. In particular, note that for the classical algorithm, a 2D torus is not sufficient to avoid contention. While Cannon's algorithm [18] does not suffer from contention on a 2D torus network, it is also not communication-optimal. The more communication-efficient "3D" algorithms [14, 4, 31, 36], which utilize extra memory and have the ability to strong scale perfectly, require a 3D torus to attain the per-processor lower bounds. For matrix multiplication algorithms with smaller exponents, the torus dimension requirements for remaining contention-free are even larger.

Range of perfect strong scaling.

We next consider subdividing the horizontal axis of Figure 2, which corresponds to the number of processors P. Because Figure 2 shows a fixed problem size, increasing P (moving to the right) corresponds to "strong scaling." We differentiate between whether or not the computation has the possibility of strong scaling perfectly: that is, for a fixed problem size, increasing the number of processors by a constant factor reduces the communication costs (and running time) by the same constant factor. Note that of the bounds, the memory-dependent per-processor bound (Equation (5)) exhibits this possibility of perfect strong scaling, as P appears in the denominator with an exponent of 1. However, as P increases, one of the memory-independent bounds eventually dominates and perfect strong scaling is no longer possible. See [9] for a discussion of this behavior given only per-processor bounds.

For direct linear algebra, Strassen-like methods and the $O(n^2)$ n-body problem, when $D \geq D_2$ and $P \leq (F/NM^{\alpha-1})^{\frac{\alpha}{\alpha-1}}$, then the memory-dependent per-processor bound dominates. When this happens, we have a perfect strong scaling range. For values of P beyond this range, the communication cost is dominated by the memory-independent per-processor bound (see [9] for further discussion). When $D_1 < D < D_2$, a smaller strong-scaling ranges exists for $P \leq (F/NM^{\alpha-1})^D$; for values of P beyond this range, the communication cost bound is dominated by contention. If $D \leq D_1$, then the contention bounds always dominate and there is no strong-scaling range. A similar analysis can demonstrate such a region of perfect strong scaling in runtime for programs that reference arrays.

Figure 3 shows this behavior for Strassen's matrix multi-

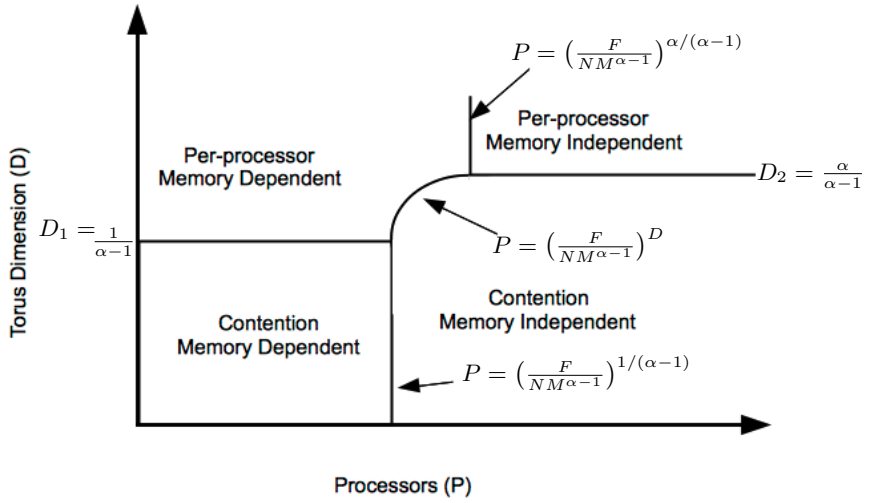

Figure 2: Relationship between the per-processor and contention communication lower bounds for direct linear algebra, Strassen/Strassen-like and the $O(n^2)$ n-body problems.

Algorithm	ω_0	$\lfloor D_1 \rfloor$	$\lceil D_2 \rceil$
Classical	3	2	3
Strassen [38]	≈ 2.81	2	4
Schönhage [33]	≈ 2.55	3	5
Strassen [39]	≈ 2.48	4	6
Vassilevska [40]	≈ 2.3727	5	7

Table 2: Torus dimensions so that communication cost is either always contention bound ($D \leq \lfloor D_1 \rfloor$) or never contention bound ($D \geq \lceil D_2 \rceil$) for a selection of matrix multiplication algorithms. The assertions regarding the last three algorithms are under some technical assumptions / conjecture, see [12].

plication (where $\alpha = (\log_2 7)/2$) given the relevant torus dimensions. For Strassen, $F/NM^{\alpha-1} = (N/M)^{\alpha-1} = P_{min}^{\alpha-1}$, where P_{min} is the minimum number of processors required to store the problem as $F = O(n^\alpha)$. Note that the lower subfigure in Figure 3 is a log-log scale, while the upper subfigure's y-axis is linear. For a good enough network ($D \geq 4$), the perfect strong scaling range is $P_{\min} < P < P_{\min}^{(\log_2 7)/2} \approx P_{\min}^{1.40}$. For a 3D torus, the perfect strong scaling range shrinks to $P_{\min} < P < P_{\min}^{3(\log_2 7-2)/2} \approx P_{\min}^{1.21}$. On 2D torus, perfect strong scaling is impossible. These three regions of network dimension ($D \geq D_2$, $D \leq D_1$ and $D_1 < D < D_2$) are illustrated in Figure 2 as being the points of transition between dominance of the various bounds. The upper portion of Figure 3 demonstrates the regions of dominance for the various network dimensions in the case of Strassen's algorithm.

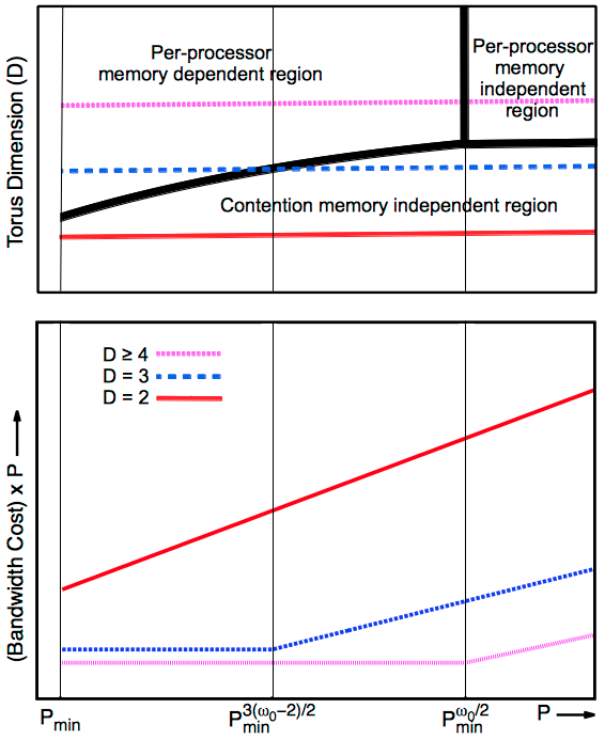

Figure 3: Communication bounds for Strassen's algorithm on D-dimensional tori. The lower plot is log-log, while the upper is linear on the y-axis. Horizontal lines in the lower plot correspond to perfect strong scaling.

5. FUTURE RESEARCH

Other Networks.

In this work, we exclusively address link contention bounds for tori and mesh networks. We suspect that results for hypercubes and certain indirect networks (e.g. fat trees) should follow easily. For indirect networks, a method for integrating router nodes into the model of computation is needs to be defined. Indirect topologies are common in datacenters as well as on-chip networks, so such an extension of the contention bounds for direct networks would be useful.

Applicability.

A network may have expansion sufficiently large to preclude the use of our contention bound on a given computation, yet the contention may still dominate the communication cost. This calls for further study on how well computations and networks match each other. Similar questions have been addressed by Leiserson and others [13, 24, 29], and had a large impact on the design of supercomputer networks. In particular, a parallel computer that uses a fat tree communication network can simulate any other routing network, at the cost of at most polylogarithmic slowdown.

Communication Efficient Algorithms.

Some parallel algorithms are network aware, and attain the per-processor communication lower bounds, when network graphs allow it (cf. [36] for classical matrix multiplication on 3D torus). Many algorithms are communication optimal when all-to-all connectivity is assumed, but their performance on other topologies has not yet been studied. Are there algorithms that attain the communication lower bounds for any realistic network graph (either by auto tuning, or by network-topology-oblivious tools)?

Acknowledgments

We thank Guy Kindler for pointing us to [17]. Research partially funded by DARPA Award Number HR0011-12-2-0016, the Center for Future Architecture Research, a member of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA, and ASPIRE Lab industrial sponsors and affiliates Intel, Google, Nokia, NVIDIA, Oracle, MathWorks and Samsung. Research is also supported by U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program DE-SC0004938, DE-SC0005136, DE-SC0003959,DE-SC0008700, DE-SC0008699, DE-SC0010200 and DOE AC02-05CH11231. Research is supported by grants 1878/14 and 1045/09 from the Israel Science Foundation (founded by the Israel Academy of Sciences and Humanities), and grant 2010231 from the US-Israel Bi-National Science Foundation. This research is supported by grant 3-10891 from the Ministry of Science and Technology, Israel. This research was supported in part by an appointment to the Sandia National Laboratories Truman Fellowship in National Security Science and Engineering, sponsored by Sandia Corporation (a wholly owned subsidiary of Lockheed Martin Corporation) as Operator of Sandia National Laboratories under its U.S. Department of Energy Contract No. DE-AC04-94AL85000. Any opinions, findings, conclusions, or recommendations in this paper are solely those of the authors and does not necessarily reflect the position or the policy of the sponsors.

6. REFERENCES

[1] Overview of the IBM Blue Gene/P project. *IBM Journal of Research and Development*, 52(1.2):199–220, Jan 2008.

[2] N. R. Adiga, M. Blumrich, D. Chen, P. Coteus, A. Gara, M. Giampapa, P. Heidelberger, S. Singh, B. Steinmacher-Burow, T. Takken, M. Tsao, and P. Vranas. Blue Gene/L Torus Interconnection Network. *IBM Journal of Research and Development*, 49(2.3):265–276, March 2005.

[3] A. Aggarwal, A. K. Chandra, and M. Snir. Communication complexity of PRAMs. *Theor. Comput. Sci.*, 71:3–28, March 1990.

[4] A. Aggarwal, A. K. Chandra, and M. Snir. Communication complexity of PRAMs. *Theoretical Computer Science*, 71(1):3–28, 1990.

[5] A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. *Commun. ACM*, 31(9):1116–1127, 1988.

[6] Y. Ajima, S. Sumimoto, and T. Shimizu. Tofu: A 6d mesh/torus interconnect for exascale computers. *Computer*, 42(11):36–40, Nov 2009.

[7] G. Ballard. *Avoiding Communication in Dense Linear Algebra*. PhD thesis, EECS Department, University of California, Berkeley, Aug 2013.

[8] G. Ballard, A. Buluç, J. Demmel, L. Grigori, B. Lipshitz, O. Schwartz, and S. Toledo. Communication optimal parallel multiplication of sparse random matrices. In *SPAA'13: Proceedings of the 25rd ACM Symposium on Parallelism in Algorithms and Architectures*, 2013.

[9] 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. In *Proceedings of the 24th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '12, pages 77–79, New York, NY, USA, 2012. ACM.

[10] G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, and O. Schwartz. Graph expansion analysis for communication costs of fast rectangular matrix multiplication. In G. Even and D. Rawitz, editors, *Design and Analysis of Algorithms*, volume 7659 of *Lecture Notes in Computer Science*, pages 13–36. Springer Berlin Heidelberg, 2012.

[11] 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.

[12] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Graph expansion and communication costs of fast matrix multiplication. *Journal of the ACM*, 59(6):32:1–32:23, Dec. 2012.

[13] P. Bay and G. Bilardi. Deterministic on-line routing on area-universal networks. In *Proceedings of the 31st Annual Symposium on the Foundations of Computer Science (FOCS)*, pages 297–306, 1990.

[14] J. Berntsen. Communication efficient matrix multiplication on hypercubes. *Parallel Computing*, 12(3):335 – 342, 1989.

[15] G. Bilardi, M. Scquizzato, and F. Silvestri. A lower bound technique for communication on bsp with

application to the fft. In *Euro-Par 2012 Parallel Processing*, pages 676–687. Springer, 2012.

[16] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, 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/.

[17] B. Bollobás and I. Leader. Edge-isoperimetric inequalities in the grid. *Combinatorica*, 11(4):299–314, 1991.

[18] L. Cannon. *A cellular computer to implement the Kalman filter algorithm*. PhD thesis, Montana State University, Bozeman, MN, 1969.

[19] E. Chan, M. Heimlich, A. Purkayastha, and R. Van De Geijn. Collective communication: theory, practice, and experience. *Concurrency and Computation: Practice and Experience*, 19(13):1749–1783, 2007.

[20] D. Chen, N. Eisley, P. Heidelberger, R. Senger, Y. Sugawara, S. Kumar, V. Salapura, D. Satterfield, B. Steinmacher-Burow, and J. Parker. The IBM Blue Gene/Q Interconnection Fabric. *Micro, IEEE*, 32(1):32–43, Jan 2012.

[21] M. Christ, J. Demmel, N. Knight, T. Scanlon, and K. Yelick. 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, 2013.

[22] M. Driscoll, E. Georganas, P. Koanantakool, E. Solomonik, and K. Yelick. A communication-optimal n-body algorithm for direct interactions. In *proceedings of the IPDPS*, 2013.

[23] M. T. Goodrich. Communication-efficient parallel sorting. *SIAM J. Computing*, 29(2):416–432, 1999.

[24] R. I. Greenberg and C. E. Leiserson. Randomized routing on fat-tress. In *Proceedings of the 26th Annual Symposium on the Foundations of Computer Science (FOCS)*, pages 241–249, 1985.

[25] J. W. Hong and H. T. Kung. I/O complexity: The red-blue pebble game. In *Proc. 14th STOC*, pages 326–333, New York, NY, USA, 1981. ACM.

[26] D. Irony, S. Toledo, and A. Tiskin. Communication lower bounds for distributed-memory matrix multiplication. *J. Parallel Distrib. Comput.*, 64(9):1017–1026, 2004.

[27] J. Jeffers, J. Jeffers, and J. Reinders. *Intel Xeon Phi Coprocessor High Performance Programming*. Elsevier Science & Technology Books, 2013.

[28] N. Knight, E. Carson, and J. Demmel. Exploiting data sparsity in parallel matrix powers computations. In *Proceedings of PPAM '13*, Lecture Notes in Computer Science. Springer (to appear), 2013.

[29] C. E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, C-34(10):892–901, 1985.

[30] L. H. Loomis and H. Whitney. An inequality related to the isoperimetric inequality. *Bulletin of the AMS*, 55:961–962, 1949.

[31] W. McColl and A. Tiskin. Memory-efficient matrix multiplication in the BSP model. *Algorithmica*, 24(3-4):287–297, 1999.

[32] J. P. Michael, M. Penner, and V. K. Prasanna. Optimizing graph algorithms for improved cache performance. In *Proc. Int'l Parallel and Distributed Processing Symp. (IPDPS 2002), Fort Lauderdale, FL*, pages 769–782, 2002.

[33] A. Schönhage. Partial and total matrix multiplication. *SIAM J. Computing*, 10(3):434–455, 1981.

[34] M. Scquizzato and F. Silvestri. Communication lower bounds for distributed-memory computations. *arXiv preprint arXiv:1307.1805*, 2014. STACS'14.

[35] E. Solomonik, E. Carson, N. Knight, and J. Demmel. Tradeoffs between synchronization, communication, and work in parallel linear algebra computations. Technical Report (Submitted to SPAA'14), University of California, Berkeley, Department of Electrical Engineering and Computer Science, 2013.

[36] E. Solomonik and J. Demmel. Communication-optimal parallel 2.5d matrix multiplication and lu factorization algorithms. In E. Jeannot, R. Namyst, and J. Roman, editors, *Euro-Par 2011 Parallel Processing*, volume 6853 of *Lecture Notes in Computer Science*, pages 90–109. Springer Berlin Heidelberg, 2011.

[37] E. Solomonik and J. Demmel. Communication-optimal parallel 2.5D matrix multiplication and LU factorization algorithms. In *Euro-Par'11: Proceedings of the 17th International European Conference on Parallel and Distributed Computing*. Springer, 2011.

[38] V. Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13:354–356, 1969.

[39] V. Strassen. Relative bilinear complexity and matrix multiplication. *Journal fűr die reine und angewandte Mathematik (Crelles Journal)*, 1987(375–376):406–443, 1987.

[40] V. V. Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing*, Proc. 45th STOC, pages 887–898, New York, NY, USA, 2012. ACM.

APPENDIX

A. DERIVATION OF FIGURE EXPRESSIONS

- **Equivalence point for per-processor bounds**
  We set the per-processor bounds equal to each other, and solve for P:

$$\frac{F}{PM^{\alpha-1}} = \Theta\left(\frac{N}{P^{1/\alpha}}\right)$$

$$P = \Theta\left(\frac{F}{NM^{\alpha-1}}\right)^{\alpha/(\alpha-1)}$$

- **Equivalence point for contention bounds**
  We set the contention bounds equal to each other, and solve for P:

$$\frac{F}{P^{\alpha-1/D}M^{\alpha-1}} = \Theta\left(\frac{N}{P^{1-1/D}}\right)$$

$$P = \Theta\left(\frac{F}{NM^{\alpha-1}}\right)^{1/(\alpha-1)}$$

- **Equivalence point for the memory-dependent per-processor and memory-independent contention bounds**

  We set the memory-dependent per-processor and memory-independent contention bounds equal to each other, and solve for P as a function of D:

$$\frac{F}{PM^{\alpha-1}} = \Theta\left(\frac{N}{P^{1-1/D}}\right)$$

$$P = \Theta\left(\frac{F}{NM^{\alpha-1}}\right)^{D}$$

B. DOMINANCE OF MEMORY-INDEPENDENT CONTENTION BOUND

CLAIM B.1. *Let Alg be an algorithm performing a computation of the form given by equation* (2) *on P processors, each with local memory of size M, and assume the input data is initially evenly distributed across processors. Then,*

$$\frac{|\mathcal{Z}|^{1/s_{\text{HBL}}}}{M} \leq \sum_{j=1}^{m} \frac{|\phi_j(\mathcal{Z})|}{M}.$$

As the minimum number of processors required to hold the problem is the right-hand side of this inequality, we conclude that the memory-independent contention bound dominates the memory-dependent contention bound as the two bounds are equivalent when $P = |\mathcal{Z}|^{1/s_{\text{HBL}}}/M$.

PROOF. To begin a proof, the HBL bound discussed in Christ et al. [21], states (with certain assumptions) that

$$|\mathcal{Z}| \leq \prod_{j=1}^{m} |\phi_j(\mathcal{Z})|^{s_j}.$$

To detail an argument from Section 2 of [21], we present several greater upper bounds on $|\mathcal{Z}|$ that will allow us to demonstrate the desired result:

$$|\mathcal{Z}| \leq \prod_{j=1}^{m} |\phi_j(\mathcal{Z})|^{s_j} \leq \prod_{j=1}^{m} \left(\max_{j=1}^{m} |\phi_j(\mathcal{Z})|\right)^{s_j}$$

$$= \left(\max_{j=1}^{m} |\phi_j(\mathcal{Z})|\right)^{\sum_{j=1}^{m} s_j} = \left(\max_{j=1}^{m} |\phi_j(\mathcal{Z})|\right)^{s_{\text{HBL}}}$$

As $\max_{j=1}^{m} x_j \leq \sum_{j=1}^{m} x_j$ if all $x_j \geq 0$,

$$|\mathcal{Z}| \leq \left(\max_{j=1}^{m} |\phi_j(\mathcal{Z})|\right)^{s_{\text{HBL}}} \leq \left(\sum_{j=1}^{m} |\phi_j(\mathcal{Z})|\right)^{s_{\text{HBL}}}$$

which proves the desired inequality if we take s_{HBL}th root of both sides and divide by M. □