SAND2015-0991C

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
The Hebrew University
odedsc@cs.huji.ac.il

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

ABSTRACT

Network topologies can have significant effect on the costs of parallel algorithms due to inter-processor communication. For particular combinations of computations and network topologies, costly network contention may inevitably become a bottleneck, even if algorithms are optimally designed so that each processor communicates as little as possible. We obtain novel contention lower bounds that are functions of the network and the computation graph parameters. For several combinations of fundamental computations and common network topologies our new analysis improves upon previous per-processor lower bounds. We consider torus and mesh topologies, universal fat-trees, and hypercubes; algorithms covered include classical matrix multiplication and direct numerical linear algebra, fast matrix multiplication algorithms, programs that reference arrays, N-body computations, and the FFT. For example, we show that fast matrix multiplication algorithms (e.g., Strassen's) run on a 3D torus will suffer from contention bottlenecks. On the other hand, this network is likely sufficient for a classical matrix multiplication algorithm. Our new lower bounds are matched by existing algorithms only in very few cases, leaving many open problems for network and algorithmic design.

1. INTRODUCTION

Good connectivity of the inter-processor network is necessary for fast execution of parallel algorithms. Insufficient connectivity provably slows down specific parallel algorithms that are communication intensive. Parallel algorithms that ignore network topology can suffer from congestion along network links, and 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 costs*, and point out cases where this cost is a performance bottleneck.

In our model a $\langle P, M, G_{Net}\rangle$-machine has P identical processors, each with local memory of size M, connected with interprocessor network G_{Net}.[1] The network G_{Net} may have P vertices where each vertex is both a processor and a router (direct networks like tori and hypercubes) or may have more than P vertices, where some vertices represent routers (indirect networks like fat-trees). Edges of G_{Net} are network links with weights corresponding to bandwidth. All weights are typically the same in a torus or hypercube, but not in a fat-tree. 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, programs that reference arrays and others (cf. [1, 4, 5, 7, 11, 17, 22, 24, 27, 30, 37]).

We demonstrate the usefulness of our novel analysis by applying it to a spectrum of algorithm and network combinations, including many of the algorithms above, with networks such as meshes and tori of any dimension, universal fat-trees, and hypercubes. We note that the current five fastest supercomputers in the world [34] have either torus or fat-tree network topologies (as of November 2014). By considering the network graphs, we introduce communication lower bounds for certain computations and networks that are tighter than what was previously known. We also show that often, but not always, the worst contention is expected across the network bisection, supporting a trend that appears in various network designs for decades, namely that a network's bisection is one of its most important parameters.

By considering the network graphs, we introduce communication lower bounds for certain computations and networks that are tighter than what was previously known. In this work, 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),

*Current affiliation: Google Inc.

[1]This model is a variant of the distributed-memory communication model (cf, [7, 12, 15]), where all-to-all connectivity is assumed, and the bandwidth-cost of an algorithm is proportional to the number of words communicated by the worst processor.

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 theorems 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., assuming no recomputation. Our new lower bounds are matched by existing algorithms only in very few cases, leaving exciting open problems for network and algorithmic design such as when scheduling heavily utilized computation kernels on (a subset of) a supercomputer.

2. CONTENTION LOWER BOUNDS

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 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/received by processor p.*

Observe that for W_{proc} there exist two types of per-processor lower bounds: memory-independent $W_{\text{proc}}(P, N)$ (cf. [5]) and memory-dependent $W_{\text{proc}}(P, M, N)$ (cf. [6, 7, 8, 17, 27]) where N is the input and output data size.

DEFINITION 2.2. *Consider a $\langle P, M, G_{Net}\rangle$-machine running an algorithm. The* contention cost W_{link} *is the maximum over edges $e \in E(G_{Net})$ 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 [25]. 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 $\langle P, M, G_{Net}\rangle$-machine. Given a computation with input and output data size N and lower bounds on the per-processor bandwidth cost $W_{proc}(P, M, N)$ and $W_{proc}(P, N)$, 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 contention cost is bounded below by*

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

and

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

where

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

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.[2] 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 memory-dependent contention bound follows. The memory-independent lower bound is derived similarly. □

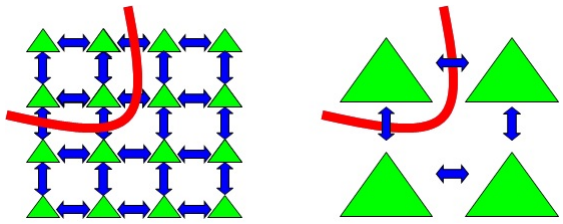

Figure 1: Computation of t = 4 processors on a 16-processor machine can be emulated as the computation of one processor on a 4-processor machine.

In the case of irregular or indirect network topologies, we may not be able to easily identify the edge expansion of the network graph. For these cases, we state a more general theorem that leads to valid but possibly weaker lower bounds. Intuitively, one can partition the processors in any way and apply the per-processor bound to the partitioned network graph to obtain a valid contention bound. While Theorem 2.3 uses subsets attaining the edge expansion to maximize the lower bound, one can also choose contention-bound subsets by hand. The following theorem states this formally for uniformly sized partitions; we omit the proof because it is nearly identical to Theorem 2.3.

THEOREM 2.4. *Consider a $\langle P, M, G_{Net}\rangle$-machine. Let $\{\mathcal{P}_i\}$ be a set of partitions of the P processors, where each partition consists of equally sized subsets of size t_i, and define*

$$L_i = \min_{S_j \in \mathcal{P}_i} |E(S_j, V \setminus S_j)|$$

[2]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 $|E(S)| = d|S|$.

to be the smallest aggregate bandwidth for any subset of processors in partitioning $\mathcal{P}_i$. Given a computation with input and output data size N, and lower bounds on the per-processor bandwidth cost $W_{proc}(P, M, N)$ and $W_{proc}(P, N)$, 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 contention cost is bounded below by

$$W_{link}(P, M, N) \geq \max_i W_{proc}(P/t_i, Mt_i, N)/L_i$$

and

$$W_{link}(P, N) \geq \max_i W_{proc}(P/t_i, N)/L_i.$$

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 $\Theta(n^3)$.[3] 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 [3, Definition 4.1]. For such computations, we have the following lower bound:

THEOREM 3.1 ([7],[27]). *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*

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

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 ([5]). *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*

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

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

Strassen-like Fast 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 [8, Section 5.1] for more details. The memory-dependent communication lower bound for Strassen-like algorithms is:

THEOREM 3.3 ([8, 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*

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

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 [8]) the number of words some processor must communicate is at least*

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

The proof is identical to [5, Theorem 2.1], 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. [17] consider programs of the following form:

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

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 [17]. 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.[4] For such computations we have the following lower bound:

THEOREM 3.5 ([17]). *Consider an algorithm performing a computation of the form given by* (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 Inequality (3)*, assuming that this linear program is feasible (see [17]).*

We restate the memory-independent bound from [17] 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

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

words where s_{HBL} is defined as in Theorem 3.5.

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 [20]. 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 next discuss per-processor communication cost bounds for the FFT and comparison sorts. A sequential communication cost lower bound of $\Omega(n \log n/ \log M)$ was given by Hong and Kung [24]. A parallel memory-independent per-processor bound has been proven for the LPRAM model [1] and the BSP model [11].

THEOREM 3.6 ([1, 11]). *The per-processor communication cost of FFT algorithm on input of size N, run on a $\langle P, M, G_{Net}\rangle$-machine with no recomputation is bounded below by*

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

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

We are unaware of a previous memory-dependent per-processor bound for FFT, although this is a straightforward extension of the existing sequential lower bounds in [24] or the one in [35] for memory hierarchy model. We provide it here for completeness.

THEOREM 3.7. *The per-processor communication cost of FFT algorithm on input of size N, run on a $\langle P, M, G_{Net}\rangle$-machine with no recomputation is*

$$W_{proc}(P, M, N) = \Omega\left(\frac{N \log N}{P \log M} - M\right).$$

PROOF. The bound follows from the same logic as the partitioning argument in Hong-Kung [24], applied to one of the processors that performs $\Theta((N \log N)/P)$ of the computations. Since (as in [24]) each M-partition may have no more than $M \log M$ vertices, the number of M-partitions is at least $N \log N/(PM \log M)$. Thus, the total per-processor bandwidth is $\Omega(M\lfloor((N \log N)/(PM \log M)\rfloor)$, and the theorem follows. □

Note that this memory dependent bound is dominated by the memory independent one, since we assume $MP \geq N$.

3.2 Small Set Expansion of Torus Networks

Torus networks are common topologies among current supercomputers (four of the current top five [34], for example). Cray's XK7 [18] and IBM's Blue Gene/P [26] machines utilize 3D tori, Blue Gene/Q a 5-dimensional torus [16], and the K computer in Japan a 6-dimensional network topology [2]. Intel Xeon Phi coprocessors rely on a ring-based (a 1-dimensional torus) on-chip communication network between cores [28]. In this section, 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 [13].

LEMMA 3.8. *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 [32]. 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 $s \leq |V|/2$. $\square$

3.3 Universal Fat-Trees

We also consider universal fat-trees [31], which are indirect networks consisting of processors connected by a binary tree of router nodes. As of November 2014, the world's fastest supercomputer [34] is China's National University of Defense Technology Tianhe-2, which has a custom interconnect with a fat-tree topology [19]. In a fat-tree network, the bandwidth capacity of links between routers varies, increasing from links connecting the processor leaves to the "fattest" links connected to the root node. Universal fat-trees are parametrized by the root links' capacity: w words of data per unit time, where the capacity of the leaf links (to processors) is normalized to 1. Given the parameter w, the capacity of each link at level $0 \leq i \leq \log P$ in the tree is $\min\{P/2^i, \mathrm{w}/2^{2i/3}\}$. This implies that the capacities of the links between subsequent levels of the tree differs by either a factor of 2 or $2^{2/3}$. Because the fat-tree network graph consists of both processor and router nodes (an indirect network) and link capacities are variable, we will not be able to apply Theorem 2.3 straightforwardly; instead we will appeal to the more general Theorem 2.4.

4. APPLICATIONS

In this section, we apply the general contention lower bounds of Theorems 2.3 and 2.4 to pairs of computations and network topologies. That is, we combine the previously known per-processor bounds of a computation with properties of the network topology to obtain novel contention bounds that are tighter in some scenarios. We first derive contention lower bounds for all the computations on D-dimensional tori (or meshes) in Section 4.1, and then in Section 4.2 we compare the bounds to determine in which scenarios each lower bound dominates. We also derive contention bounds for the computations on fat-tree topologies in Section 4.3; we discuss the comparisons among bounds for that topology more briefly, as the relationships are qualitatively similar. Table 1 presents the communication lower bounds for each of the computations on both D-dimensional tori and fat-trees with root capacity w.

We have applied similar analysis for these computations to hypercube topologies, but we omit the derivations because we obtain no tighter lower bounds than the per-processor bounds. Although the lack of tighter bounds does not imply hypercubes will never be contention bound for the these computations, the failure of this line of analysis is somewhat expected because, for example, any bisection cut of a P-processor hypercube includes at least $P/2$ links [13].

4.1 Contention Lower Bounds for Tori

In this section, we derive contention lower bounds by plugging the memory-dependent and memory-independent per-processor lower bounds [5, 8, 17, 27] 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-like, and N-body.

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:

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

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. By Lemma 3.8, for a D-dimensional torus, the denominators of the contention bounds in Theorem 2.3 are $2D \cdot t \cdot \Theta(t^{-1/D})$. Thus, the memory-dependent contention bound is:

$$W^{\mathrm{tor}}_{\mathrm{link}}(P, M, N) = \max_{t \in T} \Omega\left(\frac{F}{PM^{\alpha-1}} \cdot t^{1-\alpha+1/D}\right).$$

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 $t = P/2$,[5] and we obtain a new and tighter bound:

$$W^{\mathrm{tor}}_{\mathrm{link}}(P, M, N) = \Omega\left(\frac{F}{P^{\alpha-1/D}M^{\alpha-1}}\right).$$

The per-processor memory-independent bound is

$$W_{\mathrm{proc}}(P, N) = \Omega\left(\frac{N}{P^{1/\alpha}}\right), \tag{6}$$

and we can apply Theorem 2.3 to obtain:

$$W^{\mathrm{tor}}_{\mathrm{link}}(P, N) = \max_{t \in T} \Omega\left(\frac{N}{P^{1/\alpha}} \cdot t^{1/\alpha-1+1/D}\right).$$

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:

$$W^{\mathrm{tor}}_{\mathrm{link}}(P, N) = \Omega\left(\frac{N}{P^{1-1/D}}\right).$$

Programs that Reference Arrays.

According to Theorem 3.5, the memory-dependent per-processor bandwidth lower bound for programs defined by (2) is $W_{\mathrm{proc}}(P, M, N) = \Omega(|\mathcal{Z}|/(PM^{s_{\mathrm{HBL}}-1}))$. Similar to the derivation for the previous problems (albeit with $\alpha =$

[5] 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-torus), hence the following contention analysis holds up to a constant factor.

		Memory Dependent	Memory Independent
Direct Linear Algebra	W_{proc}	$\Omega\left(\frac{N^{3/2}}{PM^{1/2}}\right)$	$\Omega\left(\frac{N}{P^{2/3}}\right)$
	$W_{\text{link}}^{\text{tor}}$	$\Omega\left(\frac{N^{3/2}}{P^{3/2-1/D}M^{1/2}}\right)$	$\Omega\left(\frac{N}{P^{1-1/D}}\right)$
	$W_{\text{link}}^{\text{f-t}}$	$\Omega\left(\frac{N^{3/2}}{\text{w}(MP)^{1/2}}\right)$	$\Omega\left(\frac{N}{\text{w}}\right)$
Strassen and Strassen-like	W_{proc}	$\Omega\left(\frac{N^{\omega_0/2}}{PM^{\omega_0/2-1}}\right)$	$\Omega\left(\frac{N}{P^{2/\omega_0}}\right)$
	$W_{\text{link}}^{\text{tor}}$	$\Omega\left(\frac{N^{\omega_0/2}}{P^{\omega_0/2-1/D}M^{\omega_0/2-1}}\right)$	$\Omega\left(\frac{N}{P^{1-1/D}}\right)$
	$W_{\text{link}}^{\text{f-t}}$	$\Omega\left(\frac{N^{\omega_0/2}}{\text{w}(MP)^{\omega_0/2-1}}\right)$	$\Omega\left(\frac{N}{\text{w}}\right)$
$O(N^2)$ N-body	W_{proc}	$\Omega\left(\frac{N^2}{PM}\right)$	$\Omega\left(\frac{N}{P^{1/2}}\right)$
	$W_{\text{link}}^{\text{tor}}$	$\Omega\left(\frac{N^2}{P^{2-1/D}M}\right)$	$\Omega\left(\frac{N}{P^{1-1/D}}\right)$
	$W_{\text{link}}^{\text{f-t}}$	$\Omega\left(\frac{N^2}{\text{w}MP}\right)$	$\Omega\left(\frac{N}{\text{w}}\right)$
Programs Referencing Arrays	W_{proc}	$\Omega\left(\frac{F}{PM^{s_{\text{HBL}}-1}}\right)$	$\Omega\left(\left(\frac{F}{P}\right)^{1/s_{\text{HBL}}}\right)$
	$W_{\text{link}}^{\text{tor}}$	$\Omega\left(\frac{F}{P^{s_{\text{HBL}}-1/D}M^{s_{\text{HBL}}-1}}\right)$	$\Omega\left(\frac{F^{1/s_{\text{HBL}}}}{P^{1-1/D}}\right)$
	$W_{\text{link}}^{\text{f-t}}$	$\Omega\left(\frac{F}{\text{w}(MP)^{s_{\text{HBL}}-1}}\right)$	$\Omega\left(\frac{F^{1/s_{\text{HBL}}}}{\text{w}}\right)$
FFT/Sorting	W_{proc}	$\Omega\left(\frac{N\log(N)}{P\log(M)}\right)$	$\Omega\left(\frac{N\log(N)}{P\log(N/P)}\right)$
	$W_{\text{link}}^{\text{tor}}$	$\Omega\left(\frac{N\log(N)}{P^{1-1/D}(\log(M)+\log(P))}\right)$	$\Omega\left(\frac{N}{P^{1-1/D}}\right)$
	$W_{\text{link}}^{\text{f-t}}$	$\Omega\left(\frac{N\log N}{\text{w}\log(MP)}\right)$	$\Omega\left(\frac{N}{\text{w}}\right)$

Table 1: Per-processor bounds (W_{proc}) vs. the new contention bounds (W_{link}) on a D-dimensional torus and fat-trees with root capacity w for classical (dense) linear algebra, fast matrix multiplication, $O(N^2)$ N-body, a general set of programs that reference arrays, and Fast Fourier Transform (FFT) and comparison sorting.

s_{HBL}), the contention bound becomes

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

which is maximized at either $t=1$ (the per-processor bound), or $t=P/2$ (see Footnote 5). So, we obtain

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

as a memory-dependent lower bound on contention. In a similar manner, we can derive a memory-independent contention lower bound from Equation (4). Observing that the contention bound is maximized at either $t=1$ or $t=P/2$, we derive the memory-independent lower bound on contention at $t=P/2$: $W_{\text{link}}^{\text{tor}}(P,N) = \Omega(|\mathcal{Z}|^{1/s_{\text{HBL}}}/(P^{1-1/D}))$.

FFT/Sorting.

As with the previous algorithms, we apply Theorem 2.3 to the relevant per-processor bounds given in Section 3.1. As we mentioned earlier, the memory-independent per-processor bound always dominates the memory-dependent bound because we assume that $M \ge N/P$.

The per-processor memory-independent bound is

$$W_{\text{proc}}(P,N) = \Omega\left(\frac{N\log(N)}{P\log(N/P)}\right),$$

and we can apply this bound to Theorem 2.3 to obtain:

$$W_{\text{link}}^{\text{tor}}(P,N) = \max_{1\le t\le P/2} \Omega\left(\frac{N\log(N)}{P\log(Nt/P)t^{-1/D}}\right) = \frac{N\log(N)}{P}\max_{1\le t\le P/2}\Omega\left(\frac{t^{1/D}}{\log(Nt/P)}\right). \quad (7)$$

Again, when $t=1$ we obtain the original per-processor bound. Equation (7) has a stationary point at $t = P2^D/N$, but via consideration of the second derivative with respect 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 $t=P/2$:

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

4.2 Analysis and Interpretation for Tori

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 [17] and is proven in Appendix A. We note that in practice the value of constants 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 B 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 [14] does not suffer from contention on a 2D torus network, it is also not communication-optimal. The more communication-efficient "3D" algorithms [10, 1, 33, 38], 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 (see Equation (5) for example) 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 [5] for a discussion of this behavior given only per-processor bounds.

Algorithm	ω_0	$\lfloor D_1 \rfloor$	$\lceil D_2 \rceil$
Classical	3	2	3
Strassen (1969) [39]	≈ 2.81	2	4
Schönhage (1981) [36]	≈ 2.55	3	5
Strassen (1987) [40]	≈ 2.48	4	6
Le Gall (2014) [21]	≈ 2.3729	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 / conjectures, see [8].

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 [5] 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 multiplication (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.

4.3 Contention Lower Bounds for Fat-Trees

To obtain contention bounds for fat-trees, we use Theorem 2.4. Thus, we must define a set of partitions $\{\mathcal{P}_i\}$ and compute the corresponding aggregate bandwidths $\{L_i\}$. We consider partitioning the processors into complete subtrees of sizes ranging from one processor to $P/2$ processors. In this case, each subset of processors has one external link from its root node to the next level in the tree. We let $\mathcal{P}_i$ be a partition of the processors into 2^i subsets, where each subset of processors consists of a complete binary subtree of $P/2^i$ processors. In this case, the minimum aggregate bandwidth is given by the capacity of the link connected to the

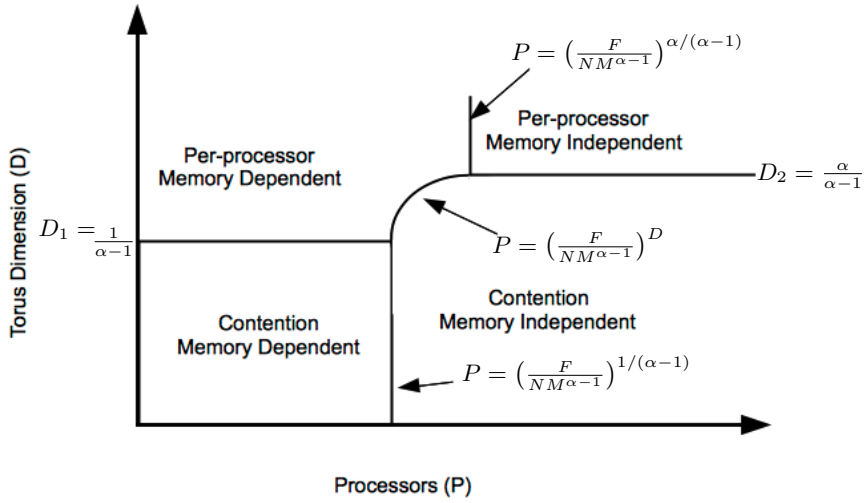

Figure 2: Relationship between the per-processor and contention communication lower bounds for direct linear algebra, Strassen/Strassen-like and the $O(N^2)$ N-body problems.

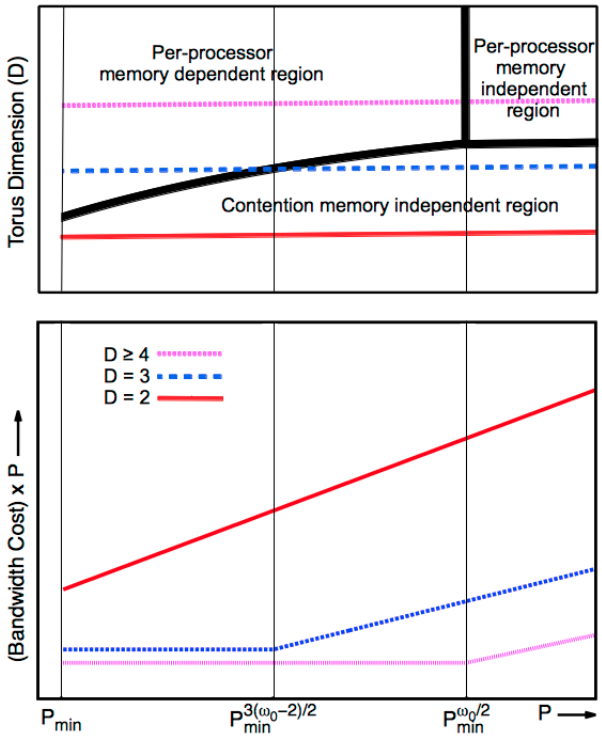

Figure 3: Communication bounds for Strassen's algorithm on D-dimensional tori. The lower plot is log-log, while the upper is linear on the y-axis. Horizontal lines in the lower plot correspond to perfect strong scaling.

root of the subtree:

$$L_i = \min\left\{\frac{P}{2^i}, \frac{\mathrm{w}}{2^{2i/3}}\right\}. \tag{8}$$

Direct Linear Algebra, Strassen-like, and N-body.

We first consider direct linear algebra, Strassen-like matrix multiplication, and $O(N^2)$ N-body algorithms with per-processor lower bounds given by Equations (5) and (6) where $1 < \alpha \leq 2$, depending on the algorithm. The aggregate bandwidth of a complete subtree of size $P/2^i$ is given by Equation (8), so by Theorem 2.4 our memory-dependent contention bound is

$$W_{\text{link}}^{\text{f-t}}(P,M,N) = \max_{1\leq i\leq \log P} \Omega\left(\frac{\frac{F}{2^i(MP/2^i)^{\alpha-1}}}{\min\{P/2^i, \mathrm{w}/2^{2i/3}\}}\right)$$
$$= \max_{1\leq i\leq \log P} \Omega\left(\frac{F\cdot 2^{i(\alpha-1)}}{P^\alpha M^{\alpha-1}} + \frac{F\cdot 2^{i(\alpha-4/3)}}{\mathrm{w}(MP)^{\alpha-1}}\right).$$

The first term is an increasing function of i because $\alpha > 1$. The second term can be either increasing or decreasing (it will be decreasing for a fast matrix multiplication algorithm with exponent $\omega_0 < 2.66$). The maximum is therefore achieved at either $i = 1$ or $i = \log P$, so in order to obtain a new lower bound we evaluate the expression at $i = 1$ to obtain $W_{\text{link}}^{\text{f-t}}(P,M,N) = \Omega\left(F/\left(\mathrm{w}(MP)^{\alpha-1}\right)\right)$. Likewise, the memory-independent contention bound is

$$W_{\text{link}}^{\text{f-t}}(P,N) = \max_{1\leq i\leq \log P} \Omega\left(\frac{N/(2^i)^{1/\alpha}}{\min\{P/2^i, \mathrm{w}/2^{2i/3}\}}\right)$$
$$= \max_{1\leq i\leq \log P} \Omega\left(\frac{N}{P}\cdot 2^{i(1-1/\alpha)} + \frac{N}{\mathrm{w}}\cdot 2^{i(2/3-1/\alpha)}\right).$$

As before, the first term is always increasing but the second term can be decreasing (this time for fast matrix multiplication with exponent $\omega_0 < 3$), so the maximum could be achieved at either endpoint. By plugging in $i = 1$ we obtain a new bound of $W_{\text{link}}^{\text{f-t}}(P,N) = \Omega(N/\mathrm{w})$.

As in the case of torus networks, of the two contention bounds, the memory-independent one dominates the memory-dependent one assuming $N < MP$, or that the data fits across all processors' memories. Relationships among the two per-processor bounds and memory-independent contention bound can be derived as in Section 4.2 for torus networks (see Figures 2 and 3). We note that the memory-independent contention bound is the tightest bound for Strassen-like ma-

trix multiplication when, for example, the fat-tree parameter w falls in the range $P^{2/3} \leq \mathrm{w} < P^{2/\omega_0} = P^{1/\alpha}$ and $P = \Omega((N/M)^{\omega_0/2})$. That is, the bound $\mathrm{w} > P^{1/\alpha}$ for a fat-tree is analogous to the value $D > D_2 = \lceil \alpha/(\alpha-1) \rceil$ for a torus. We do not obtain any tighter bounds for direct linear algebra or the N-body problem. This analysis suggests that for those computations, a fat-tree with the cheapest choice of $\mathrm{w} = P^{2/3}$ is sufficient to avoid contention becoming the communication bottleneck, though we point out that tighter contention bounds may exist that contradict this conjecture.

Programs Referencing Arrays.

The analysis for programs referencing arrays follows that of direct linear algebra, matrix-multiplication, and N-body computations. Replacing F with $|\mathcal{Z}|$, N with $|\mathcal{Z}|^{1/s_{\mathrm{HBL}}}$, and α with s_{HBL}, we obtain the contention bounds $W^{\text{f-t}}_{\text{link}}(P,M,N) = \Omega\left(|\mathcal{Z}|/\left(\mathrm{w}(MP)^{s_{\mathrm{HBL}}-1}\right)\right)$ and $W^{\text{f-t}}_{\text{link}}(P,N) = \Omega(|\mathcal{Z}|^{1/s_{\mathrm{HBL}}}/\mathrm{w})$. The dominance of the memory-independent bound follows from Claim A.1, and we can also conclude that choosing $\mathrm{w} > P^{1/s_{\mathrm{HBL}}}$ guarantees that these contention bounds do not dominate the per-processor bounds.

FFT/Sorting.

To obtain a contention bound for the FFT on a fat-tree, we combine (via Theorem 2.4) the per-processor bounds given in Theorems 3.6 and 3.7 with the aggregate bandwidth defined by Equation (8). This yields a memory-independent contention bound of

$$W^{\text{f-t}}_{\text{link}}(P,N) = \max_{1 \leq i \leq \log P} \Omega\left(\frac{\frac{N \log N}{2^i \log(N/2^i)}}{\min\{P/2^i, \mathrm{w}/2^{2i/3}\}}\right) = \max_{1 \leq i \leq \log P} \Omega\left(\frac{N \log N}{P \log(N/2^i)} + \frac{N \log N}{\mathrm{w} \log(N/2^i) 2^{i/3}}\right).$$

Again, this function is maximized at either endpoint, so to obtain a new bound we choose $i = 1$, which evaluates to $W^{\text{f-t}}_{\text{link}}(P,N) = \Omega(N/\mathrm{w})$. The memory-dependent contention bound can be derived similarly, evaluating to

$$W^{\text{f-t}}_{\text{link}}(P,M,N) = \Omega(N \log N/(\mathrm{w} \log(MP))).$$

As in the per-processor case, the memory-independent contention bound dominates the memory-dependent contention bound because $N < MP$. However, either of the two memory-independent bounds may dominate; the contention bound dominates when $\mathrm{w} < P(1 - (\log P/\log N))$. That is, for sufficiently small N (N close to P), contention is not a bottleneck even for $\mathrm{w} = P^2/3$; for sufficiently large N ($N = P^C$ for some constant C), then contention will bottleneck the computation unless w is chosen to be $\Omega(P)$.

5. DISCUSSION AND FUTURE RESEARCH

Is it always about bisection bandwidth?.

For the algorithms discussed in this paper on torus and fat-tree networks, the contention lower bound, when it differs from the per-processor bound, is maximized for $t = P/2$; that is, the contention bound corresponds to a network bisection cut. Is this always the case, or do we expect to have combinations of algorithms and machines where contention bounds dominate, but the constricting cut is not balanced? A contrived example could be when $h_s(G_{Net})$ is not a decreasing function of s; for example, two networks of processors, a large and a small one, where each of them is well connected, but the connection between the large and the small one is narrow (e.g., two racks with one router each, connected with a narrow link one to the other, where the racks contain different numbers of processors).

Applicability.

Immediate applications of the technique include combinations of other networks (e.g. dragonfly networks [29]) and other classes of algorithms for which per-processor lower bounds are known. 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 [9, 23, 31], 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.

Contention-Efficient Algorithms.

Some parallel algorithms attaining per-processor communication lower bounds have also been tuned to particular topologies (cf. [38] for classical matrix multiplication on 3D torus). Algorithmic analysis of the contention costs will likely show that the contention bounds for these and related computations are attainable. Many other 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 automatic tuning or topology-oblivious tools?

Acknowledgments

We thank Guy Kindler for pointing us to [13]. 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, 1901/14, 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. Research is supported by the Einstein Foundation. Research is supported by the Minerva Foundation. 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.

6. REFERENCES

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

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

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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/.

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

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

[15] 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.

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

[17] 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.

[18] Cray. Cray XK7 brochure, 2011.

[19] J. Dongarra. Visit to the National University for Defense Technology Changsha, China, June 2013.

[20] M. Driscoll, E. Georganas, P. Koanantakool, E. Solomonik, and K. Yelick. A communication-optimal N-body algorithm for direct interactions. In *Proceedings of IPDPS '13*, 2013.

[21] F. L. Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC 2014)*, pages 296–303, 2014.

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

[23] 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.

[24] 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.

[25] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin of the AMS*, 43(4):439–561, 2006.

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

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

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

[29] J. Kim, W. Dally, S. Scott, and D. Abts. Technology-driven, highly-scalable dragonfly topology. In *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, pages 77–88, June 2008.

[30] 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.

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

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

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

[34] H. Meuer, E. Strohmaier, J. Dongarra, and H. Simon. Top500 Supercomputer Sites, 2014. www.top500.org.

[35] J. E. Savage. Extending the Hong-Kung model to memory hierarchies. In *COCOON*, pages 270–281, 1995.

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

[37] M. Scquizzato and F. Silvestri. Communication lower bounds for distributed-memory computations. In E. W. Mayr and N. Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, volume 25 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 627–638, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[38] 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, pages 90–109. Springer Berlin Heidelberg, 2011.

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

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

APPENDIX

A. DOMINANCE OF MEMORY-INDEPENDENT CONTENTION BOUND

CLAIM A.1. *Let Alg be an algorithm performing a computation of the form given by (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. [17], 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 [17], 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. □

B. DERIVATION OF EXPRESSIONS IN FIGURE 2

- **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:

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

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