Bandwidth-optimal All-to-all Exchanges in Fat Tree Networks

Bogdan Prisacari
IBM Research
bpr@zurich.ibm.com

German Rodriguez
IBM Research
rod@zurich.ibm.com

Cyriel Minkenberg
IBM Research
sil@zurich.ibm.com

Torsten Hoefler
ETH Zurich
htor@inf.ethz.ch

ABSTRACT

The personalized all-to-all collective exchange is one of the most challenging communication patterns in HPC applications in terms of performance and scalability. In the context of the fat tree family of interconnection networks, widely used in current HPC systems and datacenters, we show that there is potential for optimizing this traffic pattern by deriving a tight theoretical lower bound for the bandwidth needed in the network to support such communication in a non-contending way. Current state of the art methods require up to twice as much bisection bandwidth as this theoretical minimum. We propose a set of optimized exchanges that use exactly the minimum amount of resources and exhibit close to ideal performance. This enables cost-effective networks, i.e., with as little as half the bisection bandwidth required by current state of the art methods, to exhibit quasi optimal performance under all-to-all traffic. In addition to supporting our claims by mathematical proofs, we include simulation results that confirm their correctness in practical system configurations.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network communications*

Keywords

all-to-all; fat tree networks; bandwidth optimality

1. INTRODUCTION AND MOTIVATION

Today's parallel computations are often arranged in a bulk synchronous programming (BSP) model [36] in which communication and computation steps alternate. For example, in the Message Passing Model as implemented by MPI [25], one can realize communication steps by sending messages between processes. Several common communication patterns, such as broadcast, are available as collective communications in MPI. This high-level specification enables a clean separation of concerns between the algorithm, that requires certain semantics, and the network architecture, that implements communication. This enables performance-portability of programs to a wide variety of different architectures and systems. Indeed, a collective programming model has enormous benefits for programmability as well as performance [9] and is thus also adopted by many other parallel programming environments [2, 21].

Of those collective communications, all-to-all is certainly the most demanding operation for the network because it needs to move $\Omega(p)$ data between p processes; it is thus also often considered the least scalable of MPI's collective operations [3]. Yet, there are many applications that require this communication pattern: Spectral codes, such as Direct Numerical Simulation [23] perform a 3D Fast Fourier Transform, utilizing large all-to-all communications for transposing the y direction over the network together with shared-memory on-node transforms of the x direction [6]. Even at the other end of the spectrum of scientific applications, data-driven parallel graph computations, such as betweenness centrality, essentially perform full all-to-all exchanges over the network [37].

All-to-all has thus been the subject of multiple research efforts optimizing it for different theoretical network models, such as LogP [29] or Postal [4], or technologies, such as InfiniBand [19] or Myrinet [39]. Our approach, however, is mostly oblivious to the network technology and current theoretical network models because we optimize for a certain network *topology*.

One important topology today is the family of fat trees [17], which form the base of several Petascale computer architectures (e.g., [38]). Fat trees con be configured in terms of bandwidth, rack layout, and cost. At the heart of the cost/performance trade-off, and thus a critical decision in supercomputer design, is the number of resources towards the root, which determines the *bisection bandwidth*. As computations requiring all-to-all exchanges are important, and as it is commonly believed that all-to-all bandwidth is bounded by the bisection bandwidth (cf. [16]), most fat tree networks are designed with full bisection bandwidth.

In this paper, we show that all-to-all only requires half bisection bandwidth due to its inherent locality. We then demonstrate how to design slimmed fat trees that have optimal cost while still providing full all-to-all bandwidth and

ICS'13, June 10–14, 2013, Eugene, Oregon, USA.

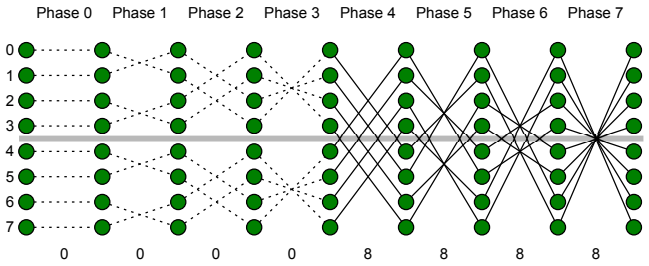

Figure 1: XOR exchange pattern between 8 communicating tasks. The first half of the phases are local and no traffic crosses the network bisection (horizontal line), whereas during the last half of the phases, all the messages cross the bisection (continuous lines). This requires a bisection bandwidth equal to the aggregate injection bandwidth.

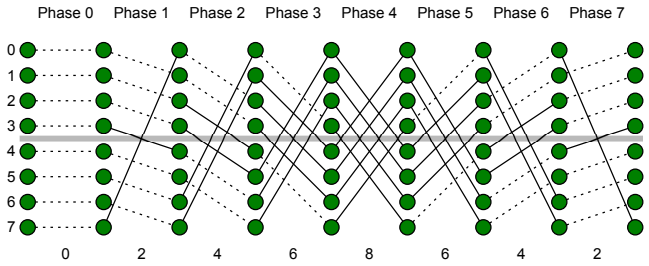

Figure 2: Linear exchange pattern between 8 communicating tasks for shift = 0. The uneven distribution of the number of messages (continuous lines) crossing the network bisection requires a bisection bandwidth equal to the aggregate injection bandwidth to enable contention-free traffic in all phases.

how to schedule all-to-all communications on general trees optimally. The main contributions of our work are:

- A general discussion of locality in the all-to-all problem.
- A construction of cost-optimal fat trees delivering full all-to-all bandwidth.
- An algorithm to compute optimal exchange patterns for a given fat tree.

We start with a description of the locality within all-to-all and intuitive examples for mappings to fat trees. We then present generic lower bounds for all-to-all communications on fat trees followed by a construction of optimal all-to-all schedules. We conclude our study with a set of simulated real-world examples.

2. THE ALL-TO-ALL PROBLEM

In this section we will discuss the intuition behind the bounds for all-to-all. The full proof and the algorithm to compute the send permutations for a given fat tree will be presented in Sections 3.1 and 3.4.

All-to-all is often referred to as "bisection-bound" (cf.[16]). While this is true asymptotically, i.e., a full-bandwidth all-to-all requires a $\Omega(p)$ bisection bandwidth on p processes, we will show that it only requires half of the actual bisection bandwidth *if one can take advantage of the inherent locality of the all-to-all problem.*

Consider an all-to-all schedule where each process $p \in P$ exchanges a single message with each other process. Let P_1 and P_2 form an arbitrary bisection of the process set P, such that $b = |P_1| = |P_2| = \frac{|P|}{2}$. In an all-to-all, each process in the set P_1 has b peers in P_1 and b peers in P_2, such that only $b = \frac{|P|}{2}$ exchanges leave each set. If P_1 and P_2 were arbitrary bisections of processes mapped to endpoints in an arbitrary topology, then only half of the messages cross any bisection of this topology. Thus, intuitively, all-to-all exchanges require only half bisection bandwidth for arbitrary topologies.

The argument can be applied recursively, i.e., partitioning P_1 into P_{11} and P_{12} etc., yielding a bandwidth-minimal construction. We now demonstrate that current all-to-all algorithms do not take advantage of this observation and thus require full bisection bandwidth for a full-bandwidth all-to-all. We then show how to utilize our observation by optimally scheduling all-to-all communications such that they require only half bisection bandwidth and cause less congestion than today's algorithms.

A typical [40] way of performing an all-to-all exchange between N processes or tasks, especially when there is no possibility of message aggregation (such as is the case for large messages), is to schedule the exchange as a set of N phases. Every process sends a single message and receives a single message in each phase (we consider that a process sends a message to itself as well). In this case, the exchange pattern can be entirely defined by a function that maps a given phase p and source task s to a unique destination task d (all three numbered from 0 to $N-1$), meaning that during phase p, the message sent by task s is destined to task d. The two widely used exchange patterns are the binary XOR exchange and the linear shift exchange [40].

In the case of XOR (Figure 1), the function is $d_{\text{XOR}} = s \text{ XOR } p$, where the exclusive or operation is performed bit by bit on the binary representations of s and p.

In the case of the linear shift exchange [30] (Figure 2), the function is $d_{\text{LIN}} = (s+p+\text{shift}) \bmod N$, where N is the total number of communicating tasks and shift is a parameter that takes a constant integer value between 0 and $N-1$.

If we partition the 8 nodes in the example all-to-all in the two equal halves separated by the horizontal median line it is obvious that both exchanges exhibit an uneven distribution of the amount of traffic traversing this bisection of the network across the different phases, with a peak required bisection bandwidth, in both cases, equal to the aggregate injection bandwidth.

Looking at the two examples, it is equally obvious that the average number of messages crossing the bisection as we have defined it in every phase is only half the total number of communicating tasks. Furthermore, there is a way to ensure that in every phase no more than this average number of messages crosses the bisection (Figure 3). This makes it possible to have uncontended per phase traffic with only half the full bisection bandwidth.

We will show in the following sections that the observations for this small example hold in general. We will prove tight lower bounds on the bandwidth required for uncontended all-to-all traffic. Furthermore, for fat tree networks in particular we will introduce a bandwidth-optimized exchange pattern that satisfies these bounds.

3. ALL-TO-ALL OPTIMIZATION

First, we define the model of the interconnection network we are considering, the *fat tree* network. Fat trees are one of

the most popular indirect network topologies used in current supercomputers.

The term fat tree has been used to refer to a broad class of different interconnection layouts. All fat trees can be described as a multi-stage tree-like topology where the width (bandwidth) of the connections increases towards the root of the tree.

An *ideal fat tree* would be a single binary tree interconnection network where the bandwidth of each link towards the root can always accommodate the aggregate capacity of the sub-tree connected to that link. These *ideal fat tree* networks are not realizable for large clusters, as the bandwidth needs to double at each level towards the root, which would require switches with either exponentially increasing number of ports or exponentially increasing bandwidth per link. The CM-5 [17] was the first machine to implement a variant of the ideal fat tree network.

However, it is possible to construct a tree-like network with an equivalent link bandwidth property by using fixed-radix switches, i.e., without increasing the number of connections per switch or their bandwidth [28, 27]. It is furthermore possible to construct networks [26] where the bandwidth towards the root can be tuned to almost arbitrary values, from the full-bisection bandwidth characteristic of ideal fat-trees to reduced bandwidth realizations that have a correspondingly reduced cost. We will discuss these and other practical implications of our method in Section 4.

For the time being, we will assume the following fat-tree model. We consider a layered single rooted tree structure that is homogeneous in that every node on a given layer has the same number of descendants. Using a notation similar to [26], we denote by $FT(L; M_1, M_2, ..., M_L)$ a tree of $L+1$ consecutively numbered layers, with 0 being the layer of the leaves and L that of the single root. Given a layer l, M_l then denotes the number of descendants that every node on layer l has. The tasks that send and receive messages are located at the leaf nodes, whereas the nodes on the other levels serve as message routers. We will assume a single task per node (The case where several tasks are present on the same node is equivalent to extending the fat tree with an additional leaf level, where every leaf corresponds to a task). Furthermore, we will assume that every upward link starting on a node on a given layer has a fixed bandwidth dependent only on the layer index. In the context of this abstract model of the network, we will now prove that the requirement of uncontended all-to-all traffic induces a tight lower bound on the bandwidth of every fat tree link.

3.1 Bandwidth-optimal link occupation in all-to-all communication over a fat tree

Given a fat tree $FT(L; M_1, ..., M_L)$ defined as above, we will show in this section that uncontended all-to-all traffic requires that every upward link originating on a layer l node as well as every downward link ending on a layer l node needs to be able to accommodate at least $B_{\min}(l)$ messages, with:

$$B_{\min}(l) = M_1 \cdot M_2 \cdot ... \cdot M_l - \left\lfloor \frac{M_1 \cdot M_2 \cdot ... \cdot M_l}{M_{l+1} \cdot M_{l+2} \cdot ... \cdot M_L} \right\rfloor . \quad (1)$$

To this end, we denote by Π_l the product $M_1 \cdot M_2 \cdot ... \cdot M_l$. Given that every leaf node corresponds to a single task, the total number of tasks is $N = \Pi_L$. Let us consider one of the nodes on level l, denoted by μ. The total number of messages originating from layer 0 descendants of this node

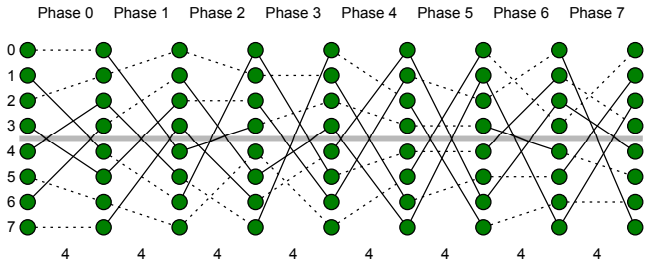

Figure 3: Optimized exchange pattern obtained for a fat tree with 2 levels and with $M_2 = 2$ and $M_1 = 4$. The link occupation optimization ensures that in every phase exactly half of the messages (continuous lines) cross the network bisection, thus making contention free traffic possible with only half the full bisection bandwidth.

during the all-to-all exchange is equal to the number of such descendants (which equals Π_l) multiplied by the total number of destinations (which equals Π_L). Out of these Π_L destinations, Π_l are located in the subtree rooted in μ and thus messages destined to these "local" tasks need *not* be routed on the link going upward from μ. All messages destined to the other tasks, located outside the subtree rooted in μ, *must* be routed on μ's upward link. Consequently, during the complete all-to-all exchange, the number of messages passing through the upward link originating on any node on level l equals $U(l) = \Pi_l \cdot (\Pi_L - \Pi_l)$.

As the exchange comprises Π_L phases, the average per-phase number of messages passing through the upward link originating on a given node on level l therefore equals $\overline{U}(l) = \Pi_l \cdot \left(1 - \frac{\Pi_l}{\Pi_L}\right)$

Given an arbitrary exchange pattern and an arbitrary phase of that exchange, the maximum number of messages passing through an upward link at level l is equal to or greater than $\overline{U}(l)$. This is equivalent to saying that at least one link will have $\overline{U}(l)$ or more messages passing through it in that phase. Thus, a prerequisite for uncontended single-phase traffic is to provide every upward link with sufficient bandwidth to transfer at least this number of messages. As such, contention-free traffic imposes a lower bound of $\lceil \overline{U}(l) \rceil$ on the number of messages that every upward link starting on a layer l node must be able to accommodate, bound that is equal to $B_{\min}(l)$ as defined in Equation (1).

Through a similar analysis, we can compute the minimum bandwidth necessary on downward links. Considering again one arbitrary node μ on level l, the total number of messages destined to level 0 descendants of this node during the entire exchange is equal to $\Pi_l \cdot \Pi_L$. Out of the Π_L sources of these messages, Π_l are located in the subtree rooted in μ and thus none of the messages originating from them traverse the downward link ending in μ. Consequently, during the complete exchange, the number of messages passing through the downward link ending on any node on level l equals $D(l) = \Pi_l \cdot (\Pi_L - \Pi_l)$, which leads to an average per-phase number of downward messages equal to $\overline{D}(l) = \Pi_l \cdot \left(1 - \frac{\Pi_l}{\Pi_L}\right)$.

Applying the same argument as before, we conclude that every downward link ending on a layer l node must be able to accommodate at least $\lceil \overline{D}(l) \rceil$ messages, which is again exactly equal to $B_{\min}(l)$ defined in Equation (1).

Next, we will introduce an exchange pattern that requires

no more than this minimum capacity, thus effectively proving that the bounds are tight. We will start by intuitively explaining the construction of the new pattern, then proceed to rigorously define it and prove its optimality.

3.2 Intuitive bandwidth-optimal exchange

Let us consider the example all-to-all exchange among 8 tasks presented in Section 2on a $FT(2;4,2)$ topology. For simplicity, let's start by optimizing the load induced by all tasks, on the two upward links ending in the root node.

As shown in Section 3.1, the key to optimizing bandwidth utilization is to balance the load on each link across all phases. Obtaining a bandwidth utilization of $B_{\min}$ across all phases is only possible if every phase exhibits this utilization. In our specific example, we are interested in $B_{\min}(1) = M_1 - \lfloor M_1/M_2 \rfloor = 2$. Thus, we must find a way to ensure that an arbitrary upward link ending on the root is used by exactly two messages in each phase. One way to achieve this is for every source task s to choose in any two consecutive phases a destination that is in the same layer 1 subtree as itself and a destination that is in the other layer 1 subtree. However, this is not enough, as all 4 tasks in a subtree could all choose to send messages outside of the subtree in the same phases. Therefore it is also necessary that the source tasks belonging to the same subtree are desynchronized among themselves in their choice of the destination subtree. A simple subtree selection function that achieves this is $((s \bmod 2)+(p \bmod 2)) \bmod 2$. Once we have selected the subtree, we also need to ensure that the actual destination within that subtree is unique over the set of sources. This can be achieved by selecting among the 4 choices in a specific subtree via the formula $((s/2) + (p/2)) \bmod 4$, where / yields the quotient of integer division. The complete destination selection function can then be defined as: $d = (((s \bmod 2)+(p \bmod 2)) \bmod 2) * 4 + ((s/2)+(p/2)) \bmod 4$. This exchange is illustrated in Figure 3 and it is bandwidth optimal as exactly 4 messages cross the bisection in each phase.

We extend this to the general case of a $FT(L;M_1,...,M_L)$ topology using the same principles. To balance load on the highest layer links, every source will send in M_L consecutive phases to M_L different subtrees. This approach is reused recursively from top to bottom.

The formalization and rigorous generalization of the approach follows in the next subsections. We first define a generic family of exchanges based on a *variable base representation* of numbers, then proceed to select among this set an exchange that we will prove to be bandwidth-optimal.

3.3 Permuted variable base exchanges

Given an ordered set of strictly positive integer numbers $(\Theta_1, \Theta_2, ..., \Theta_K)$ and given a number x with $0 \leq x < \Pi_{k=1}^{K}\Theta_k$, we define the following notation:

$$x = \Sigma_{k=1}^{K}\left(\theta_k^x \cdot \Pi_{k'=1}^{k-1}\Theta_{k'}\right), \;\; 0 \leq \theta_k^x < \Theta_k, \;\; 1 \leq k \leq K. \quad (2)$$

Basically, this equation defines a way of representing positive integers in "variable base". Unlike a normal base-n representation where each digit can take any value between 0 and $n-1$, in this case each digit may have a different range. The θ_k digits define a representation in base $(\Theta_1, \Theta_2, ..., \Theta_K)$, where the least significant digit's base is Θ_1. This representation, which applies solely to non-negative integers strictly smaller than $\Pi_{k=1}^{K}\Theta_k$, can be easily proven to be unique and uniquely identifying a certain value. That is to say:

$$\begin{aligned} &\forall x, y \;\text{ s.t. }\; 0 \leq x, y < \Pi_{k=1}^{K}\Theta_k, \\ &x = y \Leftrightarrow (\theta_k^x = \theta_k^y, \quad 1 \leq k \leq K). \end{aligned} \quad (3)$$

In the context of fat trees, we will assign a variable base representation to every leaf node, and consequently, to every task. For this, we will assign tasks with indices ranging from 0 to $\Pi_L - 1$ and choose appropriate variable bases to represent those indices in.

Given a fat tree $FT(L;M_1,...,M_L)$, let $K = L$ and let $(\Theta_1, \Theta_2, ..., \Theta_L) = (M_{\sigma(1)}, M_{\sigma(2)}, ..., M_{\sigma(L)})$, where σ is an arbitrary permutation of $(1, 2, ..., L)$. Additionally, we denote by α_k^x the digits of an arbitrary number x in the variable base $(M_1, M_2, ..., M_L)$. With these notations, we propose the following exchange pattern, which we will refer to as a *permuted-variable-base* exchange pattern. Let d be the destination of the message generated by source s in phase p. All three values s, d and p are numbers between 0 and $\Pi_{k=1}^{K}M_k - 1$. Then the function defining the pattern is:

$$\alpha_{\sigma(k)}^{d} = (\theta_k^s + \theta_k^p) \bmod \Theta_k, \quad 1 \leq k \leq K. \quad (4)$$

We will now show that this exchange model satisfies the two key properties of a valid all-to-all exchange pattern composed of phases: a source never sends to the same destination twice and no two sources send to the same destination in the same phase.

For the first property, let p_1 and p_2 be two distinct phases and let s be a certain source. We seek to prove that in this case, d_1 and d_2 obtained with Equation (4) are distinct.

Because p_1 and p_2 are positive and strictly smaller than $\Pi_{k=1}^{K}M_k$, they have a θ-representation. Furthermore, since they are distinct, according to Equation (3) there exists at least one $\tilde{k}$ with $1 \leq \tilde{k} \leq K$ such that $\theta_{\tilde{k}}^{p_1} \neq \theta_{\tilde{k}}^{p_2}$. According to Equation (4):

$$\theta_{\tilde{k}}^{s} + \theta_{\tilde{k}}^{p_1} = q_1 \cdot \Theta_{\tilde{k}} + \alpha_{\sigma(\tilde{k})}^{d_1}$$

where q_1 is an integer. Given that

$$0 \leq \theta_{\tilde{k}}^{s} + \theta_{\tilde{k}}^{p_1} < 2 \cdot \Theta_{\tilde{k}}$$

and

$$0 \leq \alpha_{\sigma(\tilde{k})}^{d_1} < \Theta_{\tilde{k}}$$

it follows that

$$-\Theta_{\tilde{k}} < q_1 \cdot \Theta_{\tilde{k}} < 2 \cdot \Theta_{\tilde{k}}.$$

As q_1 is an integer, it can only be 0 or 1. Similarly, q_2 has the same property. In the case where both are 0 or both are 1 ($q_1 = q_2 = q$), it immediately follows that $\alpha_{\sigma(\tilde{k})}^{d_1} \neq \alpha_{\sigma(\tilde{k})}^{d_2}$ from:

$$\theta_{\tilde{k}}^{s} + \theta_{\tilde{k}}^{p_1} - q \cdot \Theta_{\tilde{k}} = \alpha_{\sigma(\tilde{k})}^{d_1}$$

$$\theta_{\tilde{k}}^{s} + \theta_{\tilde{k}}^{p_2} - q \cdot \Theta_{\tilde{k}} = \alpha_{\sigma(\tilde{k})}^{d_2}$$

$$\theta_{\tilde{k}}^{p_1} \neq \theta_{\tilde{k}}^{p_2}$$

In the case where q_1 is 1 and q_2 is 0, we obtain

$$\theta_{\tilde{k}}^{p_1} - \theta_{\tilde{k}}^{p_2} = \alpha_{\sigma(\tilde{k})}^{d_1} - \alpha_{\sigma(\tilde{k})}^{d_2} + \Theta_{\tilde{k}}.$$

If $\alpha_{\sigma(\tilde{k})}^{d_1}$ were equal to $\alpha_{\sigma(\tilde{k})}^{d_2}$, then $\theta_{\tilde{k}}^{p_1} - \theta_{\tilde{k}}^{p_2} = \Theta_{\tilde{k}}$, which would lead to $\theta_{\tilde{k}}^{p_1} \geq \Theta_{\tilde{k}}$, which would constitute a contradiction. Therefore, in this case as well, $\alpha_{\sigma(\tilde{k})}^{d_1}$ must be different from $\alpha_{\sigma(\tilde{k})}^{d_2}$. The symmetric case ($q_1$ is 0 and q_2 is 1) is similar. As we proved that there exists a $\tilde{k}$ for which $\alpha_{\sigma(\tilde{k})}^{d_1} \neq \alpha_{\sigma(\tilde{k})}^{d_2}$, we proved that $d_1 \neq d_2$.

The proof of the second property is similar as the computation of the destination task as given by Equation (4) is symmetric with respect to the phase number p and the source number s.

This shows that every permutation σ of $(1, 2, ..., L)$ induces a valid exchange pattern. However, not all of these patterns optimize the maximum traffic traversing the upward links in the fat tree. In the following subsection, we will prove that the specific exchange pattern corresponding to $\overline{\sigma} = (L, L-1, ..., 2, 1)$ is bandwidth-optimal.

3.4 Bandwidth-optimal exchange pattern

As explained above, let $(\Theta_1, \Theta_2, ..., \Theta_L) = (M_{\overline{\sigma}(1)}, M_{\overline{\sigma}(2)}, ..., M_{\overline{\sigma}(L)}) = (M_L, M_{L-1}, ..., M_1)$.

We will prove that the bandwidth requirements induced at layer l by the variable-base-exchange corresponding to this particular base is equal to $B_{\min}(l)$.

We will start with the per phase bandwidth requirement per upward link. First note that all the descendants of a given node on level $\tilde{k}$ share the same α_k values for all k such that $\tilde{k} < k \leq L$.

Let p be a given phase of the all-to-all exchange. Let μ be an arbitrary node at level $\tilde{k}$. The total number of messages originating in level 0 descendants of this node during phase p is $\Pi_{\tilde{k}}$. Among these messages, some will be destined to nodes that are descendants of μ and some to nodes that are not. For a message to be destined to a descendant of μ, as noted above, its destination d must have certain fixed values for some of its α coefficients, values that depend exclusively on the position of μ. Specifically:

$$\alpha_k^d = \alpha_k(\mu), \quad \tilde{k} < k \leq L.$$

As d is obtained via Equation (4) this leads to:

$$(\theta_{L+1-k}^s + \theta_{L+1-k}^p) \mod M_k = \alpha_k(\mu), \quad \tilde{k} < k \leq L.$$

As for a given p, the θ representation of p is fixed, and θ_{L+1-k}^s, θ_{L+1-k}^p and $\alpha_k(\mu)$ are all strictly smaller than M_k, this means that the θ representation of s needs to satisfy:

$$\theta_{L+1-k}^s = t_k(\mu, p), \quad \tilde{k} < k \leq L$$

where $t_k(\mu, p)$ is a set of factors independent of s. So for a descendant s of μ to send a packet to another descendant of μ in phase p of the exchange, the θ coefficients of s from rank 1 to rank $L - \tilde{k}$ must all have values that depend exclusively on μ and p, but not on s. According to the θ representation definition in Equation (2), this is equivalent to:

$$s = r(\mu, p) + q(s, \mu, p) \cdot \Pi_{k=1}^{L-\tilde{k}} \Theta_k$$

where $0 \leq r(\mu) < \Pi_{k=1}^{L-\tilde{k}} \Theta_k$. Thus, the only sources s that send messages to other descendants of μ are those whose index yields a fixed remainder (dependent of μ and p) when divided by $\Pi_{k=1}^{L-\tilde{k}} \Theta_k$. Because the s values are consecutive among the descendants of μ and because in total there are $\Pi_{\tilde{k}}$ such descendants, the minimum number of sources that send to a destination that is also a descendant of μ is:

$$N_{min}(\tilde{k}) = \left\lfloor \frac{\Pi_{\tilde{k}}}{\Pi_{k=1}^{L-\tilde{k}} \Theta_k} \right\rfloor$$

which according to the definition of the base is equal to:

$$N_{min}(\tilde{k}) = \left\lfloor \frac{\Pi_{\tilde{k}}}{\Pi_{k=\tilde{k}+1}^{L} M_k} \right\rfloor. \tag{5}$$

The rest of the messages could potentially go outside of the subtree rooted in μ. Therefore, the maximum bandwidth (in terms of number of messages) necessary on the upward link starting on any node on level $\tilde{k}$ is:

$$B_{max}(\tilde{k}) = \Pi_{\tilde{k}} - \left\lfloor \frac{M_1 \cdot M_2 \cdot ... \cdot M_{\tilde{k}}}{M_{\tilde{k}+1} \cdot M_{\tilde{k}+2} \cdot ... \cdot M_L} \right\rfloor \tag{6}$$

which exactly equals the value we previously found for the minimum bandwidth theoretically necessary in a fat tree for contention-free all-to-all messaging.

The per-phase bandwidth requirement per downward link can be computed analogously. During a given phase, at most $\Pi_{\tilde{k}}$ messages in total are sent to the $\Pi_{\tilde{k}}$ level 0 descendants of a node μ of the fat tree on level $\tilde{k}$. It can be shown with an argument analogous to the one above that among these messages, a minimum number must originate from these same descendants, and that that minimum number is equal to $N_{min}(\tilde{k})$ in Equation (5). As such, by the same argument as before, we obtain the same upper bound as that given by Equation (6) for the bandwidth required on the downward links.

3.5 Discussion

In this section, we have derived tight lower bounds on the bandwidth requirement of the all-to-all exchange in fat tree networks. We have also introduced an optimal exchange pattern that uses no more than the minimal bandwidth.

This new exchange pattern allows for a better use of network resources, freeing bandwidth for other traffic if the tree was designed with bandwidth in excess of the minimal bound, or, alternatively, allowing for reduced-cost fat trees that perform optimally under the all-to-all.

Compared to a standard fat tree that offers full bisection bandwidth at every level, the amount of bandwidth that is left unused, or, alternatively can be removed altogether from the tree is given, for a layer l link, by $\Pi_l - B_{\min}(l)$. This bandwidth surplus is largest at the top of the tree and decreases rapidly with the layer index. Nonetheless, at the highest level we can leave unused as much as $\frac{1}{M_L}$ of the full bisection bandwidth. Given certain conditions (specifically $M_L = 2$), this means we can achieve optimal all-to-all traffic with only half the full bisection bandwidth.

Because the amount of redundant bandwidth at lower layers of the tree decreases exponentially, in practice only a few top levels will allow significant reduction of bandwidth usage while some lower levels will allow no reduction at all. This, coupled with the fact that the fully optimized pattern as described above is highly intricate, leads to an interest in performing the optimization only at the levels where reduction occurs. We have developed methods similar to the one presented above to tune the complexity of the pattern while maintaining optimality but these will be discussed in a subsequent work.

4. PRACTICAL CONSIDERATIONS

In this section we will address some issues that arise from the fact that, in practice, fat tree networks are not implemented by means of ideal fat trees. This is because in practice the exponential increase in link bandwidth that is typical of multi-layered fat trees can only be achieved by means of having an exponentially increasing number of links between a single pair of nodes. This in turn leads to a necessity of switches with a very large port count, which are not feasible.

Therefore, in practice, fat tree topologies are approximated by means of k-ary n-trees [28, 27] or more generally by means of Extended Generalized Fat Trees (XGFTs) [26]. These network designs offer an alternative that only needs fixed small-radix switches and still provides many of the properties of ideal fat trees. However, the advantages come at the expense of reduced flexibility in bandwidth configuration and, more importantly, at the expense of needing more complicated routing algorithms.

The following two subsections briefly outline how the routing needs to be adapted to take full advantage of the optimal exchange pattern and provide an estimate of the cost saving that can be achieved by reducing the bandwidth to the theoretical lower bound for optimal all-to-all exchanges.

4.1 Routing in XGFTs

In practical fat tree realizations, optimizing the exchange pattern is not sufficient to benefit from contention-free traffic. The optimal exchange pattern introduced in Section 3.4 only ensures that, in each phase, the *aggregate* bandwidth at each fat-tree-equivalent-node of the network matches the aggregate number of messages traversing that node.

This is a necessary but not a sufficient condition for contention-free message routing. In addition, the routing strategy must ensure that the messages indeed use the available bandwidth evenly, i.e., it must assign conflict-free paths for all messages in each phase.

In XGFT networks, the typical way to establish minimal deadlock-free paths is simple up*/down* routing [33]. Given a (source, destination) pair, paths are constructed by routing upwards to any of the Nearest Common Ancestor (NCA) nodes and then downwards to the destination. Once a specific NCA is chosen, the routing choices from source to NCA and from NCA to destination are uniquely determined. The specificity of a certain routing strategy then lies in the approach used to select a specific NCA for every (source, destination) pair.

Several traffic-pattern-oblivious approaches exist, from random [10, 7] or adaptive [8, 20] route selection to deterministic (e.g., modulo based) approaches such as *S*ource-*mod-k* [26, 17, 14] or *D*estination-*mod-k* [18, 31, 41].

For complex patterns such as ours however, oblivious routing can prove insufficient. A pattern-aware routing approach takes advantage of information about the communicating pairs in a network and information about the network itself and uses it to find an optimized set of routes that will minimize contention for the set of (source, destination) pairs. Pattern-aware schemes solve in some way or another a *max-flow* network problem. A variety of approaches to achieve this are presented by Kinsy et. al [15]. Among these approaches, we were able to derive optimal conflict-free routes for the bandwidth-optimal exchange that we are proposing by means of Mixed Integer-Linear Programming.

4.2 Cost effective fat tree networks

Since the optimized exchange exhibits contention free all-to-all phases in reduced bandwidth topologies, we consider the network cost savings that this exchange allows.

Given an XGFT($h : m_1, m_2, ..., m_h : w_1, w_2, ..., w_h$) [26], the number of switches necessary to build the layer l of the topology is $S(l) = w_1 \cdot ... \cdot w_l \cdot m_{l+1} \cdot ... \cdot m_h$. The number of bidirectional links that connect layer l to layer $l-1$ is $E(l) = S(l) \cdot m_l$.

To simplify the analysis, consider a simple cost reduction scenario. Let's start with a full bisection bandwidth XGFT (i.e., where $w_{l+1} = m_l, 1 \leq l < h$ and $w_1 = 1$) that additionally has $m_h = 2$. This is the case we have shown to have the highest potential for bandwidth reduction. The reduced tree, XGFT$'(h : m_1, m_2, ..., m_h : w_1, w_2, ..., w'_h)$, has exactly the same parameters as the full bisection bandwidth tree, with the exception of w'_h which equals $w_h/2$.

Given the properties shown above, we can see that, for $1 \leq l < h$, $E(l) = E'(l) = E = w_1 \cdot m_1 \cdot ... \cdot m_h$ and $S(l) = S'(l) = E/m_l$. For $l = h$ we have $E(h) = E$, $E'(h) = w_1 \cdot m_1 \cdot ... \cdot m_h/2 = E/2$, $S(h) = E/m_h$ and $S'(h) = E/(2m_h)$.

The saving potential in the number of switches equals

$$1 - \frac{\sum_{l=1}^{h} S'(l)}{\sum_{l=1}^{h} S(l)} = 1 - \frac{1/(2m_h) + \sum_{l=1}^{h-1} 1/m_l}{1/m_h + \sum_{l=1}^{h-1} 1/m_l}$$

and in the number of links equals

$$1 - \frac{\sum_{l=1}^{h} E'(l)}{\sum_{l=1}^{h} E(l)} = 1 - \frac{(h - 1/2) \cdot E}{h \cdot E} = \frac{1}{2h}.$$

For typical XGFTs interconnecting 16 to 1024 leaf nodes, we can thus *achieve a reduction in the number of switches of* $25\% - 30\%$ *and a reduction in the number of necessary links of* $12\% - 16\%$.

5. EXPERIMENTS

The following results were obtained by means of a simulation framework that is able to accurately model custom networks (including XGFTs) at a flit level [22]. The simulator provides a high level of customization and modularity, allowing the configuration of the desired model in detail. Several tests have been performed on XGFTs of 16, 32, 64, 128, 256, 512 and 1024 leaf nodes and up to 4 non-leaf levels. All these fat tree topologies had the M parameter of the topmost level equal to two ($M_L = 2$) and half bisection bandwidth. Table 1 shows the exact configurations used.

The simulations were performed using output-buffered switches with 4 kbytes of buffer space per port. The links had a bandwidth of 10 Gbit/s while their latency was either

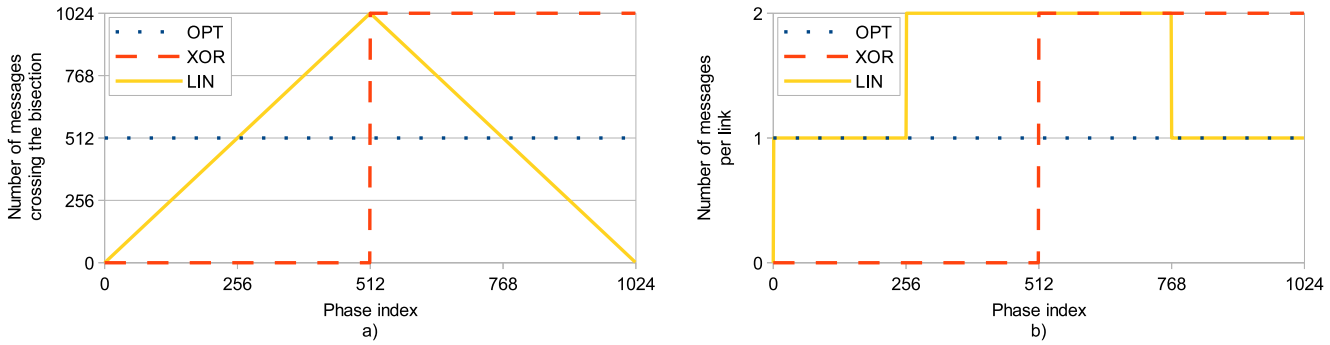

Figure 4: Bisection (top level) occupation induced by XOR, LIN and OPT exchange patterns. The topology is an XGFT with $M_L = 2$ with half bisection bandwidth and total number of leaf nodes equal to 1024. Figure a) shows the total number of messages crossing the bisection in each phase, while Figure b) shows the best value for maximum link contention (in number of messages), or best contention level, that can be obtained in each phase.

N	Network topology
16	$XGFT[3:4,2,2:1,4,1]$
32	$XGFT[3:4,4,2:1,4,2]$
64	$XGFT[3:8,4,2:1,8,2]$
128	$XGFT[3:8,8,2:1,8,4]$
256	$XGFT[4:8,4,4,2:1,8,4,2]$
512	$XGFT[4:8,8,4,2:1,8,8,2]$
1024	$XGFT[4:8,8,8,2:1,8,8,4]$

Table 1: Benchmarked topological configurations, using the notation from [26].

set to an ideal value of zero (to validate theoretical estimations) or to a realistic value of 100 ns (to estimate performance on real-life systems). In this same realistic latency scenario, switch traversal was considered to have a latency of 50 ns while adapter latency (the time between a message is issued in the MPI library and the time its first bit leaves the adapter) was set to 500 ns. Credit based flow control was used as well as wormhole switch traversal.

The messages have a constant size which, depending on the scenario, took values between 64 bytes (a single flit) and 32 kbytes, while the flit size was fixed to 64 bytes. Each acknowledgement consisted of a single flit.

The traffic pattern is that of a single all-to-all collective exchange performed by one task on each of the leaf nodes of the XGFT. The exchange is split into consecutive phases in each of which every node sends a single message and receives a single message and acknowledges it. No explicit synchronization or separation of the phases is enforced. The exact structure of each phase (the set of (source, destination) pairs) is defined according to one of three exchange models, all defined in Sections 2 and 3.4:

- XOR: the binary XOR model,
- LIN: the linear shift model with a shift value of 0,
- OPT: the optimized exchange given by Equation (4).

The metric we used is that of the time necessary for the traffic pattern to complete. However, these completion times can amount to very different values when looking at different sized networks and even at the same network with messages of different size. We are not interested in how these parameters influence the completion time, but rather in the influence of the exchange pattern within each given (network size, message size) configuration. As such, all results show the relative completion time compared to an ideal completion time that we will define in the following.

5.1 Ideal all-to-all performance

We use the following model to compute a best-case expectation for the completion time of the all-to-all exchange. Ideally, the messages travel through the network with no contention and simply incur the cumulative latency of the links, switches and adapters that they traverse. During the exchange, a given source sends $M_1 \cdot M_2 \cdot ... \cdot M_{l-1} \cdot (M_l - 1)$ messages to sources in its layer l subtree but not in its layer $l-1$ subtree, for every l. These messages need to first arrive at the root of the subtree, passing through 1 adapter, l links and $l-1$ switches, then cross the root, then travel again through 1 adapter, l links and $l-1$ switches to get to their destination. The cumulative latency for the entire path is thus $t_{\text{path}}(l) = 2 \cdot t_{\text{adapter}} + (2 \cdot l - 1) \cdot t_{\text{switch}} + 2 \cdot l \cdot t_{\text{link}}$.

Given a message that is made up of F flits, each of size S, and a link bandwidth of B, uncontended wormhole switch traversal ensures that the time it takes end-to-end, from message generation to message delivery, equals $t_{\text{path}}(l) + F \cdot S/B$.

The message needs to be acknowledged, which is equivalent to an extra one-flit message being sent along the reverse path. This leads to the total message time being: $T(l) = 2 \cdot t_{\text{path}}(l) + (F+1) \cdot S/B$.

Given the message distribution we described above, the ideal exchange time is given by

$$T = \sum_{l=1}^{L} M_1 \cdot M_2 \cdot ... \cdot M_{l-1} \cdot (M_l - 1) \cdot T(l)$$

5.2 Ideal latency analysis and measurements

To validate our simulation framework, we analyze the network contention induced by the three approaches when other performance limiting factors such as latency and latency-induced phase overlapping are factored out.

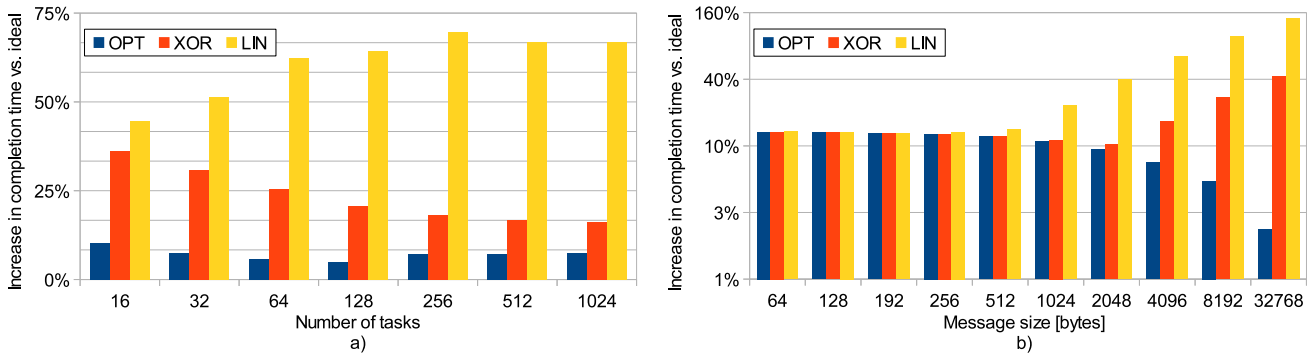

Figure 5: Completion time comparison between the three exchange patterns for varying networks sizes under fixed 4 KB message size (Fig. a) and for varying message size under fixed 512 leaf nodes network size (Fig. b). Please note the linear scale for the y axis in a) and the logarithmic scale for the y axis in b).

The XOR exchange exhibits no contention in half of its phases, namely in the phases where messages do not reach the top level of the fat tree. In the other half, N messages cross the top level, but have only half bisection bandwidth available, leading to a contention level of exactly two conflicts in every one of these $N/2$ phases. We defined the contention level as being the maximum number of messages that need to use a common link.

In the case of the LIN exchange, exactly $N - |2p - N|$ messages cross the top level of the tree in phase p, leading to a contention level of 2 yet again in exactly $N/2$ phases (the phases where $N - |2p - N| > N/2$).

In the case of the OPT exchange, exactly $N/2$ messages cross the top level of the tree in every phase, leading to uncontended traffic in every phase (Figure 4).

This implies that, in absence of latency and latency-induced phase overlapping, we can estimate theoretically that OPT will have a completion time close to the optimum, while LIN and XOR will incur a doubling of the completion time of half the phases, leading to at least a 50% increase in all-to-all completion time. This is confirmed accurately by our simulator, which yields completion times for OPT within 1% of the ideal time and completion times for XOR $50\% - 55\%$ longer than ideal, both irrespective of the message size. LIN performs even worse, with completion times between 70% and 140% larger than ideal (the performance loss increases with both the network and the message size). This is due to additional phase overlapping induced by intra-phase differences in message delivery time caused by different flows exhibiting different levels of contention (which is not the case for OPT and XOR).

5.3 Realistic latency results

We performed two sets of benchmarks to predict OPT performance on real systems. The first one is aimed at studying the influence of the network size on the performance gain induced by the optimized exchange pattern. The second on the other hand studies the influence of the size of the exchanged messages on the same performance gain.

Figure 5 a) illustrates the relative completion time of the three exchange patterns for a fixed message size of 4 kbytes and for different network sizes. We observed that in the realistic latencies configuration OPT managed to achieve completion times less than 10% higher than ideal, surpassing XOR (which is between 15% and 35% worse than ideal) and LIN (which is between 50% and 70% worse than ideal). The advantage of the optimized exchange versus XOR was especially strong for smaller topologies.

Figure 5 b) illustrates the relative completion time of the three exchange patterns for a fixed number of 512 communicating tasks (and a fixed network size) and for different sizes of the exchanged messages. We observed that for messages smaller than approximately 512 bytes, where the completion time is latency dominated, the optimization of the exchange doesn't help too much, because the impact of contention was small. For messages larger than this threshold, where the completion time was congestion dominated, OPT achieved an increasingly closer to ideal level of performance (reaching only a $2\% - 5\%$ difference for very large messages), while the non-optimized exchanges became progressively worse as the message size increased, reaching as much as a 40% performance decrease for XOR and as much as 140% for LIN, for very large messages. This confirms that our approach is asymptotically bandwidth optimal.

6. RELATED WORK

Collective optimizations are increasingly important in large-scale high-performance computing. Thus, numerous research works deal with the optimization of collective operations in general and all-to-all in particular. Most traditional approaches assume theoretical network models, such as the Postal model or the LogP model. Well-tuned algorithms exist for both models [4, 1]. However, all theoretical models (with the notable exception of LoGPC [24], which has not been used to model all-to-all algorithms due to its complexity), do not consider the network topology and thus only optimize endpoint congestion. Our work, however, uses a simple topology-specific model for optimizing all-to-all communications. Our model can be combined with existing network models to model endpoint congestion more accurately and can also be used as a blue-print for optimizing other collectives on fat trees. However, our contribution lies in the scheduling algorithm rather than the model.

Topology becomes more important with growing system size and several researchers acknowledge the fact that algorithms purely based on the mentioned theoretical models do not perform well on real systems. For example, Chan et al. [5] provide a general abstract framework for topology-aware collective communication. However, the abstract nature of this framework makes it hardly applicable to the recursive structure of fat trees. Zahavi et al. [42, 40] show that a particular routing function (a variant of up*/down*) guar-

antees under certain placement conditions full-bandwidth all-to-all exchanges implemented using typical collective permutation sequences on full bisection bandwidth fat trees. We improve upon this result by a factor of two, in that our methods applied to networks with half the full bisection bandwidth exhibit no or negligible decrease in performance by comparison. Sack and Gropp present a small set of topology-aware collectives for fat tree and torus networks in [32], however, in this latest work, they do not consider all-to-all and their approach of adding communication does not lead to improvements for all-to-all. In previous work [35], targeted at the implementation of optimized collectives in MPICH, the authors proposed the selection among a pool of implementations of different collective algorithms depending on message size and number of communicating processes, but not on any specific topology.

Other approaches for optimizing collectives, for example, improving the lower-level send/receive primitives [19], non-blocking collectives [12, 11], or hierarchical exchanges [13, 34] are completely orthogonal to the scheduling of messages. Those techniques can use our message-scheduling algorithm for improving the bandwidth of all-to-all.

7. CONCLUSIONS

The goal of this paper was to show that the current state of the art exchange patterns for the personalized all-to-all collective message exchange can be optimized in fat tree topologies to the point of using half of the bisection bandwidth required by previous proposals. To this end, we have derived a theoretical bound on the network link occupation that is intrinsically necessary and sufficient for any all-to-all exchange and we have shown that current approaches require as much as twice that proven optimally minimum. Furthermore, we introduced a message exchange pattern that we demonstrated to be bandwidth-optimal and that thus uses exactly the minimum required amount of network resources. Finally, by conducting network simulations benchmarking the proposed method against established approaches, we confirmed that in practical scenarios the optimizations we proposed achieve important reductions in network utilization, or alternatively network complexity and cost (by reducing the top level of the network such that only half the bisection bandwidth remains available), with little impact on performance. Furthermore, we demonstrated that in the half-bisection network scenario, where a cost-effective network with only the minimum amount of necessary resources was used, current methods incurred significant performance penalties, (in the 12% to 40% range for XOR and in the 12% to 140% range for LIN) whereas our method only diverged by at most 12% from optimal performance.

In conclusion, our method enables new design options for fat tree networks and offers a viable way of maintaining close to ideal application performance levels with important reductions in cost.

8. REFERENCES

[1] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman. LogGP: incorporating long messages into the LogP model - one step closer towards a realistic model for parallel computation. In *Proceedings of the 7^{th} annual ACM symposium on Parallel algorithms and architectures (SPAA)*, pages 95–105, NY, USA, 1995. ACM.

[2] G. Almasi, P. Hargrove, I. Gabriel, and T. Y. Zheng. UPC collectives library 2.0. In 5^{th} *Conference on Partitioned Global Address Space Programming Models*, 2011.

[3] P. Balaji, D. Buntinas, D. Goodell, W. Gropp, T. Hoefler, S. Kumar, E. Lusk, R. Thakur, and J. L. Traeff. MPI on Millions of Cores. *Parallel Processing Letters (PPL)*, 21(1):45–60, Mar. 2011.

[4] J. Bruck, C.-T. Ho, E. Upfal, S. Kipnis, and D. Weathersby. Efficient algorithms for all-to-all communications in multiport message-passing systems. *IEEE Trans. Parallel Distrib. Syst.*, 8(11):1143–1156, Nov. 1997.

[5] E. Chan, M. Heimlich, A. Purkayastha, and R. van de Geijn. On optimizing collective communication. In *Cluster Computing, 2004 IEEE International Conference on*, pages 145 – 155, sept. 2004.

[6] R. Fiedler. Preparing Applications for Sustained Petascale Performance, 2011.

[7] J. Flich, M. P. Malumbres, P. López, and J. Duato. Improving routing performance in Myrinet networks. In *Proc. of the 14^{th} International Parallel and Distributed Processing Symposium*, pages 27–32, Los Alamitos, CA, USA, 2000. IEEE Computer Society.

[8] P. Geoffray and T. Hoefler. Adaptive routing strategies for modern high performance networks. In *High Performance Interconnects, 2008. HOTI '08. 16th IEEE Symposium on*, pages 165–172, Aug. 2008.

[9] S. Gorlatch. Send-receive considered harmful: Myths and realities of message passing. *ACM Trans. Program. Lang. Syst.*, 26(1):47–56, Jan. 2004.

[10] R. I. Greenberg and C. E. Leiserson. Randomized routing on fat-trees. In *Proc. of the 26^{th} Annual Symposium on the Foundations of Computer Science*, pages 241–249, 1985.

[11] T. Hoefler and A. Lumsdaine. Optimizing non-blocking collective operations for InfiniBand. In *IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008*, pages 1 –8, 2008.

[12] T. Hoefler, A. Lumsdaine, and W. Rehm. Implementation and performance analysis of non-blocking collective operations for MPI. In *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, SC '07*, pages 1 –10, nov. 2007.

[13] P. Husbands and J. C. Hoe. MPI-StarT: delivering network performance to numerical applications. In *Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM)*, Supercomputing '98, pages 1–15, Washington, DC, USA, 1998.

[14] H. Kariniemi. *On-Line Reconfigurable Extended Generalized Fat Tree Network-on-Chip for Multiprocessor System-on-Chip Circuits*. PhD thesis, Tampere University of Technology, 2006.

[15] M. A. Kinsy, M. H. Cho, T. Wen, E. Suh, M. van Dijk, and S. Devadas. Application-aware deadlock-free oblivious routing. In *Proceedings of the 36th annual international symposium on Computer architecture*, ISCA '09, pages 208–219, New York, NY, USA, 2009. ACM.

[16] C. Kurmann, F. Rauch, and T. M. Stricker. Cost/performance tradeoffs in network interconnects for clusters of commodity PCs. In *Proceedings of the*

17th International Symposium on Parallel and Distributed Processing, IPDPS '03, pages 196.2–, Washington, DC, USA, 2003. IEEE Computer Society.

[17] C. Leiserson et al. The network architecture of the Connection Machine CM-5. In *Proc. of the Fourth Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 272–285, San Diego, CA, USA, June 1992.

[18] X.-Y. Lin, Y.-C. Chung, and T.-Y. Huang. A multiple LID routing scheme for fat-tree-based InfiniBand networks. *Proc. of the* 18^{th} *International Parallel and Distributed Processing Symposium*, pages 11–, 2004.

[19] A. Mamidala, R. Kumar, D. De, and D. Panda. MPI collectives on modern multicore clusters: Performance optimizations and communication characteristics. In *Cluster Computing and the Grid, 2008. CCGRID '08. 8th IEEE International Symposium on*, pages 130 –137, may 2008.

[20] J. C. Martínez, J. Flich, A. Robles, P. López, and J. Duato. Supporting fully adaptive routing in InfiniBand networks. In *Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, IPDPS '03, pages 44.1–, Washington, DC, USA, 2003. IEEE Computer Society.

[21] J. Mellor-Crummey, L. Adhianto, W. N. Scherer, III, and G. Jin. A new vision for coarray fortran. In *Proceedings of the* 3^{th} *Conference on Partitioned Global Address Space Programing Models*, PGAS '09, pages 5:1–5:9, New York, NY, USA, 2009. ACM.

[22] C. Minkenberg, W. Denzel, G. Rodriguez, and R. Birke. End-to-end modeling and simulation of high-performance computing systems. *Springer Proceedings in Physics: Use Cases of Discrete Event Simulation: Appliance and Research*, page 201, 2012.

[23] P. Moin and K. Mahesh. Direct numerical simulation: a tool in turbulence research. *Annual Review of Fluid Mechanics*, 30(1):539–578, 1998.

[24] C. A. Moritz and M. I. Frank. LoGPC: modeling network contention in message-passing programs. *SIGMETRICS Perform. Eval. Rev.*, 26(1):254–263, June 1998.

[25] MPI Forum. MPI: A Message-Passing Interface Standard. Version 3.0, September 2012.

[26] S. R. Öhring, M. Ibel, S. K. Das, and M. J. Kumar. On generalized fat trees. In *Proceedings of the* 9^{th} *International Parallel Processing Symposium*, page 37, Washington, DC, USA, 1995. IEEE Computer Society.

[27] F. Petrini and M. Vanneschi. A comparison of wormhole-routed interconnection networks. In *Proc.* 3^{th} *International Conference on Computer Science and Informatics*, NC, USA, Mar. 1997.

[28] F. Petrini and M. Vanneschi. k -ary n -trees: High performance networks for massively parallel architectures. *IPPS*, 00:87, 1997.

[29] J. Pješivac-Grbović, T. Angskun, G. Bosilca, G. Fagg, E. Gabriel, and J. Dongarra. Performance analysis of mpi collective operations. *Cluster Computing*, 10(2):127–143, 2007.

[30] S. Ranka, J.-C. Wang, and G. C. Fox. Static and run-time algorithms for all-to-many personalized communication on permutation networks. *IEEE Trans. Parallel Distrib. Syst.*, 5(12):1266–1274, Dec. 1994.

[31] C. G. Requena, F. G. Villamón, M. E. Gómez, P. López, and J. Duato. Deterministic versus adaptive routing in fat-trees. *Proc. of the* 21^{st} *Parallel and Distributed Processing Symposium, 2007*, pages 1–8, Mar. 2007.

[32] P. Sack and W. Gropp. Faster topology-aware collective algorithms through non-minimal communication. In *Proceedings of the 17th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming*, PPoPP '12, pages 45–54, New York, NY, USA, 2012. ACM.

[33] M. Schroeder, A. Birrell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterthwaite, and C. Thacker. Autonet: a high-speed, self-configuring local area network using point-to-point links. *Selected Areas in Communications, IEEE Journal on*, 9(8):1318 –1335, oct 1991.

[34] S. Sistare, R. vandeVaart, and E. Loh. Optimization of MPI collectives on clusters of large-scale SMP's. In *Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)*, Supercomputing '99, New York, NY, USA, 1999. ACM.

[35] R. Thakur, R. Rabenseifner, and W. Gropp. Optimization of collective communication operations in MPICH. *IJHPCA*, 19(1):49–66, 2005.

[36] L. G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, Aug. 1990.

[37] J. J. Willcock, T. Hoefler, N. G. Edmonds, and A. Lumsdaine. Active pebbles: parallel programming for data-driven applications. In *Proceedings of the international conference on Supercomputing*, ICS '11, pages 235–244, New York, NY, USA, 2011. ACM.

[38] M. Xie, Y. Lu, L. Liu, H. Cao, and X. Yang. Implementation and evaluation of network interface and message passing services for TianHe-1A supercomputer. In *Proceedings of the 2011 IEEE 19th Annual Symposium on High Performance Interconnects*, HOTI '11, pages 78–86, Washington, DC, USA, 2011. IEEE Computer Society.

[39] W. Yu, D. K. Panda, and D. Buntinas. Scalable, high-performance nic-based all-to-all broadcast over Myrinet/GM. In *Proceedings of the 2004 IEEE International Conference on Cluster Computing*, CLUSTER '04, pages 125–134, Washington, DC, USA, 2004. IEEE Computer Society.

[40] E. Zahavi. Fat-trees routing and node ordering providing contention free traffic for MPI global collectives. In *IPDPS Workshops*, pages 761–770. IEEE, 2011.

[41] E. Zahavi, G. Johnson, D. J. Kerbyson, and M. Lang. Optimized InfiniBand fat-tree routing for shift all-to-all communication pattern. In *International Supercomputing Conference (ISC07)*, Dresden, Germany, June 2007.

[42] E. Zahavi, G. Johnson, D. J. Kerbyson, and M. Lang. Optimized InfiniBand(TM) fat-tree routing for shift all-to-all communication patterns. *Concurr. Comput. : Pract. Exper.*, 22(2):217–231, Feb. 2010.