Fast Pattern-Specific Routing for Fat Tree Networks

BOGDAN PRISACARI, IBM Research
GERMAN RODRIGUEZ, IBM Research
CYRIEL MINKENBERG, IBM Research
TORSTEN HOEFLER, ETH Zurich

In the context of extended generalized fat tree (XGFT) topologies, widely used in HPC and datacenter network designs, we propose a generic method, based on integer linear programming (ILP), to efficiently determine optimal routes for arbitrary workloads. We propose a novel approach that combines ILP with dynamic programming, effectively reducing the time to solution. Specifically, we divide the network into smaller subdomains optimized using a custom ILP formulation that ensures global optimality of local solutions. Local solutions are then combined into an optimal global solution using dynamic programming. Finally, we demonstrate through a series of extensive benchmarks that our approach scales in practice to networks interconnecting several thousands of nodes, using a single-threaded, freely available linear programming solver on commodity hardware, with the potential for higher scalability by means of commercial, parallel solvers.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms: Algorithms, Design, Theory, Performance, Experimentation

Additional Key Words and Phrases: Extended generalized fat trees, integer linear programming, optimal routing

ACM Reference Format:
Bogdan Prisacari, German Rodriguez, Cyriel Minkenberg, and Torsten Hoefler, 2013. Fast pattern-specific routing for fat tree networks. *ACM Trans. Architec. Code Optim.* V, N, Article A (January YYYY), 25 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. MOTIVATION

The suitability of an interconnection network design is typically quantified via simple metrics such as link bandwidth/latency and bisection bandwidth. However, these metrics lead to optimistic estimations on the achievable performance, the latter depending on the exact combination of communication patterns and routing algorithm being used. Inefficient routing approaches can induce significant levels of network congestion, which will negatively impact network performance, in terms of both throughput and latency.

Several characteristics of network topologies determine the options for routing algorithms. An important feature of any interconnection network is path diversity. Path diversity denotes the number of different ways to route messages between a given source and destination and consequently quantifies the breadth of the space of possible routing algorithms and the difficulty of choosing a good algorithm. An intelligent

Author's addresses: B. Prisacari, G. Rodriguez, and C. Minkenberg, IBM Research, Zurich; T. Hoefler, ETH Zurich.
New Paper, Not an Extension of a Conference Paper.

DOI:http://dx.doi.org/10.1145/0000000.0000000

choice of the path that each message will follow will have important benefits, e.g. load balancing and congestion reduction, and will translate to an increase in workload performance. On the other hand, a suboptimal choice can introduce artificial performance limitations that are neither inherent to the workloads being accommodated nor to the network itself.

Some of the most popular choices for the interconnection fabric of large scale systems are variants of the fat tree architecture: fat trees [Leiserson et al. 1992], k-ary n-trees [Petrini and Vanneschi 1997b] or generic extended generalized fat trees (XGFTs) [Öhring et al. 1995]. In the enterprise datacenter space, fat trees are the network architecture of choice [Al-Fares et al. 2008] whereas in the high performance computing (HPC) space they are by far the dominating topology employed in clusters using Infiniband technology, clusters which account for more than 40% of the systems in the latest Top 500 list [Bogdanski et al. 2010; Top 2013]. A few prominent examples of current HPC installations using fat tree interconnects are TACC Stampede [(TACC) 2011], LRZ SuperMUC , and Pangea , the world's fastest private supercomputer, which are ranked as numbers 6, 9 and 11 in the latest Top 500 list [Top 2013], respectively.

All fat tree architectures are characterized by an amount of path diversity that increases roughly linearly with the size of the network, making the route selection problem both very important and very challenging. Currently, there is—to the best of our knowledge—no general solution to the problem of determining optimal routes for arbitrary workloads running on arbitrary fat trees of reasonable size. In practice, approximate solutions obtained either heuristically or through dynamic or adaptive approaches, where the routing decision is dependent on the current state of the network, are used to approximate optimal behavior and performance. However, for certain workloads such approaches can prove to be insufficient.

This work was originally triggered by a very specific example, described in detail in [Prisacari et al. 2013]. There, we tackled the problem of optimizing the communication pattern of the *all-to-all exchange* collective operation, encountered in many parallel programming models, for XGFT networks with less then full bisection bandwidth. Having established a theoretically optimal way of performing such an all-to-all exchange, we found that optimizing a communication pattern for a specific network topology was not sufficient: although the workload and the network guaranteed optimal performance, none of the existing routing strategies were able to sustain it. In other words, the need arose for a practical and scalable method capable of finding optimal routes for a given communication pattern and a given (XGFT) topology. This work fills this gap, providing a solution for arbitrary communication patterns on arbitrary XGFT networks..

The core idea of our method is to perform offline a highly optimized search over all possible routing algorithms, using Integer Linear Programming (ILP). Although this is not a new idea, previous attempts have failed in applying it to real systems and concluded that the method is NP-complete and impractical for all but the smallest network sizes [Elmallah and Culberson 1994]. This paper present a set of key contributions that enable us to find optimal routes for networks interconnecting several thousands of nodes in a practically feasible amount of time (in the range of hours).

After an initial survey of related work already performed in the field (Section 2), we start by introducing a novel topological characterization of XGFT networks (Section 3) that will enable us to construct an efficient ILP formulation of the optimal routing assignment problem, for arbitrary workloads (Section 4). We proceed to explain how this solution can be effectively used in practice and present a set of experimental results that illustrate the benefits that using this approach can bring in a practical scenario (Section 5). Finally, we demonstrate (Section 6) by means of a set of extensive benchmarks that our approach is able to generate optimal routing assignments using only commodity hardware and an open source, free, single-threaded linear programming solver, in a reasonable amount of time, for networks interconnecting thousands of nodes. Thus, although a polynomial time-to-solution is not guaranteed, we show that through a series of acceleration strategies (introduced in Section 4) our approach reduces the completion time to very manageable values (e.g., in the order of one hour for a network interconnecting a thousand nodes), making, to the best of our knowledge, the current work the first to achieve rendering an ILP-based optimal routing assignment generation approach practical for arbitrary communication patterns in production systems.

2. PREVIOUS WORK ON LINEAR PROGRAMMING DRIVEN ROUTE OPTIMIZATION

Finding an optimal routing for a specific communication matrix and network topology is equivalent to the well-known maximum flow problem, or its generalization, the multi-commodity flow problem [Srinivasan 1997; Leighton et al. 1991]. For the general problem, considering any directed graph, and allowing for *fractional* flows, non-integral linear programming formulations exist that are solvable in polynomial time. However, for integer flows the problem is NP-complete [Even et al. 1976].

Several works exist that do not seek to obtain the optimum integer solution but rather use the fractional solution as a basis for close-to-optimal routes and then provide upper bounds on the expected difference to an actual optimal solution. Räcke [Raecke 2002], for example, showed that it is possible to find an *oblivious* routing for any symmetric network (undirected and with equal capacities in both directions), such that any traffic pattern would only experience a maximum of a poly-logarithmic increment in contention with respect to the optimum for that specific traffic pattern. Azar [Azar et al. 2003] later corrected the bound to $O(N^{0.5})$, with N being the number of communicating nodes, and developed a linear programming formulation that can be solved with the Ellipsoid algorithm to obtain a solution in polynomial time. The proposed LP formulation, however, grows exponentially [Applegate and Cohen 2003] with respect to the network size. Applegate [Applegate and Cohen 2003] later provided an LP formulation that is polynomial in size with respect to the network size. Building on Räcke's and Azar's work, Applegate further showed that a robust routing can be obtained with little knowledge of the traffic demands. For the sample Internet Server Provider (backbone) networks studied, optimizing the oblivious case (where all source-destination pairs are considered with equal probability) was shown to be significantly more robust than optimizing for specific traffic patterns.

A more recent work [Kinsy et al. 2009] takes an additional step towards the integral max-flow problem by using a Mixed Integer LP formulation to find a set of routes for *unsplittable* flows, i.e., flows that cannot be separated across different paths. However, the solution is still not completely integral, as the flows, although unsplittable, may still be fractional. Furthermore, the author himself states that even this partial integer optimization is only feasible from a practical point of view for "problems of small size".

However, none of these works solve the actual integral max-flow problem. It might be argued that the fractional solution can be made to approximate the integral solution by using a combination of segmentation (breaking the message into smaller pieces) and *dynamic* (sprayed) routing when the messages are large enough to be subdivided into a sufficient number of smaller units. However, such a solution will still only be an approximation of the optimal routing and it will require additional hardware support (such as larger routing tables and re-sequencing queues) and introduce new problems (such as out of order arrivals and segmentation overheads). Furthermore, when a *contention-free* routing solution exists for a certain topology and traffic pattern, an approximate routing solution that has even a single unnecessary conflict will incur a noticeable performance degradation with respect to the optimum, as the communications competing for the same link will share the bandwidth of that link [Rodriguez et al. 2009a; Rodriguez et al. 2008]. These contention-caused delays will propagate throughout the entire execution, because subsequent transfers, especially in HPC applications, may depend—directly or through a chain of dependencies—on a given delayed transfer. This causes increased jitter, a well-known cause of severe performance degradation [Petrini et al. 2003; Hoefler et al. 2009].

Other works do attempt to solve the ILP problem, but they generally conclude that the approach is not feasible for practical network sizes. A notable work that falls in this category and provides an ILP formulation specifically for multi-stage networks is due to Elmallah [Elmallah and Culberson 1994]. The paper shows that, although a polynomial-time algorithm can be found for multi-stage networks having up to three levels, the problem is NP-complete for networks with more than three levels. Hence, no attempts were made to find ways of making the optimization process practically feasible.

Other works have steered away from the NP-complete integer linear programming and have concentrated on trying to find optimal routes with polynomial-time algorithms for specific networks and/or traffic patterns. For example, in the context of Clos networks there exist optimal simple routing algorithms when very specific conditions are met [Jajszczyk 2003]. However, no such results have been obtained for extended generalized fat trees.

The method we propose is radically different from previous approaches in that:

— the optimum is computed outside of the linear optimization process and the optimality is embedded in the feasibility conditions (LP constraints), such that the ILP solver serves only as a feasible solution identifier as all feasible solutions are optimal;
— the optimal routes are not computed for the entire network at once; instead, the problem is split up into smaller sub-problems that can be solved efficiently using ILP, and the optimal local solutions are subsequently combined using dynamic programming into an optimal global solution.

Finally, this is to our knowledge the first work presenting a method to solve the integral optimal routing problem for arbitrary workloads that can claim practically feasible completion times for XGFT networks with thousands of nodes and up to 6 levels.

3. EXTENDED GENERALIZED FAT TREES

As we have shown in the motivation section, fat trees are one of the most popular indirect network topologies used in the design of current HPC systems as well as datacenter networks.

The fat tree family encompasses a broad class of different interconnection layouts which can all be described as multi-stage tree-like topologies where the bandwidth of the links may increase towards the root of the tree.

An *ideal fat tree* is a k-ary tree interconnection network where the bandwidth of each link towards the root increases in powers of k. The CM-5 [Leiserson et al. 1992] was the first machine to implement a variant of the ideal fat tree network. More generically, a *full-bisection bandwidth* fat tree is a tree interconnection network in which every link can always accommodate the aggregate capacity of the sub-tree connected to that link. Such fat tree networks are not realizable for large clusters, as the bandwidth needs to increase exponentially at each layer towards the root, which would require switches with either exponentially increasing number of ports or exponentially increasing bandwidth per link.

However, it is possible to construct a tree-like network that exhibits an equivalent full bisection bandwidth property and uses fixed-capacity switches, i.e., does not require increasing either the number of ports per switch or the port bandwidth [Petrini and Vanneschi 1997b; 1997a].

Furthermore, it is possible to construct networks in which the bandwidth towards the root can be tuned to almost arbitrary values, from the full-bisection bandwidth of ideal fat trees to reduced bandwidth realizations that have a correspondingly reduced cost. These are called *Extended Generalized Fat Trees* or *XGFTs*. Öhring et al. [Öhring et al. 1995] provide a generic framework for compactly describing arbitrary XGFT topologies. In this framework, an XGFT network is completely described by two values for each tree layer, namely an M value, describing the number of direct descendants of a node at a specific layer, and a W value, describing the number of direct ancestors of a node at a specific layer. As such, the network is completely described by a set of parameters of the form XGFT$(H : M_1, ..., M_H; W_1, ..., W_H)$, where the parameter H corresponds to the number of switch layers of the tree. An XGFT$(H : M_1, ..., M_H; W_1, ..., W_H)$ is conceptually equivalent to a standard fat tree with downward branching factors $(M_1, ..., M_H)$ and upward bandwidths available to every conceptually equivalent fat tree node at every layer equal to $(W_1, W_1 \cdot W_2, ..., W_1 \cdot ... \cdot W_H)$.

3.1. A new XGFT representation

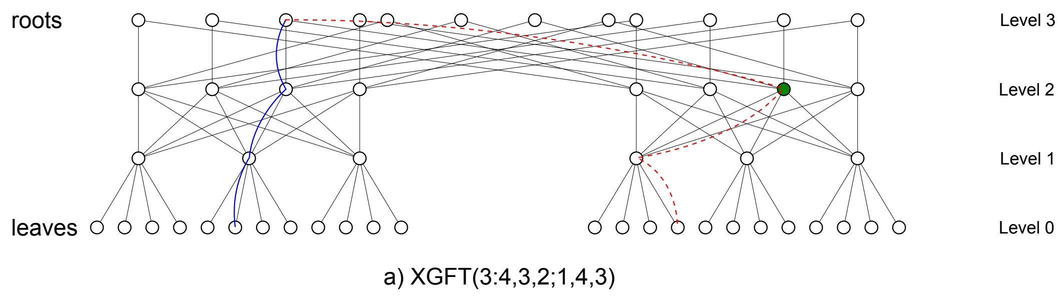
First, we introduce an alternative way of describing the XGFT topology that will enable an easier formulation and understanding of the routing optimization approach in Section 4. Instead of viewing the XGFT as before as a single, monolithic network layout, we dissociate it into two complementary aspects:

— On the one hand an XGFT is meant to conceptually mirror a regular tree where every node has a single ancestor, with the exception of the unique root, which has no ancestor.
— On the other hand, an XGFT is meant to provide a practical way of ensuring increasing bandwidth towards the root using (relatively) small radix switches. As the number of ports per switch is limited, the large fat-tree-specific bandwidth that is required of upper layer switch-to-switch connections cannot be achieved via a multitude of links. The XGFT design solves this problem by replacing the conceptual exponential-radix fat-tree switch with an exponential (towards the root) multitude of constant-radix switches. We will show that this separation of the conceptual switches can be described by a second tree that is regular but *inverted* (the root layer of this second tree corresponds to the leaf layer of the XGFT).

In the remainder of this section we will show how these two aspects are completely described by a single regular tree each, and how the XGFT itself can be described via this dual view.

Given an XGFT$(H : M_1, ..., M_H; W_1, ..., W_H)$ as described above, we consider two simple trees $MT(H : M_1, ..., M_H)$ and $WT(H : W_H, ..., W_1)$, each with the same number of layers $H + 1$ as the original XGFT. For the former, we number the layers starting with 0 from the leaf layer in increments of 1 (Figure 1b), whereas for the latter we number the layers starting with 0 from the root layer also in increments of 1 (Figure 1c). The parameters M_l of MT represent the number of descendants of each MT node on layer l, whereas the parameters W_l of WT represent the number of descendants of each WT node on layer $l - 1$.

The bijective correspondence between the standard XGFT characterization and this new dual characterization is the following. A layer l node in the XGFT is equivalent to a pair of nodes, one at layer l of MT and one at layer l of WT. The MT node is reached



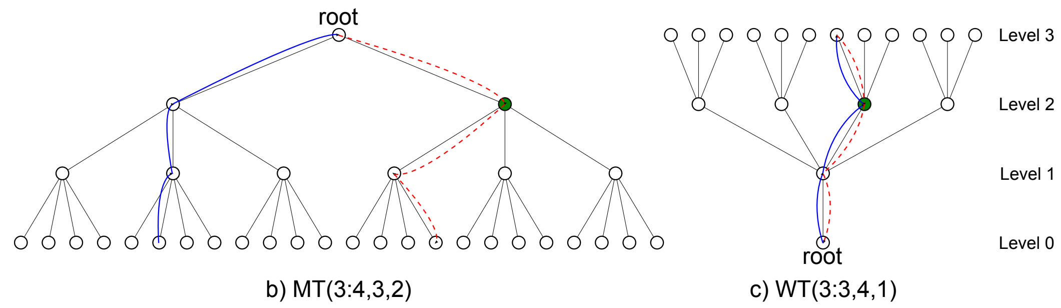


Fig. 1. Standard (Figure a) and dual (Figures b and c) representation for a 3-layer network: XGFT(3:4,3,2;1,4,3). The dual representation is composed of two simple trees, the MT tree (Figure b) and the WT tree (Figure c). The layers of the WT tree are numbered in the reverse order of that of the XGFT and MT tree. The MT tree captures the scalability properties of the fat tree that the XGFT is conceptually modeling, i.e., it captures the downward branching factors of the fat tree, with no information on the bandwidth available at each layer. The WT tree expresses how many links each node has available to the next layer (in numbering order), which equals the number of upward links for a given node in the original XGFT. Every XGFT node on layer l, n^l, corresponds to a unique pair of nodes, one in the MT tree and one in the WT tree, also at layer l in their corresponding trees: (mt^l, wt^l). The filled nodes exemplify such a correspondence. The figures also show an example path between a source-destination pair as they are reflected by both representations. The upward path is shown in a continuous line (in blue), whereas the downward path is shown dashed (in red). In the MT tree, the path between a fixed source and destination is unique. In the WT tree, the upward path and the downward path are identical, but any path connecting the root to any node on the layer where the downward path begins is a valid one.

from the unique root of the MT tree by choosing the same links moving towards lower index layers as the links that need to be chosen to reach the XGFT node when starting on any root ancestor of that node. Similarly, the WT node is reached from the unique root of the WT tree by choosing the same links moving towards higher index layers as the links that need to be chosen to reach the XGFT node when starting on any leaf descendant of that node.

Similarly to how every XGFT node on layer l is assigned a set of identifying coordinates $(l : m_H, ..., m_{l+1}, w_l, ..., w_1)$, with $0 \leq w_i < W_i$ and $0 \leq m_i < M_i$, we assign to every node on layer l in the trees MT and WT a set of coordinates $(l : m'_H, ..., m'_{l+1})$ and $(l : w'_1, ..., w'_l)$ respectively, by simply making a list of the branching choices required starting on the root of the tree to get to the node in question. Then formally, the pair of (MT,WT) nodes corresponding to an XGFT node with coordinates $(l : m_H, ..., m_{l+1}, w_l, ..., w_1)$ are the nodes with coordinates $(l : m_H, ..., m_{l+1})$ and $(l : w_1, ..., w_l)$ in their respective MT and WT trees.

Figure 1 illustrates the correspondence via the nodes that are filled. The layer 2 XGFT node that is filled corresponds to two filled nodes in the dual representation. To reach the XGFT node starting from the root layer, one would choose the rightmost link of any ancestor root, while to reach it starting from the leaf layer, one would choose first the only link available, then the second rightmost link. This choice is unique and allows the determination of the MT and WT corresponding nodes.

A link in the XGFT is represented in the dual representation via a pair of links, one belonging to MT and one belonging to WT, which are the exact links that connect the pairs of MT and WT nodes that form the dual representations of the end points of the original link in the XGFT.

Since the bijection applies to links as well as nodes, it also applies to the paths taken by messages in the XGFT. Routing can thus be seen as a synchronized movement in the two trees of the dual representation, between the MT components of the source and destination on one hand and between the WT components of the source and destination on the other. An example path in both representations is shown in Figure 1, with the upward portion of the path being shown as a continuous blue line and the downward portion as a dashed red line.

The dual representation clearly exposes the XGFT routing-related properties. Specifically, whereas in the MT tree there exists a unique route between any given source and destination, routing in the WT tree is equivalent to starting from the root, descending into the tree for a number of layers, then returning to the root. Thus, it is obvious that when moving away from the root of WT (equivalent to moving "up" in the XGFT), the number of possible ways we have to descend into the WT tree is equal to $W_1 \cdot W_2 \cdot ... \cdot W_l$, where l is the maximum descent layer, while when moving back towards WT's root (equivalent to moving "down" in the XGFT), there are no choices to be made, i.e., the path is completely determined by the choices made while descending. This property of the downward path completely being determined by the upward path is one of multiple intrinsic properties of XGFT routing which become immediately obvious in the dual representation, but are not exposed explicitly by the standard characterization.

Another example of such a property, which we will use in our approach and which, with the standard XGFT model, can require significant effort to prove [Ding et al. 2006] while being obvious in the dual representation, is the following. If during the upward part of an up-down path, on a node on layer l, the choice was made to route on the upwards link w_l, then on the downward path the message will enter the node on layer l on link w_l as well. In the dual representation, because the upward link choice is determined solely by the WT tree trajectory, and because in that tree we are taking the same path first moving away from the root and then in reverse, it is immediate that the same links will be used.

Summarizing, we have the following properties:

PROPERTY 3.1. *In an XGFT, given a message that moves from its source upwards into the tree up to layer l then descends to its destination, the total number of available paths that message can choose equals $W_1 \cdot W_2 \cdot ... \cdot W_l$.*

PROPERTY 3.2. *In an XGFT, given a message that moves from its source upwards into the tree up to layer l then descends to its destination, the message having already performed the upward portion of its path, then the downward path of the message is unique (i.e., completely determined).*

PROPERTY 3.3. *In an XGFT, given a message from a source leaf node to a destination leaf node, the message having advanced from layer l to layer $l + 1$ using the w-th upwards link from the node it was on at layer l, then on its downward path, the message will descend from layer $l+1$ to layer l using, in reverse direction, the w-th upwards link on the node it will reach on layer l.*

4. LINEAR PROGRAMMING OPTIMIZED ROUTING

The exact problem we are solving is the following. We are considering a fixed communication load that needs to be delivered using the network as a medium. In the most general case, this means that every source node S needs to send a message to every destination in a set (with repetition) D_S. The set D_S can be empty or can contain the same destination multiple times, the latter case corresponding to the situation where source S sends multiple messages to a given destination.

Messages traveling through the network can use a variety of paths. Given an assignment of a single path to each message in a workload, we define the contention level induced by the workload on an arbitrary link as the total number of assigned paths that contain that link. The problem we aim to solve is finding a path-to-message assignment that ensures that the maximum contention level across all links in the network is minimized.

This is a problem that has relevance regardless of the network type. However, for arbitrary workloads and arbitrary networks, finding such an optimum assignment is typically not possible via a heuristic approach, making it typically necessary to perform an exhaustive search of the assignment space, which is large enough to make such an approach impractical. Furthermore, even for specific classes of networks but arbitrary workloads, exhaustive search might ultimately still be necessary to obtain an optimal solution.

Nonetheless, in the latter case, an optimized exhaustive search that takes advantage of the structure of the network has potential to yield a solution in a practical time.

In the remainder of this section, we will present an approach to achieve just that in the case of arbitrary XGFT networks. We will use an integer linear programming solver as the means to perform the search (thus benefiting from the large amount of optimizations done in this field) and provide it with a specification of the routing problem that ensures rapid convergence.

4.1. Layer-by-layer route optimization

The main problem in performing an exhaustive search on the path-to-message assignment space is that the space in question is extremely large. Indeed, given a workload composed of N messages and a path diversity for any given source-destination pair of P, then the total number of possible assignments is P^N.

In an XGFT with the specification $XGFT(H : M_1, ..., M_H; W_1, ..., W_H)$, we have that $P_l = W_1 \cdot ... \cdot W_l$ for messages needing to go up in the tree up to layer l before descending back to the destination. As an example, in a small 1024-node XGFT ($XGFT(4 : 16, 8, 4, 2; 1, 16, 8, 4)$) where every node needs to send a single message to another node across the root (which is a case that is typical in practice, for example in many of the phases of an all-to-all exchange), the size of the search space is $512^{1024} \approx 10^{2700}$.

For this reason, our approach divides the network into smaller optimization domains and propagates solutions from one domain to the next, by using a dynamic programming approach. This reduces the assignment space that needs to be explored, which in turn reduces search time. However, to allow the method to work correctly, as with any dynamic programming approach, we must ensure that any solution that is found for a sub-domain and is propagated as a starting point for the next sub-domain is a partial solution of an optimum total solution.

In the case of the XGFT network, a natural way of partitioning the topology is on a per-layer basis. Specifically, messages will first be assigned routing decisions that allow them to move from layer 0 to layer 1 of the tree. Then, all messages that need to go up past layer 1 will be propagated in the network according to the previously chosen assignment and a new assignment will be chosen that allows them to go from layer 1 to layer 2, and so on up to the root.

With such an approach, the total search space is reduced to a succession of per-layer search spaces each of a size of approximately W_{l+1}^{N}, where N is the number of messages needing to go up past layer l and W_{l+1} is the XGFT W parameter at layer l. Going back to our previous example, this reduces the search space to a maximum size of $16^{1024} \approx 10^{1200}$, significantly smaller than for the original approach. The difficulty, once more, lies in ensuring the global optimality of the union of sub-solutions.

4.2. Single-layer problem formulation

The problem we want to solve is the following. Given a workload as described above, a layer l of the XGFT and a set of routing assignments for the messages in the workload for layers 0 through $l-1$, we want to generate a set of routing assignments for layer l.

The first thing to notice is that we can use the lower layer routing assignments to generate a "position" in the network for every message reaching layer l. Thus, every node at layer l will be assigned a set of messages, some of which need to be routed downwards, others routed upwards. Property 3.2 ensures that messages that are routed downwards have their entire path already decided. Therefore, routing assignments need only be generated for messages routed upwards.

The output of our algorithm must thus be, for every message that is to be routed upwards of layer l, a choice among the W_{l+1} upward links available to that message on the node it is on. This choice needs to:

(1) fulfill a local optimality criterion for the links going upwards at the current layer;
(2) ensure that the locally optimal solution is part of a globally optimal solution;
(3) since the choices made going upwards also permanently determine the downward path and since we know what links those messages will use going down, at the same layer (according to Properties 3.2 and 3.3) the current choice must also ensure Properties 1 and 2 for the downward path.

The latter property actually requires that two optimization criteria be pursued simultaneously (one for messages on their upward path and the other for the same messages on their downward path) and this is actually what makes the optimization process very difficult to achieve by other means than an exhaustive search.

4.3. Upward local optimality criterion

Given a set of n messages present on a given node of the tree at layer l that need to move to layer $l+1$ and given W_{l+1} choices in terms of links those messages can move on, the minimum achievable contention level across the links is clearly $\lceil n/W_{l+1} \rceil$. To ensure not only a minimum maximum contention level across links but also a complete load balancing of traffic, we also need to impose a lower limit on the number of messages routed on any given link of $\lfloor n/W_{l+1} \rfloor$. To understand why this is necessary, let us take the example of $n = 7$ messages needing to be distributed across $W = 3$ links. The minimum achievable contention level across the links is $\lceil n/W \rceil = 3$. However, without the strict load balancing lower limit, one acceptable assignment would be $3, 3, 1$, that is, having two links assigned 3 messages each and the third link assigned a single message. This would cause the third link to receive much fewer messages than the others. A more balanced assignment would be $3, 2, 2$. In general, a perfect load balancing of messages across links is ensured by any assignment where the difference (in number of messages assigned) between the lowest utilized link and the highest utilized link is at most 1 message. The lower bound mentioned above guarantees this property. En-

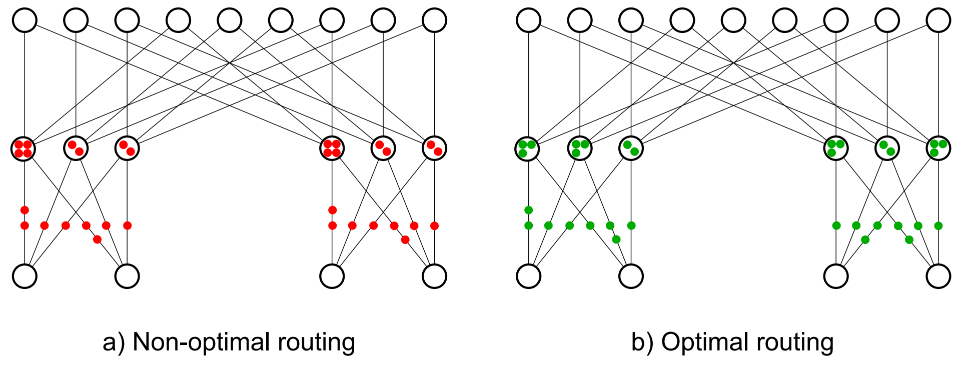


Fig. 2. Example of two route assignments for an $XGFT(2:2,2;3,3)$ network and a workload consisting of 4 messages, shown as red dots in Figure a) and as green dots in Figure b), sent from every leaf node source to leaf node destinations across the root. Both assignments are optimal for the lowest layer of links, however the assignment in Figure a) is non-optimal for the second layer (the leftmost second layer node for example has 4 messages to distribute among 3 links) while the assignment in Figure b) is optimal.

forcing it is done however at the expense of reducing the density of feasible solutions in the search space. This is the only local optimality criterion.

4.4. Global optimality preserving criteria

First, let us illustrate through an example why the local optimality conditions are insufficient, in that a locally optimal partial solution to the route assignment problem can prove to not constitute a part of a globally optimal solution. Let's take the $XGFT(2:2,2;3,3)$ illustrated in Figure 2 and let's assume a workload where every leaf node needs to send 4 messages across the root. The locally optimal upper bound for the number of messages that cross each upwards link at layer 0 is $\lceil n/W_1 \rceil$ which, given that $n = 4$ and $W_1 = 3$ is 2. One solution (illustrated in Figure 2a) that is locally optimal at layer 0 is to assign, on every node, 2 of the 4 messages to link 0 and 1 message to each of the other two upward links. Propagating the messages to layer 1, this would lead to 4 of the 6 layer 1 nodes receiving 2 messages while the remaining 2 nodes at that layer would receive 4 messages each. Given that the number of upward links at layer 1 is $W_2 = 3$, this leads to a contention level of at least 2 (the latter two nodes have to distribute 4 messages across 3 links). However, there exists a different, optimal route assignment (illustrated in Figure 2b) that induces a contention level on layer 1 upward links of only 1 (this can be achieved for example by having leaf node i, where nodes are numbered consecutively from left to right, assign 2 messages to upward link $i \mod W_1$ and 1 message to each of the remaining 2 upwards links). We have thus shown that the local optimality criterion is insufficient.

Let us now derive in the general case the optimality preserving criteria for the global solution. Given two messages at layer l, on different XGFT nodes, four conditions need to be fulfilled in order for those two messages to contend for a link at layer $l + \lambda$:

(1) the messages need to start their downstream portion of the path at a layer strictly larger than $l + \lambda$
(2) the messages need to eventually propagate to the same node in the MT tree at layer $l + \lambda$, which is equivalent to them being on layer l nodes in the MT tree that have a common ancestor at layer $l + \lambda$
(3) the messages need to eventually propagate on the same node in the WT tree at layer $l + \lambda$, which is equivalent to them being on the same layer l node in the WT tree and them picking exactly the same w-s at layers l through $l + \lambda - 1$

(4) the messages need to pick the same link $w_{l+\lambda+1}$ at layer $l+\lambda$

However, we do not know the routing assignments of the messages at layers above l. Hence, we must ensure that in the case of an optimum assignment for layers $l+1$ through $l+\lambda$, our current layer l assignment still allows for optimality for layer $l+\lambda$. The messages at layer l that, with the knowledge we currently have, can potentially contend on layer $l+\lambda$ are those messages that descend at a layer strictly greater than $l+\lambda$, that have a common ancestor in the MT tree at layer $l+\lambda$, and that are on the same WT tree node at layer l. Let's denote, for every pair $(mt^{l+\lambda}, wt^{l})$ of an MT tree ancestor at layer $l+\lambda$ and a WT tree node at layer l, the number of messages that can potentially contend at layer $l+\lambda$ as being $c^{u}_{\lambda,mt^{l+\lambda},wt^{l}}$ (the "u" standing for "upwards").

An assignment at layer l separates these messages into W_{l+1} different groups, where contention can only appear within each individual group (i.e., among messages for which the routing assignment allocated the same upward link index). The local restriction ensures only that the number of messages that have potential to contend post-assignment (i.e., the number of messages in every group) is upper bounded by $\sum_{\{mt^{l}|\text{ancestor}(mt^{l})=mt^{l+\lambda}\}}\lceil c^{u}_{0,mt^{l},wt^{l}}/W_{l+1}\rceil$. The optimal assignment however is one where the $c^{u}_{\lambda,mt^{l+\lambda},wt^{l}}$ messages that will reach past layer $l+\lambda$ are assigned evenly between groups. The condition that enforces the optimum assignment is thus that the number of messages in a group be upper bounded by $\lceil c^{u}_{\lambda,mt^{l+\lambda},wt^{l}}/W_{l+1}\rceil$.

Once more, we can additionally impose a load balancing restriction by lower bounding the same number of messages by $\lfloor c^{u}_{\lambda,mt^{l+\lambda},wt^{l}}/W_{l+1}\rfloor$. These optimality preserving restrictions need to be enforced for every λ up to the root and for each given λ, for every $mt^{l+\lambda}$.

4.5. Improving search convergence

Strong restrictions, such as the ones above, allow in principle a more aggressive pruning of invalid partial solutions during the optimization process. However, in practice we have found that past a certain point they will also diminish the density of feasible solutions in the restriction of the search space they define, making the overall completion time of the search higher.

Hence, there is potential to improve the completion time by providing slightly weaker restrictions, that are nonetheless optimality preserving (they ensure the same maximum contention level) at the expense of inducing a load across links that might be less than ideally balanced.

The local criterion is not only sufficient but also necessary, so no restriction relaxation is possible in its case. However, weaker conditions can be formulated when ensuring global optimality.

The strong restriction states that the size of a group of messages choosing the same upward link must be upper bounded by $\lceil c^{u}_{\lambda,mt^{l+\lambda},wt^{l}}/W_{l+1}\rceil$. This is equivalent to ensuring that messages that have potential to contend at layer $l+\lambda$ will be split as evenly as possible across the W_{l+1} upward links available at layer l. Looking at the WT tree, one can clearly see that once having reached layer $l+1$, each of these groups of messages will be split further, again evenly (due to the local optimality criterion that will be enforced in the next layer), into W_{l+2} groups, and so on up to layer $l+\lambda$. So all in all, the set of messages in one of the original layer l groups will have potential to be split into $W_{l+2}\cdot\ldots\cdot W_{l+\lambda+1}$ groups until contending at layer $l+\lambda$. Therefore, if that group contained n messages, the contention level those messages will induce at layer $l+\lambda$ will be $\lceil n/(W_{l+2}\cdot\ldots\cdot W_{l+\lambda+1})\rceil$.

The strong restriction imposes an upper limit on what n can be. However, we can see that increasing n from that limit to the first multiple of $(W_{l+2} \cdot \ldots \cdot W_{l+\lambda+1})$ actually has no influence on the contention level induced at layer $l + \lambda$.

Consequently, we can replace the strict upper bound derived above by $\lceil c^u_{\lambda,mt^{l+\lambda},wt^l}/W_{l+1}/(W_{l+2} \cdot \ldots \cdot W_{l+\lambda+1})\rceil \cdot (W_{l+2} \cdot \ldots \cdot W_{l+\lambda+1})$. We will call this acceleration technique *restriction relaxation*.

A second method to improve the completion time of the optimized exhaustive search is to remove superfluous restrictions. We mentioned above that the local restriction ensures by itself an upper bound of $\sum_{\{mt^l|\text{ancestor}(mt^l)=mt^{l+\lambda}\}} \lceil c^u_{0,mt^l,wt^l}/W_{l+1}\rceil$ for the number of messages in a group. If this limit is lower than or equal to either the strong upper bound or the upper bound obtained through restriction relaxation, then we can eliminate the strong/relaxed restriction altogether. We will call this acceleration technique *restriction elimination*.

The applicability of these techniques is dependent on both the structure of the topology and that of the workload so it might not always be possible to accelerate the search using these approaches. However, when it is possible, the resulting routing assignment ensures an optimal maximum contention level across links, at the possible expense of having less-than-perfect balancing of load across links.

4.6. Downward path restrictions

Until now we have only discussed about enforcing the load induced by messages moving towards the root of the tree. However, the routing decisions taken going upwards completely determine the downward path as well due to Properties 3.2 and 3.3). Ensuring an optimal maximum contention level on the downward links is equivalent to considering them as upwards links in a time-reversed message flow, where messages depart from their destination and finish at their source, and then applying the same restrictions to the exhaustive search as above.

Within the optimization step of a single layer, both the downward and upward conditions must be enforced simultaneously to ensure an overall optimum maximum link contention level.

The remainder of this section formalizes the approach described thus far.

4.7. Formal description of the optimization constraints

A message $m(id, s, d, \delta, l, mt_u, wt_u, mt_d, wt_d, w^{id}_{l+1})$ at layer l is characterized by:

— a unique id id
— its source node index s
— its destination node index d
— the layer at which is starts its downward path δ
— the current layer l
— the MT node and WT node indices at layer l on the upward path mt_u and wt_u
— the MT node and WT node indices at layer l on the downward path mt_d and wt_d
— the choice at layer l w^{id}_{l+1}

The first 4 fields are computed at the beginning and remain unchanged for the duration of the routing assignment generation across layers. The rest of the fields are initialized and updated as follows.

Initialization:

— $mt_u = s$
— $wt_u = 0$
— $mt_d = d$

— $wt_d = 0$
— $l = 0$

Update when moving to the next optimization layer, for messages that need to move upwards at that layer:

— $mt_u = ancestor(mt_u)$
— $wt_u = descendant(wt_u, w_{l+1}^{id})$
— $mt_d = ancestor(mt_d)$
— $wt_d = descendant(wt_d, w_{l+1}^{id})$
— $l = l + 1$

The optimization step at an arbitrary layer l receives as input the set of messages M that have advanced to that layer (each message missing the w_{l+1}^{id} parameter) and needs to output for each message $m(id, ...)$ the link choice w_{l+1}^{id}.

We introduce the following notations. For a given l, a given λ, a given node $mt^{l+\lambda}$ in the MT tree at layer $l+\lambda$ and a given node wt^l in the WT tree at layer l, we denote by $M^u_{\lambda,mt^{l+\lambda},wt^l}$, independent of the routing choice at layer l, the set of messages that:

— reached layer l moving upwards,
— begin their downward path at a layer strictly greater than $l+\lambda$,
— have propagated moving upwards to layer l on a node that has node $mt^{l+\lambda}$ as an ancestor in the MT tree,
— and have propagated moving upwards to layer l on node wt^l in the WT tree.

Furthermore, we denote the subset of messages in $M_{\lambda,mt^{l+\lambda},wt^l}$ that have their routing choice equal to w_{l+1} where $0 \leq w_{l+1} < W_{l+1}$, by $M^u_{\lambda,mt^{l+\lambda},wt^l,w_{l+1}}$.

We introduce analogous notations for the downward path: $M^d_{\lambda,mt^{l+\lambda},wt^l}$ and $M^d_{\lambda,mt^{l+\lambda},wt^l,w_{l+1}}$.

Formally:

$$M^u_{\lambda,mt^{l+\lambda},wt^l} = \{m(id, s, d, \delta, l, mt_u, wt_u, mt_d, wt_d, w_{l+1}^{id})| \\ \delta > (l+\lambda), ancestor^{\lambda}(mt_u) = mt^{l+\lambda}, wt_u = wt^l\} \quad (1)$$

$$M^u_{\lambda,mt^{l+\lambda},wt^l,w_{l+1}} = \{m(id, s, d, \delta, l, mt_u, wt_u, mt_d, wt_d, w_{l+1}^{id})| \\ m \in M^u_{\lambda,mt^{l+\lambda},wt^l}, w_{l+1}^{id} = w_{l+1}\} \quad (2)$$

$$M^d_{\lambda,mt^{l+\lambda},wt^l} = \{m(id, s, d, \delta, l, mt_u, wt_u, mt_d, wt_d, w_{l+1}^{id})| \\ \delta > (l+\lambda), ancestor^{\lambda}(mt_d) = mt^{l+\lambda}, wt_d = wt^l\} \quad (3)$$

$$M^d_{\lambda,mt^{l+\lambda},wt^l,w_{l+1}} = \{m(id, s, d, \delta, l, mt_u, wt_u, mt_d, wt_d, w_{l+1}^{id})| \\ m \in M^d_{\lambda,mt^{l+\lambda},wt^l}, w_{l+1}^{id} = w_{l+1} \quad (4)$$

With these notations, we then have that:

$$c^u_{\lambda,mt^{l+\lambda},wt^l} = |M^u_{\lambda,mt^{l+\lambda},wt^l}|. \quad (5)$$

and the strong condition for the upward path then is:

$$\forall\lambda \geq 0, \forall mt^{l+\lambda}, \forall wt^{l}, \forall 0 \leq w_{l+1} < W_{l+1}, |M^{u}_{\lambda,mt^{l+\lambda},wt^{l},w_{l+1}}| \leq \lceil c^{u}_{\lambda,mt^{l+\lambda},wt^{l}}/W_{l+1}\rceil, \quad (6)$$

where the local optimality criterion corresponds to $\lambda = 0$ and the global optimality conditions correspond to $\lambda > 0$.

Similarly, given the notation:

$$c^{d}_{\lambda,mt^{l+\lambda},wt^{l}} = |M^{d}_{\lambda,mt^{l+\lambda},wt^{l}}|, \quad (7)$$

the strong condition for the downward path is:

$$\forall\lambda \geq 0, \forall mt^{l+\lambda}, \forall wt^{l}, \forall 0 \leq w_{l+1} < W_{l+1}, |M^{d}_{\lambda,mt^{l+\lambda},wt^{l},w_{l+1}}| \leq \lceil c^{d}_{\lambda,mt^{l+\lambda},wt^{l}}/W_{l+1}\rceil \quad (8)$$

Should we wish to enforce strict load balancing, two more conditions need to be added: one for the upward path:

$$\forall\lambda \geq 0, \forall mt^{l+\lambda}, \forall wt^{l}, \forall 0 \leq w_{l+1} < W_{l+1}, |M^{u}_{\lambda,mt^{l+\lambda},wt^{l},w_{l+1}}| \geq \lfloor c^{u}_{\lambda,mt^{l+\lambda},wt^{l}}/W_{l+1}\rfloor, \quad (9)$$

and one for the downward path:

$$\forall\lambda \geq 0, \forall mt^{l+\lambda}, \forall wt^{l}, \forall 0 \leq w_{l+1} < W_{l+1}, |M^{d}_{\lambda,mt^{l+\lambda},wt^{l},w_{l+1}}| \geq \lfloor c^{d}_{\lambda,mt^{l+\lambda},wt^{l}}/W_{l+1}\rfloor \quad (10)$$

Should we wish to enforce the relaxed restrictions, the conditions change as follows. For the upward path:

$$\forall\lambda \geq 0, \forall mt^{l+\lambda}, \forall wt^{l}, \forall 0 \leq w_{l+1} < W_{l+1},$$
$$|M^{u}_{\lambda,mt^{l+\lambda},wt^{l},w_{l+1}}| \leq \lceil c^{u}_{\lambda,mt^{l+\lambda},wt^{l}}/W_{l+1}/(W_{l+2} \cdot ... \cdot W_{l+\lambda+1})\rceil \cdot (W_{l+2} \cdot ... \cdot W_{l+\lambda+1}), \quad (11)$$

and for the downward path:

$$\forall\lambda \geq 0, \forall mt^{l+\lambda}, \forall wt^{l}, \forall 0 \leq w_{l+1} < W_{l+1},$$
$$|M^{d}_{\lambda,mt^{l+\lambda},wt^{l},w_{l+1}}| \leq \lceil c^{d}_{\lambda,mt^{l+\lambda},wt^{l}}/W_{l+1}/(W_{l+2} \cdot ... \cdot W_{l+\lambda+1})\rceil \cdot (W_{l+2} \cdot ... \cdot W_{l+\lambda+1}), \quad (12)$$

4.8. Routing as a linear programming problem

As explained earlier in this section, we model the routing problem as an exhaustive search in the routes-to-messages assignment space. The size of the space we need to explore is immense and as such an optimized approach that is able to prune infeasible partial solutions very early is critical. In order to benefit from existing work, we model the routing problem as a linear optimization problem with binary variables and use an independently optimized linear programming solver to produce the solution.

To ensure as aggressive a pruning of the search space as possible, our method embeds the optimality of the solution not in the optimization function of the linear programming specification but in the feasibility constraints themselves. This way, any feasible solution is an optimum solution and any non-optimum solution is eliminated implicitly.

The problem formulation for layer l uses multiple binary variables named using the form $var_{l,id,w_{l+1}}$. If the solution contains a value of 1 for $var_{l,id,w_{l+1}}$, it signifies that the message with the identifier id should be routed on the upward link with index w_{l+1} at layer l. For a given linear optimization problem, l is fixed, id is the identifier of any message with $\delta > l$ and w_{l+1} takes values between 0 and $W_{l+1} - 1$.

In order to express the restrictions defined above as linear programming constraints, it is sufficient to express the values of

— $M^u_{\lambda,mt^{l+\lambda},wt^l}$
— $M^d_{\lambda,mt^{l+\lambda},wt^l}$
— $|M^u_{\lambda,mt^{l+\lambda},wt^l,w_{l+1}}|$
— $|M^d_{\lambda,mt^{l+\lambda},wt^l,w_{l+1}}|$

in terms of the variables introduced above.

The former two are not dependent on the optimization process and their exact values can be precomputed in a straightforward fashion from the distribution of messages across nodes at layer l, which is an input to the optimization process. Once these sets are established, the number of elements in each set is also readily available. The latter two can be expressed as:

$$|M^u_{\lambda,mt^{l+\lambda},wt^l,w_{l+1}}| = \sum_{m(id,\ldots)\in M^u_{\lambda,mt^{l+\lambda},wt^l}} var_{l,id,w_{l+1}} \quad (13)$$

$$|M^d_{\lambda,mt^{l+\lambda},wt^l,w_{l+1}}| = \sum_{m(id,\ldots)\in M^d_{\lambda,mt^{l+\lambda},wt^l}} var_{l,id,w_{l+1}} \quad (14)$$

This allows us to express all restrictions previously mentioned as an ILP problem. However, none of these restrictions actually enforce a property that is not related to the optimization but is intrinsic to routing itself: a message cannot be routed on more or less than one link at any given layer. This is equivalent to:

$$\forall id, \sum_{0\le w_{l+1}<W_{l+1}} var_{l,id,w_{l+1}} = 1 \quad (15)$$

Finally, as the optimality of the solution is embedded in the feasibility constraints, no optimization function is provided as part of the ILP formulation.

5. PRACTICAL RELEVANCE

In this section, we delve into some of the practical aspects and relevance of our approach. In the first part, we describe the exact steps that need to be undertaken to effectively use the method in practice. In the second part, we present how we successfully applied our approach to obtain optimal routes for an actual, relevant and challenging HPC workload that was provably optimum for the target topology, but for which existing state-of-the-art routing strategies for fat tree networks were not able to sustain the optimal level of performance. We demonstrate that in conjunction with our proposed pattern-aware route optimization, the workload was able to achieve the expected levels of performance and overall system performance was improved beyond the state of the art.

5.1. Usage of the method

The method is designed to be used in practice as follows: To achieve maximum performance, the routing optimization algorithm needs to be made aware of the exact network configuration that is targeted as well as of the workload or mix of workloads to be optimized for. While the structure of the network is typically fixed and known before-hand, a workload characterization step might be necessary to extract the patterns in the communication steps of the targeted applications. Alternatively, synthetic traffic patterns that are known to approximate typical application mixes can also be used.

The workload and network specification are then utilized to generate the routing decisions, using an offline iterative process with as many steps as layers there are in the network, as follows. For each layer of the network, starting with the lowest, an ILP formulation is generated according to the procedure detailed in Section 4. The resulting problem specification is supplied as input to an ILP solver (in our benchmarks, detailed in Section 6; we used the solver included in the freely available open-source framework GLPK [Makhorin 2013]). The solution produced by the solver is translated into a routing assignment for the current layer using once more the procedure detailed in Section 4. The routing assignment is then used to generate the workload as it propagates to the next layer and we move on to the next iteration.

The end result of this process is the complete set of routing decisions. This set is then transferred to the network, typically via programming of the routing tables.

5.2. Practical use case

Table I. Benchmarked XGFT configurations.

Number of leaf nodes	Topology specification
16	$XGFT(2:8,2:1,4)$
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	$XGFT(4:8,8,4,2:1,8,8,2)$
1,024	$XGFT(4:16,8,4,2:1,16,8,2)$
2,048	$XGFT(5:8,8,4,4,2:1,8,8,4,2)$
4,096	$XGFT(5:8,8,8,4,2:1,8,8,8,2)$
8,192	$XGFT(5:8,8,8,8,2:1,8,8,8,4)$

Each topology is specified using the the $XGFT(H : M_1, M_2, ..., M_H : W_1, W_1, W_H)$ notation introduced by [Öhring et al. 1995]. $H + 1$ is the number of network layers, the list of M parameters defines the number of descendants a node at every layer has, while the list of W parameters characterizes the bandwidth available at each layer in the tree and is equal to the number of ancestors a node has on every layer.

The need for a generic optimal routes generator emerged in the process of developing and evaluating theoretically optimal communication patterns for all-to-all message exchanges over slimmed fat tree networks.

The all-to-all message exchange is a communication pattern performed by concurrent tasks where every task needs to exchange a single distinct message with every other task. Due to the large amount of traffic it generates (the number of messages scales quadratically with the number of communicating tasks), it is one of the most challenging communication patterns to optimize and scale. The fact that it is the core message exchange in many relevant applications [Sur et al. 2004] makes its optimization all the more important. Indeed, all-to-all is the core message exchange in two of the NAS Parallel Benchmarks kernels [Bailey et al. 1995]: FT (FFT) and IS (Integer Sort), but more notably required in several physics simulations working in the spectral domain requiring distributed transpose operations, such as in GYRO (time-dependent, non linear Gyro-Kinetic-Maxwell (GKM) equations solver) [Alam and Vetter 2006], as well as full physics simulations of Global Atmospheric Models using a Double Fourier

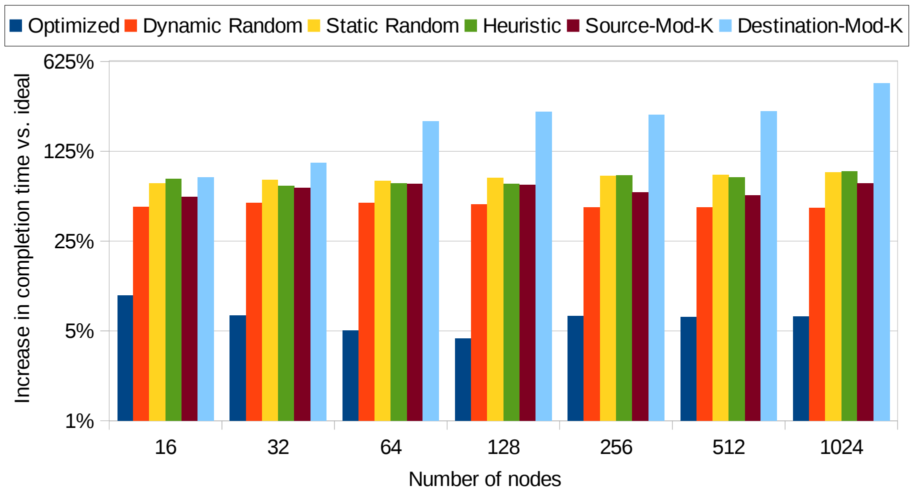


Fig. 3. Completion time comparison for the optimized all-to-all communication pattern between the optimized routing algorithm and five other routing approaches commonly used in XGFTs for varying networks sizes under fixed 4 KB message size. The scale for the y axis is logarithmic.

Series (DFS) dynamical core [Park et al. 2013], which while increasing precision and efficiency with respect to the state-of-the-art, still "suffers from the inherent scalability problem of the spectral model; namely, the transpose operation (or all-to-all communication) required for transforming wave space to grid space or vice versa" [Park et al. 2013].

In a recent work [Prisacari et al. 2013] we addressed all-to-all optimization in the context of fat tree networks. We considered a case where each leaf node of an XGFT network is assigned a single task. Furthermore, we considered the case where messages are not aggregated and where a given task only sends its own messages (it cannot act as a proxy task for another message source). Therefore, the load-optimization is performed only via the order in which every source chooses its destinations. The basic idea behind the exchange is to spread traffic as evenly as possible across the different subtrees at every layer, such that any given link of the fat tree conceptually modeled by the XGFT will be traversed in any of the phases by a minimum number of messages (if the total number of messages traversing that link in the complete exchange is n, and there are p phases, then in any given phase there are exactly $\lfloor n/p \rfloor$ or $\lceil n/p \rceil$ messages traversing that link).

This resulted in a set of highly intricate per-phase workloads that are provably bandwidth-optimal for the conceptual fat tree. Despite this optimality, when evaluated in practice, the performance of the approach showed significant degradation compared to the optimum level. The cause was the fact that the routing algorithms were not able to sustain the theoretically ideal workload performance.

[Prisacari et al. 2013] also derived the minimum amount of bandwidth required at each fat tree layer to ensure contention-free traffic in each exchange phase. The higher the layer, the more potential for bandwidth reduction exists compared to the full bisection design. In an XGFT, the bandwidth is tunable via the W parameters. The topologies that we consider all have exactly the minimum amount of bandwidth at the top layer (the one where the most reduction is possible), whereas every other layer has the full bisection bandwidth. The resulting topologies are XGFTs with as

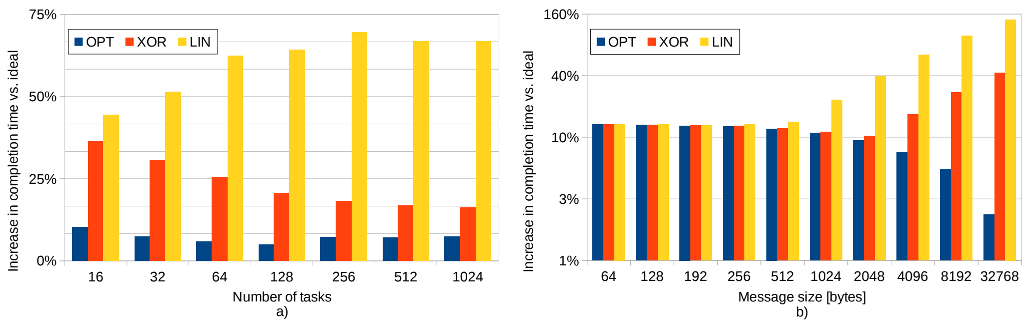


Fig. 4. [Prisacari et al. 2013] Completion time comparison between the optimized communication pattern (OPT) with optimal routing and two other commonly used patterns – XOR (binary exclusive or) and LIN (linear shift) – for varying networks sizes for fixed 4 KB message size (Fig. a) and for varying message size under fixed 512 nodes network scale (Fig. b). Note the linear scale for the y-axis in a) and the logarithmic scale for the y-axis in b).

little as 16 and as many as $8,192$ leaf nodes, with a number of layers between 3 and 6. The exact configurations are detailed in Table I.

Taking the ideal completion time as the baseline, Figure 3 shows the temporal overhead induced by several routing approaches compared to that induced by the optimal routing for a network interconnecting 512 nodes that exchange $4KB$ messages. The routing approaches that were used are some of the commonly employed strategies for fat tree networks [Zahavi 2011; Flich et al. 2012; Requena et al. 2007]:

(1) Static random: On the upwards path, the upward link is chosen at random, but the choice is fixed for every message between the same (source,destination) pair.
(2) Dynamic random: Similar to the previous approach, but with a choice that changes dynamically for every message.
(3) Heuristic routing: An optimized heuristic approach detailed in [Rodriguez et al. 2009a].
(4) Source-mod-K [Kariniemi 2006; Öhring et al. 1995].
(5) Destination-mod-K [Lin et al. 2004; Zahavi et al. 2007; Rodriguez et al. 2009b].

In all cases, the optimized routing showed a performance benefit of at least 40% compared to the best performance achieved across all other routing strategies.

Using the approach presented in this work, the artificially induced routing bottleneck was eliminated, and the measured performance matched ideal estimates in the case of large message exchanges to within a 3% difference, whereas previous approaches would be between 40% and 140% less efficient. Figure 4 shows more detailed results for the different networks shown in Table I and for different message sizes.

6. ALGORITHM PERFORMANCE EVALUATION

In benchmarking the efficiency of the approach, we will use the workload and network configurations presented in the previous section. This particular workload is especially well suited for the benchmark for several reasons. First of all, we look at the bandwidth-optimal all-to-all exchange as a set of multiple independent phases which we optimize separately. This gives us the opportunity of assessing the variability in completion time that is to be expected from the route optimization algorithm when dealing with multiple workloads. Second, each individual phase constitutes a particularly difficult to optimize traffic pattern, as showed by the performance that other routing approaches induce and as such they will test the ability of the route optimization process of finding solutions that are very rare in the search space. Finally, the workloads in this particular set satisfy the conditions necessary to make the application of the acceleration techniques described in section 4 possible. As such, we will be able to also measure the impact of the techniques on the completion time of the algorithm.

6.1. Benchmarking system description

The benchmarks were conducted using a commodity mid-level system with an Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz and 16 GB of RAM. The linear programming solver used is the freely available open-source glpsol solver that is included in the GNU Linear Programming Kit v4.50 [Makhorin 2013]. An important limitation of this solver is that it is single-threaded, so although the CPU we are using has significant potential for parallel acceleration, we do not benefit from it (using a commercial parallel solver will in all likelihood produce the ideal routing in a significantly smaller amount of time). The glpsol solver's behavior has a high degree of tunability via the numerous command line parameters it takes. Among other things, one can tune

— the selection of the branching variable (via the parameters --first, --last, --mostf, --drtom, --pcost)
— the backtracking strategy (via the parameters --dfs, --bfs, --bestp, --bestb)
— the type of heuristic cuts to be applied to the search space (via the parameters --gomory, --mir, --cover, --clique, --cuts)

For a given combination of the above parameters, it might be the case that some workloads on a given topology are a worst case for the solver, in that they take a particularly long time to optimize (much longer than the average completion time across other workloads on the same topology). However, it is often, if not always, the case that changing the parameters, and thus the heuristic of the solver, will lead to different worst case scenarios, such that across a set of parameter combinations the worst cases are eliminated altogether. As such, we analyzed several combinations of the parameters above.

Furthermore, we configured the solver:

— to use the MIP presolver (parameter --intopt),
— to use exact arithmetic (parameter --exact) and
— to consider the integer variables binary (parameter --binarize).

Lastly, we enabled the feasibility pump heuristic (parameter --fpump) which greatly accelerated the finding of feasible solutions, which in our case were also optimal solutions (the optimality is embedded in the feasibility conditions, not in any optimization function).

6.2. Results

Before presenting the results for the individual per-phase workloads making up the bandwidth-optimal all-to-all exchange, it is important to specify that the process of optimizing the entire exchange is highly parallel. The serial portion of the process (that can only be further accelerated by a parallel solver) is the single phase optimization. Given a reasonable number of execution threads and/or machines with a small amount of memory each, the determining of the route for the entire exchange can be done in an amount of time in the same order of magnitude as the time necessary to optimize a single phase. A similar rationale stands behind using different combinations of parameters for the solver for those few phases that might be more problematic to optimize. In practice, when the optimization of a workload would exceed a certain amount of time, we would in parallel start optimization processes with different sets of parameters for

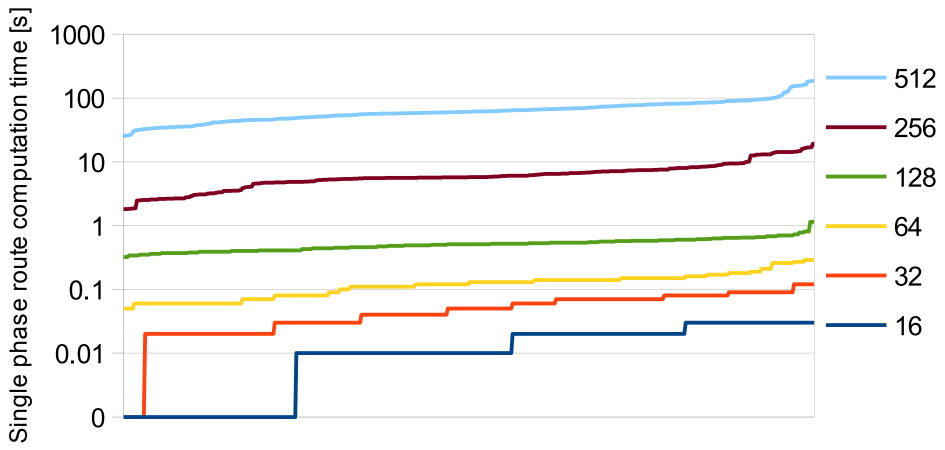


Fig. 5. Duration of the route optimization process for a workload constituting of a single phase of the bandwidth-optimized all-to-all exchange. Every line corresponds to a fixed network size (the legend shows the number of communicating nodes corresponding to that network). The completion time of the process was measured for every all-to-all phase then sorted in increasing order and plotted left-to-right to show their distribution. To enable using the same X-axis interval for all network sizes, given that the number of phases is equal to the number of communicating nodes, we use a step-wise constant graph, where the width of each step is constant for a given network size: 1 for the 512 leaf nodes XGFT, 2 for the 256 one, 4 for the 128 one and so on up to 32 for the 16 leaf nodes XGFT. The increase in completion time with the network size is faster than linear and the distribution of the completion time for a fixed network size sees significant variability, with the slowest phase to complete being up to 3 times slower than the average.

the ILP solver and halt the execution of all these parallel processes once a single one of them completes.

We will start with presenting the completion times achieved with the strongest restriction method, which, as we will see, is the slowest to complete and consequently limited in its application to larger size networks. Nonetheless, the strongly restricted approach is able to successfully generate optimal routes for XGFT networks with 1024 leaf nodes in under 30 hours for a single phase of the all-to-all workload. In order to show how the completion time of the algorithm varies with the network size, we ran the optimization process on every one of the phases of the complete all-to-all exchange, for several XGFTs with as few as 16 and as many as 512 leaf nodes. Figure 5 shows the (sorted) completion times of the optimization processes across the different phases of the all-to-all exchange. For smaller network sizes we have less phases to benchmark (the number of phases is equal to the number of communicating nodes), so in order to allow for the same X axis interval to be used for all curves, the completion times are plotted as step-wise constant graphs, with a step width inversely proportional to the total number of phases. The conclusion is that the completion time of the optimization process scales faster than linear with the network size. As we progressively double the number of leaf nodes in the XGFT, starting with 32, the average completion time for the generation of optimized routes for a single phase increases first 2.5 times, then approximately 4 times, then 10 and then another 10 times. When moving to 1024 (from 512) leaf nodes, the completion time will actually increase 100-fold, due to limitation of the ILP solver.

In the same figure one can observe that, while most workloads (each corresponding to a phase of the all-to-all exchange) have a completion time close to the average, there are a few phases for which the exchange structure seems to pose significantly

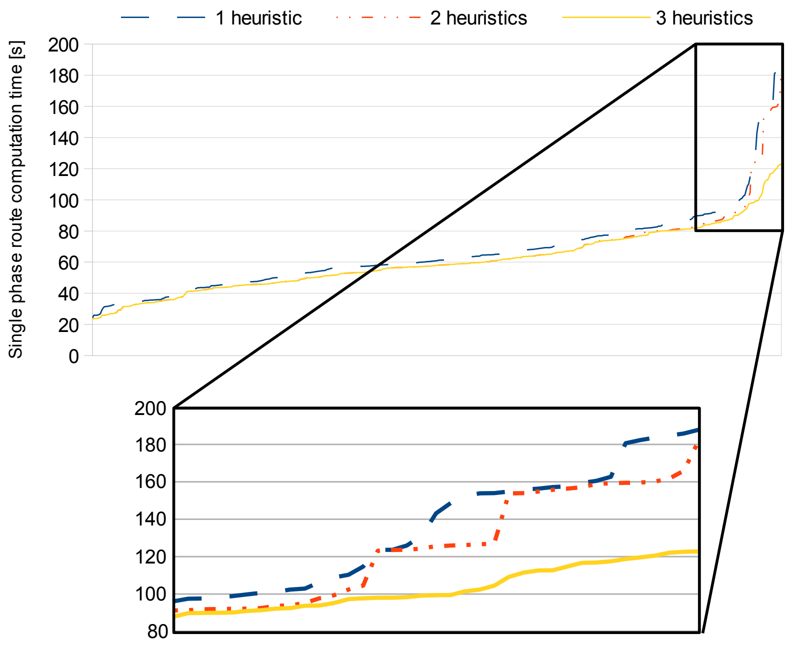


Fig. 6. Impact of salting the heuristics of the linear programming solver. The chart on top shows the completion times of the optimization process for every phase of an optimized all-to-all exchange on a 512 leaf nodes XGFT, for one, two and three combinations of solver parameters. When using more than one combination, the completion time of a phase is computed as being the minimum completion time among the different combinations of parameters used. Similarly to before, the X-axis does not represent the phases in their natural order, but instead in increasing order of optimization process completion time, so that the distribution of the completion times becomes apparent. The chart on the bottom zooms in on the tail of the distribution, i.e. the phases that take the most time to optimize. By using only two additional heuristic approaches, we are able to significantly flatten the tail, reducing the ratio between the slowest to complete optimization and the average optimization completion time from 3 to 2 - effectively a 30% reduction in the completion time of the slowest optimization process.

more problems in terms of finding the optimized routes. There might be an interest, particularly in the case where a parallel setup would be available that would allow optimization of a large number of workloads simultaneously, to reduce the completion time of these outliers, as it is these difficult to optimize workloads that would actually limit the parallel completion time. Thus, it would be beneficial to flatten the tail of the distribution. As explained above, this is possible without changing the problem formulation, simply by varying the heuristics of the linear programming solver through the usage of a slightly different combination of command line parameters.

Figure 6 shows how the distribution evolves when using two and three heuristics compared to the single heuristic case. What we can see is that by using only two additional heuristics we are able to progress from having the slowest optimization process being 3 times slower than the average to it being only twice as slow. This is particularly of interest in the case of larger networks where this decrease in completion time can mean obtaining the results hours or even days earlier.

For some types of networks and workloads, we cannot improve the completion time further. However, for other cases, where the accelerations techniques described in Section 4 are applicable, we can simplify the job of the linear solver and consequently achieve lower optimization durations. The workload we chose for our benchmarks falls

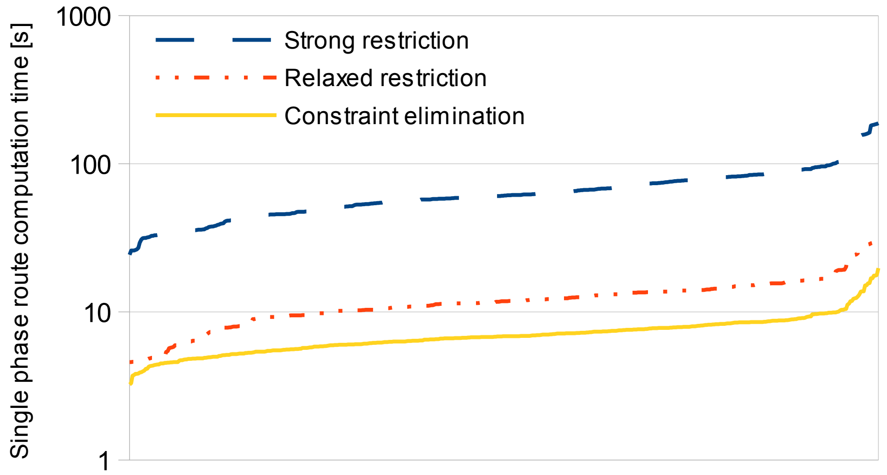


Fig. 7. Reduction in completion time of the optimization process achievable when using restriction relaxation (in red) and, in addition, restriction elimination (in yellow) compared to the standard strong restriction formulation (in blue). The benchmarked system is a 512 leaf node XGFT and the curves show the completion time of the optimization process for each of the phases of an optimized all-to-all exchange, ordered from left to right.

in this latter category and as such we can use it to evaluate the impact of the acceleration techniques.

The first acceleration technique we introduced, constraint relaxation, consisted in imposing weaker conditions in the ILP formulation, all the while preserving the optimum maximum contention level on any link property, at the expense of load not being entirely evenly balanced. The second acceleration technique, constraint elimination, consisted in removing optimality propagation conditions when they were not needed due to the relaxed limits that the first technique introduced. Figure 7 illustrates the evolution of the completion time for the 512 leaf node topology when applying constraint relaxation alone and then in conjunction with constraint elimination. We also show on the same figure the original, non-accelerated completion time for reference. What we observe is that relaxing the constraints and reducing the number of constraints brings a significant improvement. The first acceleration technique reduces the average and maximum completion time by a factor of approximately 5 while the second acceleration technique further reduces the completion time by a factor of approximately 2.

Due to this significant acceleration, we are now able to apply the method in the approximately same amount of time to networks that are 4 times larger in terms of leaf nodes: routing assignments for networks with 4096 leaf nodes can be determined in under one day. The evolution of the average completion time is shown in Figure 8.

Given a larger time budget, larger networks can be optimized as well. However, due to the faster than linear scaling, the feasible size will not increase significantly. Finally, without changing in the least the method, but by using a more complex commercial linear programming solver that has in particular the ability to perform the optimization of a single workload in parallel, further reductions in the optimization time can be envisioned.

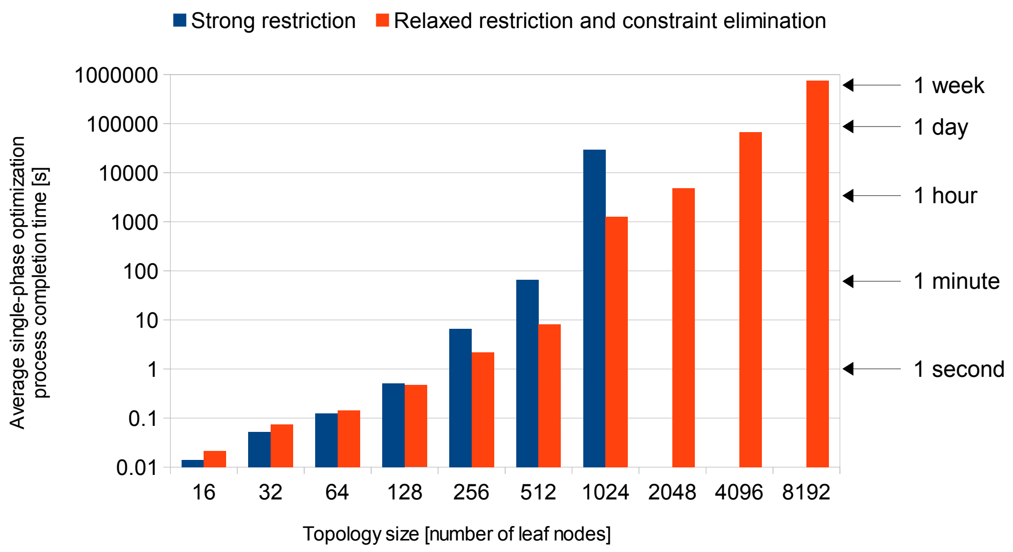


Fig. 8. Average completion times for the optimization process across the different phases of the all-to-all exchange on multiple XGFT topologies. The columns in blue show the completion time in the case where strong restrictions are enforced, while the columns in red show the completion time when both restriction relaxation and restriction elimination are employed.

7. CONCLUSIONS

We presented a method of determining an optimal assignment of routes for arbitrary communication workloads performed by concurrent tasks interconnected via arbitrary extended generalized fat tree networks. The method is based on formulating the routing problem as a set of constraints composing an integer linear programming problem and solving that problem by means of an open-source ILP solver.

As evidenced by the literature and by our own experiments, solving the integer linear programming problem for the entire network in a single step has proven to be infeasible from a computational point of view. Instead, our approach decomposes the global linear programming problem into multiple, smaller, LP formulations, each corresponding to a single layer of the fat tree network and solvable within a reasonable time frame.

Our approach ensures that the local optima can be combined into a globally optimal solution (and not just an approximation thereof), via a combination of additional global optimality-preserving constraints and a dynamic-programming-based construction of the global solution. Furthermore, to facilitate the formulation of the former, we introduced a novel generic mathematical XGFT model, referred to as *dual representation*, that reflects the routing-related properties of XGFTs in a much clearer way than existing models.

Finally, we evaluated the scalability of our approach by applying it to a very challenging workload from the routing point of view: a bandwidth-optimal all-to-all personalized exchange. Our method was able to derive the optimal routes for this workload on relatively large XGFT networks comprising up to $8,192$ nodes, using only commodity hardware and a freely available non-parallel solver.

REFERENCES

2013. Top 500 List. http://www.top500.org/list/2013/06/. (June 2013). Accessed: 2013-09-10.

Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*, Vol. 38. ACM, 63–74.

S.R. Alam and J.S. Vetter. 2006. An Analysis of System Balance Requirements for Scientific Applications. In *Parallel Processing, 2006. ICPP 2006. International Conference on*. 229–236. DOI: http://dx.doi.org/10.1109/ICPP.2006.21

David Applegate and Edith Cohen. 2003. Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '03)*. ACM, New York, NY, USA, 313–324. DOI: http://dx.doi.org/10.1145/863955.863991

Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Räcke. 2003. Optimal oblivious routing in polynomial time. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing (STOC '03)*. ACM, New York, NY, USA, 383–388. DOI: http://dx.doi.org/10.1145/780542.780599

David Bailey, Tim Harris, William Saphir, Rob van der Wijngaart, Alex Woo, and Maurice Yarrow. 1995. The NAS Parallel Benchmarks 2.0. *The International Journal of Supercomputer Applications* (1995).

Bartosz Bogdanski, Frank Olaf Sem-Jacobsen, S-A Reinemo, Tor Skeie, Line Holen, and Lars Paul Huse. 2010. Achieving predictable high performance in imbalanced fat trees. In *Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on*. IEEE, 381–388.

Zhu Ding, Raymond R. Hoare, Alex K. Jones, and Rami Melhem. 2006. Level-wise Scheduling Algorithm for Fat Tree Interconnection Networks. In *Proc. 2006 ACM/IEEE Conference on Supercomputing*. ACM, New York, NY, USA, 96. DOI: http://dx.doi.org/10.1145/1188455.1188556

E. S. Elmallah and J. C. Culberson. 1994. Multicommodity Flows in Simple Multistage Networks. *Networks* 25 (1994), pp.

S. Even, A. Itai, and A. Shamir. 1976. On the Complexity of Timetable and Multicommodity Flow Problems. *SIAM J. Comput.* 5, 4 (1976), 691–703.

J. Flich, T. Skeie, R. Juarez-Ramirez, O. Lysne, P. Lopez, A. Robles, J. Duato, M. Koibuchi, T. Rokicki, and J.C. Sancho. 2012. A Survey and Evaluation of Topology-Agnostic Deterministic Routing Algorithms. *Parallel and Distributed Systems, IEEE Transactions on* 23, 3 (2012), 405–425. DOI: http://dx.doi.org/10.1109/TPDS.2011.190

T. Hoefler, T. Schneider, and A. Lumsdaine. 2009. The impact of network noise at large-scale communication performance. In *Parallel and Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. 1–8. DOI: http://dx.doi.org/10.1109/IPDPS.2009.5161095

A. Jajszczyk. 2003. Nonblocking, repackable, and rearrangeable Clos networks: fifty years of the theory evolution. *Communications Magazine, IEEE* 41, 10 (Oct. 2003), 28–33. DOI: http://dx.doi.org/10.1109/MCOM.2003.1235591

Heikki Kariniemi. 2006. *On-Line Reconfigurable Extended Generalized Fat Tree Network-on-Chip for Multiprocessor System-on-Chip Circuits*. Ph.D. Dissertation. Tampere University of Technology.

Michel A. Kinsy, Myong Hyon Cho, Tina Wen, Edward Suh, Marten van Dijk, and Srinivas Devadas. 2009. Application-aware deadlock-free oblivious routing. *SIGARCH Comput. Archit. News* 37, 3 (June 2009), 208–219. DOI: http://dx.doi.org/10.1145/1555815.1555782

Tom Leighton, Fillia Makedon, Serge Plotkin, Clifford Stein, Eva Tardos, and Spyros Tragoudas. 1991. Fast Approximation Algorithms for Multicommodity Flow Problems. In *Journal of Computer And System Sciences*. 487–496.

Charles Leiserson, Zahi S. Abuhamdeh, David C. Douglas, Carl R. Feynman, Mahesh N. Ganmukhi, Jeffrey V. Hill, W. Daniel Hillis, Bradley C. Kuszmaul, Margaret A. St. Pierre, David S. Wells, Monica C. Wong-chan, Shaw wen Yang, and Robert Zak. 1992. The Network Architecture of the Connection Machine CM-5. In *Proc. of the Fourth Annual ACM Symposium on Parallel Algorithms and Architectures*. San Diego, CA, USA, 272–285.

Xuan-Yi Lin, Yeh-Ching Chung, and Tai-Yi Huang. 2004. A Multiple LID Routing Scheme for Fat-Tree-Based InfiniBand Networks. *Proc. of the 18^{th} International Parallel and Distributed Processing Symposium* (2004), 11–. DOI: http://dx.doi.org/10.1109/IPDPS.2004.1302913

Andrew Makhorin. 2013. GNU Linear Programming Kit. (2013). http://www.gnu.org/software/glpk/ [Online; accessed 06-June-2013].

Sabine R. Öhring, Maximilian Ibel, Sajal K. Das, and Mohan J. Kumar. 1995. On generalized fat trees. In *Proceedings of the 9^{th} International Parallel Processing Symposium*. IEEE Computer Society, Washington, DC, USA, 37.

Hoon Park, Song-You Hong, Hyeong-Bin Cheong, and Myung-Seo Koo. 2013. A Double Fourier Series (DFS) Dynamical Core in a Global Atmospheric Model with Full Physics. *Monthly Weather Review* 141 (2013), 3052–3061.

Fabrizio Petrini, Darren J. Kerbyson, and Scott Pakin. 2003. The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q. In *Proceedings of the 2003 ACM/IEEE conference on Supercomputing (SC '03)*. ACM, New York, NY, USA, 55–. DOI:http://dx.doi.org/10.1145/1048935.1050204

Fabrizio Petrini and Marco Vanneschi. 1997a. A Comparison of Wormhole-Routed Interconnection Networks. In *Proc. Third International Conference on Computer Science and Informatics*. Research Triangle Park, NC, USA.

Fabrizio Petrini and Marco Vanneschi. 1997b. k -ary n -trees: High Performance Networks for Massively Parallel Architectures. *IPPS* 00 (1997), 87. DOI:http://dx.doi.org/10.1109/IPPS.1997.580853

Bogdan Prisacari, German Rodriguez, Cyriel Minkenberg, and Torsten Hoefler. 2013. Bandwidth-optimal all-to-all exchanges in fat tree networks. In *Proceedings of the 27th international ACM conference on International conference on supercomputing (ICS '13)*. ACM, New York, NY, USA, 139–148. DOI:http://dx.doi.org/10.1145/2464996.2465434

Harald Raecke. 2002. Minimizing Congestion in General Networks. In *In Proceedings of the* 43^{rd} *IEEE Symposium On Foundations of Computer Science (FOCS)*. 43–52.

Crispín Gómez Requena, Francisco Gilabert Villamón, María Engracia Gómez, Pedro López, and José Duato. 2007. Deterministic versus Adaptive Routing in Fat-Trees. *Proc. of the* 21^{st} *Parallel and Distributed Processing Symposium, 2007* (March 2007), 1–8. DOI:http://dx.doi.org/10.1109/IPDPS.2007.370482

German Rodriguez, Rosa M. Badia, and Jesus Labarta. 2008. An Evaluation of Marenostrum Performance. *International Journal of High Performance Computing Applications, Special Issue on "Performance Characterization of the World's Most Powerful Supercomputers"* 22 (February 2008), 81–96. Issue 1. DOI:http://dx.doi.org/10.1177/1094342006085022

German Rodriguez, Ramon Beivide, Cyriel Minkenberg, Jesus Labarta, and Mateo Valero. 2009a. Exploring pattern-aware routing in generalized fat tree networks. In *ICS '09: Proceedings of the 23rd international conference on Supercomputing*. ACM, New York, NY, USA, 276–285. DOI:http://dx.doi.org/10.1145/1542275.1542316

German Rodriguez, Cyriel Minkenberg, Ramon Beivide, Ronald P. Luijten, Jesus Labarta, and Mateo Valero. 2009b. Oblivious routing schemes in extended generalized Fat Tree networks. In *CLUSTER*. IEEE, 1–8.

A. Srinivasan. 1997. Improved approximations for edge-disjoint paths, unsplittable flow, and related routing problems. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*. 416–425. DOI:http://dx.doi.org/10.1109/SFCS.1997.646130

S. Sur, Hyun-Wook Jin, and D.K. Panda. 2004. Efficient and scalable all-to-all personalized exchange for InfiniBand-based clusters. In *Parallel Processing, 2004. ICPP 2004. International Conference on*. 275–282 vol.1. DOI:http://dx.doi.org/10.1109/ICPP.2004.1327932

Texas Advanced Computing Center (TACC). 2011. Press release: "Stampede's" Comprehensive Capabilities to Bolster U. S. Open Science Computational Resources. (2011).

E. Zahavi. 2011. Fat-Trees Routing and Node Ordering Providing Contention Free Traffic for MPI Global Collectives. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*. 761–770. DOI:http://dx.doi.org/10.1109/IPDPS.2011.219

Eitan Zahavi, Gregory Johnson, Darren J. Kerbyson, and Michael Lang. 2007. Optimized InfiniBand Fat-tree Routing for Shift All-to-All Communication Pattern. In *International Supercomputing Conference (ISC07)*. Dresden, Germany.