Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks

Torsten Hoefler
Open Systems Laboratory
Indiana University
Bloomington, IN 47405
htor@cs.indiana.edu

Timo Schneider
Technical University of Chemnitz
Strasse der Nationen 62
Chemnitz 09107, Germany
timos@cs.tu-chemnitz.de

Andrew Lumsdaine
Open Systems Laboratory
Indiana University
Bloomington, IN 47405
lums@cs.indiana.edu

Abstract

Multistage interconnection networks based on central switches are ubiquitous in high-performance computing. Applications and communication libraries typically make use of such networks without consideration of the actual internal characteristics of the switch. However, application performance of these networks, particularly with respect to bisection bandwidth, does depend on communication paths through the switch. In this paper we discuss the limitations of the hardware (capacity-based) definition of bisection bandwidth and introduce a new metric: effective bisection bandwidth. We assess the effective bisection bandwidth of several large-scale production clusters by simulating artificial communication patterns on them. Networks with full bisection bandwidth typically provided effective bisection bandwidth in the range of 55-60%. Simulations with application-based patterns showed that the difference between effective and rated bisection bandwidth could impact overall application performance by up to 12%.

1 Introduction

Commodity-based clusters with central-switch-based networks have become an established architecture for high-performance computing. The use of a central switch significantly simplifies the communication model for such systems. Compared to other interconnection network topologies, such as tori or rings, the central-switch based structure has the advantage of being able to efficiently embed structured communication patterns and to support unstructured patterns as well. A network with a true crossbar as its central switch has almost ideal network properties: constant latency between all pairs of endpoints as well as full bisection bandwidth (any half of the endpoints can simultaneously communicate with the other half at full line rate).

However, although they are often treated as if they were true crossbar switches, practical central switches are generally implemented as multistage interconnection network (MINs). As a result, MINs are able to approximate, but not truly provide the latency and bisection bandwidth characteristics of crossbars. The point-to-point latency in a MIN is not constant for all port combinations (although it is usually the case that the variance is relatively low). Less obvious, but more important to application performance, is the effect of MIN architecture on network bisection bandwidth, particularly as it is seen by applications.

As with other networks, MINs can be characterized by their bisection bandwidth, which, following [9] we define as the total bandwidth between the two halves of the worst-case segmentations of a network. However, this definition of bisection bandwidth only considers the capacity provided by the hardware. It does not consider how the usage of that capacity may be affected by routing policies. That is, unlike with a true crossbar, connections must be routed in a MIN and different communication patterns may require different routing in order to achieve the rated bisection bandwidth for a MIN. If proper routing is not established in the MIN for a given communication pattern, that pattern may not be able to achieve satisfactory performance.

This issue is particularly problematic in networks, such as Infiniband, that employ static routing schemes because of the potential for mismatch between the static routes and the communication requirements of running applications. Applications can be oblivious to network parameters if the network can provide its rated bisection bandwidth for all communication patterns. Indeed, most applications and communication libraries today are written using this assumption. However, different applications have different communication patterns (and communication patterns may even vary significantly within a single application).

Given the prevalence of MIN networks in high-performance computing, a more thorough understanding of their characteristics is an important step towards more effective utilization of these resources.

Contributions

Therefore, in this paper we assess the impact of static routing on the expected bisection bandwidth for arbitrary patterns and for real applications on large-scale production clusters. We address several myths about the definition of bisection bandwidth (and full bisection bandwidth) and introduce the new application-driven definition of "effective bisection bandwidth". Furthermore, we provide a methodology and tool for cluster and network designers to characterize the theoretical performance of applications running on InfiniBand (and potentially other) MIN networks. More generally, we argue that the scalability of MINs is limited due to practical constraints (routing, hot spots).

2 Background

2.1 Network Topologies

Different network topologies with different properties have been proposed to be used in parallel computers: trees [4, 14], Benes networks [2], Clos networks [5] and many more — consult [13] for a complete overview. Crossbar switches are often used as basic building blocks today. Crossbar circuits usually implement a symmetric crossbar, i.e., the number of input ports is equal to the number of output ports. Available HPC networks, such as Myrinet, InfiniBand and Quadrics implement crossbars of sizes 32, 24 and 8 respectively. A fully connected network is not scalable because the number of necessary links grows with $O(P^2)$ for P endpoints.

The Clos network The Clos network was designed by Clos in order to build telecommunication networks with switch elements of limited size [5]. It is used today in many InfiniBand and Myrinet switches. The typical Clos network consists of three stages and is described by three parameters. Those parameters n and m and r describe the input and output port count and number of switches in the first layer respectively. The m switches in the middle stage are used to connect input to output ports. Clos networks are "strictly non-blocking" if $m \geq 2n - 1$ and rearrangably non-blocking if $m \geq n$. Most of todays networks are built with $n = m$ which makes those Clos topologies "rearrangably non-blocking" (see [5] for details). They can have the full bisection bandwidth but only for certain combinations of routing and traffic patterns. An easy routing algorithm (up*/down* [25]) can be used to ensure deadlock-free routing and multiple paths between any pair of nodes exist (failover possible).

The (fat) tree network The tree network has a fixed tree topology and has been used to connect the TMC CM-5 and Meiko CS-2 machines and many InfiniBand based cluster systems, such as the world's largest unclassified InfiniBand system (Thunderbird) at the Sandia National Laboratories. The nodes are connected at the leaves and routing is simple (go up until the target node is reachable on a down route). However, this introduces congestion near the root and limits the bisection bandwidth. Leiserson simply increased the link width at every level to avoid this congestion [14]. The resulting "fat tree" network can be designed to offer full bisection bandwidth and can be efficiently implemented in an on-chip network. However, it is not possible to construct it easily from fixed-size crossbar elements with fixed link bandwidths. A tree-like network topology that can be constructed with equally sized crossbar switches has been proposed in [15]. Similar to Clos networks, the k-ary n-tree networks [20] retain the theoretical full bisection bandwidth in a rearrangable way.

Practical installations Most modern networks, such as Myrinet, Quadrics and InfiniBand allow arbitrary network layouts but usually use Clos networks or fat trees. The main difference lies in the size of the crossbar elements and the routing strategy. Myrinet employs 16 or 32 port crossbar switches. InfiniBand and Quadrics build on 24 and 8 port switch elements respectively. The routing strategy is also very important. Myrinet uses source-based routing where every packet can define a different route. InfiniBand uses a static switch based routing scheme and Quadrics a non-disclosed adaptive scheme [19].

In the following, we pick the network with second most installations in the top 500 list (24.2% as of 11/2007), InfiniBand, to analyze the effects of changing traffic patterns and the influence of the used network topologies for existing systems and routing algorithms.

2.2 The InfiniBand Network

We focus on a practical analysis of deployed InfiniBand networks in this paper. However, our results also apply to other statically routed networks with similar topologies. The InfiniBand standard does not define a particular network topology, i.e., switches can be connected in an arbitrary way to each other. The switches have a simple static routing table which is programmed by a central entity, called the subnet manager (SM). The SM uses special packets to explore the network topology in the initialization phase. It then computes the routing table for every switch. The routing algorithm can be freely chosen by the SM. This initialization usually requires network downtime and is not performed often. Thus, the routing can be seen as static.

2.2.1 Hot-spot problems in the InfiniBand Network

Most large-scale InfiniBand networks use the k-ary n-tree topology to connect high node-counts. This topology is able to provide full bisection bandwidth, but is limited to a fixed routing table. This means that, for a given balanced routing, there is at least one pattern that delivers full bisection bandwidth, but all other communication patterns might perform significantly worse. The problem is not the number of available physical links, which is sufficient for any commu-

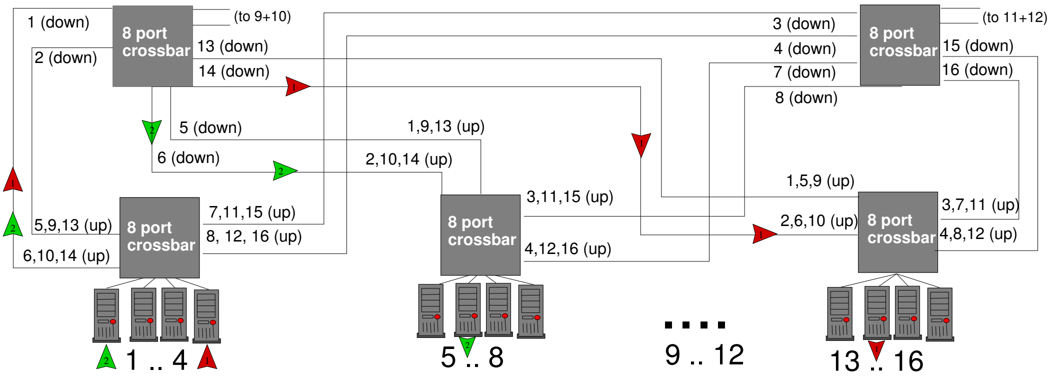

Figure 1. Statically routed fat tree example with 16 nodes and 8 port crossbars

nication pattern, it is the routing which might oversubscribe physical links even though other links remain idle. This problem has been discussed as a source for performance loss in [30, 21]. Zahavi [30] uses specialized routing tables in the switches to avoid hot-spots in the network for a certain communication pattern. This method only works if the processes are carefully placed in the network, the application pattern is known well in advance, and the system can be taken down to re-program the switches before the application run. This scenario is very unlikely; applications have to run with the pre-installed routing table that might not support their specific traffic pattern. Most routing algorithms try to distribute routes evenly over the available links, leading to a so called “balanced routing” scheme. We will illustrate the congestion with a small 16 node example network built from 8 port crossbar elements using ideally balanced routing. Figure 1 shows the network and the fixed routing tables as attributes on the physical connections (up and down routes). Figure 1 also shows a congestion case where node 4 sends packet 1 to node 14 and node 1 sends packet 2 to node 6. Even though those two node pairs are distinct and the down-route is contention-free, the static routes to the upper-left switch send both packets over a single link which causes congestion.

We call a set of $n/2$ communication partners a communication pattern. Our example network allows us to create some patterns that have full bisection bandwidth, such as (1,5), (2,6), (3,7), (4,8), (9,13), (10,14), (11,15), (12,16). But other patterns, for example (1,5), (2,9), (3,13), (4,6), (7,8), (10,11), (12,14), (15,16) show different characteristics such that every connection has a different bandwidth/oversubscription. The connections (1,5), (2,9) and (3,13) have an oversubscription of 3 and thus only one third of the bandwidth available. Other connections, such as (7,8), (10,11) and (15,16) have the full bandwidth available. The congestion points in the network (in our example the up-link between the lower and upper left switches) are called hot spots [22]. Some techniques have been introduced to avoid those hot spots, such as adaptive source-based routing or multi-path routing [16]. However, those techniques often use simple round-robin schemes to disperse the network traffic, which is very limited. Other techniques require a global view of the network [6] or significant changes to the InfiniBand network [17, 24].

2.2.2 Measuring the Effects of Fabric Congestion

To analyze the impact of the hot-spot and congestion, we performed several benchmarks on the CHiC cluster, a 528 node InfiniBand system offering full bisection bandwidth. The network topology of this system is a fat tree network built from 44 24 port leaf switches (crossbar) and two 288 port top switches (internal Clos network). We queried the routing tables and topologies of the switches with the tools `ibnetdiscover` and `ibdiagnet` and chose pairs of communicating peers that cause congestion in the fat tree such that every node is in exactly one pair (no endpoint congestion). We benchmarked the point-to-point latency and bandwidth between a pair of nodes while adding more congestion (we did this by adding sending 8MB ping-pong MPI messages between the other node pairs). To achieve comparable results, we used the well-known Netpipe [28] MPI benchmark with Open MPI 1.2.5 and OFED 1.3 to perform those measurements.

The results in Figure 2 show the transmission curves for different numbers of pairs causing congestion even though all communications have been done among independent pairs of nodes. This shows clearly that congestion and routing-based hot-spots in the fabric can have a significant impact on the communication performance. However, from a user perspective it is not trivial to know if congestion occurs because the fabric looks homogeneous.

Figure 3 shows the latency for all possible hot spot congestions for the CHiC network. The congestion might vary from 0 (no congestion) to 11 (maximum congestion) be-

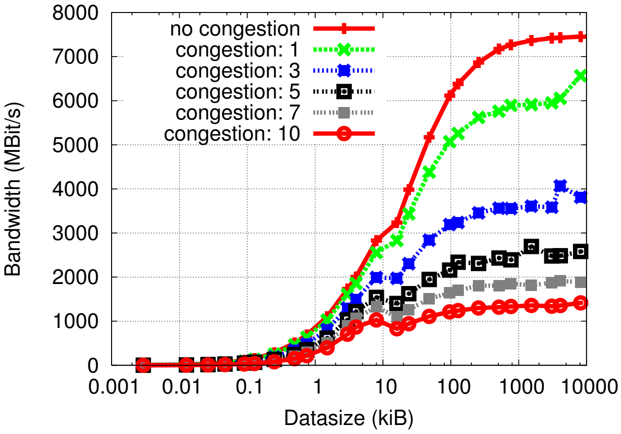

Figure 2. InfiniBand bandwidth curve in different congestion situations

cause every crossbar has 12 down- and 12 up-links. Thus, a maximum of 12 nodes can be connected to a single crossbar. We see a nearly linear increase in 0-byte transmission latency and a significant reduction of the available link bandwidth.

2.2.3 InfiniBand's Lid Mask Control

InfiniBand's Lid Mask Control (LMC) mechanism has been discussed as a solution to the hot-spot problem. It assigns multiple LIDs (InfiniBand's endpoint identifiers) to hosts and thus enables multiple routes for every pair of peers. However, the ordering constraints of the InfiniBand network prevent dispersion at the connection (queue pair) level. Thus only whole messages can be scheduled to different routes. Simple round-robin schemes have been shown to improve the worst-case network performance slightly in [29]. This "blind" way of dispersing routes does not provide near-optimal performance but finds some average performance and has not been analyzed well. It is thus unclear how much the difference to optimal routing is. Another problem with multi-path routing is that the switches need to store and evaluate the potentially very high number of routes for every packet. The worst problem however, is the growing number of endpoints (queue pairs) per process (especially on SMP or multicore systems) which is not scalable to higher node counts. Much work has been invested to limit the number of queues [26, 27] or even use alternative communication methods such as Unreliable Datagram [7, 12]. Thus, LMC-based mechanisms can be seen as counter-productive at large scale.

In this article, we focus on the analysis of single-path routed InfiniBand systems to analyze the impact of the static routing.

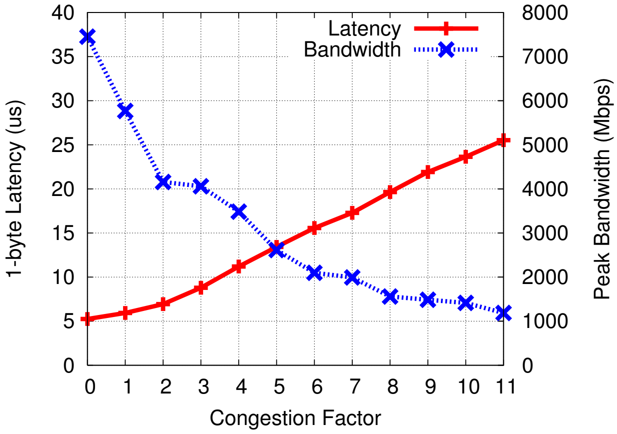

Figure 3. Latency and bandwidth decrease for InfiniBand measured in congestion situations

3 Hot Spot Analysis

While we've demonstrated that hot-spot problems exist in current InfiniBand networks, it is still not known how big the negative influence on real applications is. Our first step is to check the assumption that every communication pattern can be embedded in the network efficiently. Thus, we have to assume totally arbitrary communication patterns which reflects the worst class of applications with randomly changing communication patterns.

To examine bisection bandwidth under this assumption, we have to split P nodes into two equally sized partitions A and B. We have $\binom{P}{\frac{P}{2}}$ possibilities to do so. Furthermore we have to decide which node of partition A communicates to which node from partition B. The first node in partition A has $\frac{P}{2}$ possible partners in partition B, the second has $\frac{P}{2} - 1$ left and so on. This yields to a total of $\frac{P}{2}!$ communication schemes.

If we combine all different possibilities to split the P nodes into two groups and all different pairings between the two groups, we get

$$\binom{P}{\frac{P}{2}} \cdot \left(\prod_{i=1}^{\frac{P}{2}} i \right)$$

possible communication patterns; which is already $518, 918, 400$ in our simple 16 port network. Only a very small fraction of those patterns has full bisection bandwidth. Due to the huge number of possible patterns, there is no possibility to benchmark reasonably sized network exhaustively. Our approach is to simulate the network contention of a huge (statistically significant) number of communication patterns because this is significantly faster and easier than benchmarking.

3.1 Choosing a Network Model

Our congestion model is based on the the measurement results of Section 2.2.2. We model the network as a topology of fully connected, contention-free crossbar switches with limited port count interconnected by physical wires with bandwidth/capacity γ. Each crossbar switch has a static routing table that defines an output port to every node in the network. If we assume a specific communication pattern, we assign a usage count π to every physical connection that indicates how many connections are routed over this link. This usage count models the congestion factor in 2.2.2. The throughput per cable is thus defined as $\frac{\gamma}{\pi}$. In the following, we only discuss relative bandwidths and thus set $\gamma = 1$.

This simple model can be used to derive parameters for more complex models. The LogGP [1] model for example can be parametrized for a point-to-point connection while G is multiplied with π. The latency can be modeled as a linear function $L(\pi)$. The parameters o and g are independent of the congestion.

3.2 The Network Simulator

The design goal of our simulator is to use real existing InfiniBand systems (topologies + routing tables) to investigate contention effects on applications running on those systems. The simulator thus accepts a network structure (queried from the running system with `ibnetdiscover` and `ibdiagnet` and represented as a directed acyclic graph) and a specific communication pattern (represented by $P/2$ communication pairs) as inputs. The output is the maximum usage count $\pi(r)$ along every of the $P/2$ routes r (each route r might use multiple physical cables but the maximum congested cable mandates the transmission speed and thus the route's overall congestion). For example our example pattern (1,5), (2,9), (3,13), (4,6), (7,8), (10,11), (12,14), (15,16) would lead to the congestions 3, 3, 3, 1, 1, 1, 1, 1 respectively. The simulator assumes full duplex communication, i.e., messages using the same link in different directions do not cause congestion.

3.2.1 Related Work

Several publications, such as [3, 6, 16, 17, 20, 21], rely on network simulations. The ability to read arbitrary communication patterns and real-world topologies (not limited to fat tree or Clos) distinguishes our work from all earlier network simulations that used predefined patterns. Those predefined patterns do often not reflect the communication patterns used in high performance applications. Patterns are:

- "uniform traffic" [6, 16] which might cause congestion at the endpoints (destinations are chosen uniformly, i.e., two or more nodes might send to the same destination).
- "complement traffic" [20] where node i sends to the node with the number that equals the bit-wise (one-) complement of i. This pattern reflects a possible bisection of the network.
- "bit reversal" and "transpose" [20] are patterns that reflect all-to-all like communications which are used for some problems.
- "hot-spot traffic" [16, 21] is traffic where some percentage of the traffic is targeted at single nodes, so called hot-spots. Hot-spots should be avoided in high performance computing, thus, this pattern does not reflect typical applications.
- "localized traffic" [3] might reflect nearest neighbor communication schemes in real-world applications but the definition is vague.

Another problem with former simulations is that for each pattern, usually only the average packet latency and bandwidth is reported. However, the average values might have little meaning. Some applications can accept a wide variety of bandwidths in the communications, others can not. Fine grained applications running tightly synchronized with collective communication (lock-step) are usually only as fast as the slowest link. The average values might mislead in application modeling to just assume uniform bandwidth on every link and thus misinterpret those simulations. A more detailed application model is presented in Section 4.

3.2.2 Simulated Cluster Systems

Throughout this paper, we use four of the biggest InfiniBand cluster installations available to perform our simulations. The "CHiC" at the University of Technology Chemnitz has 528 nodes connected to 44 24-port leaf switches which are connected to two 288 port switches in the second level. The CHiC network is able to deliver full bisection bandwidth. The second larger input system is the "Atlas" system located at the Lawrence Livermore National Lab has 1142 nodes and a fat tree network with full bisection bandwidth. The "Ranger" system at the TACC uses two Sun "Magnum" 3456 port switches to connect 3936 nodes with full bisection bandwidth. The largest simulated system, the "Thunderbird" (short: Tbird) cluster, is also the largest InfiniBand installation (with the biggest number of endpoints) and its network has 4391 InfiniBand endpoints arranged in a fat tree network with $1/2$ bisection bandwidth.

3.2.3 Simulator Verification

To verify our simulation results, we implemented a benchmark that measures randomly changing "bisect" communication patterns and records the achieved bandwidths on every connection into several bandwidth classes. A "bisect" communication pattern is created as follows:

- split the network of size P into two equally sized groups A and B

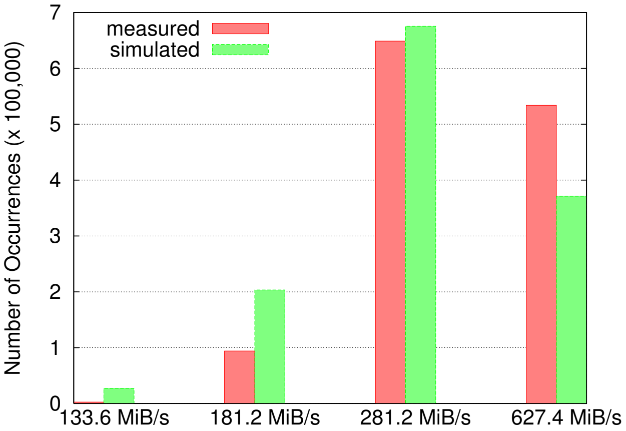

Figure 4. Simulation and Benchmark Results for a 512 node bisect Pattern with 1MiB messages in the CHiC system

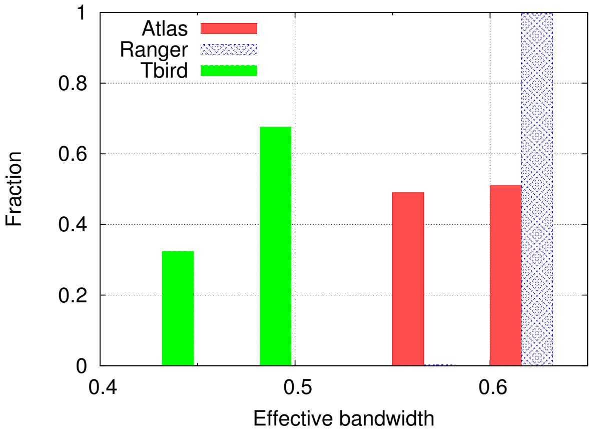

Figure 5. Bisection pattern bandwidth results for all simulated systems

- create $P/2$ pairs such that every pair consists of a node from A and B
- guarantee that no node is in more than a single pair (has more than one partner in the other group)

Our benchmark generates a random “bisect” pattern at process 0, scatters it to all P processes and synchronizes the time on all nodes with a tree-based algorithm as described in [10]. Process 0 broadcasts a starting time to all nodes that start simultaneously and benchmark the time needed to send 100 fixed-size packets between the nodes. Process 0 gathers all $P/2$ timing results and scatters a new random “bisect” pattern. This procedure is repeated for 5000 random bisect patterns. We use the Mersenne Twister [18] algorithm for to generate the random patterns. The root node records all $5000 \cdot P/2$ point-to-point bandwidth results and sorts them into 50 equally-sized bins.

The benchmark results of the full CHiC system are shown in Figure 4. This benchmark, using the full system showed very clean results with only 4 of the 50 possible bins filled. The measured link bandwidth with MPI between two nodes in the CHiC system is $\gamma = 630MiB/s$ for $1MiB$ messages. Our simple model $\frac{\gamma}{\pi}$ would predict $315MiB/s$, $210MiB/s$, $157.5MiB/s$ for a congestion π of 2, 3 and 4 respectively. The benchmark results are slightly lower than that but still reflect our expectations. Runs with fewer nodes also supported our thesis, however, the results were scattered across many bins due to background communication during those jobs.

These experiments show that our simulator design accurately reflects a real-world environment. It is usually not easily possible to perform measurements at full scale, thus we will show the effects of the network contention to applications by simulating the application traffic patterns. This will give us some estimation of how real applications are influenced by the network topology.

3.2.4 Simulating the Effective Bisection Bandwidth

To get an estimation of how much bandwidth can be expected from a random “bisect” pattern, we ran N ($N = 10^6$ for our simulation) simulations with different patterns for our three systems. Each pattern simulation resulted in $P/2$ oversubscription factors π (one per pair).

Most applications are round-based and every round exhibits a potentially different communication pattern. To model this behavior, we chose to evaluate every pattern as an entity by plotting the pattern-specific average bandwidths in a histogram. Thus, we compute the pattern-specific average oversubscriptions as $\left(\sum_{i=1}^{P/2} \pi_i\right) \cdot \frac{2}{P}$ and sorted these N results into bins. The normalized height of any histogram bin shows the fraction of mappings wich showed the specific effective bandwidth. Figure 5 shows the histograms of the bandwidth-distribution. The achieved average bandwidths are interestingly stable (nearly all patterns exhibit a similar average bandwidth) and only two bins are (nearly equally) filled for all simulated systems.

Based on the observation that all “bisect” patterns seem to have a pretty stable average bandwidth, we define the application-oriented network parameter “effective bisection bandwidth” as the average bandwidth for an arbitrary communication pattern. This most generic definition reflects applications with non-predictable and irregular communication patterns such as parallel graph algorithms as a single number. This parameter is unfortunately not easy to measure for a given network. It might be assessed by a exhaustive search or in combination with statistical methods.

The simulated “effective bisection bandwidths” are 57.6%, 55.6% and 40.6% of the full bandwidth for Ranger, Atlas and Tbird, respectively. An interesting observation is that the large Tbird system with half bisection bandwidth does not deliver a significantly worse “effective bisection bandwidth” than the Atlas system with full bisection bandwidth.

However, this analysis still does not reflect many real-world applications well. The next section explains an analysis of 5 real world application codes and simulates their communication patterns.

4 Parallel Applications Communication

To understand how the network affects applications, we are analyzing four large open source applications for the main sources of communication overhead. Later, we will use those results to derive communication patterns as input for the simulation.

Most parallel high-performance applications are written with the Message Passing Interface standard (MPI). Thus, we used the MPI profiling interface to record application communications and measure their running time as a share of the application’s running time. We analyzed point-to-point collective communication by representing the neighborhood relations in a directed graph. Collective communication calls can not be recorded that easily because the used pattern depends on the implementation. Thus, we just recorded the communicator size on which the collective operations were run. All runs were done on 64 processes with InfiniBand as the interconnection network.

Massively Parallel Quantum Chemistry Program The Massively Parallel Quantum Chemistry (MPQC) Program [11] is an open-source implementation that solves the Schrödinger equation to compute several properties of atoms and molecules. The MPQC run took 852 seconds and had 9.2% communication overhead. We were able to identify three collective routines that caused nearly all communication overhead: MPI_Reduce (67.4%), MPI_Bcast (19.6%) and MPI_Allreduce (11.9%). All routines used the full set of processes (64) as communication group.

MIMD Lattice Computation The MIMD Lattice Computation (MILC) code is used to study quantum chromodynamics, the theory of the strong interactions of subatomic physics as used in high energy and nuclear physics. We benchmarked a 9.4% communication overhead running the MILC code for 10884 seconds. More then 86% of the overhead was caused by point-to-point communication (MPI_Isend/MPI_recv/MPI_Wait) and 3.2% by MPI_Allreduce in the complete process group. We analyzed the send/receive pattern and found that every process communicates with exactly 6 other processes (neighbors).

Parallel Ocean Program The Parallel Ocean Program (POP) is an ocean circulation model. It is the ocean component of the Community Climate System Model and has been used extensively in ocean-only mode for eddy-resolving simulations of the global ocean. We measured 32.6% communication overhead for a 2294-second run. About 84% of this overhead are due to point-to-point communications and 14.1% are caused by a global MPI_Allreduce. Every process uses the point-to-point operations to communicate with 4, 5 or 6 neighbors and rank 0.

Octopus The last analyzed application, Octopus, is a part of the Time-dependent density functional theory (TDDFT) package which solved the time-dependent Schrödinger equation in real-space. The application ran on 64 nodes for 258 seconds and a communication overhead of 10.5% was measured. Most of this time was spent in MPI_Allreduce (61.9%) and MPI_Alltoallv (21.9%) on all processors.

Application Conclusions and Summary The five analyzed applications spend most of their communication time in regular neighbor or collective communications. Collective communications were reductions, broadcasts, reductions-to-all and all-to-all and usually performed with all processes in the communication context (communicator). We also identified point-to-point patterns with 4 to 6 neighbors. Thus, we conclude that we can express the network patterns of many real-world applications that exist today by simulating collective communication and nearest neighbor point-to-point patterns. The following section describes common patterns for the implementation of collective communications based on point-to-point messages.

5 Application Communication Simulation

A common way to implement collective operations is to use algorithms based on point-to-point communication. Multiple algorithms exist and are used in different scenarios. A general rule for algorithm selection is that small-message all-to-one or one-to-all operations (e.g., broadcast or reductions) use tree-like communication schemes and large versions of those operations use double-trees or pipelined (ring) communication schemes. All-to-all communication schemes (e.g., reduce-to-all or personalized all-to-all) usually implement the dissemination algorithm [8] or also a pipelined ring scheme. A more detailed analysis of algorithm selection can be found in [23].

5.1 Simulating Collective Patterns

To examine the effect of fabric congestion on applications we simulate different collective traffic patterns and record the oversubscription π per route. Most optimized collective communication patterns consist of multiple communication stages r (aka rounds, e.g., the dissemination algorithm uses $\lceil log_2 P \rceil$ rounds). Figure 6 shows the communication pattern of the dissemination and tree algorithm for 7 processes as an example. Every of those rounds reflects a different communication pattern that is performed on the network.

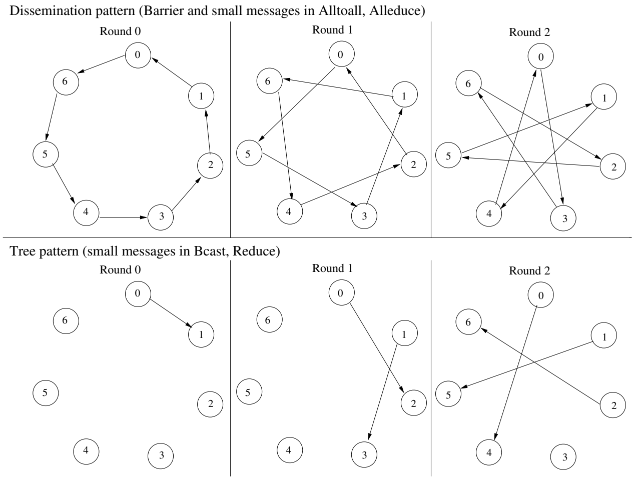
Dissemination pattern (Barrier and small messages in Alltoall, Alleduce)
Round 0
Round 1
Round 2
Tree pattern (small messages in Bcast, Reduce)
Round 0
Round 1
Round 2
0
1
2
3
4
5
6

Figure 6. Collective communication pattern for the dissemination and tree algorithm

We extended the pattern generator to also generate collective communication patterns such as dissemination, pairwise exchange, tree, pipeline, ring, scatter and gather patterns. Those patterns usually have multiple communication rounds with a specific pattern for each round.

Our simulator accepts a pattern and generates a random mapping from each rank in the input pattern to an endpoint[1] in the network. Then it simulates all communication rounds for this mapping. However, the merging of the $r \cdot P/2$ results for each multi-round pattern is not trivial because each link might have a different oversubscription factor π. Thus, we apply two strategies that determine an upper and lower bound to the communication bandwidth:

1) we use the maximum congestion factor π for every round and sum it up to the final result.

$$\pi_{sum\_max} = \frac{1}{r} \cdot \sum_{i=1}^{r} \max_{k}(\pi_{i,k})$$

This represents the most pessimistic case where all processes have to synchronize (wait for the slowest link) at the end of each round.

2) we use the average congestion factor of each round and sum them up to the final result.

$$\pi_{sum\_avg} = \frac{1}{r} \cdot \sum_{i=1}^{r} \frac{2}{P} \cdot \sum_{k=1}^{P/2} \pi_{i,k}$$

This represents, similar to our definition of the "effective bisection bandwidth", the optimistic case where no synchronization overhead occurs and every process can proceed to the next round without waiting for other processes.

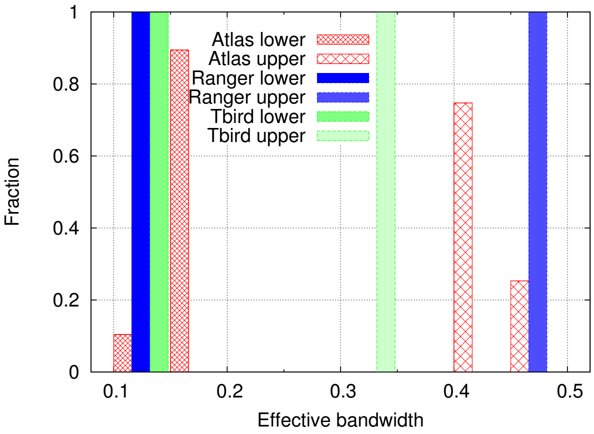

Figure 7. Dissemination pattern simulation results

The results of N different rank-to-node mappings are then combined into equally sized bins and plotted as histograms (we used $N = 10^5$ in our simulations). The normalized height of any histogram bin shows the fraction of mappings wich showed the specific effective bandwidth.

The dissemination, ring and recursive doubling simulation results are rather similar and we present only the dissemination pattern in Figure 7 due to space restrictions. The dissemination pattern uses $\lceil log_2 P \rceil$ communication rounds to perform the operation as shown in Figure 6. The lower bound in the histogram shows that the average of the minimum bandwidths of all rounds (strategy 1) is as small as 10-15%. The upper bound shows the bandwidths with around 40-50% in the optimistic estimation (strategy 2). The "effective bandwidth" (the average of the "upper" simulations) for random rank-to-node mappings of the dissemination pattern is 41.9%, 40.2% and 27.4% for the Ranger, Atlas and Tbird systems respectively.

The tree pattern, depicted in Figure 8, shows the best results because it does not leverage the network fully (each one of the $\lceil log_2 P \rceil$ rounds doubles the number of communicating peers beginning from 1 while all peers communicate from round 1 in most other patterns, cf. Figure 6). The "effective bandwidths" of the tree pattern for the Ranger, Atlas and Tbird system were 69.9%, 71.3% and 57.4% respectively.

The nearest-neighbor communication simulation results — assuming 6 simultaneous neighbor communications — are shown in Figure 9. This limits the bandwidth due to congestion at the endpoints to $1/6$. Thus, we scaled the results by a factor of 6 to isolate the effect of fabric congestion. We see a huge gap between the tightly synchronized communication (lower) and the optimistic measure (upper). The "effective bandwidths" are about 62.4%, 60.7% and 37.4% for the Ranger, Atlas and Tbird system respectively.

[1] we focus on networking effects in this work and thus we only simulate the single process per node case

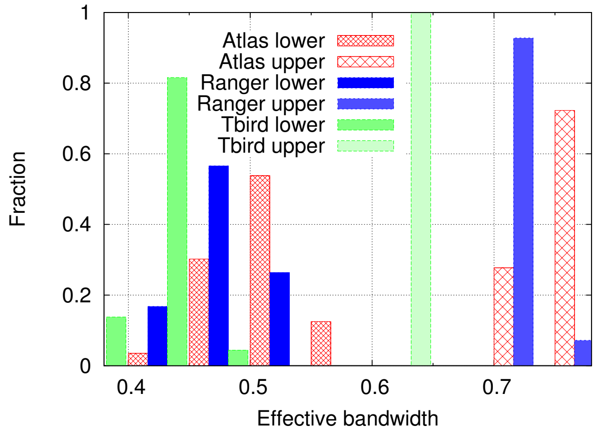

Figure 8. Tree pattern simulation results

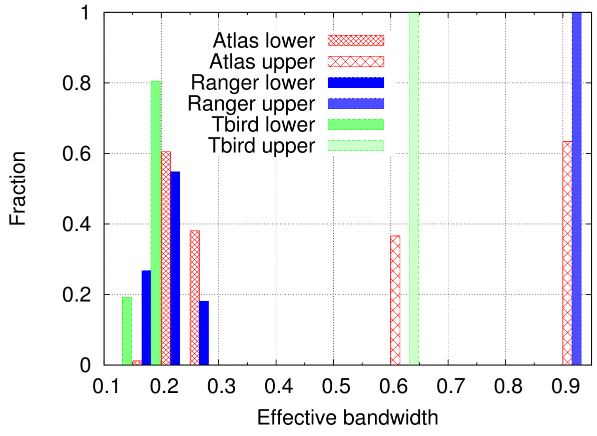

Figure 9. Nearest neighbor communication with 6 neighbors

5.2 Influence on Applications

With our simulation results and with several simplifying assumptions, we can make some approximate statements about the performance penalty due to the routing effects of the four analyzed applications. We assume that the applications are written in a scalable way such that a weak scaling problem keeps the communication overhead approximately constant at large scale and that collective algorithms like reductions, broadcast and alltoall operations are implemented in a tree-based scheme to favor small messages. We use our simulation results to extrapolate the measured communication overhead to the full effective bisection bandwidth case. Then, we calculate the difference in running time of each application. For example POP spent 27.4% of its execution time in neighbor communications with up to 6 neighbors and 4.6% in an allreduce (tree) operation. The average bandwidths for those communication patterns on the largest simulated system (Tbird) are about 37.4% and 57.4%. This means that this communication for those patterns would be 2.67 and 1.74 times faster with real full bisection bandwidth. This would decrease a hypothetical application running time of 100s to only 80.89s, i.e., would reduce the application running time to 80.89% of the original time. The following table shows an estimation of the possible application improvements on the three simulated systems (using the whole network with one process per node) assuming they could leverage full effective bisection bandwidth:

Application	Overhead	Ranger	Atlas	Tbird
MPQC	9.2%	97.23%	97.36%	96.08%
MIMD	9.4%	96.87%	97.73%	94.81%
POP	32.6%	88.32%	87.91%	80.89%
Octopus	10.5%	97.18%	97.23%	95.79%

The rated bandwidth of the Tbird system is only $1/2$ bisection bandwidth. However, we would still argue that the nearest neighbor and tree-based communication can be efficiently embedded into the network (assuming a good node-to-host mapping) such that the application effectively has (for this pattern/mapping) full bandwidth. Thus, we compare in the results in the table to full bandwidth. However, if we assume an arbitrary mapping (i.e., that the rated bandwidth is a strict limit), then the gain in application performance for all application on the Tbird system halves.

We can conclude that the effect on application performance is significant and that the usage of hot-spot avoiding routing schemes would be very beneficial for real-world applications.

6 Conclusions and Future Work

We showed that the original definition of full bisection bandwidth [9] does not take the network routing into account and might thus not be very meaningful for practical applications. Thus, we defined a more application performance oriented measure, the “effective bisection bandwidth” which takes routing into account. We demonstrated a benchmark to perform this measurement. We also propose a simulation methodology and simulate three existing InfiniBand clusters with 528, 1142 and 4391 nodes. Our results show that none of those systems achieves more then 61% of the theoretical bisection bandwidth. We also analyzed different real-world applications for their communication patterns and simulated those patterns for the three analyzed systems. A rough estimation of the communication behavior showed that the communication time of those applications could nearly be halved and the time to solution could be optimized by more than 12% with a network offering full effective bisection bandwidth.

Our results also show that an optimized process-to-node layout as offered by topological MPI communicators might result in a significant performance increase.

We are going to continue the analysis of the effects of routing in InfiniBand networks. Next steps will consider effective process placement and mapping, layout of routes and topology-aware optimization of collective operations.

Acknowledgments

The authors want to thank Jeff Squyres (Cisco), Ralf Wunderlich (FH Zwickau), Douglas Gregor (Indiana University), Patrick Geoffray (Myricom) and Christian Bell (Qlogic) for helpful comments. The authors also want to thank Frank Mietke (TUC) and Wolfgang Rehm (TUC) for granting access to the CHiC cluster system was used as an input system and to run most of the simulations. Thanks to Adam Moody and Ira Weiny (LLNL) who provided the Atlas system topology, Matt Leininger (LLNL) who provided an input file for the application MPQC, and Christopher Maestas (Sandia) who provided the input file for the Thunderbird cluster. This work was partially supported by a grant from the Lilly Endowment, National Science Foundation grant EIA-0202048 and a gift the Silicon Valley Community Foundation on behalf of the Cisco Collaborative Research Initiative.

References

[1] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman. LogGP: Incorporating Long Messages into the LogP Model. *Journal of Parallel and Distributed Computing*, 44(1):71–79, 1995.

[2] V. E. Benes. *Mathematical Theory of Connecting Networks and Telephone Traffic*. Academic Press, New York, 1965.

[3] R. V. Boppana and S. Chalasani. A comparison of adaptive wormhole routing algorithms. *SIGARCH Comput. Archit. News*, 21(2):351–360, 1993.

[4] S. A. Browning. *The tree machine: a highly concurrent computing environment*. PhD thesis, Pasadena, CA, USA, 1980.

[5] C. Clos. A study of non-blocking switching networks. *Bell System Technology Journal*, 32:406–424, 1953.

[6] Z. Ding, R. R. Hoare, A. K. Jones, and R. Melhem. Level-wise scheduling algorithm for fat tree interconnection networks. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 96, Tampa, Florida, 2006. ACM.

[7] A. Friedley, T. Hoefler, M. L. Leininger, and A. Lumsdaine. Scalable High Performance Message Passing over InfiniBand for Open MPI. In *Proceedings of 2007 KiCC Workshop, RWTH Aachen*, December 2007.

[8] D. Hengsen, R. Finkel, and U. Manber. Two Algorithms for Barrier Synchronization. *Int. J. Parallel Program.*, 17(1):1–17, 1988.

[9] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, 2003.

[10] T. Hoefler, T. Schneider, and A. Lumsdaine. Accurately Measuring Collective Operations at Massive Scale. 04 2008. Accepted for publication at the PMEO-PDS08 workshop at the IPDPS08.

[11] C. L. Janssen, I. B. Nielsen, M. L. Leininger, E. F. Valeev, and E. T. Seidl. The massively parallel quantum chemistry program (mpqc), version 2.3.0, 2004. Sandia National Laboratories, Livermore, CA, USA.

[12] M. J. Koop, S. Sur, Q. Gao, and D. K. Panda. High performance MPI design using unreliable datagram for ultra-scale InfiniBand clusters. In *Proceedings of the 21st annual international conference on Supercomputing*, pages 180–189, New York, NY, USA, 2007. ACM Press.

[13] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann Publishers, San Mateo, CA, USA, 1992.

[14] C. E. Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Trans. Comput.*, 34(10):892–901, 1985.

[15] C. E. Leiserson and B. M. Maggs. Communication-efficient parallel algorithms for distributed random-access machines. *Algorithmica*, 3:53–77, 1988.

[16] X. Lin, Y. Chung, and T. Huang. A multiple lid routing scheme for fat-tree-based infiniband networks. In *Proceedings of the 18th IEEE International Parallel and Distributed Processing Symposium (IPDPS04)*, page 11a, Sana Fe, NM, 04 2004.

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

[18] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. In *ACM Trans. on Modeling and Computer Simulations*, 1998.

[19] F. Petrini, J. Fernandez, E. Frachtenberg, and S. Coll. Scalable collective communication on the asci q machine. In *Hot Interconnects 12*, 08 2003.

[20] F. Petrini and M. Vanneschi. K-ary n-trees: High performance networks for massively parallel architectures. Technical report, 1995.

[21] G. Pfister, M. Gusat, W. Denzel, D. Craddock, N. Ni, W. Rooney, T. Engbersen, R. Luijten, R. Krishnamurthy, and J. Duato. Solving hot spot contention using infiniband architecture congestion control. In *Proceedings HP-IPC 2005*, Research Triangle Park, NC, 6 2005.

[22] G. F. Pfister and V. A. Norton. "Hot Spot" Contention and Combining in Multistage Interconnection Networks. In *ICPP*, pages 790–797, 1985.

[23] J. Pjesivac-Grbovic, T. Angskun, G. Bosilca, G. E. Fagg, E. Gabriel, and J. J. Dongarra. Performance Analysis of MPI Collective Operations. In *Proceedings of the 19th International Parallel and Distributed Processing Symposium, 4th International Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS 05)*, Denver, CO, April 2005.

[24] J. Sancho, J. Flich, A. Robles, P. Lopez, and J. Duato. Performance evaluation of up*/down* routing using virtual channels for infiniband networks. In *Actas de las XII Jornadas de Paralelismo*, Valencia, Espaa, 2001.

[25] M. D. Schroeder, A. Birell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterthwaite, and C. Thacker. Autonet: A high-speed, self-configuring local area network using pointto -point links. *IEEE Journal on Selected Areas in Communications*, 9(8), 10 1991.

[26] G. M. Shipman, T. S. Woodall, R. L. Graham, A. B. Maccabe, and P. G. Bridges. Infiniband scalability in open mpi. In *Proceedings of IEEE Parallel and Distributed Processing Symposium*, April 2006.

[27] S. Sur, M. J. Koop, and D. K. Panda. High-performance and scalable mpi over infiniband with reduced memory usage: an in-depth performance analysis. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 105, New York, NY, USA, 2006. ACM.

[28] D. Turner, A. Oline, X. Chen, and T. Benjegerdes. Integrating new capabilities into netpipe. In J. Dongarra, D. Laforenza, and S. Orlando, editors, *Recent Advances in Parallel Virtual Machine and Message Passing Interface,10th European PVM/MPI Users' Group Meeting, Venice, Italy, September 29 - October 2, 2003, Proceedings*, volume 2840 of *Lecture Notes in Computer Science*, pages 37–44. Springer, 2003.

[29] A. Vishnu, M. Koop, A. Moody, A. R. Mamidala, S. Narravula, and D. K. Panda. Hot-spot avoidance with multi-pathing over infiniband: An mpi perspective. In *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, pages 479–486, Washington, DC, USA, 2007. IEEE Computer Society.

[30] E. Zahavi. Optimized infiniband fat-tree routing for shift all-to-all communication patterns. In *Proceedings of the International Supercomputing Conference 2007 (ISC07)*, Dresden, Germany.