Parallel Processing Letters

The Effect of Network Noise on Large-Scale Collective Communications

Torsten Hoefler, Timo Schneider, and Andrew Lumsdaine
Open Systems Laboratory
Indiana University
Bloomington IN 47405, USA
{*htor,timoschn,lums*}*@cs.indiana.edu*

Received July 2009
Revised August 2009
Communicated by A.K. Jones

ABSTRACT

The effect of operating system (OS) noise on the performance of large-scale applications is a growing concern and ameliorating the influence of OS noise is a subject of active research. A related problem is that of network noise that arises from the shared use of the interconnection network by parallel processes of different allocations or other background activities. To characterize the effect of network noise on parallel applications, we conducted a series of experiments with a specially crafted benchmark and simulations. Experimental results show a decrease in the communication performance of a parallel reduction operation by a factor of 2 on 246 nodes on an InfiniBand fat-tree and by several orders of magnitude on a BlueGene/P torus. Simulations show how influence of network noise grows with the system size. Although network noise is not as well-studied as OS noise, our results clearly show that it is an important factor that must be considered when running and analyzing large-scale applications.

Keywords: network noise, operating system noise, network contention, collective communication performance.

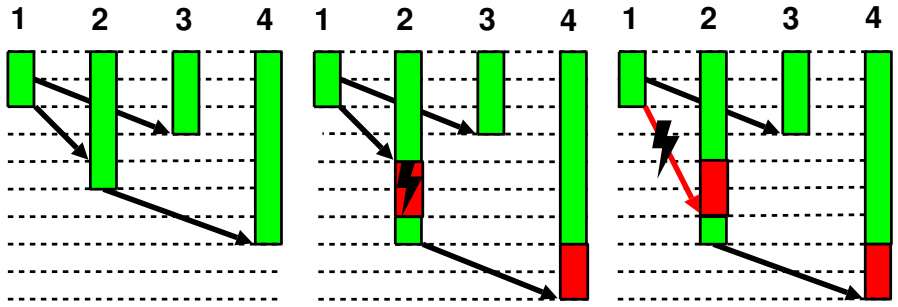
1. Introduction

The influence of external effects on the performance of large-scale parallel application has attracted recent interest [1,8,9,16,24,26]. Even though such effects usually impose a relatively small overhead to applications when run at smaller scales, they can become problematic at larger scale. For example, a single context switch every second is very unlikely to cause a measurable perturbation to a small-scale application run. However, it was shown before that such small periodic events can significantly perturb large-scale applications if they *resonate* with synchronization (caused by communication). The effect of such local perturbations can be multiplied by global (collective) communication operations.

Most existing studies focus on perturbations on the host side, that is, *operating*

system (OS) noise. Some perturbations are caused by resource-sharing between the application process and entities that belong to the computing platform such as OS daemons or monitoring processes. Other delays are caused by external events such as hardware interrupts, translation lookaside buffer (TLB) misses, context switches or cache misses—all of which result from time-sharing the main CPU among different processes. Due to the serial nature and obvious source of such overheads, it is possible to minimize their influence. One possibility would be to use low-noise operating systems specialized for high performance computing such as Catamount [17] or BlueGene Linux [22].

In addition to OS noise, *network noise* can affect parallel application performance in a similar way. Just like processes sharing a computing resource can interfere with each other, parallel processes sharing an interconnection network can suffer from similar interference. The effects of network noise manifest themselves in an application very much like OS noise. Network noise can throttle messages in the network which effectively leads to a lower bandwidth and can delay waiting processes. Figure 1 compares the influence of network noise and OS noise in a parallel application with four processes (1 . . . 4). The black arrows represent communication, green (light) areas show computation, and the red (darker) areas represent overhead caused by OS or network noise, respectively. Timeslices are indicated by horizontal dashed lines.



(a) No Noise (b) OS Noise (c) Network Noise

Fig. 1. Different sources of noise.

In the noiseless case, every process performs some computation for 1 timeslice, then all processes exchange data in a binomial tree pattern with rank 1 as the root. In this pattern, rank 1 sends data to ranks 2 and 3, and rank 2 forwards the message to rank 4. All data transmissions have a latency of 2 timeslices in our example. Between two consecutive transmissions, there is always a gap of 1 timeslice that models the sender's host overhead. Figure 1(a) shows the unperturbed (expected) case. In Figure 1(b), rank 2 can not progress and send data to rank 4 after it received the message from rank 1 because it is delayed (interrupted) by OS noise

for 2 timeslices. So both, rank 2 as well as rank 4 finish 2 slices later than in the noiseless case. Figure 1(c) demonstrates network noise: The data transfer from rank 1 to rank 2 is slowed down by congestion in the network that doubles the latency. As a result, rank 2 and 4 receive their data 2 timeslices later than in the noiseless case.

Other effects, such as the *resonance* observed with OS noise, may also amplify the effects of network noise but require complex interactions of different communications. Similar to OS noise, network noise can be absorbed in synchronization times or can accumulate in communication algorithms. Section 3 discusses absorption and accumulation of network noise in detail.

Whereas OS noise can be modeled with statistical methods [1, 9] or signal processing methods [30], modeling network noise seems much harder. Common sources of network noise are: different applications that share the same network, file I/O operations, or monitoring activities. The network topology and the network routing also play an important role in this context and make a precise prediction or modeling of the perturbation hard.

The main contributions of this work are:

(1) The design of a microbenchmark to assess the influence of random background-communication on specific communication patterns.
(2) Benchmark results for fat-tree (InfiniBand) and three-dimensional torus topologies (BlueGene/P).
(3) A simulation methodology that can be applied to arbitrary networks and communication patterns.
(4) Simulation results for existing as well as artificial fat-tree and torus systems.

We describe the benchmark and present results for different supercomputer systems in Section 2. In Section 3 we show a simulation methodology to assess the influence of network noise on existing large-scale networks. In Section 4 we present a way to generate future large-scale networks based on current established design principles and we assess their properties with regards to network noise. Finally, in Section 5, we talk about the design and simulation of large-scale torus topologies.

1.1. *Related Work*

Petrini, Kerbyson, and Pakin discuss in [26] that frequent short noise intervals together with a synchronizing collective operation such as MPI_Allreduce can cause a dramatic performance loss at large scale. In this particular case, the performance loss was caused by the resonance between the fine-grained OS noise and the synchronizing global communication.

Several studies, such as [1, 8, 9, 16, 24, 29], measure, model and quantify the influence of OS noise on parallel applications. However, no previous study considers the network as a source of additional noise.

Braccini, Del Bimbo and Vicario [4] examine the effect of network load on the

bandwidth of point-to-point TCP/IP connections by inducing load on the data link layer in small Ethernet networks. Kerbyson demonstrated the sensibility of application performance to bandwidth and latency at scale by changing the link-speed of an InfiniBand network in [18].

Badia, Labarta, and Gimenez [3] model the performance of parallel Message Passing Interface (MPI) applications based on traces. The used network model considers a varying traffic function that influences the transmission speed. However, their work does not investigate the influence of background traffic.

Sottile, Chandu, and Bader [31] also model the performance of parallel applications based on MPI traces. The authors recognize the effect of network noise and use a simple statistical model to assess the variation in latency and bandwidth. The applied model for collective communication only assumes $log_2(P)$ message transmissions on P processes. Our work extends this model by fully simulating collective algorithms and the influence of the interconnection topology. We did not choose the tracing approach to allow for simple extrapolation of communication algorithms to large process counts.

Mraz showed in [23] that OS noise can cause delays for global communication patterns. He uses a ring pattern that represents the worst case in this spectrum. Mraz concludes that the observed delays result either from other processes or from interrupt processing without considering variances in the network transmission.

In a previous work [14], we investigated the influence of static routing and congestion to large scale networks. This analysis only considered congestion in a given allocation without external perturbations or dependencies between messages in collective operations. In this work, we extend the model to give a more accurate dependency-based prediction of the running time of collective operations. We also analyze the effects of perturbations from other applications in the network.

2. Design of a Microbenchmark

We describe a microbenchmark scheme to measure the influence of network noise on collective communication. For this, we assume that communication of different applications, input/output operations, or monitoring services are the biggest source of network noise on parallel systems. Possible ways to prevent different applications from interfering are to run at most one application at any given time or to ensure that all applications use disjoint parts of the network. Although partitioning seems simple for some network topologies such as tori (one can allocate a partition of the network that is not crossed by other processes), it can be hard for topologies such as fat-trees that use a common "backplane". Partitioning a toroidal system into network disjoint allocations often comes at the cost of system utilization. Commonly used greedy allocation strategies often degenerate to a pseudo-random allocation (through fragmentation) after scheduling a sufficient number of different jobs. Thus, network noise is a very common phenomenon on current parallel systems.

We investigate MPI collective operations that play a critical role for many large-

scale applications. Many applications, such as POP [5], CTH [11] and SAGE [19] rely on global allreductions which act as implicit synchronization points. Ferreira, Bridges, and Brightwell showed in [8] that those applications are significantly influenced by OS noise. Based on those results and the application study by Rabenseifner [27], we focus our investigation on the collective operations MPI_Allreduce, MPI_Reduce and MPI_Bcast. For our studies we assume the communication of small data because large-scale applications commonly communicate very small number of elements. POP, for example, relies on a global 8-byte reduction operation.

For our benchmark, we assume that it is unlikely to get a noise-less partition. Thus, we have to accept the usual background network noise and we choose a proactive approach which generates noise on top of the normal background noise. For this, we assume that the background noise follows a normal distribution during the run of the benchmark, which enables us to consider the differences to our artificially generated noise. We call the measured noise *perturbation* in the following.

Our benchmark simulates the collective communication of an MPI application and a background noise pattern. To achieve this, we split the full allocation (MPI_COMM_WORLD) into two process-disjoint communicators: the application communicator and the perturbation communicator. The ratio between the size of the application communicator and the perturbation communicator will be called *perturbation ratio* in the following. For example, a perturbation ratio of 0.1 in an allocation of 100 processes means that 10 of the processes execute a perturbation communication and 90 other processes benchmark a collective communication. The selection of MPI_COMM_WORLD ranks for each communicator is randomized in order to model fragmented allocations. The random partitioning of MPI_COMM_WORLD is repeated before each benchmark.

The communicators are "warmed up" (cf. Gropp's comments on correct benchmarking [10]) before each use. The benchmark uses the collective benchmarking principles and the synchronization scheme proposed in [13]. Each iteration starts with a global synchronization. All processes in the perturbation communicator communicate with a random large-message[a] permutation pattern. Simultaneously, all processes in the independent application communicator measure the time that is needed to perform a specific collective MPI operation. Then, in a second step, the benchmark of the collective operation is repeated without the perturbing communication. This procedure is repeated multiple times with different communicators and perturbation ratios and analyzed statistically to show the influence of network noise. The benchmark is schematically shown in Figure 2.

2.1. *Benchmark Results on Fat-Trees*

We implemented the described benchmark scheme in the open-source tool Netgauge [12] and performed several tests on the *CHiC* cluster system. The *CHiC*

[a]We used 10 MiB in all our benchmarks.

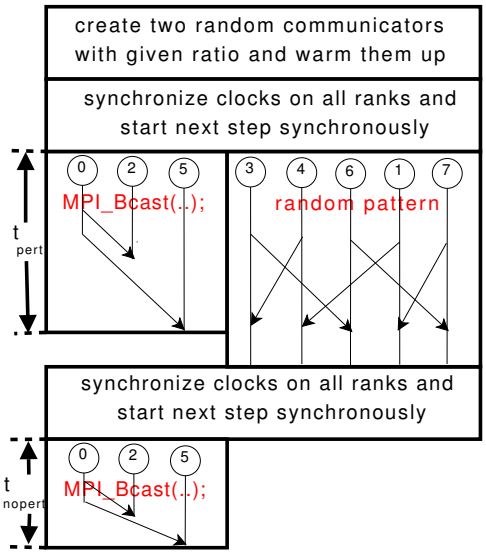


Fig. 2. One round of the used benchmarking scheme on 8 processes and a perturbation ratio of $5/8$=0.625. Two process-disjoint communicators perform perturbation and collective communication, respectively.

system comprises 528 quad-core nodes connected with a full bisection bandwidth SDR InfiniBand fat-tree network. InfiniBand uses crossbar switches with 24 ports. All benchmarks were done with Open MPI 1.2.8 and with one process per node to keep the effect of OS noise minimal. We also performed runs with multiple processes per node that showed much higher perturbations. However, it is unclear if this perturbation results from contention while accessing the local network interface, OS noise, or network noise. Thus, we present only benchmarks with one process per node which isolates the effect of network noise.

First, we prove the existence of network noise in a small allocation of 32 nodes. As described earlier, the benchmark randomly splits the 32 nodes into two disjoint communicators of varying sizes and measures an 8-byte allreduction operation in one communicator and a perturbation communication (point-to-point exchanges of 10 MiB messages) in the other communicator. We repeated the operation with 128 random splits for each perturbation ratio and recorded each measurement point. Figure 3 shows a boxplot of the different times that are needed to finish the allreduction with varying communicator sizes. The boxplot shows the distribution of single measurements for the perturbed and unperturbed run. The jump at 16 processes indicates a protocol or algorithm change in the collective implementation of Open MPI. The horizontal black bars in the middle of each box represent the sample median, while the upper and lower end of the box indicates the 25th and 75th percentile. The whiskers show the total range of the sample. The notches in the middle of the boxes indicate whether two samples show a statistically significant

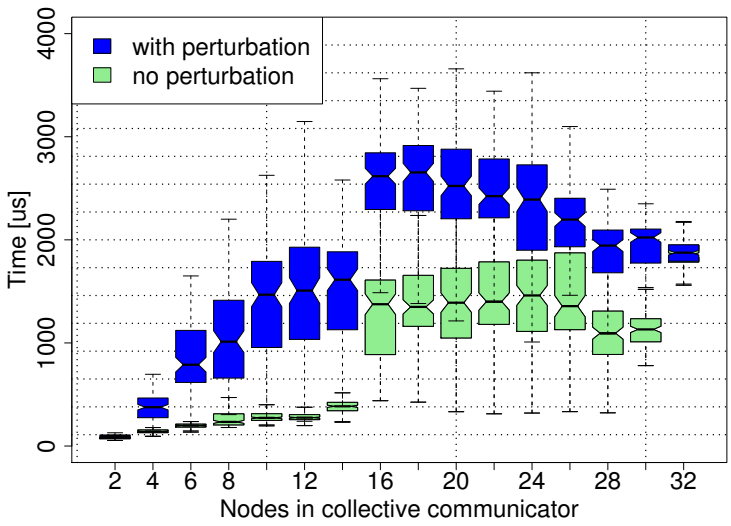


Fig. 3. Boxplot of network noise in a 32-node allocation on an InfiniBand fat-tree system while performing an allreduction operation with varying perturbation ratio.

difference. The samples are different with a 95% confidence if the notches do not overlap [7]. This shows that the effect of background contention (network noise) on randomly permuted allocations is significant on fat-tree networks with full bisection bandwidth.

In the second experiment, we aim to quantify the influence with a larger number of processes and different perturbation ratios. Figure 4 shows the relative slowdown caused by perturbation. Each data-point represents the average time of 128 measurements. The points have a high variance because of other jobs on the system (background network noise) and the relatively small set of the random mappings. We used the least squares method to fit a linear function to each set of measurements. This fit only demonstrates the general trend with growing perturbation ratio. We see that even a relatively small perturbation is able to cause significant delays of up to 50% of the latency of the measured collective operations.

We remark again that the influence on the communication performance is significant even though the perturbation communication and the collective operation run in different process-disjoint communicators. Our benchmarks also show how much small jobs on large cluster systems could be delayed due to the traffic of larger jobs or even a collection of other small jobs.

In the third experiment, we investigate the scaling of the influence of network noise with the number of used nodes in the cluster and a fixed perturbation ratio. We used the same benchmark methodology and present the scaling of an allreduction with a fixed perturbation ratio of $1/2$ in Figure 5. As expected, the effect of network noise grows with the number of communicating processes. The dent in the curve is reproducible on the *CHiC* system.

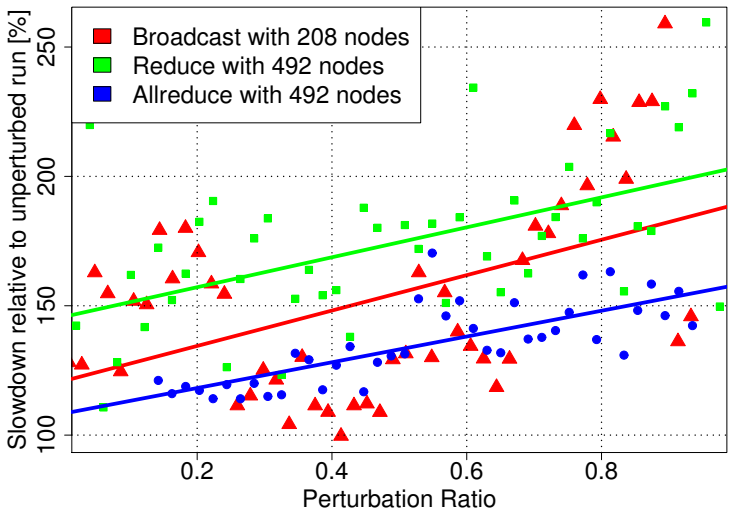


Fig. 4. Average slowdown due to network noise for fixed allocations with different perturbation ratios on an InfiniBand fat-tree.

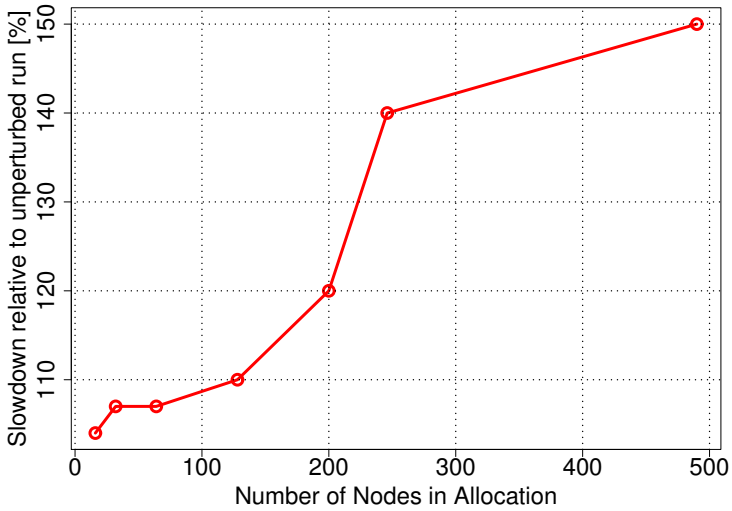


Fig. 5. Slowdown of an allreduction operation with increasing allocation size and a fixed perturbation ratio of 1/2 on an InfiniBand fat-tree with full bisection bandwidth.

2.2. *Benchmark Results on Torus Networks*

We showed that network noise has a significant effect even on full bisection bandwidth InfiniBand networks. This is due to the deterministic oblivious routing strategy in InfiniBand as explained in [14]. Assuming the same model, we would expect a much higher slowdown on toroidal networks.

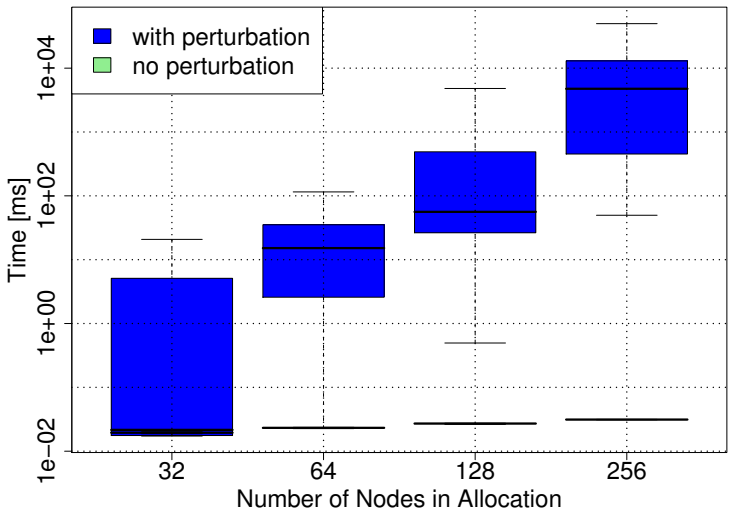


Fig. 6. Boxplot of the collective latencies with background noise and a perturbation ratio of 1/2 on the three-dimensional torus network of a BlueGene/P.

We used our microbenchmark to study the effects on Eugene: a BlueGene/P system at the Oak Ridge National Lab. Eugene consists of 1024 quad-core PPC 450 processors. We used the system in SMP mode, which means that exactly one MPI process is mapped onto each node. We hereby note that the default allocation strategy on BlueGene/P prevents network noise because the job scheduler ensures that each allocation is convex. This means that no message in one allocation will cross another allocation (if dimension-order mesh routing is used). OS noise also is not existent on BlueGene/P systems running the CNK operating system according to Yoshii et al. [33]. Thus, we must consider the following benchmark results carefully.

All measurement results on the BlueGene/P system had a very high variance in the perturbed case while the unperturbed runs were nearly noise-free. Because of the variance in the perturbation run, we plotted the graphs as boxplots and real times. We show the benchmark results of 50 runs with random mappings and a perturbation ratio of 1/2 in Figure 6. The median is represented by the middle horizontal line, the height of the box shows the 75th percentile and the upper and lower whiskers the maximal values. Note that the low variance for the runs without perturbation makes the boxes appear as horizontal lines only. The measurements reveal that perturbation communication on a random process mapping significantly influences the running time of collective communications on BlueGene/P. The large variations (up to several orders of magnitude) seem to indicate that the network scheduling could be potentially improved. It appears that the large (10 MiB) messages of the perturbation communication are able to delay small (8-byte) messages more than would be mandated by the bandwidth limits. Thus, we point out that the high delays in the benchmarks seem to be a property of this particular system

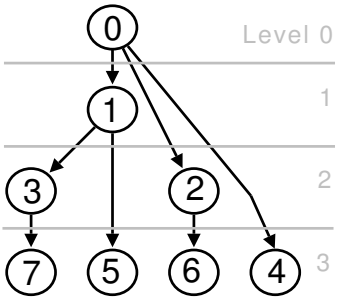


Fig. 7. Binomial tree pattern among 8 processes.

rather than a fundamental property of torus networks. We will analyze the properties of torus networks under the influence of network noise with a simulation in Section 5.

Getting the necessary large allocations to analyze large networks is often not possible or too expensive. Thus, we designed a simulation scheme to model the effects of network noise on communication patterns at large-scale machines. In the following section, we discuss our simulation methodology that enables an abstract analysis of the influence of network noise on large-scale collective communication with different underlying network topologies.

3. Simulating Network Noise

We propose a simple dependency-based simulation scheme to assess the influence of network noise to complex application communication patterns. Communication patterns can be modeled as multiple point-to-point messages with dependencies on their execution order. For example, a small-message broadcast operation is often implemented with a binomial tree. Such an algorithm usually consists of $\lceil log_2(P) \rceil$ communication rounds on P processes. The execution order mandates that each non-root and non-leaf (inner) process receives data before it sends this data to its children. A broadcast tree for 8 processes with 3 communication levels is shown in Figure 7. This scheme allows the construction of a global dependency graph where the send operations depend on previous receives at all inner processes in the tree. The communication starts at the root process (0 in Figure 7) and terminates at the children (4,5,6,7 in Figure 7). We model now the execution of this binomial tree on a 16-node fat-tree network with full bisection bandwidth built from 8-port switches. For our example we assume the random process-to-node mapping $\pi = (3, 6, 5, 13, 7, 9, 0, 10)$ of the application communicator. This means that process 0 is mapped to node 3, process 1 to node 6, process 2 to node 5 and so on. We assume now that we are in the third level of the broadcast tree from Figure 7 and thus, the communicating processes are $\mathcal{P} = \{(3, 7), (1, 5), (2, 6), (0, 4)\}$. If we apply mapping π, then the processes translate to the communicating nodes $\pi(\mathcal{P}) =$

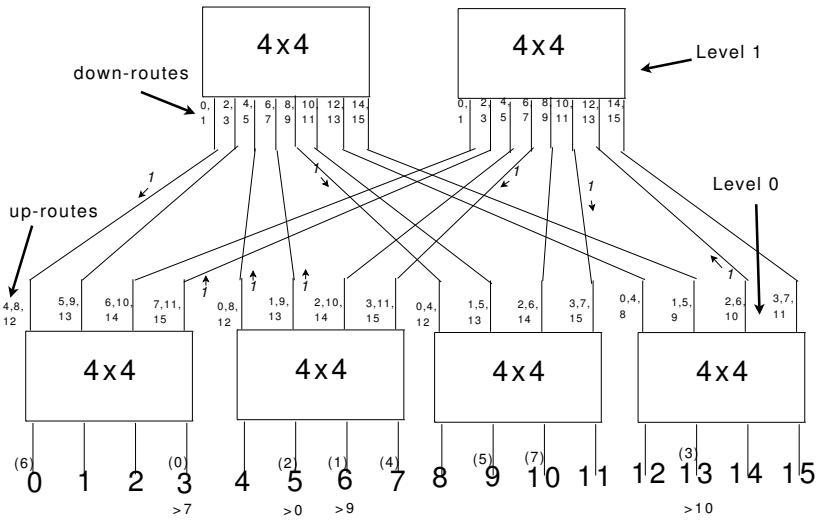


Fig. 8. Schematic example network with 8 processes performing the third level of a binomial broadcast on 8 processes. Processes are mapped randomly to nodes and process numbers are indicated in brackets (e.g., process 6 runs on node 0). Communication peers are prefixed with > and occupied links are indicated with arrows and the number of messages using the link. Each link is annotated with the targets that it routes to.

$\{(13, 10), (6, 9), (5, 0), (3, 7)\}$. We also assume an ideally balanced routing where each link between level 0 and 1 in the fat-tree serves three endpoints and each link between level 1 and 0 serves exactly two endpoints. We can now route each of the four messages separately and mark the channels that they pass. We assume, in our model, that all passed channels are occupied by the message. The node layout, routing, process assignment, and tree-message routing of the third level is shown in Figure 8.

Now we add random perturbation communication. Each node, which is not part of the application communicator, sends a message to a random peer that is also not part of the application communicator. In our example, the perturbation communication consists of the pairs $\mathcal{C} = \{(1, 4), (2, 11), (4, 15), (8, 12), (11, 14), (12, 1), (14, 8), (15, 2)\}$. Each message in the perturbation set $\mathcal{C}$ is also routed through the network. Figure 9 shows the binomial tree example from Figure 8 with added perturbation communication.

This scheme is repeated for each level of the collective communication (whereas the perturbation communication remains constant). We can now annotate the graph in Figure 7 with the maximum congestion for each message along the path that it traverses. For example, the communication from process 0 to process 4 (node 3 to node 7) in Figure 9 has a maximum congestion of 2 along the path (on the up-link between level 0 and 1).

Each edge in the communication graph is now annotated with the maximum con-

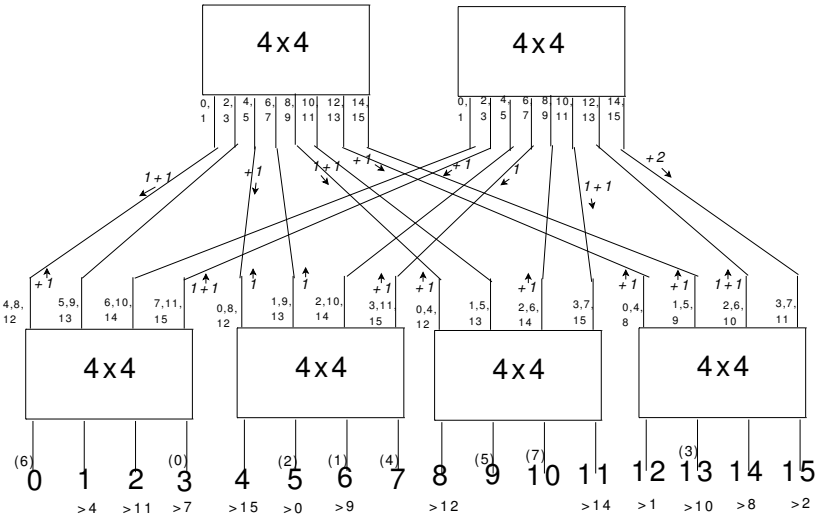


Fig. 9. Schematic example of Figure 8 with perturbation communication prefixed with "+" at the links. Communication peers are indicated with >.

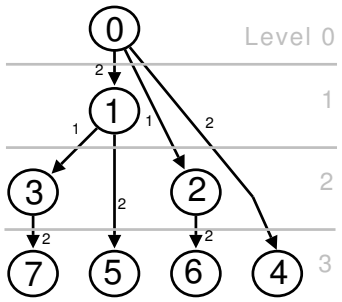


Fig. 10. Binomial tree pattern with 8 processes annotated with the maximum congestion along each path.

gestion along the corresponding logical link in the network simulation. Figure 10 shows the fully annotated binomial tree graph after a simulation of all levels. Then, a breadth first search (BFS) is started at the root node and the longest path in the dependence graph is reported as the time for this collective operation (we assume that the finishing time of the last process of the collective operation is significant). The BFS will find the longest path 0-1-3-7 with a total congestion of 5. The whole simulation is performed with and without perturbation. The congestion of the unperturbed run (3 in this example) can be simply subtracted to get the overhead caused by the perturbation.

With this approach, we model the whole communication and all dependencies.

Thus, our model also captures the effect of noise absorption and accumulation as described by Malony et al. and Wolf and Malony in [21,32]. Both effects happen in our example: The congestion in level 3 between rank 0 and 4 is absorbed (as we see in the BFS), while the congestion between rank 0 and 1, in level 1, and node 3 and 7, in level 3, accumulate.

3.1. *Simulating Real-World Installations*

To assess the influence of network noise to large-scale installations, we used InfiniBand network maps of large real-world systems (generated from `ibnetdiscover` and `ibdiagnet` output as described in [28]). We investigate several large-scale cluster systems. The *Thunderbird* (TBird) system is the largest installation with 4391 endpoints. The second largest system, the *Ranger* system at the Texas Advanced Computing Center, is connected with two 3456 port Sun Magnum switches and had a total of 3908 active endpoints during the query. The third-largest system, in our simulation, the *Atlas* system at the Lawrence Livermore National Lab, had 1142 endpoints when the network structure was queried. The *CHiC* system at the Technical University of Chemnitz, had 566 endpoints during the query and the *Odin* system at Indiana University, has 128 endpoints. All networks are based on fat-tree topologies [20] with 24-port crossbars. Thunderbird is designed with half bisection bandwidth and all other systems have full bisection bandwidth. We showed in [14] that the *effective bisection bandwidth* of the systems is significantly lower with 40.6%, 57.6% and 55.6% for Thunderbird, Ranger and Atlas, respectively. We used our network simulator [28] for those simulations, as well as for the simulations in this paper.

Figure 11 shows the average results of 1000 simulation runs of the binomial tree pattern, which is often used for small message reductions and broadcasts. We see clearly that the slowdown caused by network noise increases significantly with the network size and with the perturbation ratio. The simulations resemble the trend in the benchmark results presented in Section 2.1. However, different architectural properties of the real-world networks (e.g., CHiC is not a pure fat-tree or TBird has only half bisection bandwidth) prohibit general statements about scaling with the network size. To be able to make such statements, we generate and analyze fat-tree networks of different sizes and similar properties (full bisection bandwidth) in the next section.

4. Large-Scale Fat-Tree Networks

To simulate future large-scale networks and to evaluate the scaling with the system size, we extended our study with the simulation of large fat-tree networks. We used the recursive method described by Öhring et al. in [25] to generate extended generalized fat-trees (XGFTs). An extended generalized fat-tree XGFT$(h, m_1, ..., m_h, w_1, ...w_h)$ of height h is generated in h recursion steps. In each step s, the XGFT$(s, m_1, ..., m_h, w_1, ...w_h)$ is built by m_s copies of XGFT$(s -$

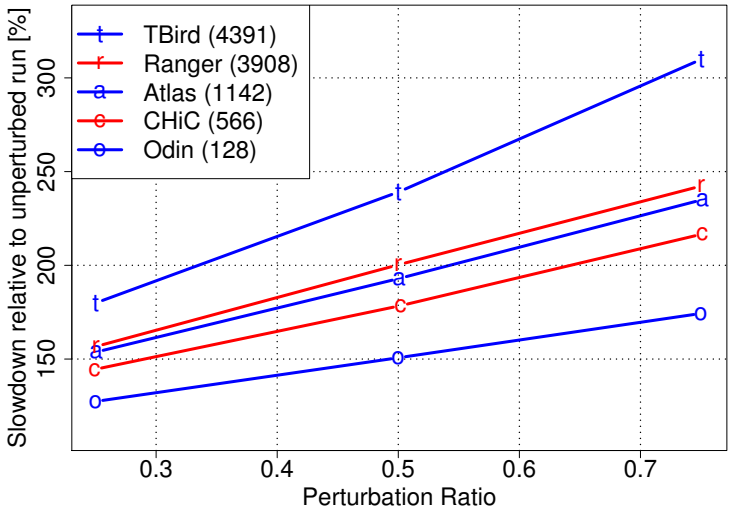


Fig. 11. Simulated networks with tree pattern.

$1, m_1, ..., m_h, w_1, ...w_h)$ and $w_1 \cdots w_s$ additional new top-level nodes. Exact construction and wiring rules can be found in [25] and are omitted due to space restrictions.

All generated trees are built from 24-port switches and are designed to have full bisection bandwidth. The generated fat-tree topologies are shown in Table 1. Figure 12 shows the topology for an exemplary XGFT$(2, 12, 6)$ with 12 leaf switches

Table 1: Layout of the simulated fat-tree networks.

Layout	**# Endpoints**	**# Switches**
XGFT(2,12,6)	144	18
XGFT(2,24,12)	288	36
XGFT(3,12,8,12,4)	1152	240
XGFT(3,12,16,12,8)	2304	480
XGFT(3,12,24,12,12)	3456	720
XGFT(4,12,12,6,12,12,3)	10368	3024
XGFT(4,12,12,12,12,12,6)	20736	6048

and 144 ports. We used the fat-tree optimized routing of OpenSM [34] together with `ibsim` to generate realistic routes for the networks.

4.1. *Simulation Results*

We analyzed the influence of network noise with growing network sizes and a fixed perturbation ratio of 1/2. Figure 13 shows the simulated influence of network noise

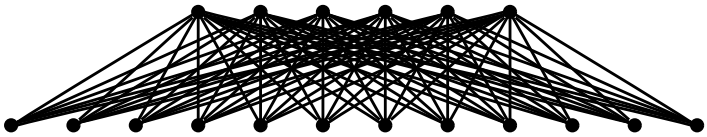


Fig. 12. An XGFT$(2, 12, 6)$ network.

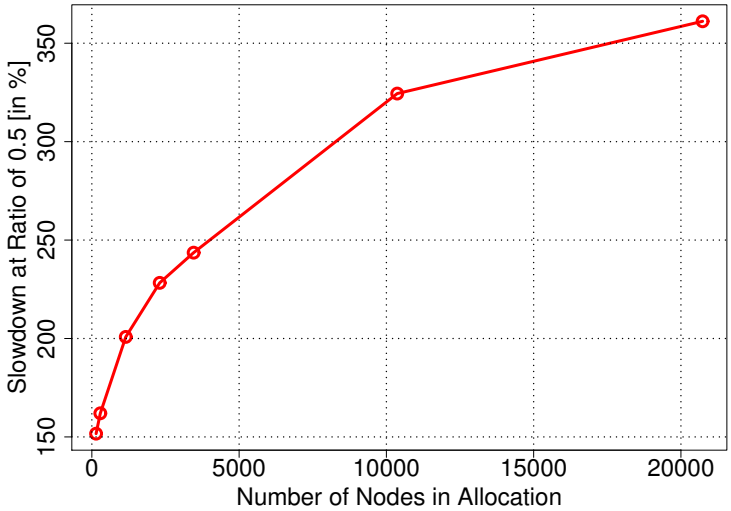


Fig. 13. Slowdown with increasing communicator size and a fixed perturbation ratio of 1/2.

to application runs on half of the nodes of fat-tree networks while the other half communicates randomly, as described in Section 3. Our simulation results resemble the trend in the benchmark results from Section 2.1

5. Large-Scale Toroidal Networks

We also generated different toroidal networks to analyze the effects of network noise on such architectures. The advantages of toroidal networks (k-ary n-cubes) with low dimensionality are discussed by Dally [6]. We analyze two- and three-dimensional implicit torus networks with different numbers of endpoints. Such k-ary n-cubes can be constructed by assigning an n-digit number to each endpoint. Each digit ranges from $0 \ldots k-1$. Figure 14 shows a 3-ary 2-cube. Two endpoint are connected if their numbers differ by exactly one digit (Hamming distance equals one) and if this digit differs by 1 (the digits of each dimension form a ring $0 \ldots k-1$, so all computations are modulo k).

We used the OpenSM dimension order routing to route the networks. Thus, our simulation resembles destination-based routing as it is used in InfiniBand. Table 2

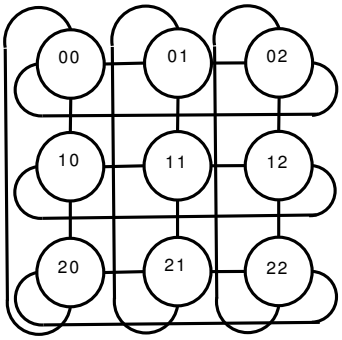


Fig. 14. 3-ary 2-cube network with endpoint numbering.

shows the detailed layouts of all simulated k-ary n-cubes.

Table 2: The simulated k-ary n-cube networks.

2-d Layout	**# Endpoints**	**3-d Layout**	**# Endpoints**
10-ary 2-cube	100	4-ary 3-cube	64
14-ary 2-cube	196	5-ary 3-cube	125
17-ary 2-cube	289	6-ary 3-cube	216
20-ary 2-cube	400	7-ary 3-cube	343
22-ary 2-cube	484	8-ary 3-cube	512
24-ary 2-cube	576	9-ary 3-cube	728
26-ary 2-cube	676	10-ary 3-cube	1000
28-ary 2-cube	784	12-ary 3-cube	1728
30-ary 2-cube	900	14-ary 3-cube	2744
31-ary 2-cube	961	15-ary 3-cube	3375
44-ary 2-cube	1936	17-ary 3-cube	4912
54-ary 2-cube	2916	18-ary 3-cube	5832
70-ary 2-cube	4900	19-ary 3-cube	6859
77-ary 2-cube	5929	20-ary 3-cube	8000
83-ary 2-cube	6889		
89-ary 2-cube	7921		

5.1. *Simulation Results*

We analyze the influence of network noise on a tree pattern with growing perturbation ratio and a fixed network size of 10000 (2d) and 8000 (3d) endpoints in Figure 15, respectively. The simulation shows that background noise can delay the communication in a tree pattern by up to 12 times. It also shows that three-dimensional torus networks are less affected than two-dimensional networks. This

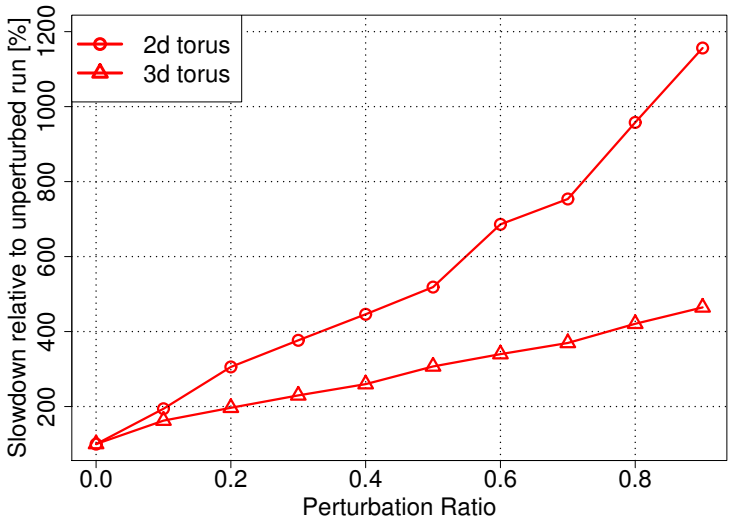


Fig. 15. The slowdown for 100-ary 2-cube and 20-ary 3-cube executing a binomial tree pattern.

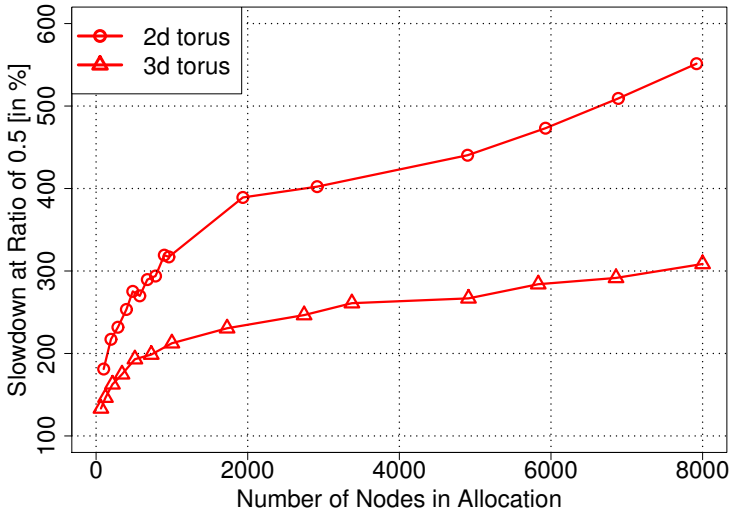


Fig. 16. The influence of background communication on k-ary 2-cubes and k-ary 3-cubes of different sizes and perturbation ratio $1/2$.

is very likely due to the higher bisection bandwidth in three-dimensional networks with the same number of endpoints.

Figure 16 shows the influence of network noise on two- and three-dimensional torus networks of different sizes. The perturbation ratio was fixed to $1/2$ in this simulation. The simulation resembles the results of the previous experiment. Applications on three-dimensional torus networks seem less affected by random network

noise then those on two-dimensional torus networks. This seems rather natural because the higher bisection bandwidth network allows to route some of the random traffic without contention while a network with a low bisection bandwidth forces the traffic on contending paths.

6. Conclusions and Future Work

We show that network noise can have a significant effect on the performance of parallel applications at all scales. Our results indicate that the slowdown grows with the network size and also affects small applications in large networks if allocations are fragmented. The main factors that influence network noise are the network type, routing scheme and the node allocation policy. We conclude that intelligent node allocation strategies are necessary to avoid fragmentation and mitigate the influence of network noise. We point out that the quality of such allocations depends on the network topology and the routing. For example, allocating convex sets of nodes like it is done on BlueGene/P systems in conjunction with dimension order routing along shortest paths effectively avoids network noise from other allocations. Similar strategies seem to be hard to find for fat-tree networks because the higher levels of the tree are shared by many endpoints.

We also point out that network noise must be considered as a source of errors in addition to OS noise when analyzing large-scale application runs.

Interesting future directions are the analysis of the effects of network noise on more topologies and mapping strategies of real-world machines and applications. This directly leads to the idea to design noise-minimizing topology-dependent communication algorithms that could be used in collective operations. We also plan to improve the simulation accuracy, which currently bases in distinct communication levels, to a more precise (and more resource-intensive) LogGP-based models [2, 15].

Acknowledgments

Thanks to Adam Moody and Ira Weiny (LLNL) who provided the Atlas system topology, Christopher Maestas (Sandia) who provided the input file for the Thunderbird cluster, and Len Wisniewski (Sun) and the TACC who provided the Ranger input. The authors also thank Frank Mietke (TUC) for supporting work with the CHiC cluster system. This work was supported by the Department of Energy project FASTOS II (LAB 07-23), a grant from the Lilly Endowment and a gift the Silicon Valley Community Foundation on behalf of the Cisco Collaborative Research Initiative. Thanks to Natalia Berezneva for editorial comments that improved the presentation of the article.

References

[1] S. Agarwal, R. Garg, and N. Vishnoi. The Impact of Noise on the Scaling of Collectives: A Theoretical Approach. In *12th Annual IEEE International Conference on High Performance Computing*, December 2005.

[2] 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.

[3] R. M. Badia, J. Labarta, and J. Gimenez. DIMEMAS: Predicting MPI applications behavior in Grid environments. In *Workshop on Grid Applications and Programming Tools (GGF '03)*, 2003.

[4] A. Braccini, A. Del Bimbo, and E. Vicario. Interprocess communication dependency on network load. *Software Engineering, IEEE Transactions on*, 17(4):357–369, 1991.

[5] K. Bryan. A numerical method for the study of the circulation of the world ocean. *J. Comput. Phys.*, 135(2):154–169, 1997.

[6] W. J. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Trans. Comput.*, 39(6):775–785, 1990.

[7] J. D. Emerson and H. Strenio. Box-plots and batch comparison. Understanding Robust and Exploratory Data Analysis, 1983.

[8] K. B. Ferreira, P. Bridges, and R. Brightwell. Characterizing application sensitivity to OS interference using kernel-level noise injection. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–12, Piscataway, NJ, USA, 2008. IEEE Press.

[9] R. Garg and P. De. Impact of Noise on Scaling of Collectives: An Empirical Evaluation. In Yves Robert et al., editors, *HiPC 2006, 13th International Conference*, volume 4297 of *LNCS*, pages 460–471. Springer, 2006.

[10] W. Gropp and E. L. Lusk. Reproducible Measurements of MPI Performance Characteristics. In *Proceedings of the 6th European PVM/MPI Users' Group Meeting*, pages 11–18, London, UK, 1999. Springer-Verlag.

[11] E. S. Hertel, Jr., R. L. Bell, M. G. Elrick, A. V. Farnsworth, G. I. Kerley, J. M. Mcglaun, S. V. Petney, S. A. Silling, P. A. Taylor, and L. Yarrington. CTH: A Software Family for Multi-Dimensional Shock Physics Analysis. In *in Proceedings of the 19th International Symposium on Shock Waves, held at*, pages 377–382, 1993.

[12] T. Hoefler, T. Mehlan, A. Lumsdaine, and W. Rehm. Netgauge: A Network Performance Measurement Framework. In *High Performance Computing and Communications, HPCC 2007, Houston, USA, Proceedings*, volume 4782, pages 659–671. Springer, Sep. 2007.

[13] T. Hoefler, T. Schneider, and A. Lumsdaine. Accurately Measuring Collective Operations at Massive Scale. In *Proceedings of the 22nd IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Apr. 2008.

[14] T. Hoefler, T. Schneider, and A. Lumsdaine. Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks. In *Proceedings of the IEEE International Conference on Cluster Computing*. IEEE Computer Society, Oct. 2008.

[15] F. Ino, N. Fujimoto, and K. Hagihara. LogGPS: A Parallel Computational Model for Synchronization Analysis. In *PPoPP '01: Proceedings of the eighth ACM SIGPLAN symposium on Principles and practices of parallel programming*, pages 133–142.

[16] K. Iskra, P. Beckman, K. Yoshii, and S. Coghlan. The Influence of Operating Systems on the Performance of Collective Operations at Extreme Scale. In *Proceedings of Cluster Computing IEEE International Conference*, 2006.

[17] S. M. Kelly and R. Brightwell. Software architecture of the light weight kernel, Catamount. In *Cray User Group Annual Technical Conference*, May 2005.

[18] D. J. Kerbyson. A Look at Application Performance Sensitivity to the Bandwidth and Latency of Infiniband Networks. In *in Proc. of Workshop on Communication Architectures for Clusters (CAC), IEEE/ACM Int. Parallel and Distibuted Processing Symposium (IPDPS)*, Rhodes, Greece, March 2006.

[19] D. J. Kerbyson, H. J. Alme, A. Hoisie, F. Petrini, H. J. Wasserman, and M. Gittings. Predictive performance and scalability modeling of a large-scale application. In *Supercomputing '01: Proceedings of the ACM/IEEE conference on Supercomputing*, pages 37–37, New York, NY, USA, 2001. ACM.

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

[21] A. D. Malony and S. S. Shende. Overhead compensation in performance profiling. In *In Proceedings of the European Conference on Parallel Processing (Euro-Par)*, pages 119–132. Springer-Verlag, 2004.

[22] J. E. Moreira, George A., Charles Archer, R. Bellofatto, P. Bergner, J. R. Brunheroto, M. Brutman, J. G. Castanos, P. Crumley, M. Gupta, T. Inglett, D. Lieber, D. Limpert, P. McCarthy, M. Megerian, M. P. Mendell, M. Mundy, D. Reed, R. K. Sahoo, A. Sanomiya, R. Shok, B. E. Smith, and G. G. Stewart. Blue Gene/L programming and operating environment. *IBM Journal of Research and Development*, 49(2-3):367–376, 2005.

[23] R. Mraz. Reducing the variance of point to point transfers in the IBM 9076 parallel computer. In *Supercomputing '94: Proceedings of the 1994 ACM/IEEE conference on Supercomputing*, pages 620–629, New York, NY, USA, 1994. ACM.

[24] A. Nataraj, A. Morris, A. D. Malony, M. Sottile, and P. Beckman. The ghost in the machine: observing the effects of kernel operation on parallel application performance. In *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 1–12, New York, NY, USA, 2007. ACM.

[25] S. R. Öhring, Ma. Ibel, S. K. Das, and M. J. Kumar. On generalized fat trees. In *IPPS '95: Proceedings of the 9th International Symposium on Parallel Processing*, page 37, Washington, DC, USA, 1995. IEEE Computer Society.

[26] F. Petrini, D. J. Kerbyson, and S. Pakin. The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q. In *Proceedings of the ACM/IEEE SC2003 Conference*, page 55. ACM, 2003.

[27] R. Rabenseifner. Automatic MPI Counter Profiling. In *Proceedings of 42nd CUG Conference*, 2000.

[28] T. Schneider and T. Hoefler. ORCS: An Oblivious Routing Congestion Simulator. Technical report, Indiana University, Computer Science Department, February 2009.

[29] M. Sottile and R. Minnich. Analysis of microbenchmarks for performance tuning of clusters. In *CLUSTER '04: Proceedings of the 2004 IEEE International Conference on Cluster Computing*, pages 371–377, Washington, DC, USA, 2004. IEEE Computer Society.

[30] M. J. Sottile. *A measurement and simulation methodology for parallel computing performance studies*. PhD thesis, Albuquerque, NM, USA, 2006.

[31] M. J. Sottile, V. P. Chandu, and D. A. Bader. Performance analysis of parallel programs via message-passing graph traversal. In *20th International Parallel and Distributed Processing Symposium, Proceedings, Rhodes Island, Greece*. IEEE, 2006.

[32] F. Wolf, A. Malony, S. Shende, and A. Morris. Trace-Based Parallel Performance Overhead Compensation. In *In Proc. of the International Conference on High Performance Computing and Communications*, September 2005.

[33] K. Yoshii, K. Iskra, P. C. Broekema, H. Naik, , and P. Beckman. Characterizing the Performance of Big Memory on Blue Gene Linux. Technical report, Argonne National Lab, March 2009. ANL/MCS-P1589-0309.

[34] E. Zahavi, G. Johnson, D. J. Kerbyson, and M. Lang. Optimized InfiniBand Fat-tree Routing for Shift All-To-All Communication Patterns. In *Proceedings of the International Supercomputing Conference 2007 (ISC07)*, Dresden, Germany.