Interference between I/O and MPI Traffic on Fat-tree Networks

Kevin A. Brown
Tokyo Institute of Technology
Tokyo, Japan
brown.k.aa@m.titech.ac.jp

Nikhil Jain
Lawrence Livermore National Laboratory
Livermore, CA, USA
nikhil@llnl.gov

Satoshi Matsuoka
Tokyo Institute of Technology
Tokyo, Japan
matsu@acm.org

Martin Schulz
Technische Universität München
Munich, Germany
schulzm@in.tum.de

Abhinav Bhatele
Lawrence Livermore National Laboratory
Livermore, CA, USA
bhatele@llnl.gov

ABSTRACT

Network congestion arising from simultaneous data transfers can be a significant performance bottleneck for many applications, especially when network resources are shared by multiple concurrently running jobs. Many studies have focused on the impact of network congestion on either MPI performance or I/O performance but the interaction between MPI and I/O traffic is rarely studied and not well understood. In this paper, we analyze and characterize the interference between MPI and I/O traffic on fat-tree networks, highlighting the role of important factors such as message sizes, communication intervals, and job sizes. We also investigate several strategies for reducing MPI-I/O interference, and the benefits and tradeoffs of each approach for different scenarios.

CCS CONCEPTS

• **General and reference** → *Performance*; • **Networks** → *Network simulations*; *Network performance analysis*;

KEYWORDS

Interference, I/O, MPI, fat-tree, network, simulation

ACM Reference Format:
Kevin A. Brown, Nikhil Jain, Satoshi Matsuoka, Martin Schulz, and Abhinav Bhatele. 2018. Interference between I/O and MPI Traffic on Fat-tree Networks. In *ICPP 2018: 47th International Conference on Parallel Processing, August 13–16, 2018, Eugene, OR, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3225058.3225144

1 INTRODUCTION

Many production high performance computing (HPC) workloads involve significant data exchange among compute nodes as well as between compute nodes and parallel file systems. Data movement is necessitated because simulation data can rarely fit on a single compute node, and due to file input/output requirements of computational simulations, which also includes checkpointing for fault tolerance. Past research has shown that such data movement on HPC systems is a source of significant performance degradation for many applications [4, 23, 24].

Studies have also shown that off-node data movement is particularly susceptible to interference on systems where the network infrastructure is shared by all running jobs [5, 25]. However, with the exception of Mubarak et al. [17], these studies have focused on network traffic generated by one of the two major sources of off-node data movement: MPI or I/O. For the dragonfly topology [8], Mubarak et al. showed that packets generated by MPI communication of one job can experience over 4000× increase in maximum latency due to interference from I/O traffic of another job. This can result in a notable performance degradation for the MPI job.

Currently, there remain several open questions regarding the nature of interference between MPI and I/O traffic. For example, the impact of the volume and frequency of MPI and I/O data movement on the resulting interference is not understood. It is also unclear if a high bisection bandwidth topology such as fat-tree [12] suffers from such interference. A deep understanding of these aspects can guide optimization efforts as well as configuration of I/O subsystems, network infrastructure, and communication libraries.

In order to develop a better understanding of the aforementioned topics, this paper characterizes the effects of interference between MPI and I/O traffic on the performance of each of these traffic types. Our characterization explores several factors that may impact the interference, including message sizes, communication intervals, system allocations, and task placement. We also evaluate the efficacy of several optimization strategies in mitigating the performance impact of MPI-I/O interference. We focus on the fat-tree topology because of its wide-spread use in more than half of the top 500 supercomputers [16] and its low-diameter, high bisection bandwidth properties. In order to avoid being limited by the constraints posed by deployed systems, e.g. the placement of I/O servers, we use the CODES simulation framework to conduct our study [18].

One recurring finding in this study is that MPI traffic is more adversely affected than I/O traffic by interference. Network congestion caused by high-intensity I/O workloads often renders the interference due to MPI traffic inconsequential once an I/O-congestion threshold has been reached. Our results indicate that I/O traffic typically experience modest slowdown of up to 18% due to MPI interference, with the extreme case exhibiting 1.9x slowdown. In

ICPP 2018, August 13–16, 2018, Eugene, OR, USA

ACM ISBN 978-1-4503-6510-9/18/08…$15.00
https://doi.org/10.1145/3225058.3225144

contrast, MPI traffic is typically expected to observe 1.6x-3.2x slowdown, but the slowdown can be as high as 7.6x in some cases.

We also find that careful job and I/O server placement strategies can be highly effective in minimizing MPI-I/O interference on fat-tree networks. Further, we show that I/O throttling can significantly reduce the interference experienced by MPI traffic while incurring less than 18% performance penalty for I/O traffic.

2 BACKGROUND AND RELATED WORK

In this section, we introduce topics relevant to this work, and discuss prior work related to them.

2.1 I/O Traffic

Parallel file systems are integral to HPC machines as they provide persistent storage which can be accessed from all compute nodes. High performance parallel file systems, such as Lustre [3], have reached capacities of up to 72 PB and 1 TB/s aggregate read throughput [2]. Such storage infrastructures are used for several purposes, e.g. storing input/output data of computational simulations, checkpointing, and big data workflows [9, 10, 15, 21].

Lustre splits I/O requests into fixed-sized chunks in order to better pipeline and parallelize requests. The chunk size is configurable by the administrator and defaults to 4 MB in the latest version of Lustre. Notable I/O optimization studies have shown how I/O requests can cause congestion on the storage servers as well as on the network, specifically on links that are directly connected to I/O servers [14]. This congestion degrades the performance of I/O jobs, and various techniques such as I/O prefetching, buffering, scheduling, and throttling have been proposed to mitigate this degradation. However, none of these studies have evaluated the effect of I/O congestion, and that of mitigation techniques on non-I/O jobs that share the network.

2.2 MPI Traffic

Inter-process communication is required for many large-scale applications to broadcast initial parameters, share updated data, combine intermediate results, etc. Most HPC applications use the Message Passing Interface (MPI) for communication and their performance can be sensitive to communication latency. Optimized MPI libraries attempt to ensure that sensitive communication tasks on the application's critical path are not delayed unnecessarily. However, it is common to observe congestion due to MPI traffic within the same application which can degrade performance [4, 24]. Several surveys of representative MPI workloads have indicated that HPC applications generate a wide range of message sizes from small (few KBs) to large (few MBs) at a wide range of communication frequencies [7, 13]. These characteristics of application communication patterns impact the congestion caused by MPI traffic, and therefore, should be considered when characterizing MPI performance.

2.3 Fat-tree Networks

Fat-tree is a tree-based network topology with constant aggregate bandwidth across all levels of the tree [11]. Logically, it can be seen as a complete k-ary tree with edges increasing in capacity as we move up towards the root of the tree, thereby providing full bisection bandwidth. Most modern fat-tree networks are implemented as extended generalized fat-trees [20], depicted in Figure 1, in which several smaller components are used to represent the connectivity at higher levels. These can be built from commodity hardware and identical components such as fixed-radix routers.

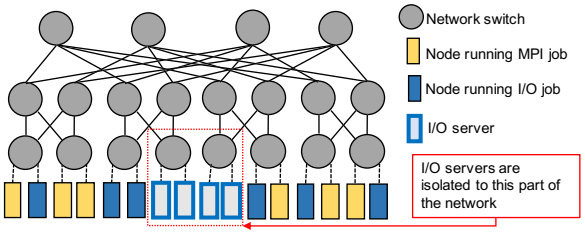


Figure 1: Fat-tree network with isolated I/O servers

Theoretically, fat-tree networks can provide congestion- and interference-free routing for pair-wise communication across the system since they provide full bisection bandwidth. However, the routing algorithm and process communication must be coordinated to prevent overlap of network paths between all active communication pairs. This is impractical and generally not applicable to production workloads. In practice, congestion occurs on full-bisection fat-tree networks that service production workloads [6].

2.4 Network Sharing and Interference

In most fat-tree systems, e.g. Tianhe-2, Stampede, and TSUBAME3, the network infrastructure is used for both I/O and MPI traffic, i.e. I/O traffic between compute nodes and I/O servers must traverse the same network links as MPI traffic. In most cases, this configuration is more economical than providing each compute node with an additional, independent connection to the storage system.

When the network is shared between I/O and MPI, I/O congestion can impact the performance of MPI traffic since congestion reduces the effective network bandwidth. At the same time, congestion caused by MPI traffic can also interfere with I/O traffic. Mubarak et al. [17] confirmed and quantified the effect of MPI-I/O interference on MPI performance in the presence of checkpointing I/O traffic on dragonfly networks. However, their study is specific to dragonfly, and cannot be used to quantify the degradation experienced on fat-tree networks. Further, to the best of our knowledge, no work has quantified the effect of MPI traffic on I/O performance.

It is typically assumed that I/O bottlenecks will only exist between the I/O servers and disk because (i) data movement on storage devices is slower than on modern networks, and (ii) I/O traffic involves n compute nodes accessing m I/O servers, where $n >> m$. However, with the advent of novel storage architectures such as burst buffers and in-memory file systems, current expectations may not hold for future systems. A more detailed investigation of inter-job interference is necessary in order to better understand its implications for the throughput of future systems.

2.5 CODES Simulation Framework

CODES is a framework for studying HPC interconnects, storage systems, and applications using parallel discrete-event simulations [18]. CODES provides high-fidelity, packet-level network models for several interconnect topologies such as fat-tree, torus and dragonfly.

It also provides a variety of MPI and I/O traffic generators that can drive these models. CODES has been used for several recent studies related to communication on different network topologies. These studies have provided validation results showing that CODES can predict performance similar to real-world systems [7, 17, 19].

3 EXPERIMENTAL SETUP

MPI and I/O traffic from different jobs running on a system interact over shared network links. Once messages are injected by nodes of different jobs on the network, they are decomposed into network packets that are structurally indistinguishable from one another. Nevertheless, the degree of slowdown due to interference from an opposing job varies and depends on the opposing job's behavior, which determines its traffic pattern and intensity. This is because interference depends on the volume of packets at each point in the network, and different jobs inject packets into the network at different rates. The rate of packet injection for a given job is a result of the sizes and frequencies of the messages being sent over the network. Therefore, characterizing the interference between different types of jobs (viz. MPI and I/O for our work) requires understanding how the communication pattern of one type will interact with that of another type.

3.1 Machine Configuration

As stated earlier, we use CODES-based network simulations to conduct our experiments, and generate synthetic traffic patterns that capture the salient features of typical I/O and MPI workloads in HPC. Unlike on real systems, network simulations allow for more fine-grained monitoring of the network traffic without inadvertently perturbing the application's behavior or performance. Our simulated system consists of 1,296 nodes that are interconnected by a three-level, InfiniBand-like fat-tree network with 12.5 GB/s links. Each node is connected to one of 72 36-port level-1 (or leaf) switches. Of the 1,296 nodes, 72 function as I/O servers and 1,224 function as compute nodes. This partitioning is similar to the Cab cluster at Lawrence Livermore National Laboratory [1]. Unless otherwise specified, all I/O servers in our system are grouped and connected to one of four leaf switches. These four leaf switches are split between the two halves of the network, similar to the system depicted in Figure 1. By default, the I/O and MPI job are each assigned 612 randomly selected nodes. In summary, each of 1,296 nodes of the simulated system has one of the following three roles in our environment:

- MPI client: a compute node that generates MPI traffic
- I/O client: a compute node that generates I/O traffic
- I/O server: a service node that receives I/O traffic from I/O clients. This can either be an LNET router [3] or an actual I/O server with attached storage.

3.2 Simulation Workload

We use synthetic communication patterns for the I/O and MPI job. The chosen patterns are designed to capture the characteristics of production HPC workloads. Since our aim is to develop a broader understanding of inter-job interference, we believe that results from well-chosen synthetic patterns are more generalizable than those from specific applications [7, 19]. The workload in this study uses samples of small, medium, and large message sizes with moderate and high traffic intensities. This allows us to see trends in how different sizes and intensities respond to interference.

MPI job: The MPI job consists of one process per node. MPI processes are paired randomly, and each process sends a fixed amount of data to its partner at a fixed average interval, which is computed as the time between the completion of the previous message and the sending of the next message. The message sizes and intervals used in each experiment are chosen to reflect a cross-section of the MPI traffic patterns found in applications running on HPC systems [7, 9]. MPI message sizes range from 4 KB to 4 MB, and the intervals range from 1 μs to 100 ms across the various experiments in our study.

I/O job: Using the Lustre I/O-forwarding pattern, each I/O client randomly selects an initial I/O server for its first request. Subsequent requests from that client are sent in a round-robin manner to other I/O servers. Unless otherwise stated, each I/O client writes ~4 GB (1000 4 MB requests) for experiments reporting I/O performance. We use a negligible interval between I/O requests for cases that represent checkpoint-style I/O traffic, where each client writes all of its data in an unbroken stream of requests. In other cases, we use non-negligible request intervals to represent jobs where I/O data is written in pieces, e.g. periodic visualization data output during a scientific simulation. Previous studies have demonstrated this range of I/O patterns on HPC systems [10, 15, 21].

3.3 Methodology

The communication performance of a job is represented in its transfer latency, i.e., the time taken to complete sending an MPI message or I/O request. Interference between MPI and I/O traffic is quantified by comparing the transfer latency for a job when it uses the system exclusively to the transfer latency for the same job when it shares the system with another job. We refer to the exclusive system case as the **baseline run** and the shared system case as the **interference run**.

The transfer latency for every MPI message and I/O request is recorded during each simulation. For consistency, the duration of each run is long enough to collect approximately 1000 data points per node for the chosen job, i.e. I/O or MPI. Initially, 50 warm-up messages/requests are sent from each client before the communication times are recorded. The message/request intervals are varied randomly within a ±5% bound to account for system noise and variations in communication time across the clients. Hence all intervals reported in this paper represent the average delay between consecutive messages/requests, and the delay varies uniformly around the arithmetic mean with a maximum variation of ±5%.

In order to characterize the interference, we evaluate the effects of three MPI message sizes at different message intervals interacting with three I/O request sizes at different request intervals. We also alter the number of MPI and I/O clients to investigate how the interference is impacted by the size of the job. These characterization results are presented in Section 4. Thereafter, we evaluate the efficacy of three different performance optimization strategies that could mitigate MPI-I/O interference: changing job placement, changing I/O server placement, and I/O throttling. The results of these evaluations are presented in Section 5.

		INTERVALS									
		50 ms	10 ms	5 ms	1 ms	500 µs	100 µs	50 µs	10 µs	5 µs	1 µs
SIZES	4 KB					✓	✓	✓	✓	✓	✓
	512 KB			✓	✓	✓	✓	✓	✓		
	4 MB	✓	✓	✓	✓	✓	✓				

Table 1: Message/request sizes and their respective intervals evaluated in our main survey.

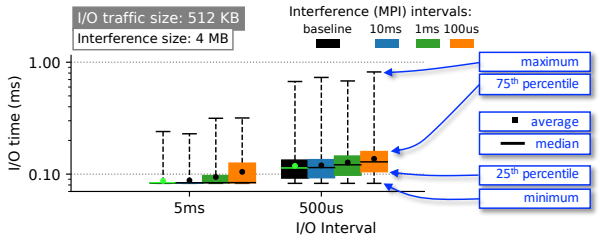


Figure 2: An example chart showing the format and describing the main elements of charts used in this paper.

4 CHARACTERIZATION OF INTERFERENCE

In order to characterize the interference between MPI and I/O traffic, we should not only capture the extent of the interference, but should also understand how the distinctive features of the traffic patterns result in different interference patterns. This section presents the resulting trends when different MPI and I/O traffic patterns interfere with each other. The traffic patterns, depicted in Table 1, are defined by the message/request size and its interval. We discuss the performance trends of I/O and MPI jobs individually in different sections since the features are unique to the two types of jobs. The final two subsections evaluate the impact of job sizes on inter-job interference and discuss the reasons for the differences in the performance trends, respectively.

Figure 2 illustrates an example of how the results will be presented in this section. This example figure shows the performance of two I/O traffic patterns in the presence of three MPI interference patterns. MPI's performance is not shown in this figure; it will be reported in a similar manner in separate charts. The baseline bars show the I/O request time when there is no MPI interference. Variation in I/O request time due to MPI interference is depicted by a change in the following attributes of a colored bar relative to its corresponding baseline bar:

- the height of the average mark, the circular dot
- the height of the median mark, the horizontal line
- the height and the length of the bar

4.1 Performance of the I/O Job

Figure 3 shows the I/O performance when different I/O request sizes interact with different MPI message sizes. Maximum and minimum values are omitted in this figure to ensure clarity. Each row of charts shows the performance for a different I/O request size; request size increases from top to bottom. Within a row, results due to increasingly larger interfering (MPI) message sizes are shown as we move from left to right.

Chart A shows no significant changes in the position of the dots and bars within any cluster, indicating that the performance of 4 KB I/O requests is not affected by 4 KB MPI background interference. However, when the size of the interfering messages increases to 4 MB, as shown in Chart C, the latency of 4 KB requests increases significantly. This demonstrates that the latency of small I/O requests is sensitive to high-intensity MPI interference. The 4 KB I/O request experiences the greatest average slowdown of all I/O request sizes, with the most significant slowdown occurring when 4 MB MPI messages are sent at 100 μs intervals, highlighted by the rightmost column of chart C. The average request time increases from 3.07 μs to 5.9 μs due to MPI interference, a 1.9× slowdown.

In Charts D-I, there is a significant increase in the I/O times for the rightmost three clusters. As this increase also occurs for the black baseline bars, which show the runtimes when there is no interference, the increase is not due to the presence of MPI traffic. Rather, the increase happens when the I/O interval is reduced (as we move from left to right within a chart) and I/O packets are injected into the network at a higher rate. This results in I/O-congestion causing the increase in I/O request time. We will refer to the I/O interval around which this rapid increase in request time happens as the *I/O-congestion threshold*. The 4 KB I/O traffic scenarios (Charts A-C) do not cross their I/O-congestion threshold for the I/O intervals we tested. We find that the I/O-congestion threshold is between 500-100 μs for 512 KB requests and between 5-1 ms for 4 MB requests.

When the interfering MPI message size is 512 KB (Charts B, E, H) and 4 MB (Charts C, F, I), an increase in the MPI traffic intensity causes a corresponding increase in the I/O request times before the I/O-congestion threshold is reached. For 4 MB I/O versus 4 MB MPI (chart I), this trend is depicted within each cluster of bars by a gradual increase in the positions of colored bars for I/O intervals of 50 ms, 10 ms, and 5 ms. Once we cross the I/O-congestion threshold at 1 ms I/O intervals, the interfering MPI traffic does not show a significant impact on I/O performance.

Figure 4 presents a more detailed view of the case in which 4 MB I/O requests interact with 4 MB MPI messages. The inclusion of additional intervals shows that the impact of interference caused by MPI gradually declines as the I/O interval approaches the I/O-congestion threshold. At the 2 ms I/O interval, there is minimal increase in the average and 75th percentile time within the cluster. Nevertheless, after the I/O-congestion threshold is reached, even the baseline is higher than the interference runs prior to the threshold. The detailed trends seen here can be used to represent the behavior of different I/O request sizes as Figure 3 suggests that different I/O request sizes exhibit a similar pattern around their I/O-congestion threshold. Overall, we conclude that if the time per I/O request is more important than the volume of requests for an I/O job, it is better to send less frequent requests despite the higher relative impact of interference from MPI.

4.2 Performance of the MPI Job

The performance results for MPI traffic in the presence of I/O interference are shown in Figure 5. Similar to Figure 3, each row of charts shows the performance for a different MPI message size; within a row, results due to increasingly larger interfering (I/O) request sizes are shown. The horizontal arrangement of dots within

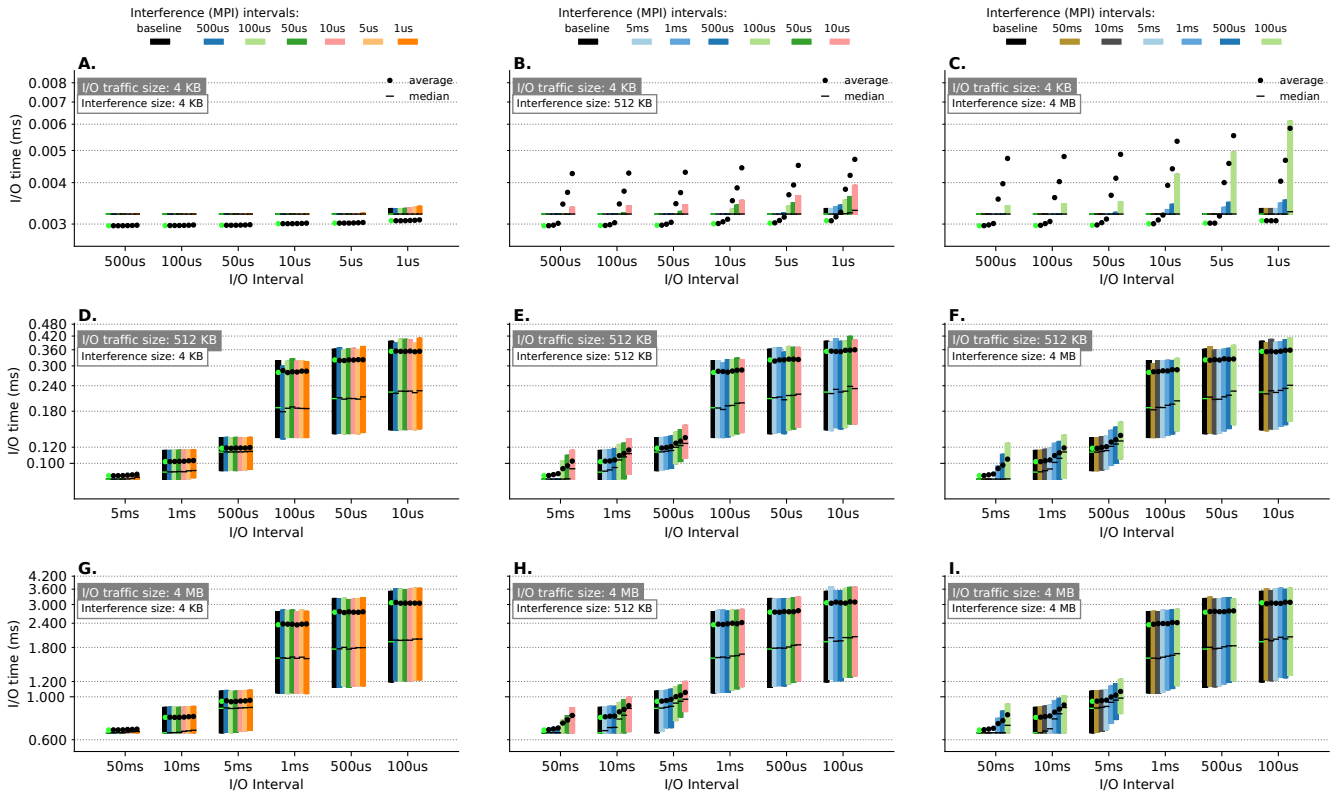


Figure 3: Performance of three I/O request sizes being interfered by three MPI message sizes (see Figure 2 for feature description). A difference in the height of the features of the colored bars (interference runs) relative to the features of the black bar (baseline run) in the same group indicates a performance variation due to interference. Small-sized I/O requests are most impacted by high-intensity MPI traffic. I/O self-congestion obfuscates the impact of MPI traffic for several other cases.

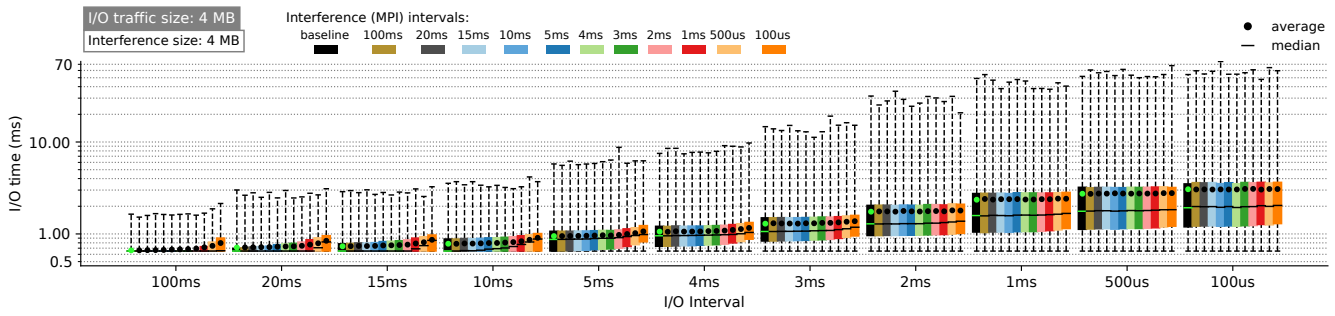


Figure 4: A detailed look at the performance of 4 MB I/O requests with interfering (MPI) message size of 4 MB.

each cluster and the absence of visible bars in Chart A indicate that there is relatively no MPI performance variation for 4 KB messages in presence of 4 KB I/O requests irrespective of their intervals. However, as the size of I/O requests increases in Charts B and C, MPI slowdown is observed with a maximum slowdown of 7.6× due to 4 MB I/O requests.

The effect of MPI self-congestion is visible in the changing height of the baseline bars across clusters in Charts D-I, i.e., for MPI message sizes 512 KB and 4 MB. However, unlike the I/O case, MPI slowdown due to self-congestion does not prevent inter-job interference from further inducing notable MPI slowdown. This suggests that regardless of the intensity of MPI messaging, it is still vulnerable to further slowdown due to I/O traffic.

Chart C shows that the overall height of each cluster decreases as the MPI interval is reduced as we move from left to right in

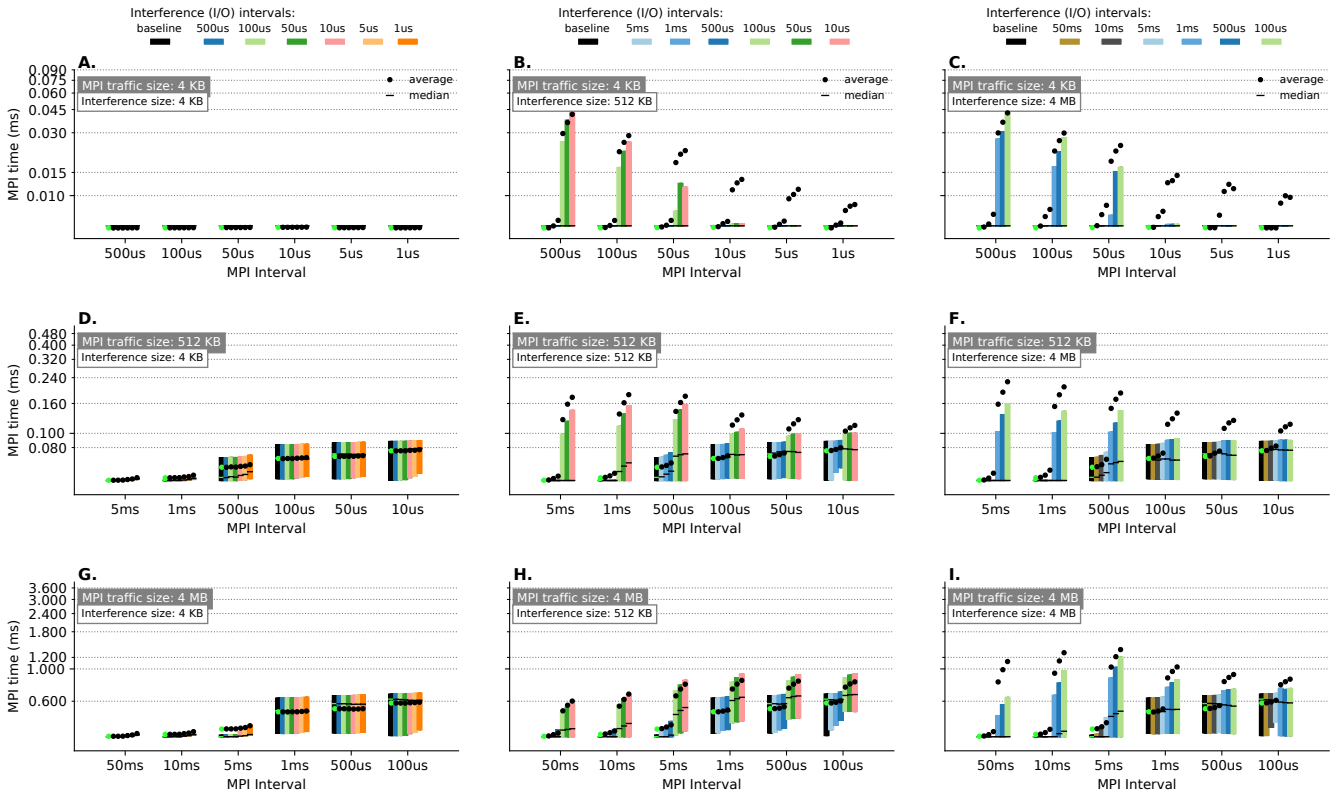


Figure 5: Performance of three MPI message sizes being interfered by three I/O request sizes: except when I/O requests are very small, latency for MPI messages is impacted significantly by interfering I/O traffic.

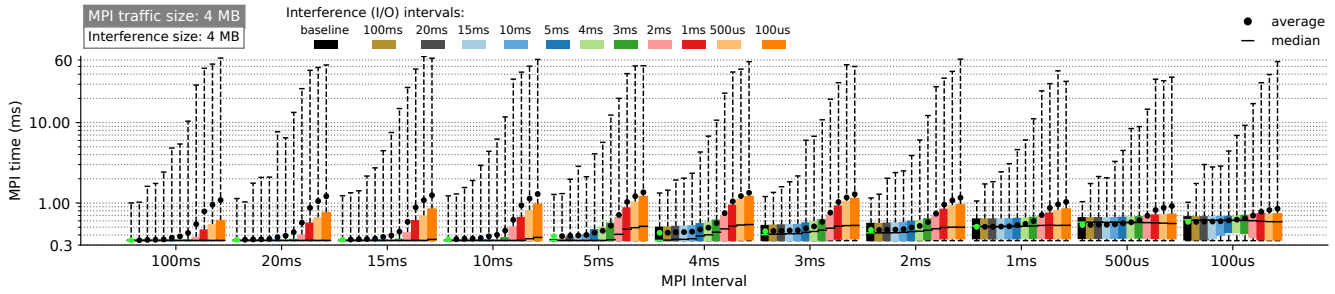


Figure 6: A detailed look at the performance of 4 MB MPI messages with interfering (I/O) request size of 4 MB.

that chart. This, surprisingly, indicates that reducing the interval between MPI messages reduces the average slowdown due to interference. This phenomenon is visible in all charts where the I/O interference size is 512 KB (charts B, E, H) or 4 MB (charts C, F, I). Higher I/O injection rates and larger I/O request sizes cause more network congestion and result in more I/O packets stalled and waiting in the buffers of network switches. Thus when MPI messages are queued infrequently, they often end up waiting behind long queues of I/O packets. We hypothesize that reduction in MPI messaging interval results in MPI packets preempting some of the I/O packets, which in turn slows down injection rate for I/O traffic. With less I/O packets in the system, MPI packets spend less time queued in the network and are transferred faster, which results in overall better MPI performance.

To get a more detailed view of MPI performance characteristics, we study the scenario in which 4 MB MPI messages are transferred alongside 4 MB I/O requests in Figure 6. As described in the previous paragraph, reducing the MPI interval improves the MPI performance during high-intensity interference. When the I/O interference interval is 100 μs, the average message time for 4 ms MPI interval is 1.34 ms while the average message time for 100 μs MPI interval is 0.85 ms. All MPI intervals larger than 4 ms experience a maximum slowdown above 300% of the baseline times when the I/O interval is 100 μs.

Overall, the results so far can be summarized as follows:

- Small interfering requests/messages (4 KB of MPI or I/O) have minimal impact on MPI or I/O performance.
- For I/O performance, the impact of interference is moderate if the I/O interval is larger than the I/O-congestion threshold. Below the threshold, interference has negligible effect.
- For MPI performance, interference causes variations even when there is a moderate amount of interfering I/O traffic. The most significant impact is observed when the I/O traffic has passed its I/O-congestion threshold.
- Overall, the impact of interference is higher for MPI than I/O.

4.3 Varying Job Sizes

The results presented so far are for the system configuration in which the compute nodes are evenly split between jobs with I/O and MPI clients. This configuration is useful for making a fair comparison between how each traffic type reacts to interference from the other traffic type. However, nodes are usually unevenly divided among I/O and MPI jobs. Hence, we present results for a set of experiments in which we alter the job sizes in order to understand how a job's scale affects the interference it experiences. Three node allocation sizes are tested: *small*, *medium*, and *large* using 25%, 50%, and 75% of all compute nodes for the analyzed job, respectively. All compute nodes in the system are occupied by either an I/O client or an MPI client and a reduction in the size of one job means an increase in the size of the other job.

Based on the I/O performance trends presented in Section 4.1, checkpoint-style I/O traffic, with request sizes in MBs and intervals close to zero, is not affected by MPI interference as its interval is below the I/O-congestion threshold. We therefore choose to focus on other I/O scenarios that are noticeably affected by the presence of MPI traffic in these results: I/O request size of 4 MB and intervals between 2 ms and 128 ms. The MPI performance evaluation is done for MPI messages between 4 KB and 1 MB with a 500 μs interval in the presence of worst-case I/O traffic, i.e. checkpoint-style traffic.

4.3.1 I/O Jobs. The I/O request times are presented in Figure 7A and the resulting interference trends are shown in Figure 7B. The baseline times reported in Figure 7A for 2 ms-interval requests show that times for I/O requests in medium and large jobs are approximately 1.8× and 3× higher than the time for the small I/O job, respectively. Increasing the job's size results in more I/O requests being sent to the 72 I/O servers, thereby increasing the self-congestion and mean time per request. Because of the increase in I/O congestion as the job size increases, medium and large jobs are less affected by interference than the small job, as shown in Figure 7B. Among small jobs, scenarios with infrequent requests are most sensitive to interference. Overall, highest relative increase of 18% in I/O request time due to MPI traffic is observed for the scenario with small I/O job and 128 ms intervals.

4.3.2 MPI Jobs. The performance trends in Figure 8 indicate that MPI messages are affected significantly by I/O traffic for all message sizes and allocations. Among allocations, for all message sizes, relatively lower interference is seen for the large MPI jobs than for the small and medium jobs. The lower number of I/O clients when the MPI job is large means that there are less I/O packets being injected into the network and that the paths of I/O packets are less likely to overlap with those of MPI traffic. Even with a large MPI job, the slowdown due to I/O interference ranges between 1.6-3.2x of the baseline performance. Figure 8 also shows that MPI jobs with small message sizes are affected the most by I/O.

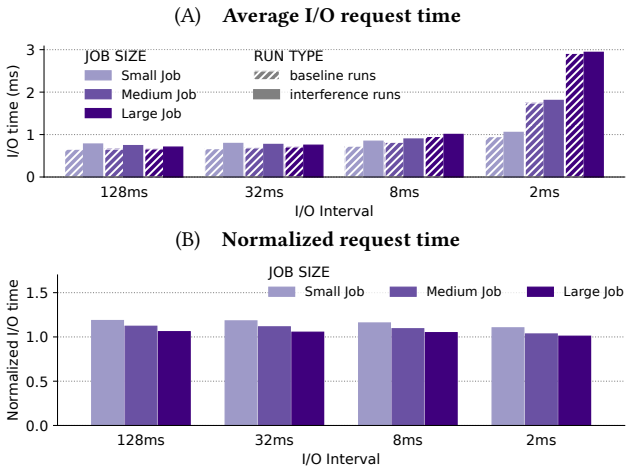


Figure 7: Impact of job sizes on I/O performance. Remaining nodes are running an MPI job.

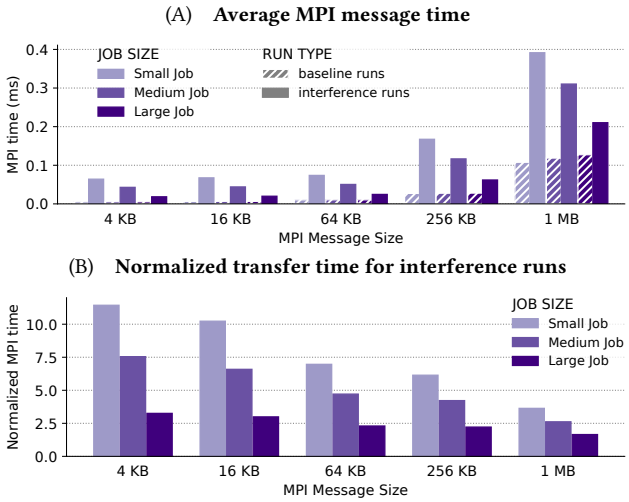


Figure 8: Impact of job sizes on MPI performance. Remaining nodes are running an I/O job.

4.4 Discussion

Section 4.1 demonstrated that 4 MB I/O request traffic reaches its I/O-congestion threshold at 2 ms intervals for the medium job. However, Figure 7 indicates that 4 MB I/O traffic pattern has passed its I/O congestion threshold at 2 ms for the large job while the small job has not reached its threshold for the same interval. In other words, larger jobs will reach their congestion thresholds more quickly (i.e. at higher intervals) than smaller jobs for a given request size.

In analyzing the differences between the I/O and MPI interference trends, it must be re-stated that there is a key difference between the traffic patterns of the two jobs: each MPI client has a single, unique destination while all I/O clients write data to the same set of few I/O servers in a round-robin manner. In our environment, there is a 1:1 mapping from MPI client to its pair and a 612:72 mapping from I/O clients to I/O servers for a medium job. Thus, the MPI traffic is dispersed more evenly across the network while I/O traffic is focused at switches servicing I/O servers, causing congestion where the paths from different I/O clients converge. This disparity is the reason for the following phenomena:

- The average baseline time of a given I/O request size is higher than the average baseline for an MPI message of the same size and moderate interval.
- MPI congestion is not as extreme as I/O congestion, which is primarily responsible for the severe I/O performance degradation below the I/O-congestion threshold.

The interference impact of each traffic type can also be attributed to the difference in traffic patterns. I/O congestion not only affects I/O traffic, but also delays MPI packets that traverse the paths on which I/O packets are concentrated. Since I/O packets spend longer time on the network, there is a higher probability that they will block MPI packets. The MPI congestion is not as extreme, but we see that it has the potential to slowdown I/O requests when the MPI traffic is relatively heavy. This was reported in Figure 3C with small I/O request sizes versus large MPI message sizes and in Figure 7B with small I/O job size versus the large MPI job size; the latter being a common situation on HPC systems.

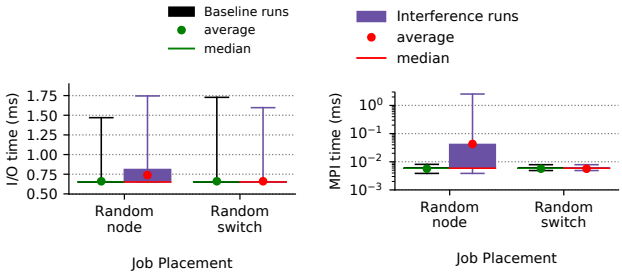
Both I/O and MPI results show that with interference, runtimes have very high maximum values and vary over large ranges with mean values significantly higher than their respective medians. Figures 4 and 6 illustrate examples of these runs. This trait is indicative of a heavy tail distribution and can be a concern for quality-of-service (QoS) and synchronization.

5 MITIGATION STRATEGIES

So far, we have presented results on the impact of I/O-MPI interference on the performance of MPI and I/O jobs. Next, we explore techniques that can potentially mitigate the effect of such interference for the most impacted scenarios. In particular, we explore three interference mitigation strategies: (i) modifying the placement of jobs to nodes, (ii) relocating the I/O servers, and (iii) throttling I/O traffic. We use the following representative configurations to report the results obtained in this section: for impact on I/O performance, we use 4 MB I/O requests sent at 128 ms intervals with interference from 4 MB MPI messages sent at intervals of 500 μs; for MPI performance, we simulate sending 4 KB messages at 500 μs intervals with interference from checkpoint-style I/O, i.e. 4 MB requests with a negligible interval between requests.

5.1 Job Placement

Past work has shown that placement of jobs (i.e. the allocation policy for assigning nodes to jobs) impacts congestion and interference observed on a system, and thus can affect job performance. This is because the placement determines the nodes that inject/consume data in the network, which along with the routing scheme impacts how the job's traffic is spread across the network links. All results presented in Section 4 used a topology-oblivious job placement in which application tasks are randomly mapped to compute nodes across the network as is done on most production systems. We now show the impact of a switch-level placement scheme on the I/O-MPI interference. In this scheme (called Random-switch), instead of randomly assigning nodes to jobs, nodes are allocated at the granularity of switches i.e. all nodes connected to a switch are assigned to the same job, either the I/O job or the MPI job. The tasks within a job are mapped sequentially to the nodes on the randomly assigned switches.



(A) **I/O performance** (B) **MPI performance**

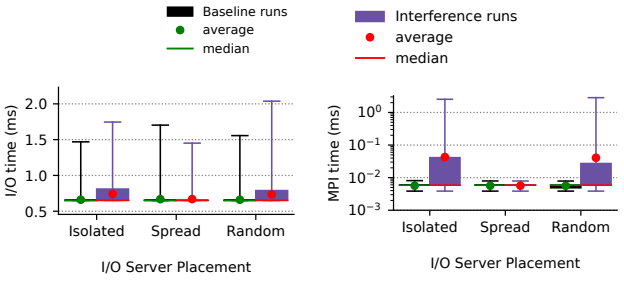
Figure 9: Impact of job placement on performance.

5.1.1 Performance Results. Figure 9A and Figure 9B show I/O and MPI performance, respectively, for executions with and without I/O-MPI interference for the Random-node and Random-switch job placement schemes. With the Random-switch placement, we see that the I/O and MPI times for the baseline and interference runs are similar, indicating that there is minimal slowdown due to interference for this type of mapping. In contrast, for the Random-node placement, the average I/O request time increases by 11.8% and the standard deviation increases from 8% to 19% when I/O-MPI interference is present. MPI messages experience an average slowdown of 800% with the Random-node placement, and the maximum message time increases by two orders of magnitude.

The Random-switch scheme leads to performance improvements because all links connected to switches with I/O servers and clients carry only I/O traffic; the same assertion holds for switches with MPI clients. Therefore, MPI-I/O interference on links between level-1 and level-2 switches is avoided. Further, when using the popular FTREE routing algorithm, dedicated ports of level-3 switches are responsible for forwarding traffic for a given level-1 switch. Thus, the links connecting to the level-3 switches that are used to relay I/O traffic do not carry MPI traffic since MPI clients do not share level-1 switches with I/O servers in this configuration.

5.2 I/O Server Placement

During our analysis of I/O performance and I/O-congestion threshold, we observed that aggregating I/O servers on some switches resulted in heavy usage of a few down links connected to different level-2 switches in the network. This created congestion throughout the system and impacted performance significantly. Thus, our second mitigation strategy explores the efficacy of the following I/O server placement schemes:



(A) **I/O performance** (B) **MPI Performance**

Figure 10: Impact of I/O server placement on performance.

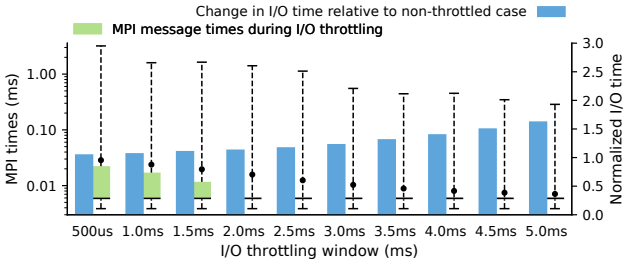


Figure 11: Impact of I/O request throttling on performance.

- **isolated-target**: In this scheme, dedicated switches are used to host I/O servers. This I/O server placement scheme has been used for all other experiments in this work as this is typically used in production HPC systems. For our simulated system, four leaf switches – two from each half of the network – are dedicated to I/O servers.
- **spread-target**: In this scheme, I/O servers are assigned in a round-robin fashion to all leaf switches. For our system, one I/O server is assigned to each of the 72 leaf switches. Each I/O server is connected to port 18 of a switch, therefore, the I/O servers are spread across the network.
- **random-target**: I/O servers are randomly distributed across the system with no restrictions.

Regardless of the I/O server placement scheme, I/O clients always send data to I/O servers in a round-robin manner as is commonly done in production systems. We have not varied I/O server selection scheme to ensure that our results are consistent and allow for a fair comparison.

5.2.1 Performance Results. Figure 10 shows the impact of MPI-I/O interference on performance when the I/O server placement is varied. We find that the spread-target placement of I/O nodes is best at mitigating interference for both types of jobs. Further, the isolated-target and random-target server placements result in similar slowdown for both jobs.

As stated earlier, a visualization of I/O traffic over the network for the isolated-target scheme showed heavily used links throughout the system. The random-target scheme does not improve the situation as the I/O traffic still heavily uses few down links connected to different level-2 switches. However, when the I/O servers are connected to the 18th port of every level-1 switch in the spread-target scheme, we observe that only four of the 72 level-2 switches (one in each pod) are used for sending down I/O traffic to I/O servers. By parsing the routing tables of all switches, we found that the traffic destined to the servers connected to the 18th port of every leaf switch is routed via the 18th level-2 switch in the pod of the destination server. As a result, the I/O traffic is confined to a subset of the network and thus does not interfere with the MPI traffic. This insight can guide the placement of LNET routers and burst buffers to leverage the I/O traffic isolation due to the routing algorithm.

5.3 Throttling I/O Traffic

I/O throttling has been presented as a solution to reducing I/O congestion and preventing low-priority I/O workloads from monopolizing the system's I/O bandwidth [14, 22]. Throttling I/O traffic refers to the process of regulating the rate at which I/O requests are sent or processed. The characterization of I/O performance in Section 4.1 showed how I/O congestion can be more detrimental to I/O performance than interference caused by MPI traffic. Further, the interference caused by I/O congestion can lead to over 8× slowdown in latency of MPI messages. Reduction in I/O congestion can therefore also reduce the degradation in MPI performance without significantly affecting I/O performance.

We investigate the effect of I/O throttling on the performance of both I/O and MPI jobs in this section. The random-node job placement and isolated-target I/O server placement are used for these results. Each I/O client writes 4.3 GB of total data, which is broken up into 4 MB units and sent to I/O servers in a round-robin manner. This operation is analogous to writing checkpoint data to disk, and hence the total checkpointing time is more important than the time to send individual 4 MB requests. I/O throttling is achieved by varying the time between initiating consecutive I/O requests. We refer to this gap as the *throttle window*. The *throttle window* represents the duration between the start times of two consecutive requests. Note that this is different from request *interval*, which is used to represent the duration between the completion time of a request and the start time of the next request. The MPI traffic consists of 4 KB messages being sent at 500 μs intervals. Results are shown in Figure 11.

The baseline I/O performance is the time taken for all I/O clients to finish their data writes without any I/O throttling and with the MPI job running in the background. We find that the I/O completion time remains unchanged for all throttle windows below 2 ms while MPI performance begins to improve when the I/O throttle window is >1.5 ms. At 2 ms, the throttling window becomes longer than the mean I/O request time by approximately 250 μs. With 3 ms throttling, the checkpointing time increases by only 18% while the average MPI latency improves by over 200%. The 3 ms window results in an average I/O request time of 1.3 ms and an effective I/O interval of 1.7 ms, which is around the I/O-congestion threshold for 4 MB requests.

These results indicate that I/O throttling can potentially be used to reduce MPI-I/O interference if a relatively small loss in I/O performance is acceptable. Additionally, the I/O-congestion threshold may be an effective guide for choosing appropriate throttling windows.

6 CONCLUSION

Resource contention over the network can be a major issue for jobs whose performance depends on efficient inter-process communication and/or I/O operations. Our characterization of the interference between I/O and MPI traffic on the fat-tree network proves that I/O traffic is less sensitive to interference than MPI traffic. Nevertheless, our results show that intensive MPI jobs can slow the performance of certain I/O traffic by up to 1.9×, while the largest slowdown for MPI traffic is 7.6×. Small I/O jobs with a more typical I/O traffic pattern reported a 18% slowdown. This is important because modern HPC systems typically run a larger percentage of MPI jobs than I/O jobs.

For I/O traffic, we identified the presence of I/O-congestion threshold in various scenarios. We found that this threshold varies based on the size of the request and the scale of the I/O jobs, and it was discovered to be an important cause of slowdowns for both I/O and MPI performance.

In order to mitigate the impact of I/O-MPI interference, we presented and evaluated three different strategies: job placement, I/O server placement, and I/O throttling. We found that each of these strategies can reduce interference and improve performance, especially for MPI jobs. The placement strategies demonstrate how the architecture and routing of fat-tree networks can be exploited to isolate I/O traffic on a shared network.

ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-CONF-751958). This work was supported by JST CREST Grant Number JPMJCR1303, Japan, and performed under the auspices of Real-World Big-Data Computation Open Innovation Laboratory, Japan.

REFERENCES

[1] [n. d.]. Cab: Intel Xeon system in Livermore Computing. http://computation.llnl.gov/computers/cab.

[2] 2018. High-Performance Storage List. Virtual Institute for I/O. http://www.vi4io.org

[3] 2018. The Lustre Filesystem. http://lustre.org/ Accessed: April 01, 2019.

[4] Abhinav Bhatele, Nikhil Jain, Katherine E. Isaacs, Ronak Buch, Todd Gamblin, Steven H. Langer, and Laxmikant V. Kale. 2014. Improving application performance via task mapping on IBM Blue Gene/Q. In *Proceedings of IEEE International Conference on High Performance Computing (HiPC '14)*. IEEE Computer Society. LLNL-CONF-655465.

[5] Abhinav Bhatele, Kathryn Mohror, Steven H. Langer, and Katherine E. Isaacs. 2013. There Goes the Neighborhood: Performance Degradation due to Nearby Jobs. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '13*. ACM Press. https://doi.org/10.1145/2503210.2503247

[6] Kevin A. Brown, J. Domke, and S. Matsuoka. 2015. Hardware-Centric Analysis of Network Performance for MPI Applications. In *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*. 692–699. https://doi.org/10.1109/ICPADS.2015.92

[7] Nikhil Jain, Abhinav Bhatele, Louis H. Howell, David Böhme, Ian Karlin, Edgar A. León, Misbah Mubarak, Noah Wolfe, Todd Gamblin, and Matthew L. Leininger. 2017. Predicting the Performance Impact of Different Fat-tree Configurations. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17)*. ACM, New York, NY, USA, Article 50, 13 pages. https://doi.org/10.1145/3126908.3126967

[8] J. Kim, W. Dally, S. Scott, and D. Abts. 2009. Cost-Efficient Dragonfly Topology for Large-Scale Systems. *IEEE Micro* 29, 1 (Jan 2009), 33–40.

[9] Thorsten Kurth, Jian Zhang, Nadathur Satish, Evan Racah, Ioannis Mitliagkas, Md. Mostofa Ali Patwary, Tareq Malas, Narayanan Sundaram, Wahid Bhimji, Mikhail Smorkalov, Jack Deslippe, Mikhail Shiryaev, Srinivas Sridharan, Prabhat, and Pradeep Dubey. 2017. Deep Learning at 15PF: Supervised and Semi-supervised Classification for Scientific Data. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17)*. ACM, New York, NY, USA, Article 7, 11 pages. https://doi.org/10.1145/3126908.3126916

[10] Rob Latham, Chris Daley, Wei keng Liao, Kui Gao, Rob Ross, Anshu Dubey, and Alok Choudhary. 2012. A case study for scientific I/O: improving the FLASH astrophysics code. *Computational Science & Discovery* 5, 1 (2012), 015001. http://stacks.iop.org/1749-4699/5/i=1/a=015001

[11] C.E. Leiserson. 1985. Fat-trees: Universal Networks for Hardware-efficient Supercomputing. *IEEE Trans. Comput.* C-34 (Oct 1985), 892–901.

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

[13] Edgar A. Leon, Ian Karlin, Abhinav Bhatele, Steven H. Langer, Chris Chambreau, Louis H. Howell, Trent D'Hooge, and Matthew L. Leininger. 2016. Characterizing Parallel Scientific Applications on Commodity Clusters: An Empirical Study of a Tapered Fat-tree. In *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC '16)*. IEEE Computer Society, Article 78, 12 pages. LLNL-CONF-681011.

[14] Qing Liu, Norbert Podhorszki, Jeremy Logan, and Scott Klasky. 2013. Runtime I/O Re-Routing + Throttling on HPC Storage. In *Presented as part of the 5th USENIX Workshop on Hot Topics in Storage and File Systems*. USENIX, San Jose, CA. https://www.usenix.org/conference/hotstorage13/workshop-program/presentation/Liu

[15] Huong Luu, Marianne Winslett, William Gropp, Robert Ross, Philip Carns, Kevin Harms, Mr Prabhat, Suren Byna, and Yushu Yao. 2015. A Multiplatform Study of I/O Behavior on Petascale Supercomputer. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '15)*. ACM, New York, NY, USA, 33–44. https://doi.org/10.1145/2749246.2749269

[16] Hans Meuer, Erich Strohmaier, Jack Dongarra, and Horst Simon. 2009. "Top500 Supercomputer Sites". http://www.top500.org.

[17] Misbah Mubarak, Philip Carns, Jonathan Jenkins, Jianping Kelvin Li, Nikhil Jain, Shane Snyder, Robert Ross, Christopher D. Carothers, Abhinav Bhatele, and Kwan-Liu Ma. 2017. Quantifying I/O and Communication Traffic Interference on Dragonfly Networks Equipped with Burst Buffers. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE. https://doi.org/10.1109/cluster.2017.25

[18] Misbah Mubarak, Christopher D. Carothers, Robert B. Ross, and Philip Carns. 2016. Enabling Parallel Simulation of Large-Scale HPC Network Systems. *IEEE Trans. Parallel Distrib. Syst.* (2016). https://doi.org/10.1109/TPDS.2016.2543725

[19] M. Mubarak and Robert B. Ross. 2017. *Validation Study of CODES Dragonfly Network Model with Theta Cray XC System*. Technical Report ANL-MCS-TM-369. Argonne National Laboratory. http://www.mcs.anl.gov/papers/MCS-TM-369.pdf

[20] S.R. Öhring, M. Ibel, S.K. Das, and M.J. Kumar. 1995. On Generalized Fat Trees. In *Proceedings of the 9th International Parallel Processing Symposium, 1995*. 37–44.

[21] Sarp Oral, James Simmons, Jason Hill, Dustin Leverman, Feiyi Wang, Matt Ezell, Ross Miller, Douglas Fuller, Raghul Gunasekaran, Youngjae Kim, Saurabh Gupta, Devesh Tiwari Sudharshan S. Vazhkudai, James H. Rogers, David Dillow, Galen M. Shipman, and Arthur S. Bland. 2014. Best Practices and Lessons Learned from Deploying and Operating Large-Scale Data-Centric Parallel File Systems. In *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE. https://doi.org/10.1109/sc.2014.23

[22] Yingjin Qian, Xi Li, Shuichi Ihara, Lingfang Zeng, Jürgen Kaiser, Tim Süß, and André Brinkmann. 2017. A Configurable Rule Based Classful Token Bucket Filter Network Request Scheduler for the Lustre File System. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17)*. ACM, New York, NY, USA, Article 6, 12 pages. https://doi.org/10.1145/3126908.3126932

[23] Edgar Solomonik, Abhinav Bhatele, and James Demmel. 2011. Improving communication performance in dense linear algebra via topology aware collectives. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11)*. ACM, New York, NY, USA.

[24] Venkatram Vishwanath, Mark Hereld, Vitali Morozov, and Michael E. Papka. 2011. Topology-aware Data Movement and Staging for I/O Acceleration on Blue Gene/P Supercomputing Systems. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11)*. ACM, New York, NY, USA, Article 19, 11 pages. https://doi.org/10.1145/2063384.2063409

[25] Xu Yang, John Jenkins, Misbah Mubarak, Robert B. Ross, and Zhiling Lan. 2016. Watch Out for the Bully! Job Interference Study on Dragonfly Network. In *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis*. 750–760. https://doi.org/10.1109/SC.2016.63