Assessing the Interference Between Inter-node Communication and Network I/O Traffic

(Unrefereed Workshop Manuscript)

Kevin Brown[1,a)] Nikhil Jain[2] Abhinav Bhatele[2] Alfredo Gimenez[2] Kathryn Mohror[2]
Satoshi Matsuoka[1] Martin Schulz[2]

Abstract: Parallel file systems are used by supercomputers to support a range of applications that require concurrent access to high-performance shared storage for data workflow and resilience. The design of most of these systems result in the logical storage network sharing the same physical network infrastructure that is used for inter-process communication. Resource sharing in this manner on shared systems is a potential area of contention, which can be significant for communication and I/O intensive applications. We assess the interference caused by inter-process communication on the I/O throughput to parallel file system when they both traffic share the same network resources. For our experiments, we used *miranda_io* and IOR I/O benchmarks for generating I/O traffic, and we used *pF3D* FFT kernel and NBP FT MPI benchmark for generating inter-node communication traffic. Our preliminary results from running I/O and communication benchmarks simultaneously indicate that inter-process communication does not affect the performance of typical I/O workloads.

1. Introduction

The complete system architecture of a supercomputer typically comprises of tightly-coupled, high-performance components. Each subsystem is designed to meet the needs of distributed application and highly parallel workloads that require low-latency and/or high throughput. The I/O subsystem, for example, is engineered using low-latency, high-bandwidth communication channels and distributed storage hardware that is capable of servicing thousands of I/O requests per second.

The I/O subsystem involves both on-node and off-node storage devices that are used to feed data into the application, to stage intermediate results and checkpoints, and to store the application's output for consumption by the user or other applications. The complexity of the storage subsystem is hidden behind the overlaying file systems, which provides an efficient method of managing the data on the storage devices. File system operations may involve main memory, multiple communication mediums and network interconnects, and numerous storage devices. Hence, bottlenecks within the I/O subsystem are often difficult to diagnose.

Parallel file system designs also vary greatly to meet the requirements of the systems that they support and the intended workloads. Furthermore, the trend of these system is to adapt to new hardware and technologies that improve the performance of I/O operations. However, one common feature of many parallel file systems is to share the infrastructure that is also used for applications' inter-process communication. While this is usually done for economic reasons, it increases the potential for interference between inter-process communication and network I/O traffic.

A better understanding of the network interference is crucial for many applications that use parallel I/O for performing their work or ensuring resilience. Additionally, understanding this interference can enable improved I/O performance without negatively affecting application communication performance.

The challenges with understanding the I/O-communication interference stem from many design and operational issues. Such issues include: (1) a single I/O operation can span multiple nodes and involve multiple shared component of the system, (2) I/O benchmarking is usually done without the presence of realistic communication workloads and ignores other activities on the system, and (3) richer storage-memory hierarchies that invalidate traditional disk-based experiments since storage is being coupled with main memory for increasing performance.

We investigate the interference between I/O traffic and MPI communication to quantify its effect on I/O throughput and to expose the points in the systems that are most susceptible to such interference. Our results show that, for regular I/O workloads, there is negligible interference caused by MPI traffic on Lustre I/O operations. Our results are based on experiments on two supercomputers running *miranda_io* and IOR (for generating I/O workloads) and that pF3D-comm and NPB FT kernel (for generating MPI traffic).

[1] Tokyo Institute of Technology, 2-12-1 W8-33 Oo-okayama, Meguro-ku, Tokyo 152-8550, Japan
[2] Lawrence Livermore National Laboratory, 7000 East Ave., Livermore, CA 94550-9234, U.S.A.
[a)] brown.k.aa@m.titech.ac.jp

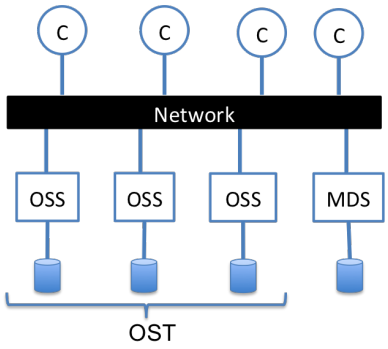


Fig. 1 Diagram of a typical cluster with luster nodes.

2. Communication Infrastructure and Storage

2.1 Network Infrastructure

Low-latency, high-speed networks are available in most supercomputers. These networks are used to accelerate inter-process MPI communication in distributed applications that run on supercomputers. Such networks connect hundreds or thousands of compute and management nodes, switches, storage servers, etc.

Efficient data communication is achieved by utilizing advanced topologies and routing algorithms. Topologies such as the fat-tree, torus, and tofu are constructed using InfiniBand, high-speed ethernet, or proprietary technologies to provide high-throughput and low-latency.

2.2 Storage

Local Storage: Some systems use on-node storage to provide temporary storage for local processes, staging checkpoints, and handling bursty I/O. Local processes can use the file system on these local storage devices without needing to access the network, ensuring interference-free performance. However, remote processes usually have no direct access to these storage devices.

Network Storage Lustre [1] is one of the most widely used parallel file systems for providing access to network storage in HPC systems. Figure 1 shows the design of a typical Lustre cluster. The files are stored on the object storage targets (OSTs) and presented via the the object storage servers (OSSs). The OSTs collectively present a unified file system to the compute nodes (clients) and the meta-data server (MDS) are used to track file layout information. A single file may be spread across multiple OSSs, and hence, multiple storage devices (OSTs).

The process of accessing files stored on Lustre involves multiple network requests. Lustre uses the Lustre Network protocol (LNET) to communicate between nodes in the the cluster. LNET operates above the network layer and each Lustre cluster.

2.3 MPI Communication

Inter-process communication is essential in HPC applications since it facilitates distributed processing. MPI is the most widely used standard for inter-process computation in HPC. Libraries that implement the MPI standard optimize communication over the network hardware and provide a unified interface for performing communication.

Efficient MPI libraries implement communication optimization algorithms for inter-process communication. However, these algorithms do not usually consider external interference.

3. Resource Contention and Interference

One way for large-scale HPC systems to achieve high utilization is by providing simultaneous access to many different users and many different applications. While this means more efficient usage of the system's resources, it also increases contention for shared resources such as network bandwidth and shared storage devices. Contention for the storage devices can occur within an application (during parallel I/O operations) and between applications (when multiple applications access the storage device simultaneously). Similarly, contention for network resources can be caused by processes within a single application as well as processes across applications.

Modern MPI libraries and parallel I/O infrastructure are designed to minimize bottlenecks in inter-process communication and parallel I/O operations, respectively. However, the I/O infrastructure is agnostic of the MPI network activities even though the network used for MPI communication is typically partially/completely shared by the I/O subsystem. The reverse also holds true for MPI libraries.

The bottlenecks within MPI libraries and the bottlenecks within the I/O subsystem have been the subject of numerous studies. However, the interference between MPI communication and I/O traffic over the network has not been explored in much detail.

4. Experiment Setup

To understand the potential impact of interference between MPI communication and I/O traffic, we analyze the performance of representative communication-bound application kernels and I/O-bound application kernels running on supercomputers at the Tokyo Institute of Technology (TITECH) and the Lawrence Livermore National Laboratory (LLNL). We design test cases that expose the performance of Lustre I/O operations with and without interference from MPI network traffic.

4.1 System

In TITECH, we use 256 nodes of the TSUBAME2.5 system and, in LLNL, we use 300 nodes of the Catalyst system for our experiments. The system specification for TSUBAME2.5 and Catalyst are shown in Table 1 .

For the TSUBAME2.5, the Lustre file system used in these experiments are connected to the same network as the compute nodes. We use the same set of compute nodes for all experiments (per system) with the node-process placements described in Section 4.3.

4.2 Benchmarks

miranda_io is chosen as the I/O benchmark to run on Catalyst. *miranda_io*is the I/O kernel of Miranda [2], a hydrodynamics code that was developed at LLNL. In each *miranda_io*benchmark, each process writes 111 MB of data to a file and then reads the

	TSUBAME2.5	Catalyst
# of Nodes	1442	324
OS	SLES SP3	TOSS
CPU per node	2 x Intel Xeon X5670 (12 cores total)	2 x Intel Xeon E5-2695v2 (24 cores total)
Memory per node	56GB	128GB
On-node scratch (SSD)	51GB	800GB
Network Infrastructure	QDR Infiniband	QDR InfiniBand
Network Topology	Fat-tree (16 nodes per edge switch)	Fat-tree (18 nodes per edge switch)
Size of Lustre file system used	1.5PB	2.0PB

Table 1 The composition of each test cases (run/trial).

equivalent amount of data from a file. For our experiments, the benchmark has been modified to provide additional timing information and to ensure that each application writes to and reads from its own file.

miranda_io is used for experiments on Catalyst with 10 processes per node over 150 nodes. Associated experiments have shown that this is sufficient for generating our expected workload.

IOR [3] is a tool for benchmarking parallel file systems that was also developed at LLNL. A variety of file management options are offered by IOR. We chose to use the POSIX API, with each processing writing and reading 100MB to it's own file. The full configuration options are listed in Section aA.1. IOR is used for experiments on TSUBAME2.5, using 10 processes per node over 128 nodes.

pF3D-comm is the communication kernel of the pF3D laser-plasma interaction code[4] developed at LLNL. The kernel is configured to perform 2D-FFT on a 3D Cartesian grid. We modified the code to include additional timers and to synchronize after each 2D FFT operation.

We use pF3D-comm for experiments on Catalyst with all-to-all message sizes of 32KB and 4KB. This benchmark was ran with 16 processes per node over 150 nodes, totaling 2048 processes.

NPB FT kernel is a 3D fast Fourier Transform benchmark in the NAS Parallel Benchmark suite (NPB) **?**. We used problem size E over 128 nodes on TSUBAME2.5 with 8 processes per node, totaling 1024 processes. pf3d-comm was not available to be ran on TSUBAME2.5.

4.3 Test Cases

Table 2 shows the test cases executed on TSUBAME2.5 and Catalyst. As noted in Section 4.2, the benchmarks that were ran on TSUBAME2.5 were NPB FT (for communication) and IOR (for IO), while pf3d-comm (for communication) and *miranda_io* (for IO) were ran on Catalyst.

		Run/Trial					
		R1	R2	R3	R4	R5	R6
Benchmark	Communication				✓	✓	✓
	I/O	✓	✓	✓	✓	✓	✓

Table 2 The composition of each test cases (run/trial).

Runs R1-3 test the I/O performance without MPI interference while R4-6 test the I/O performance in the presence of MPI interference. For each trial, the I/O benchmark is ran for approximately 7-10 minutes, i.e., 30 iterations of IOR on TSUBAME2.5 and 25 iterations of *miranda_io* on Catalyst. The communication benchmarks, NPB FT and pF3D-comm, are configured to run continuously while the I/O benchmarks are running for trials R4-6.

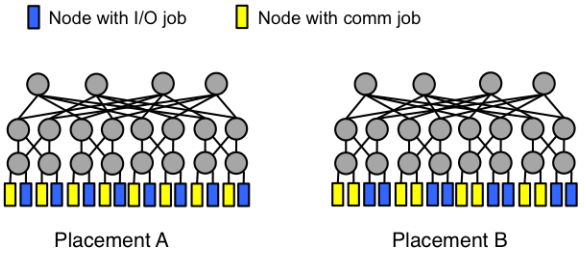


Fig. 2 The process-node mapping (node placement strategy) used in our experiments.

Two process-node placements were used for each test case. The placements, shown in Figure 2, were chosen to investigate the variation in interference based on task placement.

5. Results

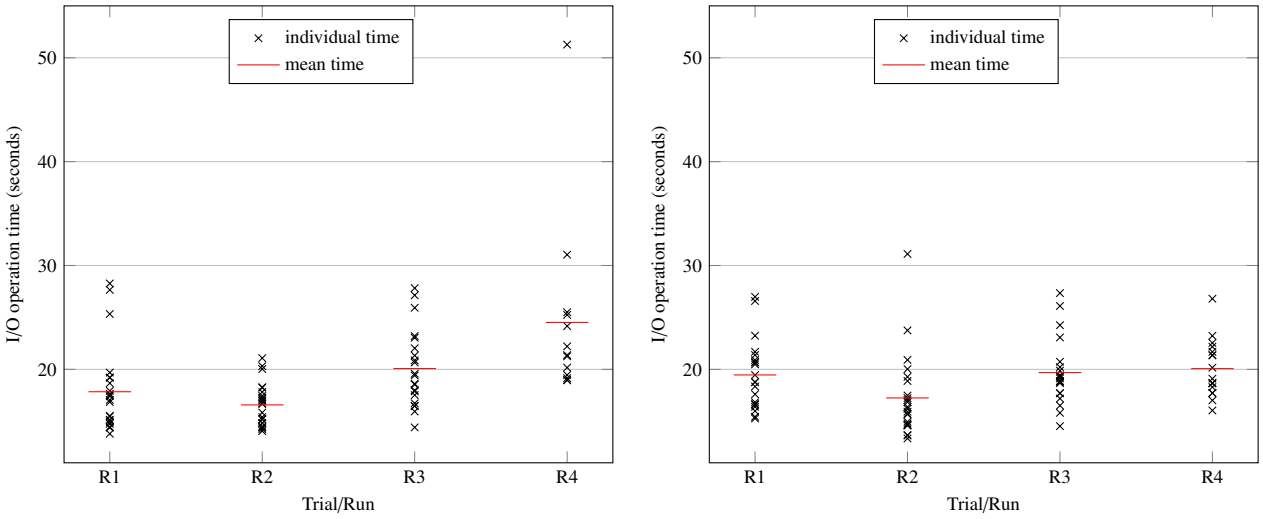
The I/O performance results are shown in Figures 3, 4, 5, and 6. Write performance is shown Figures 3 and 4 and read performance is shown in Figures 5 and 6. The time for each read and write operation is individually plotted for every iteration of the I/O benchmarks and the mean of operation time for each run is indicated on the charts. As mentioned in **??**, IOR and *miranda_io* are the I/O benchmark used for TSUBAME2.5 and Catalyst, respectively. The standard deviations for the trials are shown in Tablestable:std-data

R1, R2, and R3 are the trials without communication kernels and R4, R5, and R6 are the trials with MPI traffic being generated by our communication kernels. There are no R5 and R6 trials for TSUBAME2.5 because there was not enough time to run these trials.

The results show no distinct nor consistent change in the I/O operation times across all the trials, indicating that I/O traffic is not significantly impacted by the presence of MPI activity for these experiments. The noticeable spikes occur in cases with and without our communication benchmarks, confirming that even these outliers were not caused by MPI interference.

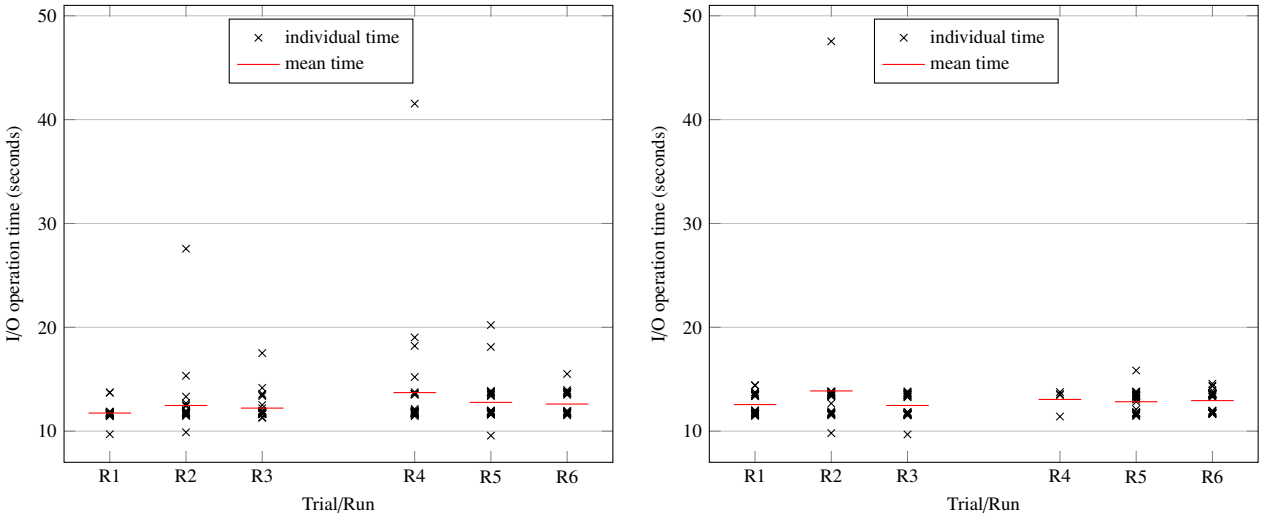
6. Discussion

The variations in the read and write times provided in the Section 5 were similar to the variations caused by system noise, instead of being caused by the MPI traffic that was generated in



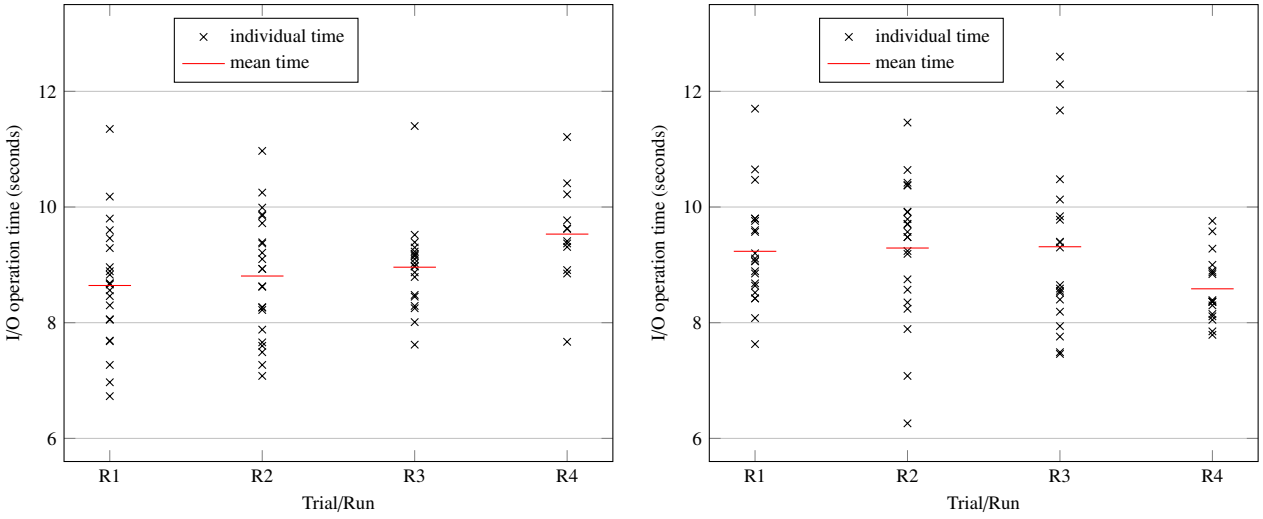
(a) Placement A (b) Placement B

Fig. 3 IOR performance when writing to Lustre file system on TSUBAME2.5



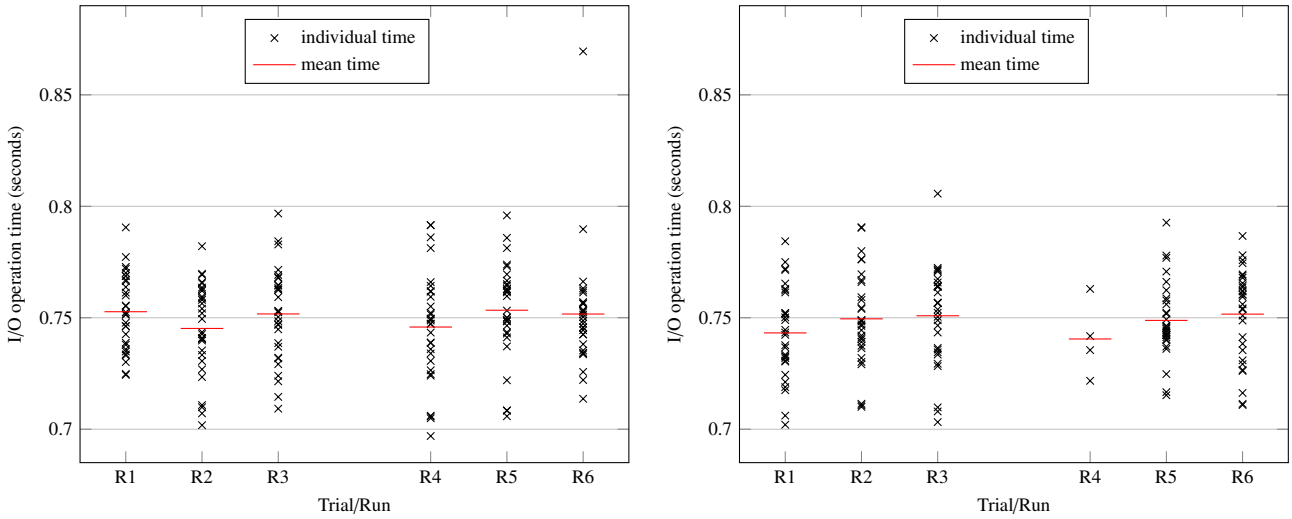
(a) Placement A (b) Placement B

Fig. 4 miranda_io performance when writing to the Lustre file system on Catalyst



(a) Placement A (b) Placement B

Fig. 5 IOR performance when reading from Lustre file system on TSUBAME2.5



(a) Placement A (b) Placement B

Fig. 6 miranda_io performance when reading from Lustre file system on Catalyst

TSUBAME2.5 Write Standard Deviation (s)

Run	Placement-A	Placement-B
1	4.0798	3.318
2	2.0081	3.9568
3	3.7168	3.1864
4	8.7604	2.7212

TSUBAME2.5 Read Standard Deviation (s)

Run	Placement-A	Placement-B
1	1.0748	0.9359
2	1.0413	1.2056
3	0.7601	1.4885
4	0.8813	0.5816

Catalyst Write Standard Deviation (s)

Run	Placement-A	Placement-B
1	0.6559	1.0148
2	2.969	6.4464
3	1.2174	1.0805
4	5.5699	1.1061
5	2.0233	1.0395
6	1.0907	0.9924

Catalyst Read Standard Deviation (s)

Run	Placement-A	Placement-B
1	0.017	0.0205
2	0.0206	0.0213
3	0.0209	0.022
4	0.0248	0.0172
5	0.0217	0.0169
6	0.0267	0.0205

Table 3 Standard deviation of I/O times (in seconds).

this experiment. The standard deviations were low (excepting in cases with abnormal spikes) and there were no notable differences in the times for cases with MPI traffic versus cases without MPI traffic.

Variations caused by system noise

In looking more closely at this issue, we ran IOR without network I/O, i.e., we wrote to the file system on the local SSD. As with the other cases, we ran trials with and without generating MPI activity on the system. The results of these trials are shown in Figure 7. The variations in the runtimes that we noted when writing to Lustre are also present when writing to SSD. This confirms that this variation is not caused by network I/O traffic interfering with MPI traffic.

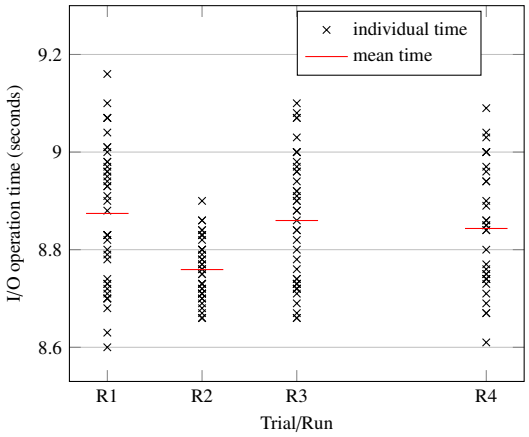


Fig. 7 IOR performance when writing to the on-node SSD on TSUBAME2.5 with placement B

Generated traffic is less than the peak network bandwidth

TSUBAME2.5 and Catalyst use InfiniBand QDR networks and have a peak single-rail bandwidth of 40Gbps, respectively. At the same time, each IOR and *miranda_io* process generates 111MB and 100MB of traffic, respectively, for each operation. Hence for the case of *miranda_io*, one node with 10 processes would generate 1.1GB over a 6 seconds read operation, equating to a rate of approximately 1.5 Gbps. This rate, while higher than the rate of the communication benchmarks, is much less than the peak bandwidth of each system. Furthermore, both systems use the fat-tree network topology, which supports full-bisection bandwidth and reduces bottleneck when multiple nodes are communicating simultaneously.

Generated I/O traffic seems representative of regular workloads

The amount of I/O traffic generated by our experiments was similar to the usual workload of a production supercomputer.

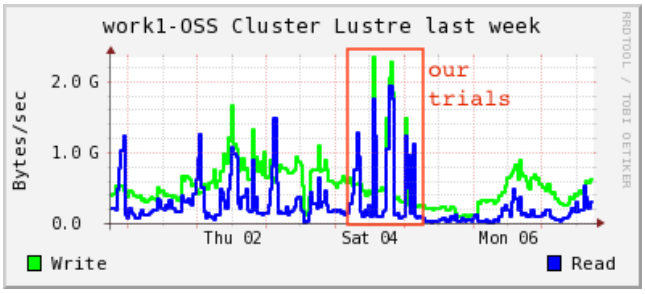


Fig. 8 Traffic measurements recorded for the /work1 file system on TUSBAME2.5 during our experiments. Measurements were reported at 6-minutes intervals over the period of one week.

This graph in Figure 8 shows the read and write throughput for the `/work1` Lustre file system that was used in our experiments on TSUBAME2.5. The section within the red box indicate the period of time when our experiments were conducted. The increase in traffic during our experiments was not much larger than some other periods earlier the same week, based on the 6-minutes measurement granularity of TSUBAME2.5's monitoring system.

7. Related Work

Extensive studies have been done related to parallel I/O performance from the aspect of measurement [5], modeling [6], and interference [7]. However, all of these work focus on interference on I/O performance cause by other I/O activities. None of these nor other surveyed literature investigate MPI traffic interference on I/O performance.

To the best of our knowledge, ours is the first work to investigate the interference caused by MPI traffic on I/O performance in this manner.

*Edited one Extensive studies have been done to assess parallel I/O performance from the aspect of measurement [5], modeling [6], and interference [7]. However, all of these works focus on interference on I/O performance caused by other I/O activities. None of these nor other surveyed literature to investigate MPI traffic interference on I/O performance.

To the best of our knowledge, ours is the first work to investigate the interference caused by MPI traffic on I/O performance in this manner.

8. Conclusion

Even though Lustre file system traffic and inter-node MPI traffic both utilize the same network infrastructure, our experiments have shown that regular I/O-bound workloads are not impacted by the presence of MPI traffic over the network. The transfer rate of typical I/O workloads on our system is well-below the capacity of modern high-speed HPC networks with full bisection bandwidth, like those of TSUBAME2.5 and Catalyst. Therefore, optimizations efforts for I/O-bound applications should not be focused on the negligible network interference from communication-bound applications. These efforts should focus on other aspects of the I/O subsystem.

For future work, we aim to identify the types of workload that can generate significant network interference for parallel I/O operations. We will also assess the impact of bottlenecks in other parts of the I/O subsystem.

Acknowledgments This research was supported by JST, CREST (Research Area: Advanced Core Technologies for Big Data Integration).

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-XXXXXX).

References

[1] Lustre: Lustre Parallel File System, `http://lustre.org/`.
[2] Lawrence Livermore National Laboratory: Miranda, `https://wci.llnl.gov/simulation/computer-codes/miranda`.
[3] Lawrence Livermore National Laboratory: Parallel filesystem I/O benchmark, `https://github.com/LLNL/ior`.
[4] Bhatele, A., Mohror, K., Langer, S. H. and Isaacs, K. E.: There Goes the Neighborhood: Performance Degradation Due to Nearby Jobs, *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, New York, NY, USA, ACM, pp. 41:1–41:12 (online), DOI: 10.1145/2503210.2503247 (2013).
[5] Chang, Y.-T. S., Jin, H. and Bauer, J.: Methodology and Application of HPC I/O Characterization with MPIProf and IOT, *Proceedings of the 5th Workshop on Extreme-Scale Programming Tools*, ESPT '16 (2016).
[6] Groot, S., Goda, K., Yokoyama, D., Nakano, M. and Kitsuregawa, M.: Modeling I/O Interference for Data Intensive Distributed Applications, *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13 (2013).
[7] Zhang, X. and Jiang, S.: InterferenceRemoval: Removing Interference of Disk Access for MPI Programs Through Data Replication, *Proceedings of the 24th ACM International Conference on Supercomputing*, ICS '10 (2010).

Appendix

A.1 IOR Configuration File

```
IOR START
   testFile=/path/to/file/on/lustreFS
   useO_DIRECT=1
   api=POSIX
   repetitions=1000
   verbose=2
   transferSize=10m
   blockSize=10m
   segmentCount=10
   writeFile=1
   readFile=1
   filePerProc=1
   randomOffset=0
   maxTimeDuration=8
IOR STOP
```