Benefit of DDN's IME-FUSE for I/O intensive HPC applications.

Eugen Betke[1] and Julian Kunkel[2]

[1] Deutsches Klimarechenzentrum, Hamburg, Germany,
betke@dkrz.de
[2] University of Reading, Reading, United Kingdom,
j.m.kunkel@reading.ac.uk

Abstract. Many scientific applications are limited by I/O performance offered by parallel file systems on conventional storage systems. Flash-based burst buffers provide significant better performance than HDD backed storage, but at the expense of capacity. Burst buffers are considered as the next step towards achieving wire-speed of interconnect and providing more predictable low latency I/O, which are the holy grail of storage.
A critical evaluation of storage technology is mandatory as there is no long-term experience with performance behavior for particular applications scenarios. The evaluation enables data centers choosing the right products and system architects the integration in HPC architectures.
This paper investigates the native performance of DDN-IME, a flash-based burst buffer solution. Then, it takes a closer look at the IME-FUSE file systems, which uses IMEs as burst buffer and a Lustre file system as back-end. Finally, by utilizing a NetCDF benchmark, it estimates the performance benefit for climate applications.

Keywords: Lustre, FUSE, evaluation, flash-based storage

1 Introduction

The dilemma of conventional high-performance storage systems based on HDDs is that they must maximize the throughput to reduce application run times and at the same time they shall minimize the provided bandwidth to reduce costs. The first requirement is often prioritized to the detriment of the second one, which typically ends up in the oversizing and in a low average usage of the bandwidth procured. The prioritization is motivated by the requirement to process large performance peaks particular due to checkpoint/restart workloads, that often occur in large-scale applications. However, since these systems are optimized for sequential I/O, data-intense workloads that are not following this pattern are unable to saturate the network – reducing the effective utilization.

Traditional parallel file systems can be deployed on flash-based storage instead of HDDs, increasing performance for random workloads. A nice work in this direction was done in [3]. Typically, data is accessed via POSIX interfaces

The final publication is available at Springer via https://doi.org/10.1007/978-3-030-02465-9_9

but can be accessed using MPI-IO [14]. MPI-IO is a widely accepted middleware layer for parallel I/O that relaxes the POSIX semantics and is designed for parallel I/O. In an alternative storage architecture, a burst buffer [8,11] is placed between compute nodes and the storage. Acting as an intermediate storage tier, it's goal is to catch the I/O peaks from the compute nodes. Therefore, it provides a low latency and high bandwidth to the compute nodes, but also utilizes the back-end storage by streaming data constantly at a lower bandwidth.

In-memory systems, like the Kove® XPD® [7], provide byte-addressable storage with better latency, endurance and availability as flash chips. Flash-based systems, like DDN IME [13], are also byte-addressable, but have different characteristics than an in-memory storage, for example, flash offers a better costs per gigabyte ratio.

Accessing a fast storage over a POSIX compliant file system or MPI-IO interface is an interesting option for many users, because neither changes in source code, nor software recompilation is required as long as it doesn't degrade the performance too much. Closed source and pre-compiled applications could also benefit from that. For that purpose, DDN developed a fuse module (IME-FUSE) which uses IME as a burst buffer and stores data on a parallel file system. In this evaluation we used Lustre as back-end.

Our **contributions** are: 1) we investigate peak performance of IME-native and IME-FUSE, and compare it to Lustre, 2) we estimate the performance behaviour for HPC applications, that access data using NetCDF library.

This paper is structured as follows: Section 2 discusses related work, then Section 3 describes the test environment. Section 4 and 5 show the test setup and performance results. Finally, the paper is summarized in Section 6.

2 Related Work

Relevant state-of-the-art can be grouped into performance optimization, burst buffers to speedup I/O and in-memory storage solutions.

Optimization and tuning of file systems and I/O libraries is traditionally an important but daunting task as many configuration knobs can be considered in parallel file system servers, clients and the I/O middleware. Without tuning, typical workloads stay behind the peak-performance by orders of magnitude. With considerable tuning effort a well fitting problem can yield good results: [15] reports 50% peak performance with a single 291 TB file. In [4] MPI-IO and HDF5 were optimized and adapted to each other, improving write throughput by 1.4x to 33x.

Many existing workloads can take benefit of a burst buffer as a fast write-behind cache that transparently migrates data from the fast storage to traditional parallel file system. Burst buffers typically rely on flash or NVRAM to support random I/O workloads. For flash based SSDs, many vendors offer high-performance storage solutions, for example, DDN Infinite Memory Engine (IME) [2], IBM FlashSystem [5] and Cray's DataWarp accelerator [1]. Using comprehensive strategies to utilize flash chips concurrently, these solutions are

powerful and robust to guarantee availability and durability of data for many years.

The integration of Cray DataWarp burst buffer into the NERSC HPC architecture [10] increased the I/O performance of Chumbo-Crunch simulator by 2.84x to 5.73x, compared to Lustre. However, for the sake of efficient burst buffer usage, the serial simulator workflow had to be split into single stages (i.e., simulation, visualization, movie encoding), which then were executed in parallel. The research group at JSC uses DDN IME burst buffer [12] and GPFS to identify requirements for the next HPC generation. The main purpose is to accelerate the I/O performance of the NEST ("NEural Simulation Tool"). The preliminary IOR experiments show, that I/O performance can be increased upto 20x. BurstFS [16] uses local NVRAM of compute nodes, instead of dedicated remote machines. An elaborated communication scheme interconnects the distributed NVRAM and provides a contiguous storage space. This storage is allocated at beginning and exists for the lifetime of the job. In the experiments, BurstFS outperforms OrangeFS and PLFS by several times.

The usage of DRAM for storing intermediate data is not new and RAM drives have been used in MSDOS and Linux (with tmpfs) for decades. However, offered RAM storage was used as temporary local storage and not durable and usually not accessible from remote nodes. Exporting tmpfs storage via parallel file systems has been used mainly for performance evaluation but without durability guarantees. Wickberg and Carothers introduced the RAMDISK Storage Accelerator [18] for HPC applications that flushes data to a back-end. It consists of a set of dedicated nodes that offer in-memory scratch space. Jobs can use the storage to pre-fetch input data prior job execution or as write-behind cache to speedup I/O. A prototype with a PVFS-based RAMDISK improved performance of 2048 processes compared to GPFS (100 MB/s vs. 36 MB/s for writes). Burst-mem [17] provides a burst buffer with write-behind capabilities by extending Memcached [6]. Experiments show that the ingress performance grows up to 100 GB/s with 128 BurstMem servers. In the field of big data, in-memory data management and processing has become popular with Spark [19]. Now there are many software packages providing storage management and compute engines [20].

The Kove XPD [7] is a robust scale-out pooled memory solution that allows aggregating multiple Infiniband links and devices into one big virtual address space that can be dynamically partitioned. Internally, the Kove provides persistence by periodically flushing memory with a SATA RAID. Due to the performance differences, the process comes with a delay, but the solution is connected to a UPS to ensure that data becomes durable in case of a power outage. While providing many interfaces, the XPD does not offer a shared storage that can be utilized from multiple nodes concurrently.

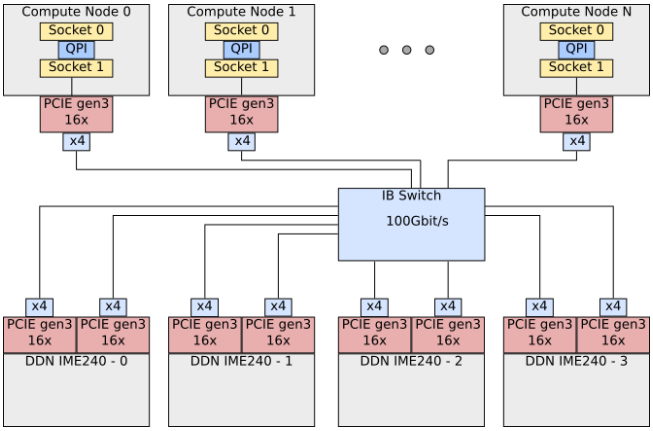


Fig. 1: DDN test cluster

3 Test environment

DDN provided access to their test cluster in Düsseldorf on which 10 nodes could be used for testing. Each node is equipped with two Sandy Bridge processors (8 cores, E5-2650v2 @2.60GHz) and 64 GB RAM. They are interconnected with a Mellanox Connect-X-4 card providing 100 Gb/sec (4x EDR). As storage, a DDN ES14K (Exascale 3.1) with two metadata servers and Lustre 2.7.19.12 is provided; additionally, an IME system consisting of 4 servers is provided. The flash native data cache of IME acts as a burst buffer and is drained to the Lustre system, the performance reported with IOR is 85 GB/s in write mode. The DDN IME provides byte-addressable flash-based storage space with high performance characteristics. It can be addressed directly (IME-native) in a fast and efficient way, but DDN also provides a number of convenient solutions, that require less integration effort. (1) The applications can be re-linked to the MPI-IO implementation with IME support, which was developed by DDN. (2) Then, DDN provides a fuse module (IME-FUSE) with IME support, which are convenient ways to access a shared storage. Both file systems are POSIX compliant and can be used by the applications without any source code modification, recompilation, or re-linking. In the conducted tests, IME is used via its FUSE mount and backed by the DDN Lustre. We assume during the write experiment, data is kept inside the burst buffer and not written back, albeit we cannot ensure this. The DDN cluster is a experimental system with a lightweight software setup. Especially, the exclusive access to the IME was not guaranteed, so that some results could be affected by other users. Therefore, we don't draw conclusions from outliers, since we don't know the origin of them.

3.1 Benchmarks

As our primary benchmark, IOR [9] is used varying access granularity, processes-per-node, nodes and access pattern (random and sequential). The official version of IOR allows us to measure the real performance without considering open/close times (see Equation (1)). To synchronize the measurements and capture time for open, close and I/O separately, inter-phase barriers are turned on (IOR option -g). The DDN version (IME-IOR) supports IME-native interface, but doesn't allow measuring real I/O performance. Therefore, the performance values include open/close times (see Equation (2)).

$$\text{perf}_{\text{Lustre, IME-FUSE}} = \frac{\text{filesize}}{t_{\text{io}}} \tag{1}$$

$$\text{perf}_{\text{IME-native}} = \frac{\text{filesize}}{t_{\text{total}}} = \frac{\text{filesize}}{t_{\text{open}} + t_{\text{io}} + t_{\text{close}}} \tag{2}$$

Since the IOR benchmarks does not support NetCDF, and HDF5 is only supported with limited configuration of the pattern, additionally, the NetCDF-Bench has been used[3]. This benchmark uses the parallel NetCDF interface to read/write patterns on a 4D dataset into a NetCDF4/HDF5 file. It decomposes a domain geometry of (t,x,y,z), e.g., (100,16,64,4) across the processes of an MPI parallel program. The processes partition the geometry in x and y direction and one time step is accessed per iteration of each parallel process. Various options to control the optimizations and data mappings from NetCDF are exported by the benchmark (chunking vs. fixed layout, unbound dimensions, chunk size, pre-filling).

Finally, to measure performance of individual operations to investigate variability, the sequential benchmark `io-modelling` is used[4]. It uses a high-precision timer and supports various access patterns on top of the POSIX interface.

4 Experiment Configuration

On the DDN cluster, we use NetCDF-Bench, IOR, and IME-IOR to measure the IME's throughput, and use `io-modelling` for testing variability. Each test configuration is repeated 10 times. All experiments are conducted with block sizes 16, 100, 1024, and 10240 KiB.

To find the performance limits of the test system we use the IOR benchmarks. For that purpose, we conduct a series of experiments with various parameters, where we measure the performance for {read, write} × {random, sequential} × {POSIX, MPIIO} × {Lustre, IME-FUSE, IME-native} × {collective, independent}. The stripe count on Lustre is twice as large as the number of nodes.

The purpose of NetCDF-Bench is to investigate the I/O behaviour of typical scientific application, that access large variable through NetCDF4. In the

[3] `https://github.com/joobog/netcdf-bench`
[4] `https://github.com/JulianKunkel/io-modelling`

experiment, we varied the following parameters: {Lustre, IME-FUSE} × {read, write} × {chunked, contiguous} × {collective, independent}.

With `io-modelling` benchmark we looked at the variability of individual I/O accesses {Lustre, IME-FUSE} × {read, write} × {random, sequential}.

4.1 Open/close times

The time of open/close reduces the reported performance of IME. They are dropped whenever possible for two reason. Firstly, in our experiments the test file size is variable (filesize = 100·blocksize·NN·PPN), it affects small experiments more than the larger ones. Additionally, it should be noted, that for production runs, larger files and capacities are assumed, reducing this overhead. Unless otherwise stated, the performance reported in this paper was measured without open/close times.

The goal of our evaluation is to systematically investigate the scaling behavior of the DDN IME's, IME-FUSE and Lustre. In the following experiments we use 1-10 client nodes (NN) and 1-8 processes per node (PPN) to push hardware to the limits. On each compute node only one CPU is used, that is connected directly to the Infiniband adapter, to avoid the QPI overhead. To provide reliable results, each experiment was repeated 10 times.

4.2 Performance

Table 1 shows the best and the average performance values that were observed with IME-IOR during the test runs on a single node and on 10 nodes for random and sequential I/O. Based on average performance for random I/O with NN=1 and PPN=8, 10 client nodes can achieve a throughput of 61 GB/s and of 80 GB/s for write and read, respectively. As Table 1 shows, the measured write performance is similar to expected values, which indicates that the compute nodes are the bottlenecks. But the measured read performance is significantly lower than expected. This indicates, that the bottleneck here are the IMEs. The same considerations apply to sequential performance.

5 Evaluation

IME-native (Figures 2 and 3): Characteristic for IME-native is that for each block size, there is a linear dependency between read and write accesses. The performance behavior for each block size can be approximated by a linear function and that small block sizes tend to have better write behaviour.

The complete set of performance results for random I/O is shown in Figure 3. Firstly, it confirms the linear scalability. Secondly, there is also no regression of the curves, probably because the experiment setup couldn't push the IMEs to the limits. Further observations are: 1) writing small blocks is more efficient than reading small blocks; reading large blocks is more efficient that writing large blocks, 2) performance increases with increasing access granularity. 3) with 1 or

4 PPN the available network bandwidth is not utilized. With PPN=8, we are close to the available network bandwidth for 1 and 10 MiB accesses. Hence, the I/O path involves relevant latencies.

Lustre (Figure 4a): Firstly, a single node can profit from caching, when reading data. In this case observable performance can rise up to 37 GiB/s (not shown in the figure). The caching effects disappear for NN > 1, hence we ignore them in further discussion. Secondly, the read performance don't exceed 17.4 GiB/s, and is achieved with NN=10, PPN=8, BS=100 KiB. This is a

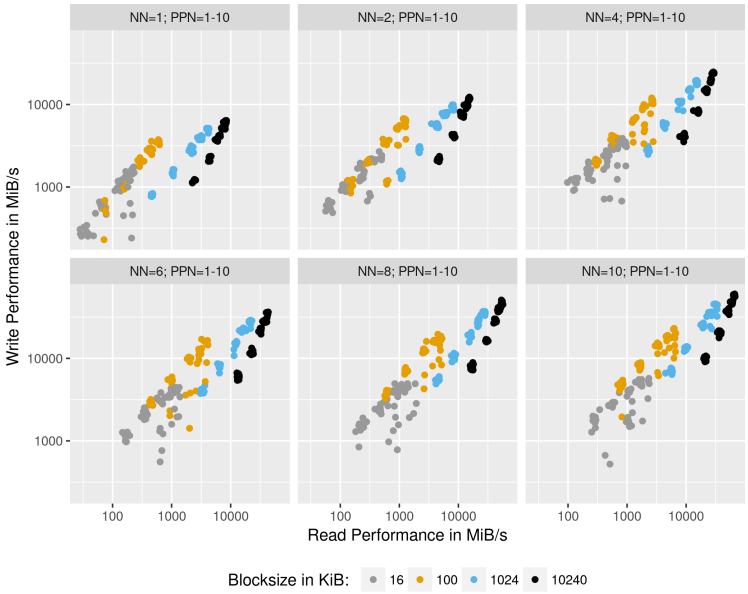


Fig. 2: Random access performance depending on blocksize and PPN

NN	PPN	Best Performance in [MiB/s] read	Best Performance in [MiB/s] write	Mean Performance in [MiB/s] read	Mean Performance in [MiB/s] write	I/O type	File size in [MiB]
1	1	2,560	1,240	2,400	1,180	rnd	1000
1	1	2,290	1,230	2,000	870	seq	1000
1	8	8,500	6,390	8,100	6,120	rnd	8000
1	8	8,700	6,380	7,100	4,530	seq	8000
10	1	22,300	10,700	21,200	10,000	rnd	10000
10	1	23,200	10,800	22,200	8,430	seq	10000
10	8	67,500	60,200	65,300	58,400	rnd	80000
10	8	67,500	62,900	61,700	54,300	seq	80000

Table 1: The best and mean performance measured with IME-IOR (blocksize: 10MiB) (NN: number of nodes; PPN: processes per node).

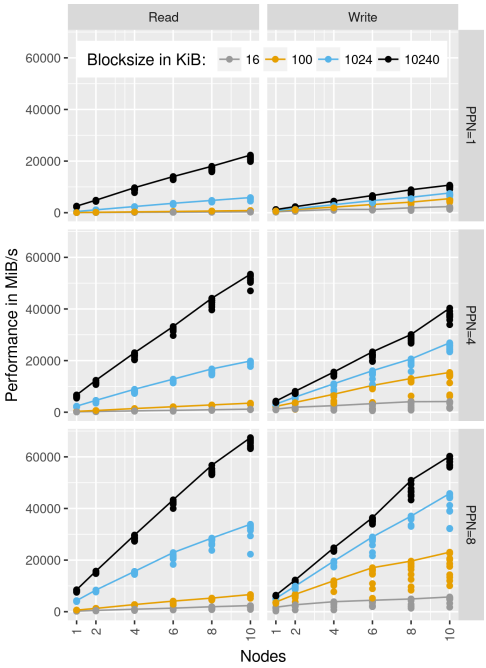


Fig. 3: IME-native random I/O performance (lines go through max. values)

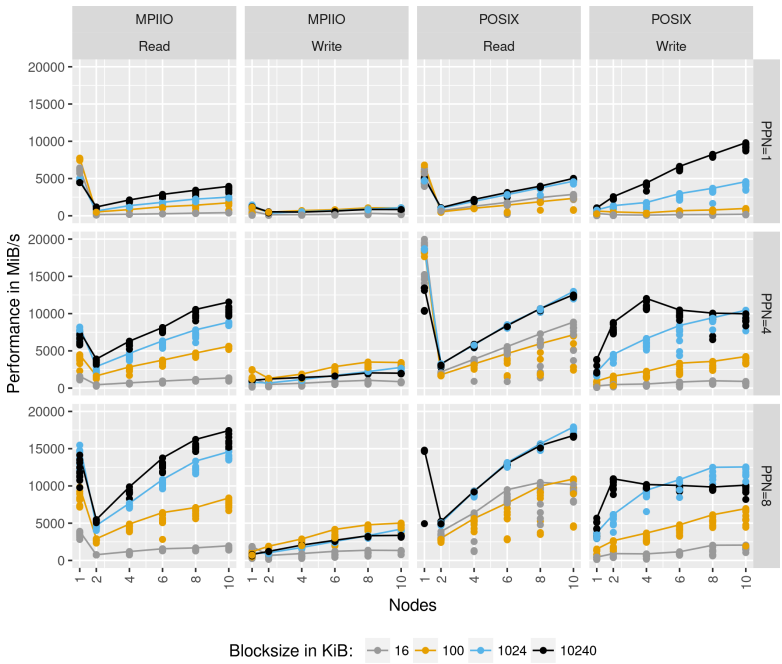
contra-intuitive, because usually large block size show better performance. The best write performance is 11.8 GiB/s, and is achieved with NN=4, PPN=6, BS=1000 KiB. This measurement and the incrementally flattening curve indicate a poor scalability of Lustre. Generally speaking, Lustre has a lot of internal overhead, especially to make it POSIX compliant, e.g. distributed lock management. Thirdly, a particular striking point is the result for MPI-IO write performance. It is significantly lower than for other configurations. For this behaviour we have no explanation at the moment. It is also a confusing result, because it is in contradiction to our later experiment with NetCDF-Bench (Figure 5). NetCDF4 uses MPI-IO as back-end, but achieves better results.

IME-FUSE (Figure 4b): The file system shows a linear scalability, similar to the IME-native, but provides less I/O performance, especially for reading. This is probably caused by the FUSE overhead, which includes moving I/O requests from user space to kernel space, and then from kernel space to IME-FUSE.

5.1 Application Kernel Using HDF5

In this experiment, the HDF5 VOL development branch (date 2016-05-09), NetCDF 4.4.1 and NetCDF-bench is used. Several values for the 4D data geometry of raw integer data have been explored. For each block size we did 100 measurements. The configuration parameters are summarized in Table 2.

(a) Lustre



(b) IME-FUSE

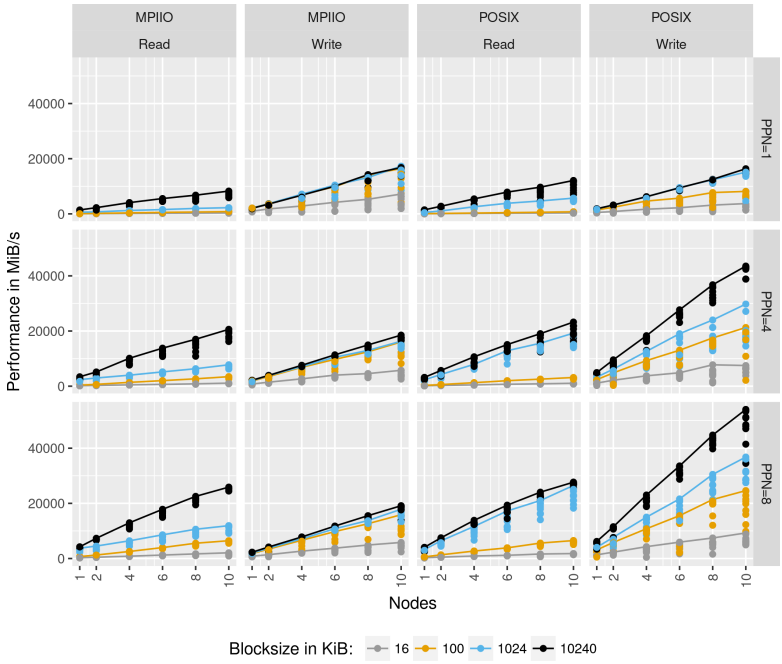


Fig. 4: Random I/O performance (lines go through max. values)

Parameter (-d) (t:x:y:z)	Data size [in GiB]	Block size [in KiB]
(100:16:64:4)	0.5	16
(100:16:64:25)	3.1	100
(100:16:64:256)	7.8	1024
(100:16:64:2560)	78.1	10240

Table 2: NetCDF-Bench configuration used in during the benchmark.

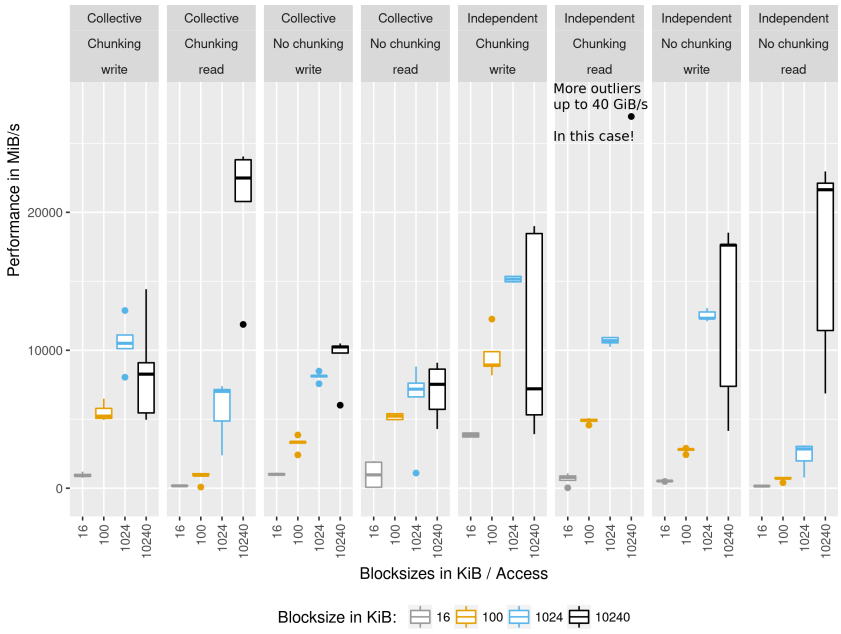


Fig. 5: NetCDF performance for Lustre (similar to IME-FUSE)

In the experiments, we use 10 client nodes and 8 processes per node to access a shared file. All experiments were conducted with fixed dimension sizes only, since the unlimited/variable dimensions are not supported in combination with independent I/O in NetCDF4. Figure 5 shows the results. Generally, as expected, independent chunked I/O was a good configuration.

Lustre vs. IME-FUSE: Generally, the performance looks very similar for Lustre and IME-FUSE, that is why we only included the picture for Lustre. There are a few differences: (1) Collective I/O without chunking causes large variability while reading 16 KiB blocks, (2) and better performance while writing 10 MiB blocks on Lustre. (3) If chunking is enabled and independent I/O is used, then 10 MiB block sizes can be read with a low variability. The best performance achieved for collective read is 23 GiB/s write 14 GiB/s, and for independent read 40 GiB/s and write 18 GiB/s.

Chunking vs. no chunking: Read performance suffers a lot on both file systems, if chunking is enabled for small blocks. The probability, that several NetCDF processes access the same chunk, increases for small block sizes. In this case, the processes have to load the whole chunk on each node into memory, even if only a small part of it is required. Such inefficient access patterns can lead to unnecessary data transfer over the network, i.e. when large parts of the data are pre-loaded, but aren't unused. This doesn't apply to large block sizes. Therefore, we can observe performance advantages.

Independent I/O vs. collective I/O: If chunking is enabled, collective I/O degrades the performance. If chunking is disabled, it improves I/O for small blocks and degrades I/O of large blocks.

Caching: For large block sizes (10204 KiB) independent chunked read performance outperforms the write performance. We suppose that cache is responsible for this performance speed-up.

5.2 Performance variability with individual I/Os.

This experiment is conducted measuring timing of 10,000 or 1,024 individual I/Os with a single process on IME test cluster on IME-FUSE and Lustre. Figure 6 shows the qualitative difference between the file systems. The figure shows the density (like a smoothened histogram) of the individually timed I/Os. We observe 1) the read operations on Lustre are faster than using IME-FUSE – this is presumably due to client-side caching. 2) the random acceleration of IME improves write latencies/throughput for IME.

6 Conclusion

IME is a burst buffer solution, that is completely transparent to applications and to users. These properties make it beneficial for random workloads. Read performance depends whether data is located on the IME flash or on Lustre. The data migration policy is usually hidden from the users, so that read behaviour is not known in advanced. There is an API though to allow users to stage data explicitly.

For large access sizes and processes per node, IME was able to nearly saturate the network. We did not achieve better performance with IME in all test scenarios, particularly, for the NetCDF benchmark. The reason for the suboptimal performance gain of IME compared to Lustre may be due to: 1) the access pattern caused by NetCDF4 with HDF5 has a considerable overhead; 2) the Lustre storage from DDN is already well optimized; 3) the small and experimental laboratory setup that we used for testing. We expect a significant performance gain once more clients access IME. Further large-scale investigation is necessary.

Acknowledgment

Thanks to DDN for providing access to the IME test cluster and to Jean-Thomas Acquaviva for the support.

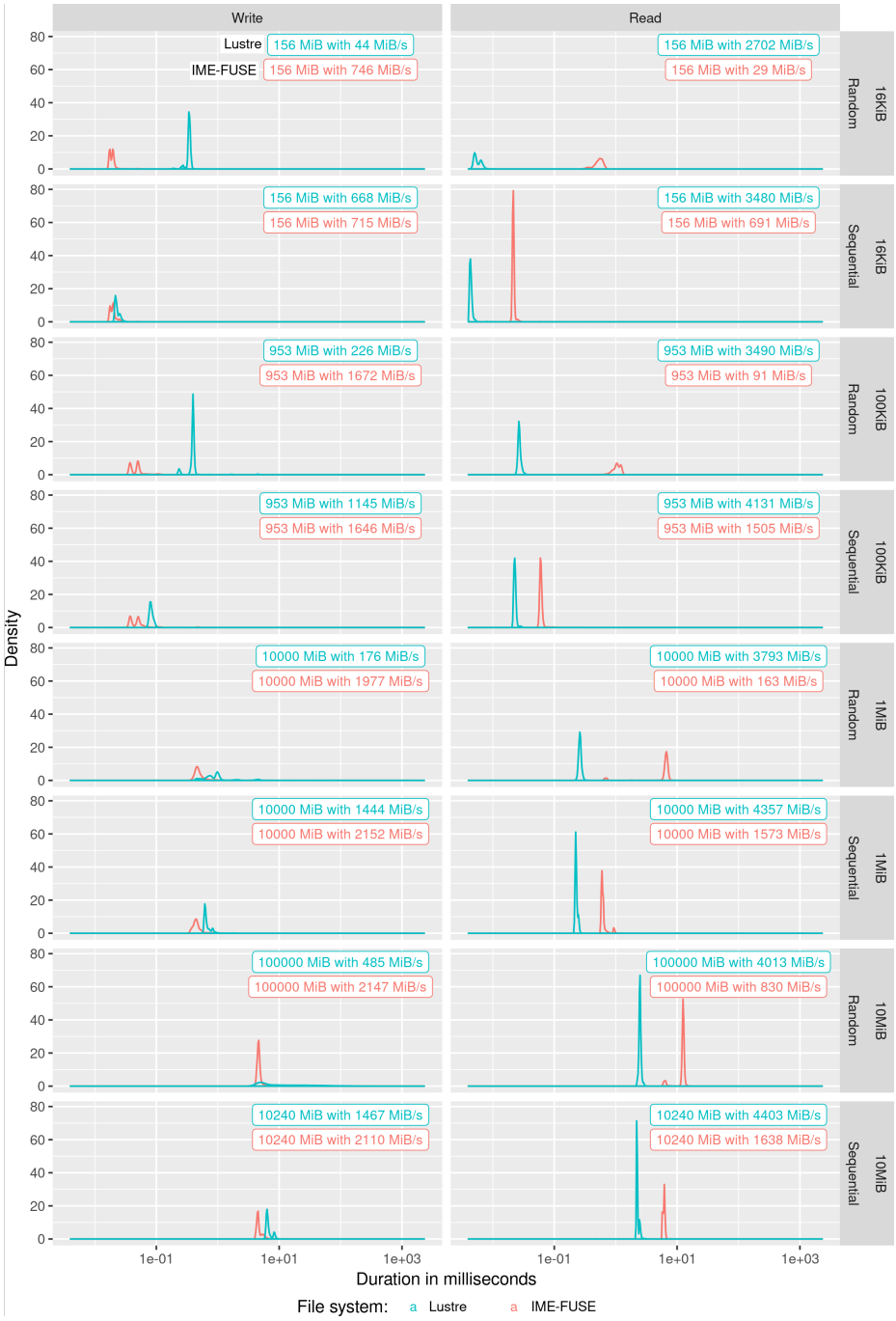


Fig. 6: Density of timing individual I/O operations

References

1. Cray: CRAY XC40 DataWarp Applications I/O Accelerator. http://www.cray.com/sites/default/files/resources/CrayXC40-DataWarp.pdf
2. DDN: Worlds's most advanced application aware I/O acceleration solutions. http://www.ddn.com/products/infinite-memory-engine-ime14k
3. Hebenstreit, M.: Performance Evaluation of Intel SSD-Based Lustre Cluster File Systems at the Intel CRT-DC. Tech. rep., Intel (2014), http://www.intel.com/content/dam/www/public/us/en/documents/performance-briefs/lustre-cluster-file-system-performance-evaluation.pdf
4. Howison, M., Koziol, Q., Knaak, D., Mainzer, J., Shalf, J.: Tuning HDF5 for Lustre File Systems. In: Workshop on Interfaces and Abstractions for Scientific Data Storage (IASDS10), Heraklion, Crete, Greece, September 24, 2010 (2012)
5. IBM: Flash Storage. http://www-03.ibm.com/systems/storage/flash
6. Jose, J., Subramoni, H., Luo, M., Zhang, M., Huang, J., Wasi-ur Rahman, M., Islam, N.S., Ouyang, X., Wang, H., Sur, S., et al.: Memcached design on high performance RDMA capable interconnects. In: 2011 International Conference on Parallel Processing. pp. 743–752. IEEE (2011)
7. KOVE: about xpress disk (xpd). http://www.hamburgnet.de/products/kove/Kove-XPD-L3-4-datasheet.pdf (2015)
8. Liu, N., Cope, J., Carns, P., Carothers, C., Ross, R., Grider, G., Crume, A., Maltzahn, C.: On the role of burst buffers in leadership-class storage systems. In: In Proceedings of the 2012 IEEE Conference on Massive Data Storage (2012)
9. Loewe, W., McLarty, T., Morrone, C.: IOR Benchmark (2012)
10. Ovsyannikov, A., Romanus, M., Straalen, B.V., Weber, G.H., Trebotich, D.: Scientific Workflows at DataWarp-Speed: Accelerated Data-Intensive Science using NERSC's Burst Buffer (2016), http://conferences.computer.org/pdswdiscs/2016/papers/5216a001.pdf
11. Romanus, M., Parashar, M., Ross, R.B.: Challenges and considerations for utilizing burst buffers in high-performance computing. arXiv preprint 1509.05492 (2015)
12. Schenck, W., El Sayed, S., Foszczynski, M., Homberg, W., Pleiter, D.: Early Evaluation of the Infinite Memory Engine" Burst Buffer Solution, pp. 604–615. Springer International Publishing, Cham (2016), http://dx.doi.org/10.1007/978-3-319-46079-6_41
13. Storage, D.: Burst buffer & beyond; I/O & Application Acceleration Technology. DDN Storage (9 2015)
14. Thakur, R., Gropp, W., Lusk, E.: On Implementing MPI-IO Portably and with High Performance. In: Proceedings of the Sixth Workshop on I/O in Parallel and Distributed Systems. pp. 23–32 (1999)
15. The HDF Group: A Brief Introduction to Parallel HDF5. https://www.alcf.anl.gov/files/Parallel_HDF5_1.pdf
16. Wang, T., Mohror, K., Moody, A., Sato, K., Yu, W.: An Ephemeral Burst-buffer File System for Scientific Applications. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 69:1–69:12. SC '16, IEEE Press, Piscataway, NJ, USA (2016), http://dl.acm.org/citation.cfm?id=3014904.3014997
17. Wang, T., Oral, S., Wang, Y., Settlemyer, B., Atchley, S., Yu, W.: BurstMem: A high-performance burst buffer system for scientific applications. In: Big Data (Big Data), 2014 IEEE International Conference on. pp. 71–79. IEEE (2014)

18. Wickberg, T., Carothers, C.: The RAMDISK storage accelerator: a method of accelerating I/O performance on HPC systems using RAMDISKs. In: Proceedings of the 2nd International Workshop on Runtime and Operating Systems for Supercomputers. p. 5. ACM (2012)
19. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M.J., Shenker, S., Stoica, I.: Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. pp. 2–2. USENIX Association (2012)
20. Zhang, H., Chen, G., Ooi, B.C., Tan, K.L., Zhang, M.: In-memory big data management and processing: A survey. IEEE Transactions on Knowledge and Data Engineering 27(7), 1920–1948 (2015)