Hybrid Flash Arrays for HPC storage systems

An alternative to Burst Buffers

Torben Kling Petersen, PhD
HPC Storage Solutions
Seagate Technology
60 Norden Road, Maidenhead, Berkshire, SL6 4AY, UK
torben.kling.petersen@seagate.com

John Bent, PhD
Seagate Government Solutions
2300 Dulles Station Boulevard
Suite 230, Herndon, VA, USA
John.Bent@seagategov.com

***Abstract*— Cloud and high-performance computing storage systems are comprised of thousands of physical storage devices and uses software that organize them into multiple data tiers based on access frequency. The characteristics of these devices lend themselves well to these tiers as devices have differing ratios of performance to capacity. Due to this, these systems have, for the past several years, incorporated a mix of flash devices and mechanical spinning hard disk drives. Although a single media type will be ideal, the economic reality is that a hybrid system must use flash for performance and disk for capacity. Within the high-performance computing community, flash has been used to create a new tier called burst buffers which are typically software managed, user visible, wed to a particular file system, and buffer all IO traffic before subsequent migration to disk. In this paper, we propose an alternative architecture that is hardware managed, user transparent, file system agnostic, and that only buffers small IO while allowing large sequential IO to access the disks directly. Our evaluation of this alternative architecture finds that it achieves comparable results to the reported burst buffer numbers and improves on systems comprised solely of disks by several orders of magnitude for a fraction of the cost.**

Keywords*— HPC storage, I/O performance, PD-RAID, benchmarking, flash acceleration, Burst Buffer, tiered storage

I. Introduction

Performance of high-performance computing (HPC) storage systems has always been the main selection criteria when procuring a new file system for a new or existing high performance computing cluster. As the amounts of data required for many application workflows [1, 2] is increasing exponentially, the need for higher-performance storage systems is correspondingly increasing. This applies to academic, research and commercial HPC users and the patterns by which different types of users accessing persistent data are homogenizing. This is true across a large range of scientific computing from computational fluid dynamics calculations for the next airliner, weather simulations, high-fidelity physics simulations, video analytics, computational chemistry, health sciences and genomics, material science and energy centric exploration.

However, not all applications are written with data access in mind and a significant majority have I/O patterns that are not optimal for modern storage systems. In addition, most, if not all, large scale HPC systems are intended for many different concurrent users, many using different applications and data sets. From the storage solution point of view, this means a mix of large streaming I/Os, multiple un-aligned streams, small block writes and very random I/O overall. When a single application accesses storage with large sequential IO streams, storage performance is high and consistent. More random access patterns however can very quickly deteriorate performance by orders of magnitude [3]. Unfortunately, any mix of random and sequential workloads brings the entire system down to the performance level of random resulting in storage systems that almost chronically underperform [4].

This has led to many users looking at using flash based storage and its presumed advantages over traditional spinning storage media in terms of superior bandwidth and IOPS per dollar. *Burst buffers* [5] are the most well-known example of flash accelerated storage in HPC but is only one possible model of flash acceleration. The other two variants are compute node local flash acceleration [6] and hybrid storage solutions [7].

II. Flash Acceleration Concept

The basic idea of flash acceleration is to provide data intensive applications the means of accessing relevant data faster and with lower latency compared to traditional disk based file system. Current compute and storage architectures allows for adding flash acceleration in many different places. Each possible placement has both advantages and drawbacks and will provide acceleration in different ways. The three possible placements are *local* placement within compute nodes, in dedicated nodes often referred to as a *burst buffer,* and embedded within the storage system. We will now describe each of these in more detail.

A. Compute Node Local Flash Acceleration

The basic concept here is to equip all compute nodes in a cluster with direct attached flash storage or similarly use custom flash enhanced nodes in a larger SMP to facilitate a built-in tier of very low latency storage enabling extremely fast data access. An obvious problem here is that, for the most part, the data is not shared between different nodes or the access to data suffers from non-deterministic variable delayed access on remote nodes like the NUMA problems on multi-socket machines. Another problem is that this direct-attached flash effectively couples the failure domains of data and compute;

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

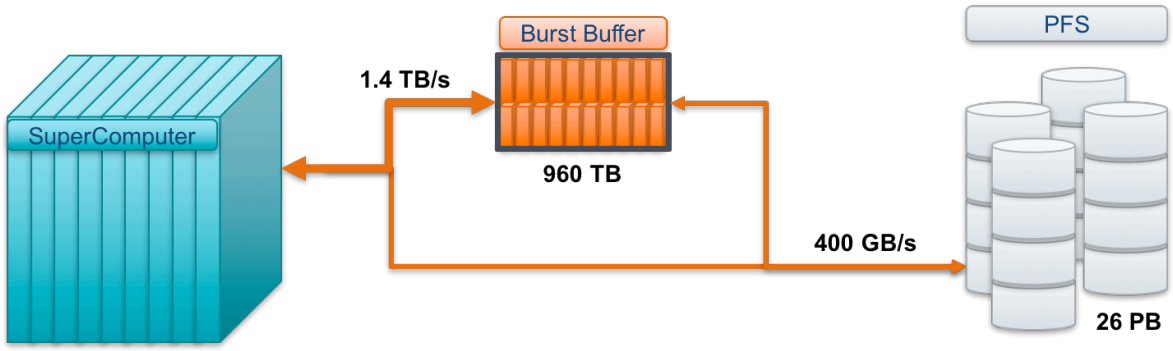

Figure 1 - Conceptual drawing of a network attached burst buffer and a backend parallel file system

when a compute node fails, its data is lost. With decoupled flash, there is implicit redundancy between the data stored in the flash and the application's running state. This is especially true for checkpoint IO in which a running application can survive the independent loss of its checkpoint data and the checkpoint data can be preserved for restarted applications [8].

However, this approach can be extremely beneficial for embarrassingly parallel applications where little or no data communication between different data nodes are required.

B. "Burst Buffers"

The A burst buffer has multiple different definitions depending on the user and ranges from server side flash acceleration such as Cray's DataWarp [9] to a separate network attached tier such as DDN's IME [10]. In this paper, we define it as a separate storage tier sitting between the compute system(s) and the main storage volume using the same high speed interconnect. The concept here is to provide a very fast storage volume off-loading the compute and then an asynchronous migration of that data to a more permanent media, usually a parallel file system. While a burst buffer can be beneficial for specific workloads such as check pointing and pre-staging of rapid access data, data migration essentially invalidates the concept. In addition, most common burst buffers require custom clients, optimized I/O libraries requiring a recompilation of the applications and separate paths to the flash layer and creating very complex architectures.

In the example above, the Burst Buffer has 1/20th of the capacity of, and approximately 4x the performance of, the main storage system. Both systems are connected to the same high speed interconnect and the relative performance is gated by each storage tier.

C. Storage Flash Acceleration

While there are several different approaches to flash acceleration with in the storage tier., the basic concept stays the same. By combining a large number of HDDs as the main capacity volume, a few SSDs can be used to accelerate specific storage components. While not unique, Seagate has for many years included SSDs in the modular storage enclosures for off-loading the Write Intent Bitmaps (WIBs) and standard Linux journals and thereby improving the performance of the parallel file system running on these nodes.

Storage side flash acceleration of the actual workload is a different matter entirely. For workload acceleration, the I/O software stack must be able to handle data ingestion and subsequent migration to the HDD layer in a fully transparent manner. While there are several examples of fully integrated architectures with this functionality, most notably ZFS with its L2ARC and ZIL components, the end result is not suitable for typical HPC workloads due to the overhead of the design.

III. Burst Buffer Problem Statement

In the burst buffer example above, the basic problems are that while it's possible to write data to the flash tier at very high performance (1.6 TB/s), the capacity limitations mean that writing can be done at full speed for slightly less than 11 min. After that the data must be migrated of the back-end tier. Unless all I/O activity between the compute and the backend storage tier stops completely, the burst buffer data migration will have to share the bandwidth and would get a fair share of 200 GB/s. At this performance, it would take more than 83 min to migrate all data.

Basically, the solution would only be able to run at full performance 13% of the time. While this scenario is an extrapolation of a use case that in real life, most likely, is more dynamic, the basic tenement stands. And as the cost of said burst buffer can be as high as 50% of the total storage cost, one might speculate that there must be a better way.

IV. File System Implications

The most common solution used today to provide supercomputing data storage is parallel file systems. While there are many different file systems on the market today that can (at least to a certain degree) provide parallel access to data, only two are relevant to large systems and the performance requirements seen today: Lustre and IBM Spectrum Scale.

While this paper is not focusing on the file systems mentioned above, any flash acceleration solution would have to interact with either or preferably both alternatives.

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

Seagate recently [11] announced a new model of flash acceleration of parallel file systems called Nytro Intelligent I/O Manager (NytroXD) that tries to address this problem. Physically NytroXD is a hybrid array that combines a small number of flash with a larger quantity of HDD. Logically NytroXD works as a reliable block device. Seagate currently packages it as the scalable storage unit within both Lustre and Spectrum Scale systems. To accelerate storage workloads, NytroXD inspect incoming I/O and routes small I/O to the flash and large IO to the HDD as is shown in Figure 2.

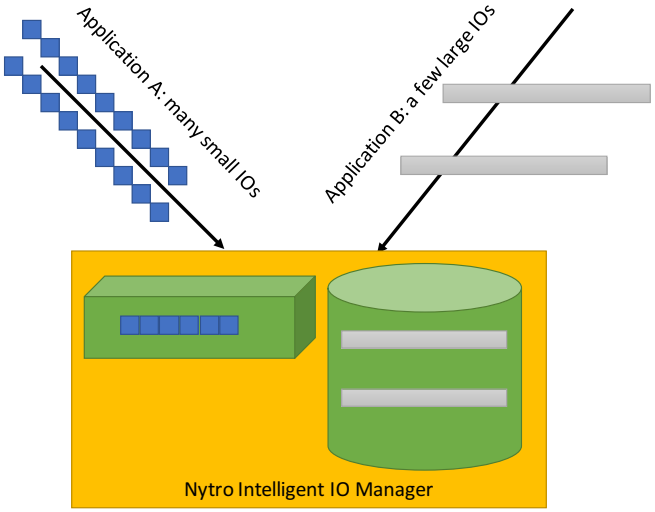

Figure 2 - The Nytro Intelligent I/O Manager routes small I/O to flash and large IO to HDD.

Note that this intelligent internal routing helps with both *temporal* and *spatial* fragmentation. Temporal fragmentation occurs when multiple applications simultaneously perform IO to a shared storage system. In these situations, even applications which are issuing large sequential IO with the reasonable expectation of high storage bandwidth can receive unexpectedly low performance [12]. This is because the storage system cannot solely service those large IOs but must simultaneously intersperse servicing IOs from a different application which might be doing smaller, lower performing, IOs. This is more generally referred to as the well-known phenomenon of storage interference.

Spatial fragmentation occurs due to normal file system aging in which data is written and deleted thereby leaving gaps throughout the storage allocation area [13]. That storage fragmentation degrades storage performance has been known for many decades [14]. This degradation is due to the fact that even though an application is issuing a large IO which should achieve high HDD bandwidth, the actual layout of the data onto the storage cannot place it in one contiguous physical location thereby rendering a logical large I/O effectively into a set of small physical I/Os. By separating small and large IO at the block layer, NytroXD can mitigate the performance penalties of both temporal and spatial fragmentation.

V. Aims

The aim of this study is to examine the validity of the Nytro Intelligent I/O Manager with regards to I/O intensive workloads and the benefits it provides for both Lustre and Spectrum Scale file system based storage solution.

VI. Material and Methods

A. *Storage Subsystem*

Our study utilized a Seagate L300N solution for the main benchmarking. The system consists of the following components:

- A 5 RU, 84 slot storage array with 2 embedded application controllers.
- SSU HDD: 6TB MakaraBP
- SSU SSD: Seagate ST800FM0183
- A system management server
- A Lustre metadata server
- 16 Lustre client nodes
- An EDR infiniband interconnect and a GbE based management network

B. *Parallel file system*

This study utilized back ends installed with the following storage configuration:

- Lustre 2.5.1 running on Seagate GridRAID (parity de-clustered RAID)
- Seagate NytroXD version 3.0.10.0

The following Lustre client tunings were used:

- `max_rpcs_in_flight=256`
- `max_dirty_mb=1024`
- `max_pages_per_rpc=1024`
- `read_ahead_step=4`
- `checksums=0`
- `lru_size=0`
- `max_read_ahead_mb=1024`
- `max_read_ahead_per_file_mb=1024`
- `lnet.debug=0`

C. *Software*

All I/O tests were done using IOR-2.10.3. The following procedures were used:

Aligned I/O:

Random Write, Random Rewrite and Random Read

Unaligned I/O:

Random Write, Random Rewrite and Random Read

Transfersizes = 5,000, 10,000, 20,000, 40,000, 80,000, and 160,000 bytes

D. *NytroXD*

For each transfer size, NytroXD was tuned to filter the corresponding block sizes alternatively tuned off when running against the disk array.

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

Figure 3 - Effects of NytroXD on Write and Re-Write performance using aligned I/O

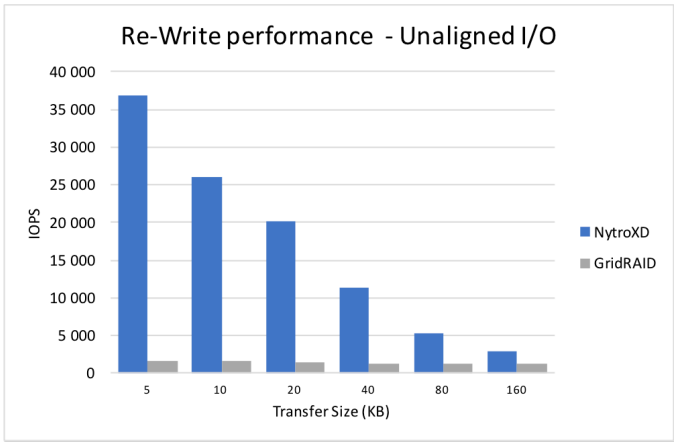

Figure 4 - Effects of NytroXD on Write and Re-Write performance using unaligned I/O

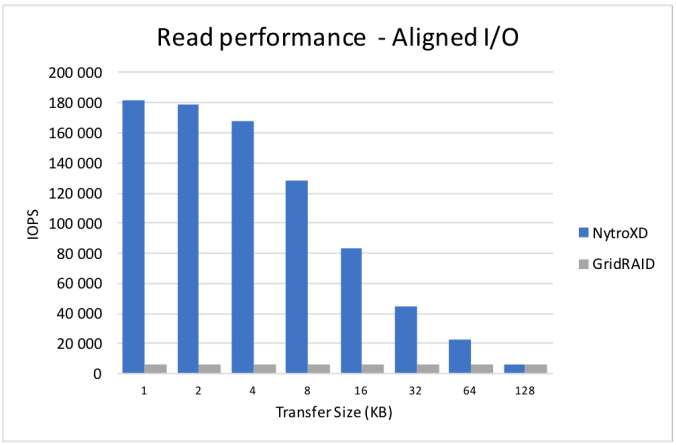

Figure 5 - Comparison of read performance from the NytroXD tier using aligned and un-aligned I/O

VII. Results

As this paper is limited in size, we've focused on synthetic benchmarks to assess the possible I/O enhancements of NytroXD. Figure 3 shows the improvement that NytroXD provides for aligned writes and aligned re-writes. The write case measures performance to newly created files. The re-write case measures the performance of overwriting already existing files. Re-write is an important measure for two reasons. One, re-write allows us to isolate the data performance because it involves less communication with the Lustre metadata server. Two, it is a common user workload just as that seen in even-odd checkpointing [15] and multi-level checkpointing [16] in which users maintain multiple checkpoint files and merely

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

overwrite the older ones. Overall, the write performance, from an IOPS point of view, is several orders of magnitude higher using NytroXD compared to the performance without Nytro as shown in the figures as GridRAID. GridRAID is the underlying parity declustered block driver in Seagate systems [17] that is still used under NytroXD across the HDDs. As expected the re-write performance is higher for NytroXD than is the write performance. Note that the results without NytroXD show little difference between write and re-write. This is because the effect of the degradation due to extra metadata messaging is not shown since the performance without NytroXD is so low.

Figure 4 shows the performance for unaligned writes and unaligned rewrites. Unaligned IO is more challenging and achieves lower performance than aligned IO due to locks imposed due to false sharing [18] and some unavoidable read-modify-writes. However, as expected NytroXD achieves several orders of magnitude improvement over the default system.

While the solution is specifically intended as a write performance accelerator, read performance is, as expected, also significantly enhanced as is shown in Figure 5. Again NytroXD shows orders of magnitude improvement and again shows its ability to benefit more so from aligned IO than does the default (as was seen earlier in writes as well).

To validate that the performance improvements were due to NytroXD small block isolation and to assess whether the IOR runs were focused on the desired block size bracket, the NytroXD I/O Analyzer (NXD I/O Histogram) was used to asses the actual blocks being delivered to NytroXD. Figure 6 is a snapshot of the of block sizes received by NytroXD from the 8 kB test. Interestingly, although the IOR parameters stated that all writes and subsequent reads should be exactly 8kB in size, there seem to be a small number of writes and reads that are 4kB and some that are larger than 8 kB. Whether this is due to the benchmark tool or the I/O analysis is unclear at this time. More generally however this does validate that NytroXD does receive and does handle correctly the small IOs.

We do have some prelimary results with real applications, HYDRA and COSMO. However, our evaluation of these results is still ongoing as both have yielded somewhat unexpected results. COSMO did not show any acceleration as a result of NytroXD but subsequent analysis of the actual I/O using the NytroXD Histogram function reveal that the authors of COSM have solved the small block I/O problem by coalescing the writes in application specific I/O libraries to enable large block I/O thereby bypassing the problem of small block I/Os. We note that NytroXD is an approach that should hopefully free computational scientists from having to modify their codes for the sake of IO performance.

HYDRA did see a beneficial increase in performance with about a 10% decrease in averall application execution time. During these tests I/O analysis revealed that almost 90% of all I/Os were larger than 1 MB. While 10% might seem small, the tests were done on a limited sized test system and was unable to fully stress the I/O subsystem to its full potential.

```
# ./nytrocli64 /xd show histogram
************************************************
     Seagate NytroXD Management Utility
     Version 3.0.10.1 (2017.02.01)
     Copyright (c) 2017 Seagate Technology LLC.
************************************************
Histogram :
==================
Num Reeds < 4K                  = 0
Num Reads 4K                    = 56796
Num Reads 4K+1 — 8K             = 456385
Num Reads 8K+1 — 16K            = 55
Num Reads 16K+1 — 32K           = 31
Num Reads 32K+1 — 64K           = 4
Num Reads 64K+1 — 128K          = 0
Num Reads 128K+1 — 256K         = 0
Num Reads 256K+1 — 512K         = 0
Num Reads 512K+1 — 1M           = 0
Num Reads 1M+1 — 2M             = 0
Num Reads 2M+1 — 4M             = 0
Num Reads 4M+1 — 8M             = 0
Num Reads 8M+1 — 16M            = 0
Num Reads 16M+1 — 32M           = 0
Num Writes < 4K                 = 0
Num Writes 4K                   = 23978
Num Writes 4K+1 — 8K            = 499318
Num Writes 8K+1 — 16K           = 19954
Num Writes 16K+1 — 32K          = 1962
Num Writes 32K+1 — 64K          = 127
Num Writes 64K+1 — 128K         = 39
Num Writes 128K+1 — 256K        = 6
Num Writes 256K+1 — 512K        = 2
Num Writes 512K+1 — 1M          = 0
Num Writes 1M+1 — 2M            = 0
Num Writes 2M+1 — 4M            = 0
Num Writes 4M+1 — 8M            = 0
Num Writes 8M+1 — 16M           = 0
Num Writes 16M+1 — 32M          = 0
```

Figure 6 - Example of an I/O Histogram during a IOR run with 8 kB block sizes.

VIII. DISCUSSION

As this paper covers the initial tests and findings with the Seagate Nytro Intelligent I/O Manager, there is not yet a lot of empirical data to analyze. The current results, albeit based on synthetic benchmarks, do however, indicate that the NytroXD filter driver and the small block, flash based partitions do work. For smaller block sizes, (4 -16 kB) the increase in performance compared to the standard parity de-clustered RAID set on the disk arrays, is 30 - 40x as is shown in Figures 3-5. Interestingly enough, using re-write (i.e. random small writes and updates to the already established file from the original write operation) is almost twice as fast for aligned I/O block sizes whereas for unaligned I/O, the benefit is around 10 - 15%. Comparing aligned and un-aligned I/O shows, as expected, that the aligned I/Os are significantly more efficient that the un-aligned ones. This pattern is similar for both reads and writes. The observed performance of the NytroXD solution is what one would expect from flash based storage systems. As the current solution utilizes 1 SSD per storage volume (Lustre OSTs), the maximal performance is gated by the individual drive performance whereas in the case of a networked attached burst buffer, most the performance is not realized as the bottleneck is moved to the network layer. Even contemporary EDR infiniband of 100 Gbit OmniPath, cannot deliver enough

performance to saturate the drives, especially not when using NVMe based infrastructures.

Comparing NytroXD to a traditional network based burst buffer is however, not easy as the over-all throughput (regardless of block size) is faster on an all flash tier compared to a disk tier. Moreover, most, if not all, of the performance seen using a separate network based flash tier is negated by the need for data movement to the HDD based storage tier, where the compute system will be competing for bandwidth to the back end storage with the data migration from the network based flash tier, the flash acceleration should be seen from the perspective of the Achilles heel of parallel file systems, namely small block I/O.

Disk based systems using a parallel file system such as Lustre® or IBM Spectrum Scale®, still provides the best price/performance and price/capacity for multi PB storage systems. And these solutions are by no means slow. Over the last 2 years, no less than 5 deployments reliably deliver performance on more than a TB/s (1,000 GB/s). While this performance is achieved by using streaming I/O rather than a mixed I/O workloads, NytroXD has the potential to augment parallel filesystems to continue to deliver superior streaming performance while also accommodating a more typical mixed I/O workload.

IX. Related Work

There are three possible architectures to use flash storage to accelerate HPC workloads. One is dedicated burst buffer nodes inserted between compute nodes and the storage system, a second is compute node local burst buffers, and a third is to do as NytroXD does and embed flash transparently within the storage system.

The academic literature is rife with burst buffer research [5, 6, 19] for dedicated burst buffer architectures. In addition, there are commercial product offerings from both Cray [9] and DDN [10]. SCR [20] and ADIOS [21] are two earlier works that harness neighbor DRAM for checkpointing and as such are examplars of compute node local burst buffers. As commercial NVRAM products, such as Intel's 3D XPoint, emerge, we suspect that burst buffer research will resume a more active exploration of this architectural choice. In contrast to these two approaches, NytroXD chooses the transparently embedded approach because Seagate wants to minimize user disruption and required modifications to user workflows.

Recently there has been some research [22] in comparing and contrasting the relative advantages and disadvantages of these three approaches in terms of failure domains, bandwidth, throughput, usability, and various other metrics. We believe this is critical research to ensure that HPC sites procure the most cost-effective solutions and urge the community to continue this research.

X. Conclusion

While the results reported here are early results and based on code that still have many tuning parameters to be explored, the initial results do clearly indicate a beneficial effect on performance. While the NytroXD archutecture is not entirely devoid of data movement, most data migration to the HDD based volumes are done asynchronously and does not impact the overall performance of the storage systems. We note that excessive use of the SSD partition could lead to bottlenecks and performance hits seen in a number of tests performed outside this study which will be an area of future exploration. While none of these tests are conclusive, they do point to the fact that proper and careful I/O profiling of each application is critical to achieve the best optimizationsof workflows. As storage solutions utilizing the NytroXD flash acceleration becomes more wide spread, we expect a clearer picture to arise.

Acknowledgment

The following people are acknowledged for their contribution to this paper: Cedric Husianycia, Bill Loewe, Prasad Balakrishnan, Scott Milk, Rex Tanakit and Eugene Birkine.

References

[1] LANL, NERSC, and SNL, "APEX Workflows," https://www.nersc.gov/assets/apex-workflows-v2.pdf, 2016.

[2] SAGE, "Data Storage for Extreme Scale: The SAGE Project Technical White Paper," http://sagestorage.eu/sites/default/files/Sage White Paper v1.0.pdf, 2016.

[3] J. Bent, G. Gibson, G. Grider, B. McClelland, P. Nowoczynski, J. Nunez, and M. Wingate, "PLFS: A Checkpoint Filesystem for Parallel Applications," in SC09, Portland, Oregon, 2009.

[4] Y. Liu, R. Gunasekaran, X. Ma, and S. S. Vazhkudai, "Automatic Identification of Applications I/O Signatures from Noisy Server-Side Traces" in 12th USENIX Conference on File and Storage Technologies (FAST '14), Santa Clara, California, 2014.

[5] N. Liu, J. Cope, P. Carns, C. Carothers, R. Ross, G. Grider, A. Crume, and C. Maltzahn, "On the role of burst buffers in leadership-class storage systems" in 28th IEEE MSST conference, 2012.

[6] G. Wang, M. Kathryn, A. Moody, W. Yu, and K. Sato, "BurstFS: A Distributed Burst Buffer File System for Scientific Applications" in SC15, Austin, Tx, 2015.

[7] J. F. Kovar, "Hybrid Flash Arrays: 13 Vendors Pushing Capacity, Performance Boundaries," *CRN*, 2015.

[8] J. Bent, S. Brad, N. DeBardeleben, S. Faibish, U. Gupta, D. Ting, and P. Tzelnic, "On the non-suitability of non-volatility" in 7th USENIX Workshop on Hot Topics in Storage and File Systems, HotStorage '15, Santa Clara, CA, 2015.

[9] Cray. "DataWarp," http://www.cray.com/datawarp.

[10] DDN. "IME," http://www.ddn.com/products/infinite-memory-engine-ime14k/.

[11] Seagate. "NIIOM," http://www.seagate.com/gb/en/about-seagate/news/clusterstor-system-with-flash-acceleration-pr-master/.

[12] Y. Liu, R. Gunasekaran, X. Ma, and S. S. Vazhkudai, "Server-side Log Data Analytics for I/O Workload Characterization and Coordination on Large Shared Storage Systems" in SuperComputing 2016, Salt Lake City, UT, 2016.

[13] K. A. Smith, and M. I. Seltzer, “File system aging—increasing the relevance of file system benchmarks” *ACM SIGMETRICS Performance Evaluation Review,* vol. 25, no. 1, 1997.

[14] J. E. Shore, “On the external storage fragmentation produced by first-fit and best-fit allocation strategies” *Communications of the ACM,* vol. 18, no. 8, pp. 4333440, 1975.

[15] L. Kumar, M. Mishra, R. Joshi, P. Ezhilchelvan, and A. Romanovsky, "Checkpointing in Distributed Computing Systems" *Concurrency in Dependable Computing*: Springer Publishing, 2002.

[16] K. Mohror, A. Moody, G. Bronevetsky, and B. R. de Supinski, “Detailed Modeling and Evaluation of a Scalable Multilevel Checkpointing System” *IEEE Transactions on Parallel and Distributed Systems*, no. 01, 2014.

[17] J. Fragalla, “Improving Lustre® OST Performance with ClusterStor GridRAID,” in HPC Advisory Council, Stanford, 2014.

[18] M. Moore, "Lockahead: Early Experience and Performance using Optimized Locking" in Cray User Group, Redmond, 2017.

[19] J. Bent, S. Faibish, J. Ahrens, G. Grider, J. Patchett, P. Tzelnic, and J. Woodring, "Jitter-free co-processing on a prototype exascale storage stack" in 28th IEEE Symposium on Massive Storage Systems and Technologies, MSST, San Diego, USA, 2012.

[20] A. Moody, G. Bronevetsky, K. Mohror, and B. R. De Supinski, "Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System" in Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, 2010.

[21] H. Abbasi, J. Lofstead, F. Zheng, K. Schwan, M. Wolf, and S. Klasky, "Extending I/O through high performance data services." pp. 1-10, in 2009 IEEE International Conference on Cluster Computing and Workshops, 2009.

[22] L. Cao, B. Settlemyer, and J. Bent, “To share or not to share: Comparing burst buffer architectures” in Spring Simulation Multi-Conference, 2017.

978-1-5386-3472-1/17/$31.00 ©2017 IEEE