The Role of SSD Block Caches in a World of Networked Burst Buffers

Torben Kling Petersen, PhD
Cray
Billdal, Sweden
tpetersen@cray.com

Bill Loewe, PhD
Cray
Pleasanton, CA, USA
bloewe@cray.com

Abstract—**The concept of burst buffers has revitalized discussions about how to improve the performance of HPC storage solutions and parallel filesystems. However, current burst buffer implementations are not capable of solving all I/O problems facing users today. As a result, HPC researchers are investigating new paradigms to accelerate the performance of HPC storage in the near-term.**

New concepts such as server-side and storage-side flash acceleration enable flash to deliver strategic performance enhancements at a reasonable cost. Cray's flash acceleration offering, the NXD storage appliance, uses an SSD cache that is transparent behind the block controller in a disk array. As SSDs are typically organized in separate storage tiers, for example in Cray's DataWarp solution, the use of additional SSDs in the HDD tier might seem redundant. The perception of SSD redundancy in the parallel filesystem persists with the introduction of features such as Lustre's Data on MDT.

In this paper, we challenge this perception by identifying workloads that uniquely benefit from the deployment of SSDs in various locations within the storage stack. We report the results of experiments that measured the performance of different I/O workloads with and without NXD. We explore the novel use of SSDs in Cray NXD to deliver the benefits of storage-side flash acceleration in a world of networked SSDs, offering a unique, performance-oriented feature set for the HPC realm. Finally, we consider planned enhancements that may significantly improve the flash acceleration concept.

Keywords-component; Flash, Burst Buffer, HPC Storage accelleration.

I. INTRODUCTION

Over the last five years, the HPC world as well as the Big Data community and the growing market of AI, have demonstrated that, within these areas, the role of storage is becoming a significant bottleneck to compute. While streaming I/O (such as check pointing and time series data) is adequately serviced by parallel filesystems running on comparably inexpensive enterprise hardware, new applications and new designs in I/O intensive data access have shown that small block, random access is not well managed by current designs for enterprise filesystems.

To address this challenge, flash-based storage is commonly recommended for small, random I/O handling. However, the largest hurdle to wider adoption today is cost, although there is hope this will change for the better in the near future.

Currently, there's also a bit of confusion about what constitutes a burst buffer. The most common design is a set of arrays, with either SAS SSDs or NVMe drives, controlled by a set of servers on the network between the compute and the parallel filesystem backend. While this layout may be practical for some I/O workloads, such as transient data or checkpointing, it suffers from the problem of excessive data movement. For most I/O operations, data needs to move to and from the burst buffer volume which, in many cases, significantly impacts network bandwidth. In addition, a networked burst buffer usually constitutes a separate namespace and mount point in order to create and maintain the tiered architecture.

The authors of this paper consider two different models of burst buffers: server-side flash acceleration (such as Cray DataWarp[1] that will not be covered in this paper) and storage-side flash acceleration (such as Cray NXD[2]). With regard to NXD, it is important to remember that while the end goal is to accelerate application file I/O, NXD acts by accelerating block I/O. Within this paper, we consider the capability of NXD to offer improved handling of small, random I/O, based on the results of recent testing.

II. AIMS OF THIS STUDY

The main topic of this study was to further categorize the ability of NXD to improve application performance and manage a mixture of large streaming I/O in conjunction with small, random I/O. Basic test results, using synthetic benchmarks, have been presented elsewhere[3]. In addition, initial tests of end user applications have also been performed.

III. Material and Methods

For the majority of reported tests, the following system setup was used. The end-user runs were performed on a similar setup in collaboration with Atos in Angiers, France.

A. Compute

- 16 nodes, dual sockets
- Intel® Xeon® CPU E5-2630 v3 @ 2.40GHz
    - 16 Cores (32 CPUs w/HT)
- 64GB Memory
- OPA (IB for the Angiers tests)
- CentOS Linux
    - release 7.2.1511 (3.10.0-327.el7.x86_64)
- Lustre 2.7.19.8 client

B. Storage solution

ClusterStor L300N

- 82x Seagate 6TB Enterprise SAS HDDs
- 2x Seagate (ST3200FM0033) SSDs
- OPA (IB for the Angiers tests)
- Lustre 2.7.19.12.x8-51
- NXD Version 3.1.0.3 (2017.12.20)

C. Software

- IOR-2.10.3[4]
- HDF5 SWMR [5]
- HYDRA
- COSMO
- CDI-PIO
- FESOM[6]

D. Experimental Setup

1) Sequential IOR

To assess the impact of NXD on high bandwidth streaming I/O, a standard IOR benchmark aimed at saturating the storage backend was run with NXD turned on and turned off. The following parameters were used:

Sequential IOR:
```
IOR -b 32g -t 1m -F -k -m -e -v -v -C -i 5 -o $OUTFILE
```

2) Mixed IOR

To understand the impact of NXD when performing a concurrent I/O schema, using both large streaming and small random I/O, a set of solo and combined runs with different load factors were performed with and without NXD. The following parameters were used:

Sequential IOR:
```
IOR -b 32g -t 1m -F -k -m -e -v -v -C -i 10 -o $OUTFILE
```
Random IOR:
```
IOR -b 280m -t 4k -F -k -m -e -v -v -C -i 10 -z -o $OUTFILE
```

3) Concurrent IOR, Increasing Random I/O

To assess how an increasing number of small random I/O threads would affect the I/O pattern, a number of concurrent IOR runs (with and without NXD) were performed using 32, 96, 160 and 224 I/O threads with the above parameters for mixed IOR.

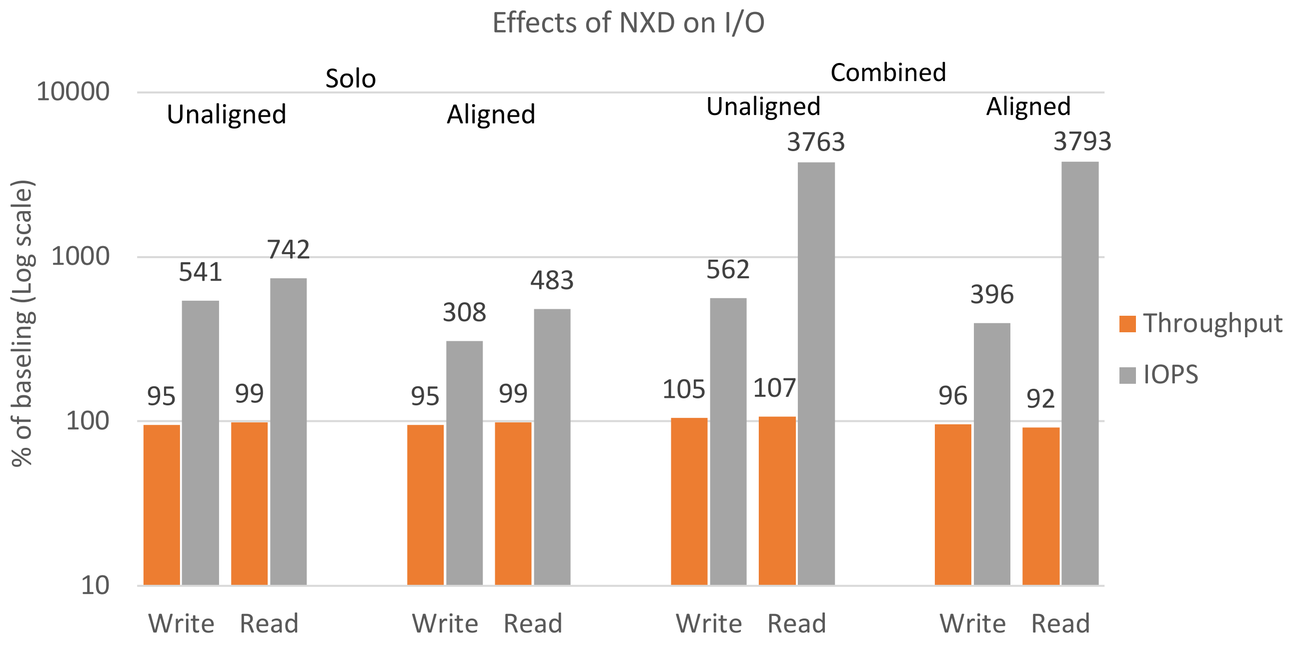


Figure 1. Mixed IOR (Experiment 2) - All results above are indicated as a percentage over baseline. Baseline is defined as the performance (throughput in the case of streaming I/O and IOPS in the case of random I/O) without NXD.

4) *SWMR*

Using SWMR in write mode, 1-16 nodes (single thread per node) were compared with and without NXD. The following parameters were used:

```
swmr write --niter 2500 --testdatafile $IN_FILE $OUT_FILE
```

IV. Results

1) *Sequential IOR.*

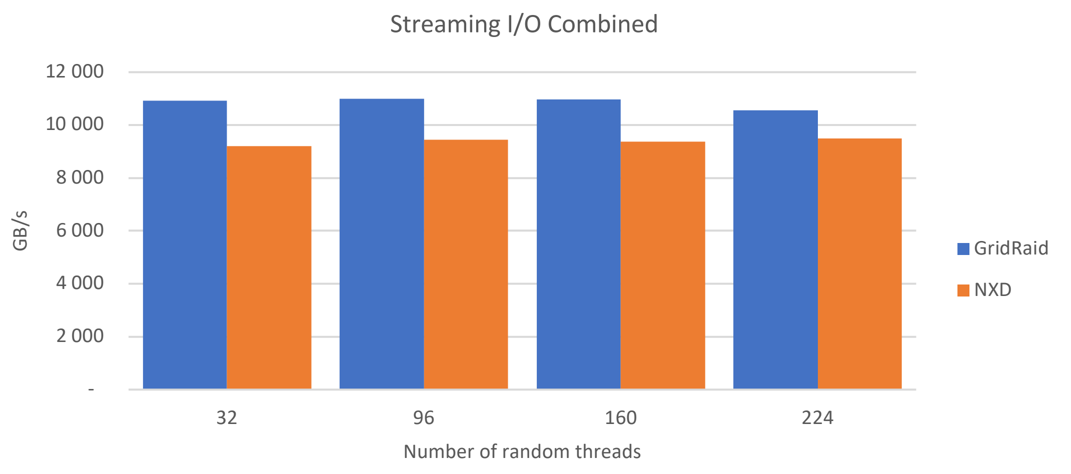
NXD	Write (MB/s)	Read (MB/s)
Disabled	13 143	11 182
Enabled	13 014	11 154

Table 1 – Streaming Write and Read performance with and without NXD.

Results for Experiments 2-4 are described in Figures 1-3.

V. Discussion and Conclusions

NXD is a transparent filter that intercepts small blocks in the I/O chain and redirects them to system SSDs, thereby increasing the performance of small I/O instead of forcing it to use a 1 MB RPC to write a few KBs. In the initial experiment (Table 1), neither the write nor read performance of streaming I/O appears to have been affected by NXD being enabled or disabled. This behavior was mirrored in the experiments using end-user applications (Table 2) when comparing runs with or without NXD enabled. However, when running two applications with different loads (medium and large size datasets), the positive effect of NXD became more pronounced as load increased.



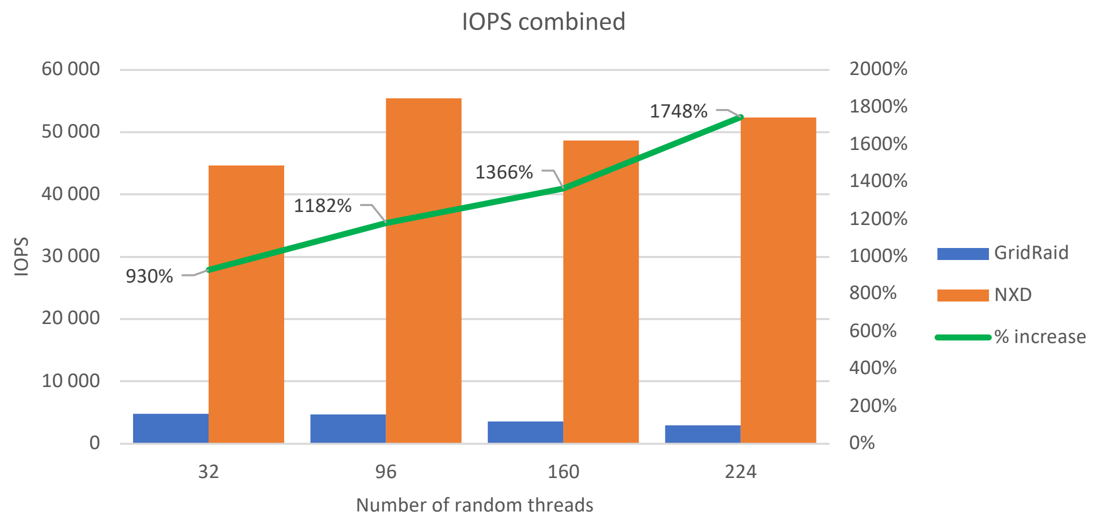


Figure 2. Concurrent IOR, increasing load of random I/O (Experiment 3) – Effects of NXD on concurrent streaming and random I/O expressed as throughput (GB/s) [top] for the former and percent increase in IOPS for the latter [bottom].
During combined sequential and random writes, overall throughput is slightly reduced but the random I/O shows orders of magnitude improvement.

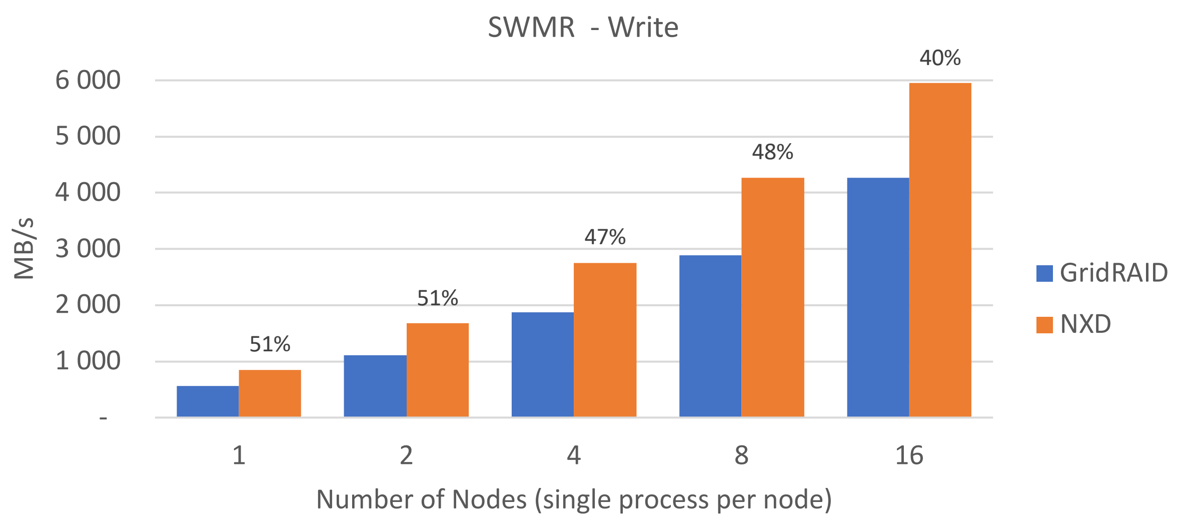


Figure 3. SWMR (Experiment 4) – Improvement of write performance (MB/s) using an increasing number of single threaded clients. The percentage increase for each group is also indicated.

In the experiments that focused on concurrent I/O, the streaming component seems to incur a slight, but significant reduction (between 1-8%). However, random, small block I/O more than makes up for the loss of streaming performance with a several order of magnitude improvement in performance (Figure 1).

When focusing on this part of the I/O spectra, the results indicate that despite the increase in small block, random I/O activity, the penalty seems stable at 10-15% while the IOPS improvement exhibits a linear increase with the number of random I/O threads (Figure 2).

Upon closer inspection at how multiple threads of small random I/O affect the overall performance of the storage system, it appears that the tradeoff might significantly benefit certain applications. While not entirely "transparent", this acceptable decrease in streaming performance allows for a far greater benefit as the frequency of small I/O and, presumably, small jobs have an even chance of completing rather than being "pushed to the side" by the dominant streaming I/O.

This effect is supported by the end-user results (Table 2). Although the data does not show orders of magnitude performance increases, but rather a modest 1-10%, it still demonstrates the positive effect of NXD. In the case of HYDRA and FESOM, the benefits of NXD appear to improve as the storage system load increases.

Application	Work load type	Large block range	Small block range	NXD impact	Overall CS Performance
Hydra	Write Intensive Large Block	4M-8M – 99.9%	<4K – 0.1%	Minimal	No Degradation
Hydra	Write intensive Large block	4M-8M – 99.9%	<4k – 0.1%	Perf Gain 2.6% to 6%	No Degradation StripeSize : 1M and the gains started showing up.
CDI-PIO (DKRZ)	Write Intensive Mixed Mode	1M – 90%	< 512 – 10%	Perf Gain of 10%	Performance Improvement with NXD Enabled / Histogram OFF and StripeSize – 1M / RPC 512/7
FESOM (DKRZ)	Read Intensive Small Block	10% large reads	90% Small Read	Perf Gain of 6%	No Degradation. No tuning, no pre-load
FESOM (DKRZ)	Read intensive Small block,	10% random large block	90% small Read (mostly seq.)	Perf gain of 10% over a run of continuous 100 iterations.	No Degradation CS performs around 10% Sample size is 9GiB
COSMO	Write intensive large blocks	1 MB	Mostly 4K	1% with stripe count 1 and stripe size 4m	No degradation over 7 runs

Table 2 – Effects of NXD on user application (using an earlier NXD version). For HYDRA and FESOM, two different tests were run with medium and large sized data sets.

One might argue that 10% is not a significant performance increase, but for sites that contend with a substantial backlog of jobs or sites in which time to completion is the predominant concern, then 10% is a notable gain, especially considering the incidental implementation cost of NXD compared to a full-fledged burst buffer. Readers should note that the user applications for which performance results are reported in this paper were tested on a much earlier codebase than the other described experiments. We are confident that re-running user application tests on a newer codebase would yield significantly better results.

Regarding the final set of experiments, SWMR is, essentially, a test to measure single threaded performance of a Lustre system. As single threaded I/O performance is Lustre's Achilles heel, the fact that NXD can increase performance by ~40-50% is very encouraging (see Figure 3). Anecdotally, we also ran an UNTAR operation followed by re-taring the data; NXD reduced the time to complete the TAR to 43% while the UNTAR operation was unaffected.

At first impression, the results presented in this study may seem random and somewhat inconsistent. However, if one looks deeper, a specific pattern emerges. The initial tests clearly demonstrate that NXD, when running large streaming I/O, is entirely transparent and does not affect sequential reads and writes. However, the results of tests that measure concurrent IOR require closer inspection. When saturating sequential I/O competes with a significant number of small random I/O operations, streaming performance is negatively impacted, but, the degree of performance loss seems to remain constant regardless of how many processes cause interference. This behavior is expected as the system now handles a much larger number of write threads and the controller's resources are somewhat limited. There does seem to be a direct relationship between the number of random, small threads and the IOPS measured in the performance tests. The main conclusion to be drawn from these results may be different from previous assumptions that NXD shields large sequential throughput from random I/O. The current tests seem to indicate that NXD managed SSD block caches, in fact, allows a greater portion of small I/O processes to complete despite the overwhelming load produced by large sequential writes. NXD appears able to preferentially "help" small random block I/O operations to be executed despite the pressure produced by large sequential I/O.

To conclude, NXD has the potential to significantly improve application execution, especially in mixed I/O environments while being considerably less expensive than a traditional burst buffer.

FUTURE WORK

NXD is under active development and a number of new features are planned. Within the current version of NXD, several tuning options are available. Further study of end-user applications is warranted to gain further insight into I/O behavior.

ACKNOWLEDGMENT

This work could not have been completed without the support of several people in our management chain including, but not limited to, Rex Tanakit and Charlie Carroll. In addition, the extended NXD team invested significant time and effort reviewing and discussing the test results, and they provided valuable insights into the finer points of small block, random I/O behavior. We wish to thank the following people for their help: Prasad Balakrishnan, Dipak Ghosh, Marc Roskow, Nathan Rutman, Wolfgang Szoecs and Vipin Verma.

Last, but not least, we wish to thank Sheila Barthel for helping correct and improve the "no nativ Engrish speeker's" many linguistical errors.

REFERENCES

[1] http://www.cray.com/datawarp

[2] Kling Petersen, T., and Balakrishnan, P.: 'Flash Acceleration of HPC Storage - Nytro Intelligent I/O Manager', (Seagate Technologies, 2017).

[3] Kling Petersen, T., and Bent, J.: 'Hybrid flash arrays for HPC storage systems: An alternative to burst buffers'. Proc. 2017 IEEE High Performance Extreme Computing Conference (HPEC) 2017.

[4] http://sourceforge.net/projects/ior-sio/

[5] https://github.com/ulrikpedersen/swmr-testapp/SWMR - Single Write Multiple Read

[6] http://fesom.de