The Search for Missing Parallel I/O Performance on the Cori Supercomputer (Extended Abstract)

Matt Bryson, Alex Sim (advisor), Suren Byna (advisor), John Wu (advisor)†
Lawrence Berkeley National Laboratory
National Energy Research Scientific Computing Center (NERSC)†
Email: mbryson@ucsc.edu {asim,sbyna,kwu†}@lbl.gov

***Abstract*—With scientific computing approaching exascale rapidly, I/O is becoming the main cause of bottlenecks in High Performance Computing (HPC). To solve this, next generation systems such as the NERSC Cori Supercomputer are equipped with I/O nodes equipped with NVRAM, otherwise known as burst buffers (BB). BB systems are designed to provide more I/O throughput than traditional Parallel File Systems (PFS) that rely on magnetic storage. This has not been the case with the Cray Burst Buffer (CBB) which is performing at 11.07% of peak performance when confronted with the popular HDF5 file library. We use the Vector Particle in Cell (VPIC) I/O kernel to benchmark the system and find potential optimization strategies. By changing the I/O access pattern of VPIC I/O kernel we are able to improve performance up to 4.6 times in some configurations.**

I. Introduction

Burst Buffers bridge the gap between memory and disk resources on modern supercomputers [1]. Burst buffers are IO nodes backed by solid state storage and provide compute nodes inside an HPC access to fast persistent storage [2]. The latest supercomputer at the National Energy Research Scientific Computing Center (NERSC) called Cori is equipped with a BB system produced by Cray [3]. The announced peak bandwidth of the burst buffer on Cori Phase 1 (Cori P1) system is 750 GB/s and the capacity is 875TB. The Cori P1 burst buffer is split into 144 burst buffer nodes that can store data in parallel with each stripe providing 6 GB/s peak performance. While the IOR benchmark in sequential mode showed 700 GB/s performance, tests from the Scientific Data Management group at Lawrence Berkeley National Lab with parallel I/O using HDF5 on 48 burst buffer nodes showed 41 GB/s I/O performance, which is 15% of the peak performance for the used 48 nodes. This arises the question, Where is the parallel I/O time being spent?

II. The Vector Particle in Cell I/O Kernel

Vector Particle in Cell is an advanced plasma physics codes that generates trillions of particles [4]. Because of the massive amount of data that VPIC generates, storing it's data without creating an I/O bottleneck is a challenge [4]. To work on I/O techniques to accelerate this code, Surendra Byna et al. created an I/O kernel that simulates that output of the full simulation [4].

The VPIC I/O kernel is a critical tool in our analysis of the Cori supercomputer for several reasons. Modifying the source code of the kernel gives us a level of control not possible with other profiling tools. Additionally, VPIC utilizes the popular HDF5 scientific file API, which is a file APIs that has performed poorly on the Cori burst buffer. We use our control of the kernel to gather various metrics on the Cori supercomputer.

III. Results

A. Exploratory VPIC-IO Runs

The initial results with the unmodified VPIC-IO kernel were important to confirm our suspicions that the Cray Burst Buffer system was not performing optimally with collective I/O. The results of these initial tests are shown in Figure 1. As we can see, both systems perform similarly on small scale tests, but as scale increases the Cori PFS drastically outperforms the DataWarp system. The best performing test on the Cori Lustre PFS was the 16K run with 32 processes per node, writing at 176.78 GB/s. This is 25.25% of the peak bandwidth of the system. Comparing a similar run on the burst buffer system shows that the system is only running at 11.07% of peak bandwidth, writing at 61.75 GB/s. This is significantly less than the IOR benchmark result, showing that the burst buffer is not handling collective I/O efficiently.

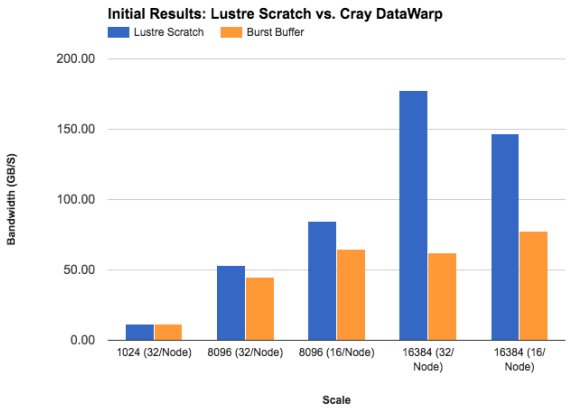

Fig. 1: Cray DataWarp v. Cori Lustre Scratch Exploratory Results

B. Aggregator Matching

We determined that when the number of BB nodes were not divisible by the number of aggregator nodes data would be written unevenly from the aggregator nodes to the burst buffer nodes, resulting in sub-optimal performance. Our matched configuration used 64 burst buffer nodes and our unmatched configuration used 65. The results, shown in Figure 2, show increased performance. Darshan logs, which provide detailed performance data per process, show higher I/O variance. The DVS counters, which list the amount of data written per process, show twice the data being written by one process. Matched burst buffer configuration can result in as high as a 86% increase in performance by decreasing this variance and spreading writes evenly.

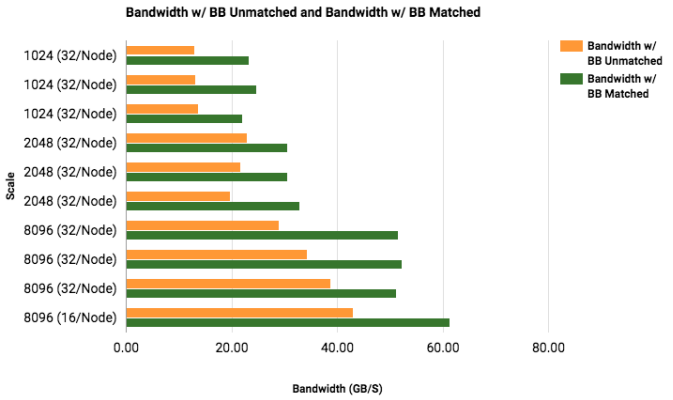

Fig. 2: Matched v. Unmatched Burst Buffer & Aggregator Nodes

C. Independent I/O

Because of solid state disks ability to absorb small I/O requests, and due to our lack of insight into DataWarp's inner workings, we choose to run tests with independent I/O. Independent I/O runs increased performance 4.6 times that of the original unmatched runs. This occurred because the DataWarp and HDF5 mechanisms that were causing the bottleneck were avoided with the independent I/O run. Additionally, having a greater number of writing processes saturated the CBB more effectively. These results are showing in Figure 3.

IV. Conclusion

With Cray DataWarp under performance is common with typical I/O optimization strategies, such as the utilization of collective I/O. By ensuring aggregator nodes were divisible by burst buffer nodes, we were able to improve performance when using collective I/O. Independent I/O performs well on this system when compared to collective I/O by avoiding the collective I/O mechanisms of DataWarp and HDF5 and saturating the BB nodes. Because of the massive improvement seen with independent I/O over collective, current applications

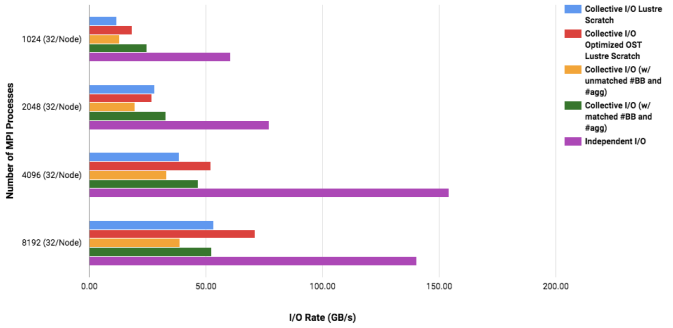

Fig. 3: Independent I/O v. Collective I/O

seeking to use any DataWarp or related solid state technology for HPC should choose to use independent I/O over collective I/O.

References

[1] 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 *Mass Storage Systems and Technologies (MSST), 2012 IEEE 28th Symposium on*. IEEE, 2012, pp. 1–11. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6232369

[2] M. Funk, "The What And Why Of Burst Buffers." [Online]. Available: http://www.theplatform.net/2015/05/19/the-what-and-why-of-burst-buffers/

[3] "Cori." [Online]. Available: http://www.nersc.gov/users/computational-systems/cori/

[4] S. Byna, J. Chou, O. Rbel, H. Karimabadi, W. S. Daughton, V. Roytershteyn, E. Bethel, M. Howison, K.-J. Hsu, K.-W. Lin, and others, "Parallel I/O, analysis, and visualization of a trillion particle simulation," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 59. [Online]. Available: http://dl.acm.org/citation.cfm?id=2389077