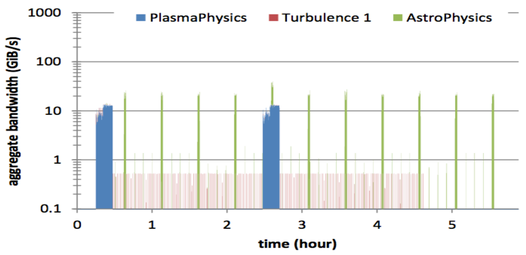
Improving VPIC I/O Performance Using Burst Buffers

Matt Bryson[1], Alex Sim[2], Suren Byna[2], K. John Wu[2]
[1]California Lutheran University, [2]Computational Research Division, Lawrence Berkeley National Lab

Introduction/Previous Work/Why NERSC need?

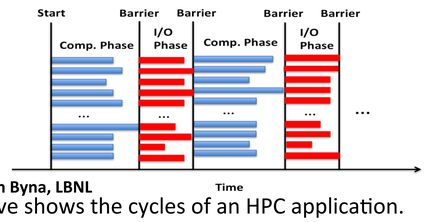
- High performance computers (HPCs) have many points of failure, so frequent check pointing is required to allow application restart
- Applications process more data and generate results that can not be stored in memory, so results must be frequently written to disk
- Parallel file systems will soon be slower than their computational (HPC) counterparts, so HPC's will spend longer than ever on I/O phases
- HPCs will have larger data sets and much more memory, so they will have more data to store to disk than ever
- Computational efficiency can be greatly improved by shortening or allowing computation to continue during I/O phases

Problem



Credit: Liu Cope, IEEE Symposium 2012

This graph shows the nature of the I/O ran on high performance computer. The large spikes are these applications writing out to disk, either to checkpoint or perform an incremental save operation.



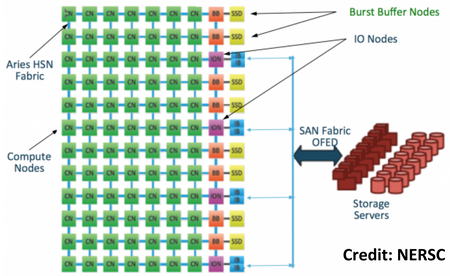
Credit: Suren Byna, LBNL [6]

The graphic above shows the cycles of an HPC application.

Previous Work

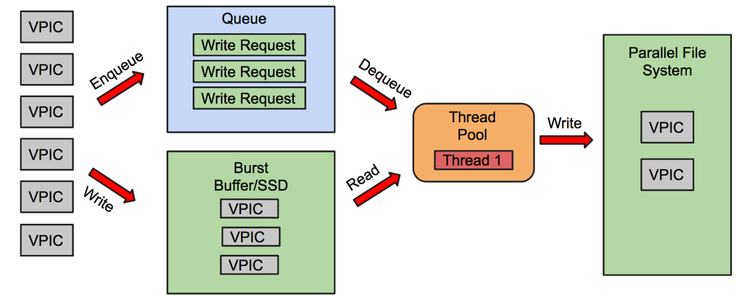
- Simulation of HPC system with burst buffers
- Experimental approaches optimizing application I/O
- Experiments with burst buffers compared against existing storage systems
- Implementation of burst buffer systems both commercially (Cray Datawarp, EMC)
- Live visualization using burst buffer co-processing
- Studies and analysis of location of solid state storage in different parts of the HPC
- Use case analysis, testing, and implementation (ie., out of core memory)

Approach

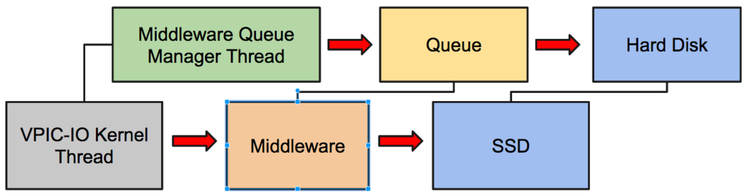


Credit: NERSC

The most common approach to this bursty I/O problem places the SSDs in the I/O nodes along with middleware, which is effectively I/O forwarding software.



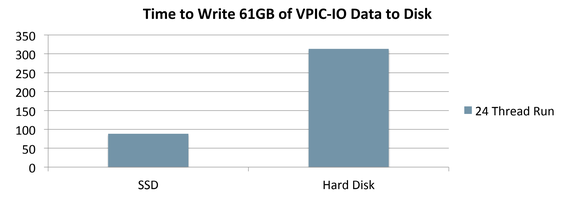
The diagram above shows the flow of data through the middleware that I designed for VPIC. Data is generated by VPIC, written to SSD, and an I/O request is put on a queue. As time is available a separate thread writes this data to it's final destination in disk. It's able to do this with chunks of the file, rather than the entire file since file access is handled through H5Part, which is an extension of HDF5.



The diagram above shows the interactions of my middleware with the I/O kernel and the thread that it spawns. The I/O kernel generates VPIC data, passes it to the middleware, which then uses it's queuing engine and H5Part to write the data to temporary, and then permanent storage asynchronously.

Experimental Setup

- Running on Alva, NERSC Edison Test Bed
- Burst Buffers are made up of 6 pairs of striped SSDs, each with approximately .9 GB/S
- Nodes have 24 cores each, 14 compute nodes available
- Tests will be ran using a single instance of the VPIC-IO kernel on 2 nodes, 4 nodes and 8 nodes
  - Between 12 & 24 cores per node will be used due to possible SLURM memory use limitations
  - One pair of striped SSDs will be used
  - Final data destination will be the global scratch parallel file system
- As show on the chart below, writing to SSD with 24 threads (or 61GB of data) took 88.16 seconds. The de-queue operation to hard disk took 313.24 seconds. Without this middleware implementation, the main thread would have had to write data for ~313 seconds, instead of ~90.
- More experimentation on larger sets is needed to prove these results.



Work in Progress

- Full experimentation set to generate graphs and data showing definitive results
- I/O thread timing and MPI integration
- Thread and I/O optimizations
- Multiple scale experimentation (larger data set, more threads generating data)
- Generalization of software for multiple applications
- Add compatibility to NERSC's XC40 Supercomputer, Cori

Acknowledgements

I would like to thank my mentors Alex Sim & Suren Byna, as well as K. John Wu & Elizabeth Bautista. Additionally, I would like to thank the TRUST REU program, Aimee Tabor, and Tiffany Reardon, at the University of California Berkeley and Lawrence Berkeley National Laboratory. This work was supported by the National Science Foundation through grant CCF-0424422 and the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Contact

Matt Bryson
Lawrence Berkeley National Lab
University of California Berkeley
Email: mbryson@lbl.gov

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, 2012, pp. 1–11.
[2]M. Romanus, R. Ross, and M. Parashar, "BB Use Case Discussion."
[3]N. Hemsoth, "Burst Buffers Flash Exascale Potential," HPC Wire. [Online]. Available: http://www.hpcwire.com/2014/05/01/burst-buffers-flash-exascale-potential/.
[4]"DataWarp Applications I/O Accelerator." Cray.
[5]S. Andersson, S. Sachs, C. Tuma, and T. Schütt, "DataWarp: First Experiences." .
[6]J. Bent, "EMC IOD BURST BUFFER API ."
[7]F. Isaila, J. Garcia, J. Carretero, R. B. Ross, and D. Kimpe, "Making the case for reforming the I/O software stack of extreme-scale systems," Preprint ANL/MCS-P5103-0314, Argonne National Laboratory, 2014.
[8]W. Zhang, "[Burst Buffer] Related Work & Discussion." .
[9]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 Mass Storage Systems and Technologies (MSST), 2012 IEEE 28th Symposium on, 2012, pp. 1–5.
[10]X. Ouyang, S. Marcarelli, and D. K. Panda, "Enhancing Checkpoint Performance with Staging IO and SSD," in 2010 International Workshop on Storage Network Architecture and Parallel I/Os (SNAPI), 2010, pp. 13–20.
[11]K. Sato, K. Mohror, A. Moody, T. Gamblin, B. R. de Supinski, N. Maruyama, and S. Matsuoka, "A User-Level InfiniBand-Based File System and Checkpoint Strategy for Burst Buffers," in 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2014, pp. 21–30.
[12]S. Byna, J. Chou, O. Rübel, 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, 2012, p. 59.
[13]M. Funk, "The What And Why Of Burst Buffers," The Platform. [Online]. Available: http://www.theplatform.net/2015/05/19/the-what-and-why-of-burst-buffers/. [Accessed: 24-Jul-2015].