The Search for Missing Parallel I/O Performance on the Cori Supercomputer

Matt Bryson[1,2,3], Suren Byna(advisor)[3], Alex Sim(advisor)[3], K. John Wu(advisor)[3]

[1]California Lutheran University, [2]Baskin School of Engineering at UCSC, [3]Computational Research Division, Lawrence Berkeley National Lab

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.

Problem

- The IOR Storage Benchmarking Tool with sequential I/O showed that the Cori Burst Buffer could achieve 700 GB/S
- When tested with the Cori Burst Buffer, VPIC, which uses a parallel I/O pattern, only performed at 15% of optimal (41 GB/S)
- There is extra time being spent on parallel I/O

Background

- More data is being generated by HPC applications, making I/O systems the bottleneck for future HPCs
- To solve this, non-volatile storage is being integrated in the HPC memory/storage hierarchy
- HPC applications currently are not optimized for non-volatile storage

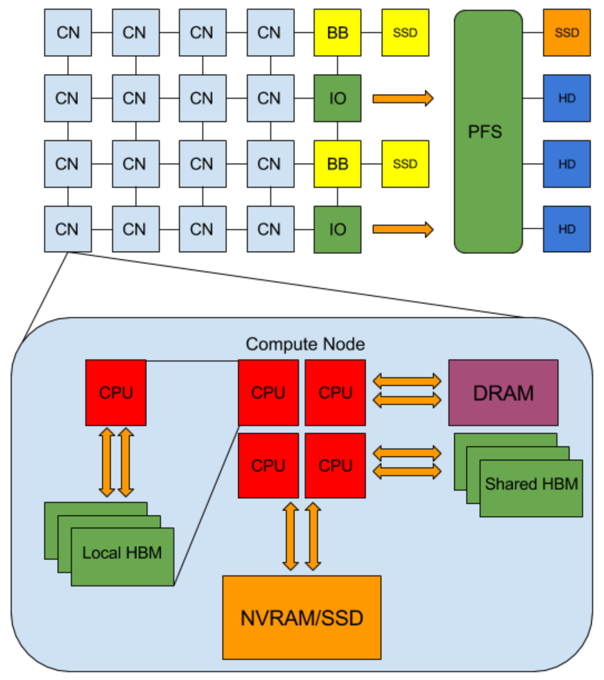

Fig. 1 The figure above is an abstract diagram of a HPC equipped with non-volatile storage in its various forms. The phase 1 Cori Supercomputer is equipped with BB nodes (in yellow) but not node local NVRAM.

Vector Particle in Cell (VPIC)

- Vector Particle in Cell (VPIC) is an advanced plasma physics code that simulates interactions of trillions of particles in space weather, such as solar flares interacting with the earth's magnetosphere.
- A recent VPIC simulation generated approximately 40 TB of data files per time step, causing the application to be I/O bound.
- VPIC utilizes MPI-IO which is a file access pattern that shows lower than expected performance on the CBB. Combined with our ability to change its I/O configuration makes it a useful tool in our search to find the lost I/O time.
- We use the VPIC I/O kernel, rather than the full code. This removes the computational step and allows us to use it for I/O profiling.
- VPIC writes out several I/O stages into an HDF5 object based file. I/O is written collectively from many threads to a single file.

Results

- Fig 2 shows the I/O performance of the VPIC I/O Kernel on the Cori Lustre Parallel File System compared to that on the CBB.
- Fig 3 shows the performance benefit of ensuring that MPI-IO aggregators match or are able to be divided evenly by BB node count.
  - When BB nodes are not divisible by MPI-IO aggregators, one aggregator needs to write more data; this slows the system.
- Fig 4 compares the primary I/O modes we used to benchmark and optimize the system.
  - •Collective I/O in all cases includes optimizations that result in 1 MPI-IO stream per node.
  - •Optimized Lustre PFS differs by uses fewer OSTs. This resulted in a more optimal result in our tests.
  - •Matched BB runs ensures I/O streams divide evenly amongst BB nodes to avoid stragglers, which results in better performance.
  - •Independent I/O writes 1 I/O stream per process, with no MPI-IO collection. This results in more variance amongst processes, but better overall performance.

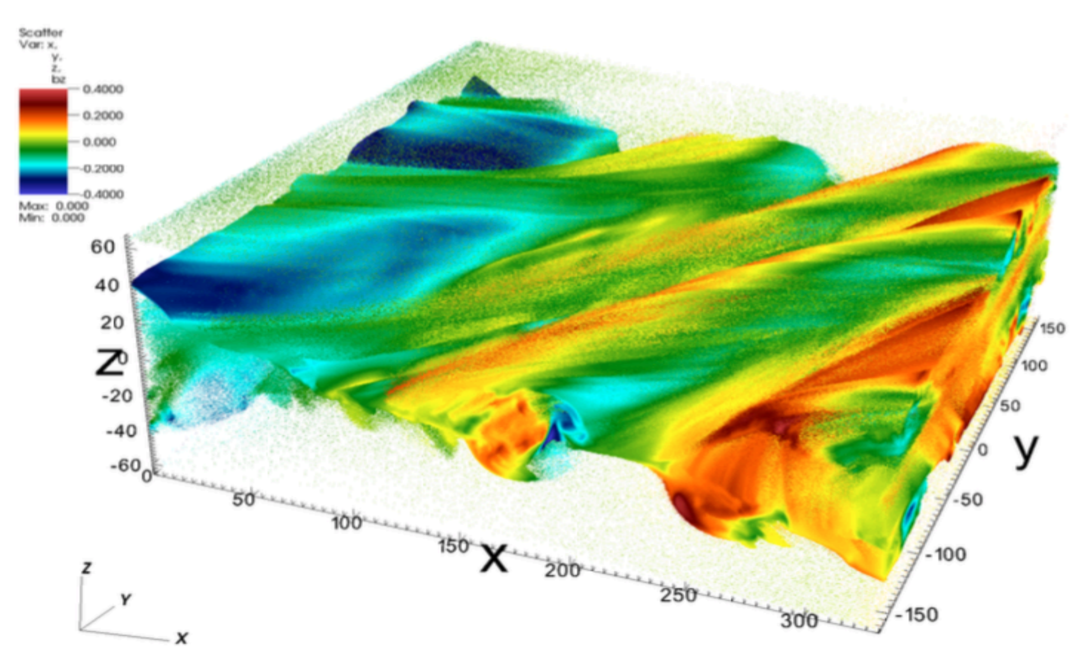

Fig 2. Mapped magnetic field data in the z dimension (bz) for all highly energetic particles (Energy>1.5) in physical space (x, y, and z dimensions). The plots use a linear color scale. The range is restricted to [0.4, 0.4] for bz. Credit: Oliver Rübel

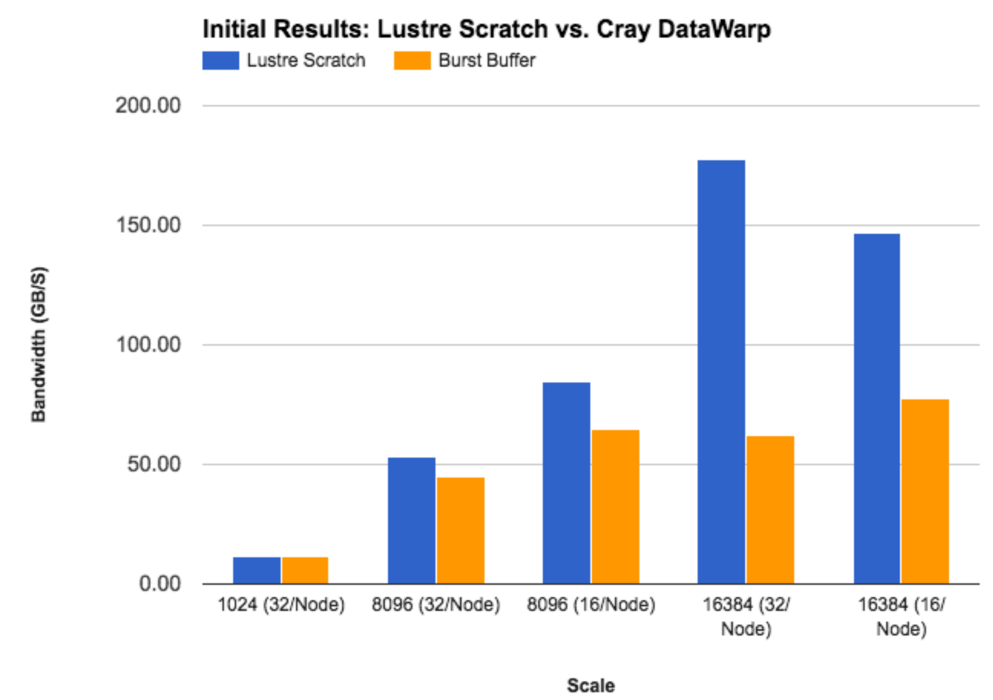

Fig 3. Comparison of I/O initial performance for the VPIC-IO kernel on the Cori PFS and Cori Burst Buffer.

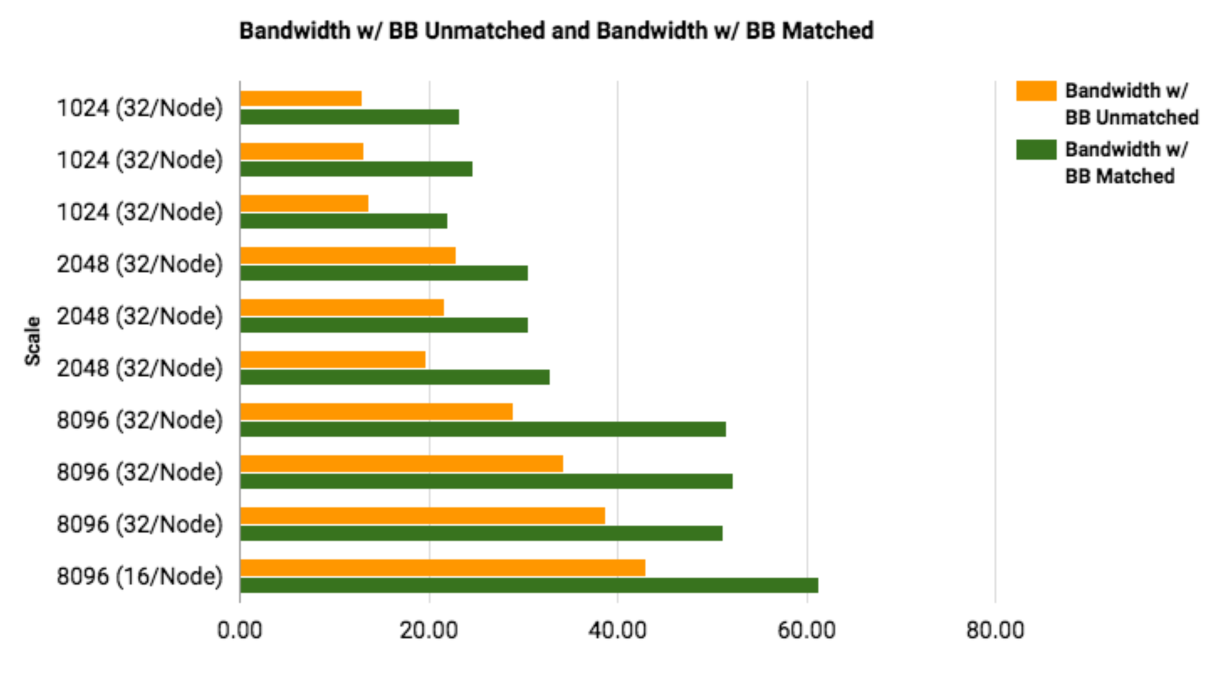

Fig 4. I/O performance (I/O rate in GB/s) on the Cori Burst Buffer with matched and unmatched numbers of BB nodes and MPI-IO aggregators

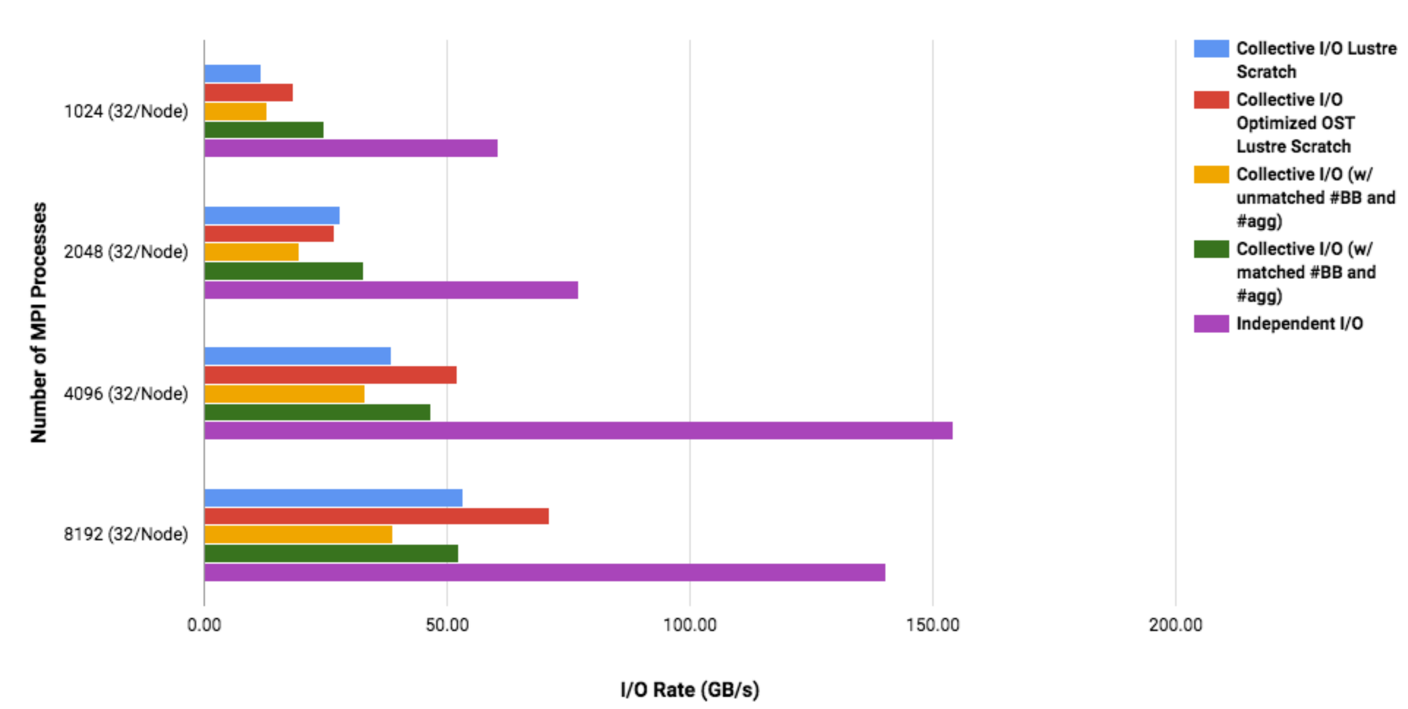

Fig 5. Performance comparison with several I/O configurations on both the Cori PFS and Cori Burst Buffer System. Blue & Red: MPI-IO collective runs on PFS. Yellow & Green: Matched & Unmatched Numbers of MPI-IO Aggregators to Burst Buffer Nodes. Purple: Independent I/O

Software & Tools

- Vector Particle in Cell I/O Kernel (VPIC)
  - •Instrumented to show performance difference with different HDF5 optimizations: Independent I/O, Collective I/O
- SLURM Scheduler
  - •Changed burst buffer configuration options to generate matched & unmatched cases
- Darshan Logs & Performance Counters
  - •Darshan logs provided individual process I/O time information. This showed variance between writing I/O threads
  - • Performance counter script showed the number of writing threads and data written

Conclusions

- Collective buffering on the Burst Buffer when using MPI-IO is a bottleneck for overall performance. Performance on the Cori burst buffer is increased when used in Independent I/O mode.
  - • Independent I/O saturates the BB more effectively than collective I/O.
- When aggregator nodes are not divisible by BB nodes, parallel I/O time is lost due to one process writing twice the amount of data. By ensuring divisibility approximately 50% performance increase can be achieved.
- The largest bottleneck with the CBB is collective (MPI-IO) operations. Because of this bottleneck in the collective buffering mode, alternative I/O access patterns, such as independent I/O, should be used to achieve better performance. Work must be done to create a collective I/O pattern that is optimal for Burst Buffer.

Acknowledgements

I would like to thank my mentors Alex Sim & Suren Byna, as well as K. John Wu, Elizabeth Bautista, Glenn Lockwood, Wahid Bhimji, and Debbie Bard for their support, feedback, and contributions. This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internship (SULI) program.
This work was also supported by the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

More Info