Recent Progress in Tuning Performance of Large-scale I/O with Parallel HDF5

M. Scot Breitenfeld[†], Kalyana Chadalavada[‡], Robert Sisneros[‡], Surendra Byna[*], Quincey Koziol[†], Neil Fortner[†], Prabhat[*], and Venkat Vishwanath[§]

[*]Lawrence Berkeley National Laboratory, USA. Email: {sbyna,prabhat}@lbl.gov
[†]The HDF Group. Email: {brtnfld,koziol,nfortne2}@hdfgroup.org
[‡]National Center for Supercomputing Applications, USA. Email: {kalyan,sisneros}@illinois.edu
[§]Argonne National Laboratory. Email: venkat@anl.gov

Large-scale scientific simulations running on hundreds of thousands of cores produce massive amounts of data that often needs to be stored in files. Analysis applications run on thousands of cores to access data files in order to extract useful information. Both, simulation and analysis codes, require high-level I/O libraries that offer superior data access performance for writing and reading data to/from parallel file systems. In this work-in-progress talk, we will present our recent work in tuning HDF5 parallel I/O library [6] that achieved 2-10× performance improvement over default configurations.

HDF5 is a versatile data model that can represent a number of complex data objects and a wide variety of metadata. HDF5 provides a software library that can run on a variety of computing systems ranging from laptops to massively parallel systems. The HDF5 file format is portable and comes with a high-level application programming interface for several high-level programming languages. Parallel I/O is a special feature of the HDF5 library that offers various I/O optimizations for parallel computers. HDF5 implements parallel I/O by exploiting features of MPI, including collective communication and I/O, along with internal algorithmic optimizations that enable high-performance application I/O. When an application requests a collective I/O operation, HDF5 generates two MPI datatypes, one that describes the application memory region and another that describes the region in the HDF5 file to access. The versatility and flexibility of the HDF5 library attracted numerous applications in reading and writing scientific data.

While HDF5 offers various optimizations and features, choosing the right combinations is necessary to obtain good I/O performance. In our recent work, we tuned two applications on different parallel systems: MOAB [1] on Mira at Argonne National Lab (ANL) and VPIC [3] on Blue Waters at the National Center for Supercomputing Applications (NCSA). Since HDF5 optimizations vary based on the underlying parallel file system and the data access patterns of applications, our study uses distinct optimization strategies for each application.

MOAB (**M**esh-**O**riented dat**AB**ase, [1], [5] is a software package for representing and evaluating mesh data. MOAB's file format for storing the mesh data structure uses HDF5 [2]. In this work, we studied the performance of a MOAB application, where each process reads data at different coordinates. It is challenging to obtain good performance when the locality of the data accesses is low. To counter this, we developed an improved hyperslab selection algorithm in HDF5 that merges multiple non-contiguous hyperslabs into one large hyperslam. This feature reduced the number of read calls by a MOAB application and achieved 10X performance over the original implementation.

Vector Particle-in-Cell (VPIC) is a highly optimized code for simulating plasma physics phenomenon. In this work, we have used the VPIC-IO kernel, that mimics the data fields of a magnetic reconnection simulation [4]. The number of particles written by the kernel scales as the number of processes increase. As the number of processes grow to hundreds of thousands, writing to the same file can be become a performance bottleneck. We studied the scalability of VPIC-IO kernel running on ≈300,000 MPI processes and writing up to 10 trillion particles that amounts to ≈290 TB shared file. We will show that proper distribution of write load among processes and among I/O servers achieve 2× better performance than default configuration. This amounts to ≈53% of the available I/O rate using 160 Lustre Object Storage Targets (OSTs) of the Blue Waters system. We will also present testing a new feature of HDF5, called multi-dataset writes, where multiple HDF5 datasets can be written to the file without a collective call between each dataset write operation. We are exploring further optimizations such as Lustre wide striping support to maintain the compute node to OST ratio reasonably.

References

[1] MOAB:Mesh-Oriented datABas. https://bitbucket.org/fathomteam/moab.
[2] Parallel Reading of an H5M File. http://trac.mcs.anl.gov/projects/ITAPS/wiki/MOAB/H5Mnotes.
[3] K. J. Bowers, B. J. Albright, L. Yin, B. Bergen, and T. J. T. Kwan. Ultrahigh performance three-dimensional electromagnetic relativistic kinetic plasma simulation. *Physics of Plasmas*, 15(5):7, 2008.
[4] S. Byna, A. Uselton, Prabhat, D. Knaak, and H. He. Trillion Particles, 120,000 cores, and 350 TBs: Lessons Learned from a Hero I/O Run on Hopper. In *Proceedings of 2013 Cray User Group*, May 2013.
[5] T. J. Tautges, R. Meyers, K. Merkley, C. Stimpson, and C. Ernst. MOAB: a mesh-oriented database. SAND2004-1592, Sandia National Laboratories, Apr. 2004. Report.
[6] The HDF Group. HDF5 user guide. http://hdf.ncsa.uiuc.edu/HDF5/doc/H5.user.html, 2010.