1. ABSTRACT

Parallel I/O is an unavoidable part of modern high-performance computing (HPC), but its system-wide dependencies mean it has eluded optimization across platforms and applications. This can introduce bottlenecks in otherwise computationally efficient code, especially as scientific computing becomes increasingly data-driven. Various studies [4] have shown that dramatic improvements are possible when the parameters are set appropriately. However, as a result of having multiple layers in the HPC I/O stack and its own optimization parameters, tuning can be a very complex problem. Additionally, optimal settings do not necessarily translate between use cases, since tuning I/O performance can be highly dependent on the individual application, the problem size, and the compute platform being used. Tunable parameters are exposed primarily at three levels in the I/O stack: the system, middleware, and high-level data-organization layers. HPC systems need a parallel file system, such as Lustre, to intelligently store data in a parallelized fashion. Middleware communication layers, such as MPI-IO, support this kind of parallel I/O and offer a variety of optimizations, such as collective buffering. Scientists and application developers often use HDF5, a high-level cross-platform I/O library that offers a hierarchical object-database representation of scientific data. One solution to this problem is using empirical optimization techniques, also called auto-tuning. Auto-tuning has been investigated as a solution to problems of this type as it is an autonomous, portable, and scalable approach [3].

In order to assign best-possible parameter sets, the autotuner must have a set of high-performing configurations for indicative test cases. To traverse the intractably large parameter search space, we chose these sets via a genetic algorithm heuristic, which we found to produce well-performing configurations after a suitably small number of test runs. Since different applications inevitably invoke different I/O patterns, in this project we propose benchmark-guided auto-tuning covering all three layers of the I/O stack. To this end, we have worked towards a benchmark framework that identifies well-performing parameter sets for a given system and problem size. I/O parameters are specified in an XML configuration file that is read by the H5Tuner library. Since the parameters are not embedded in the application, it is easy to try different settings without changing or recompiling the application. Currently, H5Tuner is implemented as a dynamic library, which is preloaded before the HDF5 and MPI libraries. It intercepts application HDF5 function calls, adjusts parameters based on the configuration file contents, and then calls the stock HDF5 functions. The framework also includes H5Evolve, built on Pyevolve [2], which uses a genetic algorithm to find well-performing parameter sets. It has a discrete set of values for each of the tunable parameters and uses crossover and mutation functions to intelligently search for a well-performing set. As the runtime of the application may not be the only output of interest, we have also developed H5PerfCapture, an extension to Darshan [1]. H5PerfCapture uses the same dynamic library approach as H5Tuner, and captures performance characteristics of the HDF5 and MPI-IO function calls: time taken to read/write data and metadata, number of bytes read/written from/to the disk by the application, etc. These values are tabulated, compressed, and written to log files as the application terminates. This information offers insights into the specifics of a particular I/O stack, and will be used to further understand and address I/O bottlenecks. To date, we have auto-tuned three application-based I/O benchmarks (VPIC-IO, GCRM-IO, and VORPAL-IO) with this framework, running on two HPC systems (Hopper at LBNL and Ranger at TACC). Speedups of 3.3X–16.7X were achieved with the auto-tuned I/O parameters in comparison to the default values. Our work shows that auto-tuning parallel I/O parameters in HDF5 applications can improve I/O performance without requiring hands-on optimization or code changes. We plan to use our extensible framework to further explore I/O performance issues and tuning opportunities, and have begun work on H5Recorder/Augmenter to automatically construct I/O application kernels from full HDF5 applications.
2. ACKNOWLEDGMENTS

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. Some architectural concepts were drawn from I/O benchmarking work with LLNL under Contract B592421. H. Luu is partially supported by NSF grant 0938064. This research used resources of the National Energy Research Scientific Computing Center and of the Texas Advanced Computing Center.

3. REFERENCES


