DOE Award #: DE-SC0004948

Name of the Recipient: The HDF Group

Project Title: Damsel: A Data Model Storage Library for Exascale Science

PI: Quincey Koziol, HDF Group

Team Members: Argonne National Laboratory, The HDF Group, North Carolina State University

Date of the report: November 26, 2014

Final Technical Report

Project Goal

The goal of this project is to enable exascale computational science applications to interact conveniently and efficiently with storage through abstractions that match their data models. The project consists of three major activities: (1) identifying major data model motifs in computational science applications and developing representative benchmarks; (2) developing a data model storage library, called Damsel, that supports these motifs, provides efficient storage data layouts, incorporates optimizations to enable exascale operation, and is tolerant to failures; and (3) productizing Damsel and working with computational scientists to encourage adoption of this library by the scientific community.

Background

Computational science applications have been described as having one of seven motifs (the "seven dwarfs"), each having a particular pattern of computation and communication. From a storage and I/O perspective, these applications can also be grouped into a number of data model motifs describing the way data is organized and accessed during simulation, analysis, and visualization. Major storage data models developed in the 1990s, such as Network Common Data Format (netCDF) and Hierarchical Data Format (HDF) projects, created support for more complex data models. Development of both netCDF and HDF5 was influenced by multi-dimensional dataset storage requirements, but their access models and formats were designed with sequential storage in mind (e.g., a POSIX I/O model). Although these and other high-level I/O libraries have had a beneficial impact on large parallel applications, they do not always attain a high percentage of peak I/O performance due to fundamental design limitations, and they do not address the full range of current and future computational science data models.

It is well recognized that a different approach, one that leverages the lessons and best practices learned from previous approaches, is needed to achieve the scalability required from high-level I/O and storage libraries to fulfill the promise of exascale systems. As the International Exascale Software Project (IESP) report observes, "The purpose of I/O by an application can be a very important source of information that can help scalable I/O performance when hundreds of thousands (to millions) of cores simultaneously access the I/O system." In other words, the high-level view of the data model is overlooked rather than exploited. Also, the data layout used in these codes and how that layout interacts with I/O software used to save the data to or read the data from storage systems are highly relevant. Arguably, the model of building "verticals" with customized interfaces and formats for the data model motifs of computational science is the next important step in enabling usable and high performance exascale I/O, and this model represents the underlying approach of our project.

Project Accomplishments

As specified in the Damsel proposal's statement of work, this project was planned as a comprehensive collaboration between some of the most experienced I/O middleware development teams in the country, at all levels of the I/O stack, from MPI-I/O (the NWU and Argonne teams), HDF5 and PnetCDF (the HDF Group and Argonne teams), and domain-specific layers such as MOAB, FLASH and GCRM (the NCSU, Argonne and NWU teams, and outside collaborators). As such, we planned out an aggressive replacement for both PnetCDF and HDF5, enhancements to MPI-I/O features, and outreach to DOE simulation teams, visualization applications and computing facilities.

Unfortunately, in attempting to achieve these lofty goals, we have fallen significantly short, mainly due to our overestimation of what we could accomplish in the time and funding limits available. We have not designed or implemented a new file format for Damsel, data indexing and querying mechanisms for the Damsel interface, conversion tools between PnetCDF/HDF5 and Damsel, fault tolerance mechanisms, self-tuning files based on access patterns, or many of the other planned objectives. We have instead created a flexible, higher-level of abstraction I/O middleware library, built on top of HDF5, explored how that library could support several DOE applications, such as FLASH, GCRM, and applications that use the MOAB library and explored ideas about how to index scientific data. As a whole, this project has laid a foundation for a follow-on project that could advance the concepts we explored, but has not created software that is useable in a production environment as currently implemented.

Technical Activities

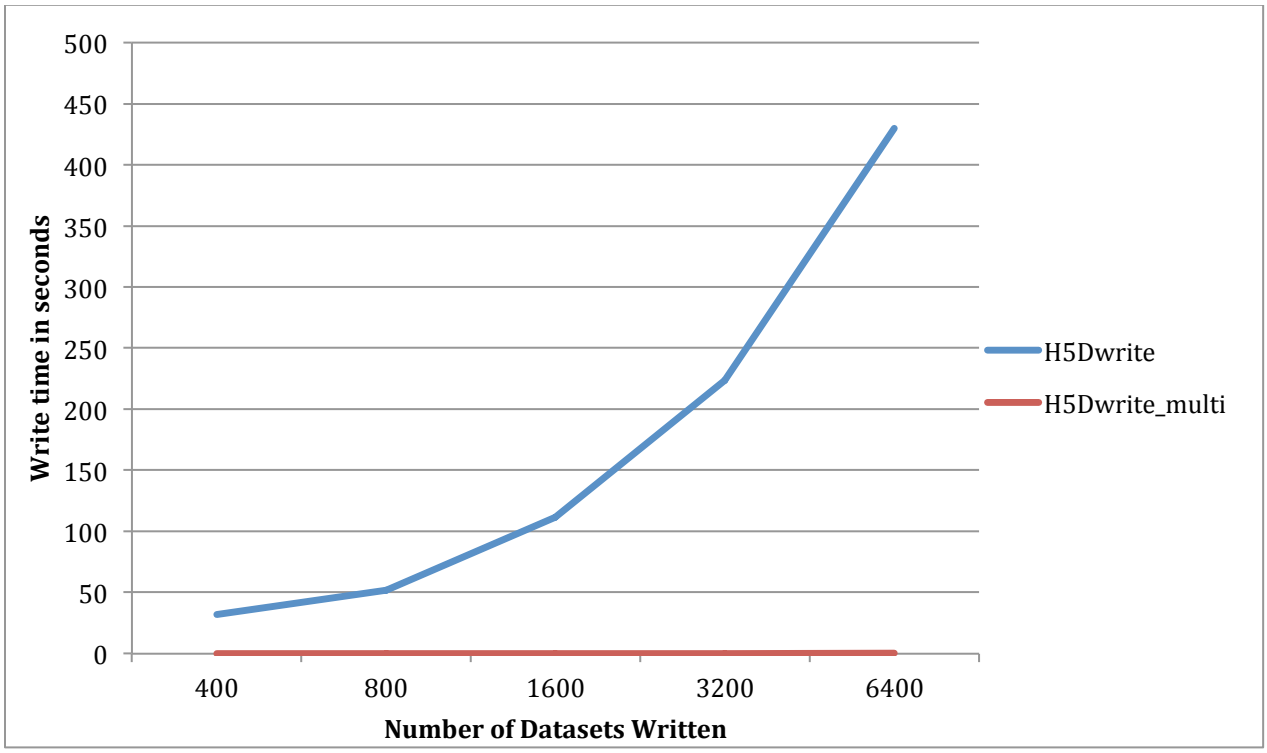
As the sole institution from the commercial realm on the Damsel project, the HDF Group team leveraged our experience in producing high-quality software for the high-performance computing (HPC) arena and concentrated our efforts on the Damsel programming model and interface design and the core implementation of the Damsel library. As implemented, Damsel depends on HDF5 for storing its information, so we also enhanced the HDF5 library with features that supported Damsel's advanced data and programming models. As the other institutions focused on applying Damsel to HPC applications such as FLASH and GCRM, we focused on implementing the underlying components, meeting in the middle with a project-wide collaboration at the data model and programming interface layers.

The Damsel data model is designed to be very flexible, allowing applications to store both structured and unstructured meshes and rectilinear array data in a flexible, hierarchical manner that includes metadata to annotate components in the model. The fundamental classes in the data model include mesh entities (such as vertices, edges, polygons and polyhedral), entity sets (which aggregate entities into larger units), and tags (a user-defined value for a given entity, such as pressure or temperature). Over the course of the project, the whole Damsel team met several times face-to-face each year and regularly via telecons to create, optimize, refine and revise the data model, with the HDF Group team participating in all discussions and decisions. A full description of the final Damsel data model can be found here: http://cucis.ece.northwestern.edu/projects/DAMSEL/damsel_datamodel.html.

The HDF Group was primarily responsible for the implementation of the Damsel library, including a collaborative effort to design the programming interface with the rest of the Damsel project teams. The Damsel library is implemented in C and uses the HDF5 library for storing data. Key features explored in the Damsel implementation are its support for both blocking and nonblocking I/O operations, hierarchical mesh specification, and support for sub-file storage. Implementation of the Damsel library was primarily performed on several Linux distributions (Redhat, Ubuntu and Debian), but the package was also tested by developers on MacOSX systems, to verify that it was portable to non-Linux environments as well. Best practices for software development were used during the development process, including integration of daily regression tests in the NSF Middleware Initiative's Metronome environment (which also enabled testing in many other operating system environments), a standardized build system, use of a source code version repository, and the usual array of wikis and mailing lists for collaboration between developers. A description of the Damsel programming API and both simple and complex examples of its use are

available here: http://cucis.ece.northwestern.edu/projects/DAMSEL/damsel_api.html and http://cucis.ece.northwestern.edu/projects/DAMSEL/#DamselUsecases.

Finally, in order to support parallel I/O for the collections of datasets produced by Damsel, the HDF Group implemented a new feature in HDF5: support for collective parallel I/O on multiple HDF5 datasets with a single operation. Adding this feature to HDF5 allowed the datasets created by Damsel to be accessed in a single read or write call, by creating a single MPI datatype and file view and a single final call to MPI_File_write_at_all. The graph below shows the results from writing multiple HDF5 datasets individually vs. a single call to the new multi-dataset write call:



As shown above, multiple calls to write data to HDF5 datasets scale exponentially, as $O(n^2)$, whereas a single call to write all datasets at once is performed in effectively constant time, as O(1). Similar results for reading multiple datasets in many vs. one calls were also observed. Not only does this feature support multiple dataset I/O for Damsel, but this functionality is also available to any HDF5 application that writes multiple datasets in parallel, benefitting many applications beyond those using Damsel directly.

Summary

The Damsel project, as currently implemented, falls short of the goals proposed. We have implemented an I/O middleware library that provides an interface for storing scientific data at a higher level of abstraction than HDF5 and PnetCDF, and explored how to use that library with several DOE applications (see collaborations listed below), but without many of the advanced research aspects proposed (such as actively reconfiguring the file's storage based on its access pattern, live querying and indexing of the data stored, etc). The Damsel library successfully delivers a working prototype of the Damsel data model, but without a significant further effort, probably with a smaller team that was more focused on just this project, future work on Damsel proper is not justified. Arguably, the most valuable work products from the project are the Damsel data model, which has had significant effort invested and would be valuable to apply to another project, and the multi-dataset I/O enhancements to HDF5. Additionally, the teams involved in the Damsel project will be able to apply the lessons learned and ideas generated to future work, which hopefully can be delivered in a more useful form for applications.

Web Access

The project web page, http://cucis.ece.northwestern.edu/projects/DAMSEL, contains a description of the Damsel library, including the programming model and C application programming interface reference manual. Several case studies are available, ranging from several fundamental data entities to real production applications. It also includes the I/O optimizations developed, example codes, and references to the studied cases.

The internal software development repository is running on a SVN server at Argonne National Lab. and its trac/wiki page is https://trac.mcs.anl.gov/projects/damsel. This platform provides an easy and secure platform for collaborative software development from parties at remote locations. Registered users will be able to access all development history and internal user discussion.

Software Release

The source code and instructions for building Damsel version 1.0.0 are publicly available from the project web page, here: http://cucis.ece.northwestern.edu/projects/DAMSEL/#DamselSourceCodeDownload.

Collaboration and Outreach

We collaborated with the Geodesic Grid I/O team led by Karen Schuchardt at Pacific Northwest National Laboratory to improve the parallel I/O performance of the Global Cloud Resolving Model (GCRM) framework. GCRM is supported by the DOE SciDAC program as one of the major climate simulation application frameworks. The geodesic grids used by GCRM cover the entire earth surface with clouds with the dimensions of longitude, latitude, and altitude. A 4km grid resolution run will contain 42M horizontal cells and generate about 0.3TB data for each snapshot, assuming 100 vertical layers and a modest number of 3D variables. A case study has been created to describe the data model and layout for geodesic grids in Damsel, including source code, illustrative figures, input data, and expected outputs, here: http://cucis.ece.northwestern.edu/projects/DAMSEL/GCRM_write.html

We collaborated with Dr. Anshu Dubey and Christopher Daley, application scientists from the ASC / Alliances Center for Astrophysical Thermonuclear Flashes at the University of Chicago on mapping the FLASH simulation code to the Damsel interface. The FLASH code is used to study the surfaces of compact stars such as neutron and white dwarf stars, and the interior of white dwarfs. FLASH uses an AMR-based domain decomposition method to partition the data and is a difficult model to map to most I/O middleware interfaces, but mapping it to Damsel's interface is straightforward. A case study has been created to describe the data model and layout for AMR grids in Damsel, available here: http://cucis.ece.northwestern.edu/projects/DAMSEL/damsel_usecase_flash_detail.html

Presentations and Publications

None, from the HDF Group, as we focused on the Damsel software implementation during the span of the project.