In situ data analysis and I/O acceleration of FLASH astrophysics simulation on leadership-class system using GLEAN

Venkatram Vishwanath[1], Mark Hereld[1], Michael E. Papka[1], Randy Hudson[2], G. Cal Jordan IV[2], and Christopher Daley[2]

[1] Math and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439
[2] FLASH Center for Computational Science, University of Chicago, Chicago, IL 60637

E-mail: venkatv@mcs.anl.gov

Abstract.

The performance mismatch between computing and I/O components of current-generation HPC systems has made I/O a critical bottleneck for scientific applications. It is therefore critical to make data movement as efficient as possible and, inorder to facilitate simulation-time data analysis and visualization to reduce the data written to storage. These issues will be of paramount importance to enabling us to glean novel insights from simulations. We present our work in GLEAN, a flexible framework for data-analysis and I/O acceleration at extreme scale. GLEAN leverages the data semantics of applications, and fully exploits the diverse system topologies and characteristics. We discuss the performance of GLEAN for simulation-time analysis and I/O with FLASH astrophysics simulations at scale on leadership-class systems.

1. Introduction

Today's largest computational systems are providing unprecedented opportunities to advance science in numerous fields, such as climate sciences, biosciences, astrophysics, computational chemistry, high-energy physics, materials sciences, and nuclear physics. Current Department of Energy leadership-class machines such as the IBM Blue Gene/P (BG/P) supercomputer at the Argonne National Laboratory, and the Cray XT system at the Oak Ridge National Laboratory consist of a few hundred thousand processing elements. In the case of FLASH [3], a multiphysics multiscale simulation code with a wide international user base, the Intrepid BG/P supercomputer at the Argonne Leadership Computing Facility (ALCF) is enabling scientists to better model, validate, and verify various phenomena in fields including thermonuclear-powered supernovae and high-energy density physics.

While the computational power of supercomputers keeps increasing with every generation, the I/O systems have not kept pace, resulting in a significant performance bottleneck. The *ExaScale Software Study: Software Challenges in Extreme Scale Systems* puts it this way: "Not all existing applications will scale to terascale, petascale, or on to exascale given current application/architecture characteristics" citing "I/O bandwidth" as one of the issues [5]. The *International Exascale Software Project Roadmap* notes that "management, analysis, mining, and knowledge discovery from data sets of this scale is a very challenging problem, yet a critical one in Petascale systems and would be even more so for Exascale systems" [1].

Simulation-time data analysis and visualization have been widely recognized as key components in future systems to reduce the data being written to storage, as well as provide faster insight. Two approaches have been proposed: *in situ analysis*, where the analysis runs on the same resource as the simulation, and *co-analysis*, where the data is moved to a dedicated coupled resource over a high-speed network for analysis. Proposed solutions in visualization toolkits [4] [6] require modification to the simulation code. Additionally, a hybrid approach is extremely attractive for applications, wherein coarse-grained analysis is performed in situ to identify regions of interest that are moved out to the co-analysis resource for fine-grained analysis.

In this paper, we describe GLEAN, a flexible and extensible framework that takes application, analysis, and system characteristics into account to facilitate simulation-time data analysis and I/O acceleration. The GLEAN infrastructure hides significant details from the end user, while at the same time providing a flexible interface to the fastest path for their data and analysis needs and, in the end, enabling scientific insight. It provides an infrastructure for accelerating I/O, interfacing to running simulations for co-analysis, and/or an interface for in situ analysis with zero or minimal modifications to the existing application code base. We present our work on simulation-time analysis and I/O acceleration for FLASH simulations on ALCF systems using GLEAN.

The remainder of the paper is organized as follows. We present a brief overview of GLEAN in Section 2. In Section 3, we evaluate the effectiveness of GLEAN with FLASH for I/O and analysis at scale. We present our conclusions in Section 4.

2. GLEAN Architecture

In designing GLEAN we are motivated to improve the performance of applications that are impeded by their own demanding I/O and analysis requirements. We strive to provide these with zero or as little overhead as possible to the system.

Figure 1 shows an overview of GLEAN. Traditionally, a simulation uses I/O libraries to write their data out to storage. This data is later postprocessed. In GLEAN, the simulation running on a compute resource may invoke GLEAN directly or transparently through I/O libraries such as Parallel NetCDF and HDF or via the GLEAN API. Custom analyses including fractal dimension (FDIM) and Lagrangian coherent structures can be applied in situ on the compute resource. We note that as we leverage the data models and structures of the application, in situ analysis does not need any memory copy. Additionally, one could move the simulation data out to the analysis nodes for staging, where they can be transformed, reduced, analyzed using tools including ParaView, and written to storage. GLEAN is implemented in C++ leveraging MPI and threads and provides interfaces for Fortran and C-based parallel applications. It provides a flexible and extensible API that can be customized to meet the needs of the application. We describe our GLEAN design in terms of three principal concerns: network topology, data semantics, and asynchronous data staging.

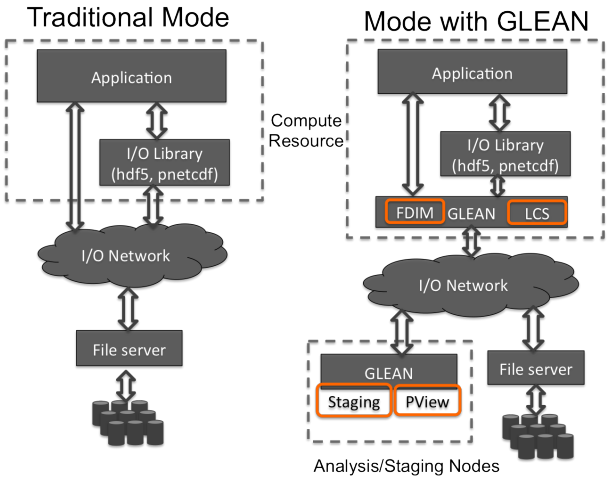


Figure 1. Relationships between GLEAN and principal components of an HPC application.

Network topology: As we move to systems with heterogeneous and complex network topologies, effective ways to fully exploit their heterogeneity are critical. BG/P has five networks with varying throughputs and topologies. An important goal in GLEAN is to leverage the topologies to move the data out of

the machine as soon as possible, thus enabling the simulation to continue its computation. On BG/P, MPI collective I/O uses the BG/P collective network; in GLEAN, we leverage both the 3D torus network and the collective network for moving the data out of the machine. Additionally, we leverage the topology to reduce the synchronization overheads of collective operations - critical in scaling to large core counts.

Application data semantics: A key design goal in GLEAN is to make application data semantics a first-class citizen instead of just viewing data as buffers and/or files. This enables one to apply various analytics to the simulation data at run-time to reduce the data volume written to storage, transform data on-the-fly to meet the needs of analysis, and enable various I/O optimizations that leverage the application's data models. In GLEAN, we have codesigned a FLASH AMR class that captures the in-memory data schema used by FLASH simulation and associated data structures to manage it. Additionally, we have mapped relevant pNetCDF and HDF5 APIs to relevant GLEAN APIs, thus enabling us to seamlessly interface with FLASH simulations using these libraires.

Asynchronous data staging: Asynchronous data staging blocks the simulation only for the duration of copying data from the compute nodes to the dedicated staging nodes. The data staging serves as a burst buffer for the simulation I/O that can be written out asynchronously while the computation proceeds. A key distinguishing characteristic of GLEAN's data staging is that it leverages the data models and semantics of applications for staging instead of viewing data as files and/or buffers. On the staging nodes, GLEAN runs as an MPI job and communicates with the compute nodes. On BG/P, this communication is over sockets—the only way to communicate between BG/P compute nodes (CNs) and the external I/O network. Further, each staging node is designed with a thread pool wherein each thread handles multiple connections via a poll-based event multiplexing mechanism. The data semantics enables GLEAN to transform the data on the fly to various I/O formats.

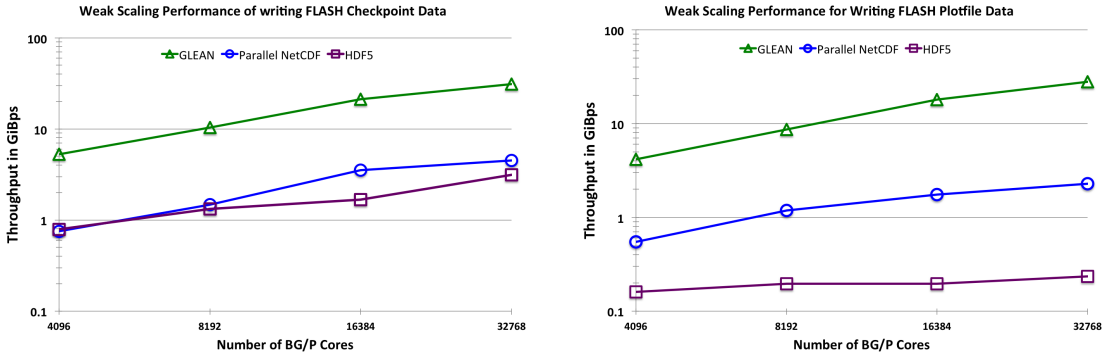
3. Evaluation with FLASH

FLASH [3] is an adaptive mesh, parallel hydrodynamics code developed to simulate astrophysical thermonuclear flashes, such as Type Ia supernovae, Type I x-ray bursts, and classical novae. It solves the compressible Euler equations on a block-structured adaptive mesh. FLASH provides an adaptive mesh refinement (AMR) grid using a modified version of the PARAMESH package and a uniform grid to store Eulerian data. We worked closely with FLASH developers to capture the AMR hierarchy and data semantics of FLASH in GLEAN. Additionally, by embedding GLEAN in the pNetCDF and HDF5 I/O libraries used by FLASH, we are able to interface with FLASH without modifying the simulation code. In Section 3.1, we discuss the performance of GLEAN to accelerate the I/O of FLASH, and next discuss in situ analysis of FLASH using GLEAN to compute the fractal dimension. These evaluations were performed on the ALCF infrastructure consisting on the Intrepid BG/P supercomputer, Eureka data analysis cluster and the file servers all interconnected via a five-stage Myrinet Clos network.

3.1. FLASH I/O Acceleration

For our study, we used the Sedov simulation included in the FLASH distribution. Sedov evolves a blast wave from a delta-function initial pressure perturbation [2]. The Sedov problem exercises the infrastructure (AMR and I/O) of FLASH with minimal use of physics solvers. It can, therefore, produce representative I/O behavior of FLASH without spending too much time in computations. The application is configured to have 16^3 cells per PARAMESH block. We choose to advance the solution over four time steps and produce I/O output at every single step so that I/O dominates the application runtime. The I/O in each step consists of checkpoint files for restart purposes and plot files for analysis. A plot file is a user-selected subset of mesh variables stored in single precision. In these experiments a checkpoint file contains all ten mesh variables in double precision, and a plot file contains three selected variables in single precision.

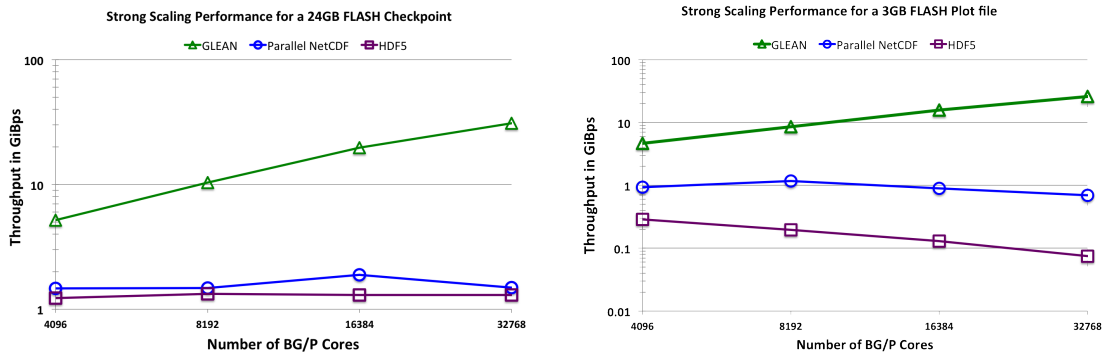
We compare the performance of pNetCDF, HDF5, and GLEAN for writing out the checkpoint and plot data of FLASH. We configured FLASH to use both pNetCDF and HDF5 with collective I/O. On the staging nodes (Eureka), GLEAN can be configured to write out data asynchronously using pnetcdf or to transform the data on the fly to HDF5 and write it out. This strategy is possible because GLEAN captures the data semantics of FLASH. As we scaled from 4,096 cores to 32,768 cores in powers of two, the number of Eureka nodes used for staging data in GLEAN was 12, 24, 36, and 72, respectively.



(a) (b)

Figure 2. Weak scaling results for FLASH.

In our weak scaling study, each run wrote five checkpoints and six plotfiles. For 4,096 cores, each checkpoint file is 12 GiB; and for 32,768 cores, it is 96 GiB. From Figure 2(a), we see that at 32,768 cores the maximum throughput achieved by HDF5 is 3.13 GiBps and pNetCDF is 4.4 GiBps. GLEAN is able to sustain an observed throughput of 31.3 GiBps—a 10-fold improvement over HDF5 and a 7-fold improvement over pNetCDF. For plot file data, at 4,096 cores, each file is 1.8 GiB; and at 32,768 cores, it is 16 GiB. In plot files, the data written per process is smaller than the checkpoint data. At 32,768 cores, GLEAN is able to sustain an observed throughput of 27.2 GiBps—a 117-fold improvement over HDF5 and a 12-fold improvement over pNetCDF. At these scales, the efficacy of GLEAN becomes even more evident.



(a) (b)

Figure 3. Strong scaling results for FLASH.

Figure 3 depicts the strong scaling performance for writing out 3.4 GiB plot files and 24 GiB checkpoint files as we scale the number of processes from 4,096 to 32,768 processes. For plot files, at 4,096 processes, GLEAN achieves a 5-fold improvement over pNetCDF and 16-fold improvement over HDF5. At 32,768, GLEAN achieves a 37-fold improvement over pNetCDF and a 347-fold improvement over HDF5 and sustains an average throughput of 26 GiBps. With checkpoint files, GLEAN is able to sustain an observed throughput of 31 GiBps.

Thus, effectively exploiting the network topology of BG/P, leveraging the data semantics of the applications, and staging asynchronous data are critical as we scale to larger core counts. These approaches will be of increasing importance as we move to exascale architectures with larger core counts and smaller per core memory footprints.

3.2. In Situ Fractal Dimension Analysis for FLASH

In FLASH, fractal dimension helps illustrate the degree of turbulence in a particular time step, as well as identify the variation of turbulence across sub regions in the domain. We compared the time spent in fractal dimension computation of FLASH on 2,048 BG/P cores, using box-counting for five variables at the highest refinement level for three isovalues using a parallel postprocessing analysis with pNetCDF on 20 analysis nodes and in situ analysis with GLEAN. cwThe results showed that in situ analysis yields a 14-fold improvement over the parallel postprocessing tool (0.8 seconds compared with 11.6 seconds). With postprocessing, the time to read the necessary data from storage dominates the analysis time. Additionally, since GLEAN is embedded in the pNetCDF layer, the in situ analysis requires no modification to the FLASH simulation code.

4. Conclusions

Simulation-time data analysis and data staging is critical to mitigate the storage bottlenecks faced by applications and to generate insight expeditiously. The GLEAN infrastructure hides significant details from the end users, while at the same time providing them with a flexible interface to the fastest path for their data and analysis needs and, in the end, enabling scientific insight. It leverages the diverse network topologies of the system and data semantics of applications. GLEAN provides an infrastructure for accelerating I/O, interfacing to running simulations for co-analysis, and/or an interface for in situ analysis with zero or minimal modifications to the existing application code base. Nonintrusive integration is achieved by seamlessly embedding GLEAN in higher-level I/O libraries such as pNetCDF and HDF5.

For FLASH simulations on leadership-class systems, we successfully demonstrated both weak scaling and strong scaling for I/O using GLEAN. We achieved up to 31 GiBps observed I/O throughput for FLASH—multifold improvement over pNetCDF and HDF5. Additionally, we demonstrated in situ fractal dimension computation for FLASH and obtained a 14-fold improvement over a postprocessing approach. We believe this is a significant step toward scaling the performance of applications on current large-scale systems and will provide insight for the design of I/O and analysis architectures for exascale.

Acknowledgments

We gratefully acknowledge the use of the resources of the Argonne Leadership Computing Facility at Argonne National Laboratory. This work was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357 and an Argonne National Laboratory Director's Postdoctoral Fellowship. This work has also been supported in part by the DOE-supported ASC / Alliance Center for Astrophysical Thermonuclear Flashes at the University of Chicago under contract B523820.

References

[1] J. Dongarra, P. Beckman, and T. Moore. International exascale software project roadmap. Technical report, DOE and NSF, November 2009.

[2] FLASH user guide. `http://flash.uchicago.edu/website/`.

[3] B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, and H. Tufo. FLASH: An adaptive mesh hydrodynamics code for modelling astrophysical thermonuclear flashes. *Astrophysical Journal Supplement*, 131:273–334, 2000.

[4] Kitware. ParaView. `http://www.paraview.org/`.

[5] V. Sarkar. Exascale software study: Software challenges in extreme scale systems. Technical report, DARPA, September 2009.

[6] VisIt. `http://wci.llnl.gov/codes/visit/`.