

Toward simulation-time data analysis and I/O acceleration on leadership-class systems

Venkatram Vishwanath*

Mark Hereld†

Michael E. Papka‡

Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439

ABSTRACT

The performance mismatch between computing and I/O components of current-generation HPC systems has made I/O the critical bottleneck for scientific applications. It is therefore critical to make data movement as efficient as possible, and, to facilitate simulation-time data analysis and visualization to reduce the data written to storage. These 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 acceleration with 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 [6], a multiphysics, multiscale simulation code with a wide international user base, the Intrepid BG/P supercomputer at 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 [10]. 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." [4].

Simulation-time data analysis and visualization has been widely recognized as a key component in future systems to reduce the data

being written to storage as well as provide faster insight. There have been two proposed ways to achieve this: *in situ* 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. These two approaches are not mutually exclusive and may be used in combination as appropriate. Each presents special advantages and has limitations. *In situ* analysis uses simulation data structures while they are in memory and so obviates the need to move or store the data. On the other hand, *in situ* analysis must be laid out in advance, excluding the opportunity for data exploration before it is written into an analysis result. Furthermore, memory-intensive analysis operations such as temporal analysis that require multiple time steps of the simulation to be stored for computation are made impractical by the already high memory pressure of simulations. *Co-analysis* can provide extremely flexible machinery that enables analysis of temporal variations and other memory-intensive operations while offloading this work from the simulation (similar to how GPUs are used now). This flexibility comes at the expense of additional hardware resources, namely the analysis nodes and related infrastructure. Additionally, one could employ a hybrid approach 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 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. Non-intrusive integration is achieved by seamlessly embedding GLEAN in higher-level I/O libraries such as `pnetcdf` and `hdf5`.

The novel contributions of our paper include the following:

- A flexible infrastructure for simulation-time analysis and I/O with non-intrusive integration with applications;
- Demonstration of simulation-time analysis at scale with applications; and,
- Demonstration at scale with applications to yield multi-fold improvement over their current I/O mechanism.

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 applications at scale. Section 4 covers related work to provide the reader with context for the reported results and design choices. Finally, we present our conclusions in Section 5.

*e-mail:venkatv@mcs.anl.gov

†e-mail:mhereld@mcs.anl.gov

‡e-mail:papka@mcs.anl.gov

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 post-processed. In GLEAN, the simulation running on a compute resource may invoke GLEAN directly or transparently through I/O libraries such as parallel-netcdf and hdf5, or via the GLEAN API. Custom analyses including Fractal Dimension (FDIM) and Lagrangian Coherent Structures (LCS) can be applied in situ on the compute resource. We would like to 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, 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.

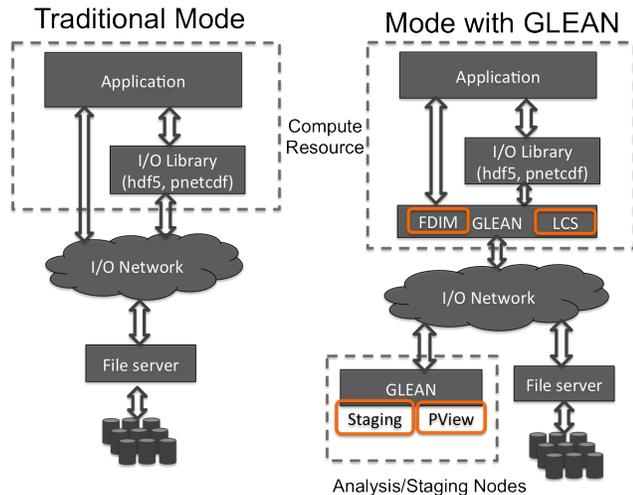


Figure 1: Relationships between GLEAN and principal components of a HPC application.

2.1 Exploiting network topology for I/O

As we move towards systems with heterogenous and complex network topologies, effective ways to fully exploit their heterogeneity is critical. BG/P has different networks with varying throughputs and topologies. BG/Q is expected to have a more complex network topology. 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 on with its computation.

On BG/P, MPI collective I/O uses the BG/P collective network, while in GLEAN, we leverage both the 3D Torus network as well as 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. Figure 2 shows GLEAN's selection of 8 aggregator groups formed within 64 BG/P nodes (pset). In determining the importance of selection of aggregator group partitions, we found that even for a single *pset*, there is at least a factor of three in performance from the worst case we tested to the configuration that

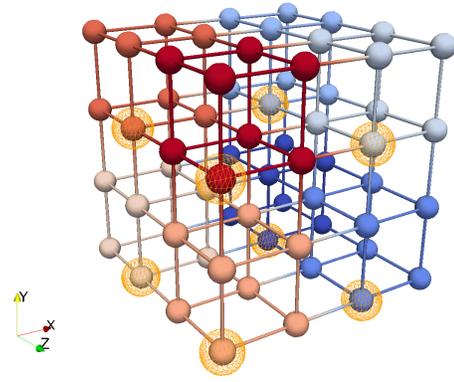


Figure 2: Aggregation groups formed within a pset(64 Compute Nodes) by GLEAN with aggregator nodes highlighted

GLEAN uses. In each aggregator group, the node where the aggregation is performed is chosen such that the aggregator nodes are distributed across the collective network in a *pset*.

2.2 Leveraging 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. Toward this effort, we have worked closely with FLASH [6], an astrophysics application, to capture its adaptive mesh refinement (AMR) data model. We have interfaced with PHASTA [9], which uses an adaptive unstructured mesh, to add unstructured grid support to GLEAN. Our initial targets have been chosen to cover a wide gamut of data models used in simulations. Figure 3 depicts the AMR data model used by FLASH. In GLEAN, working closely with FLASH developers, we have co-designed a FLASH AMR class that captures the in-memory data schema used by FLASH simulation and associated data structures used 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 pnetcdf and hdf5.

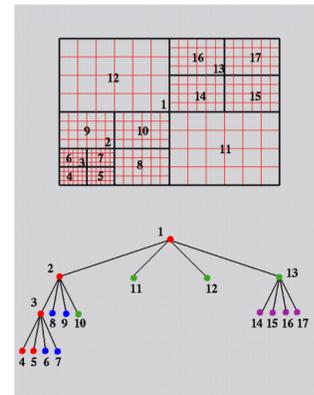


Figure 3: FLASH AMR

2.3 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 ahead. 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. In case of 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 APPLICATIONS

We have integrated GLEAN with several applications and report its performance at leadership class scale with realistic data patterns. The applications selected span a wide-gamut of data models, and enabled us to experiment with several integration schemes. In the following we describe our methods and results for GLEAN as used by two applications: FLASH and PHASTA.

3.1 FLASH I/O Acceleration

FLASH [6] 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 PARA-MESH package and a Uniform Grid (UG) 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. 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 5-stage Myrinet CLOS network.

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 [5]. 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 PARA-MESH 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 data out asynchronously using either pnetcdf, or transformed on-the-fly to hdf5 and written out. This 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.

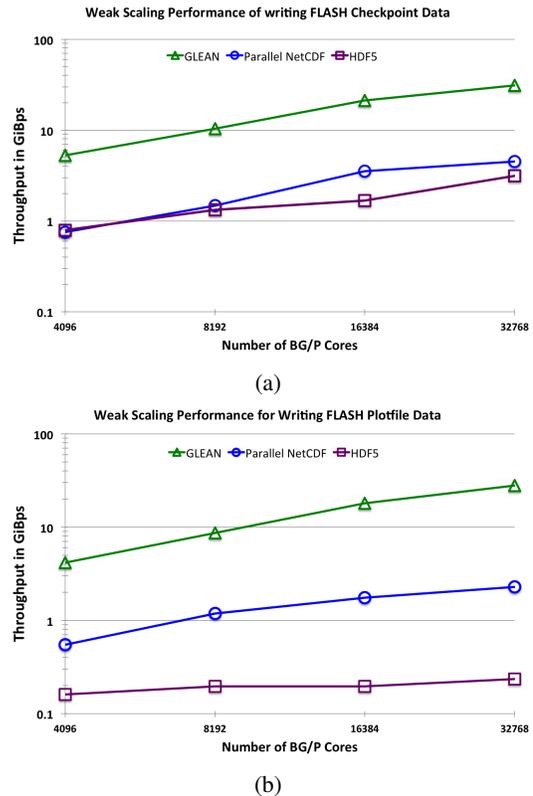
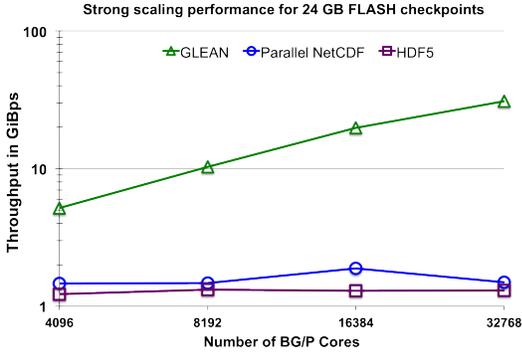


Figure 4: Weak scaling results for FLASH.

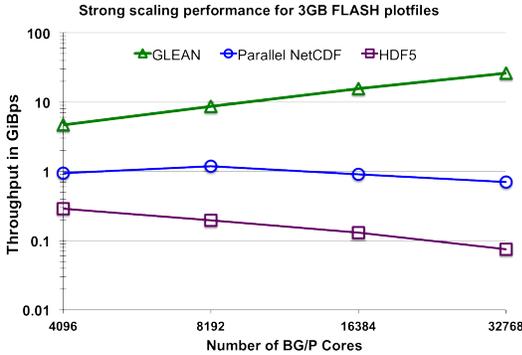
In our weak scaling study, each run wrote five checkpoints and six plotfiles. For 4,096 cores, each checkpoint file is 12 GiB; and for 32,768 cores, it is 96 GiB. From Figure 4(a), we see that at 32,768 cores the maximum throughput achieved by hdf5 is 3.13 GiBps and pnetcdf is 4.4 GiBps. GLEAN is able 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 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.

Figure 5 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 4096 to 32,768 processes. For plot files, at 4096 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. In case of 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 asynchronous data staging are critical as we scale towards larger core counts. These approaches will be of increasing importance as we move towards future exascale architectures with larger core counts and smaller per-core memory footprints.



(a)



(b)

Figure 5: Strong scaling results for FLASH.

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. The fractal dimension computation is designed with MPI and uses several optimizations including nearest neighbor communication. Table 1 compares the time spent in fractal dimension computation of FLASH using box-counting for five variables at the highest refinement level for three iso-values using a parallel post-processing analysis using pnetcdf on 20 analysis nodes, and in situ using GLEAN on 2,048 BG/P cores. From table 1, we see that in situ yields a 14-fold improvement over the parallel post-processing tool. In case of post-processing, the time to read the necessary data from storage dominates the analysis time. Additionally, as GLEAN is embedded in the pnetcdf layer, the in situ analysis requires no modification to the FLASH simulation code. We plan to build a suite of in situ analyses in GLEAN for FLASH. This would also help identify the regions of interest that need further fine-grained analysis which can be then be moved over to the staging/analysis nodes.

Table 1: Fractal Dimension Analysis of FLASH

	Post Processing	in situ with GLEAN
Analysis time in secs	11.6	0.8

3.3 Co-visualization of PHASTA

The PHASTA code [9] performs computational fluid dynamics (CFD) using a finite element discretization using a LES-based turbulence model. PHASTA uses an adaptive, unstructured tetrahedral grid. The number and locality of elements changes frequently, based on solution characteristics and load balancing. Grid and field

structures are stored in dynamically-allocated memory, due to frequent adaptivity updates.

We describe our success with GLEAN at enabling simulation-time analysis and visualization of PHASTA running on 160K cores of Intrepid (40 racks - entire machine) with ParaView [7] running on 100 Eureka nodes. To achieve this we worked closely with PHASTA developers to capture its unstructured tetrahedral mesh data model. Additionally, to facilitate visualization of the PHASTA data using ParaView, we added support for ParaView’s visualization meshes. On Eureka, GLEAN was able to transform on-the-fly PHASTA’s staged data into ParaView’s mesh format. ParaView is one of the highly used visualization toolkits on leadership class systems. Simulations using GLEAN would be able to visualize data at run-time using ParaView.

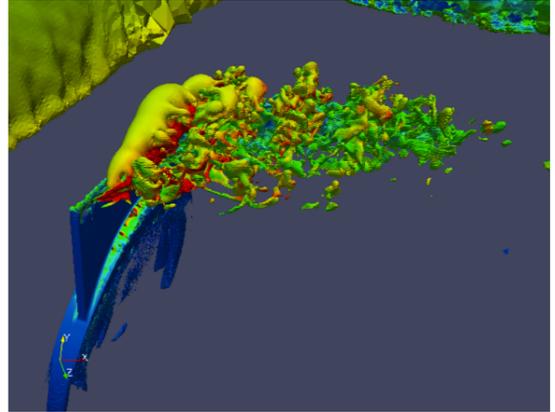


Figure 6: Co-visualization of PHASTA

Figure 6 depicts a simulation-time co-visualization of PHASTA. For a 3.3 billion element case (100GiB), GLEAN was able to transfer the data from Intrepid to Eureka and sustain 48GiBps. This enabled one to apply multiple ParaView filters to achieve simulation-time visualization of PHASTA. This is critical for understanding the health of a simulation as well as visual debugging its state. GLEAN enabled PHASTA developers to visualize live a 160K cores simulation.

4 RELATED WORK

The ADIOS high-level I/O library [8, 2] has demonstrated high-performance I/O for applications on large-scale systems. To use ADIOS, applications need to be modified via a light-weight API. Additionally, ADIOS writes data out into a custom BP format. The library provides several post-processing tools to convert the BP format into formats including HDF5 and netCDF. As the I/O bottleneck becomes even more critical in future, this conversion time could become significant. In GLEAN, we support the commonly used I/O libraries and do not need any offline data conversion. ADIOS supports asynchronous data staging via Decoupled and Asynchronous Remote Transfers (DART) [3] and DataStager [1]. DART is an asynchronous communication and data transport substrate for large-scale parallel applications. It uses one-sided communication mechanisms including RDMA for extracting and transporting data. To the best of our knowledge these mechanism supports multi-dimensional arrays. In GLEAN we strive to capture a range of data models including AMR and Adaptive unstructured grids.

Over the years there have been many visualization systems built to run in tandem with simulation, often on supercomputing resources. Recent examples include a visualization and delivery system for hurricane prediction simulations and a completely in-

egrated meshing-to-visualization system for earthquake simulation [11]. These systems are typically lightweight and specialized to run a specific type of visualization under the given simulation framework. Our approach differs in that we link the codes and run on the simulation nodes, directly accessing the simulation data structures in memory without any changes to the simulation. Proposed solutions in visualization toolkits [7] [12] require modification to the simulations code and we strive to provide non-intrusive integration as well as an infrastructure where one could use the aforementioned toolkits for analysis. Additionally, our goal is to provide a flexible infrastructure combining both in situ as well as co-analysis.

5 CONCLUSIONS

Simulation-time data analysis and data staging is critical to mitigate the storage bottlenecks faced by applications and to generate insight expeditiously. GLEAN is a flexible and extensible framework taking 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 them with a flexible interface to the fastest path for their data and analysis needs, and in the end scientific insight.

Table 2: Modes accomplished with GLEAN

Infrastructure	Application	Mode
Co-analysis	PHASTA	Visualization with ParaView
Data Staging	FLASH, S3D	I/O Acceleration
In Situ	FLASH	Fractal Dimensions
In Flight	MADBench2	Histograms

In GLEAN, we fully leverage the diverse network topologies of the system and data semantics of applications. We have successfully demonstrated (Table 2) in situ analysis, co-analysis, data staging and in flight analysis for a number of the applications at scale (up to 160K cores) on leadership class systems and achieved up to 48 GiBps for data movement. 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 systems.

ACKNOWLEDGEMENTS

We gratefully acknowledge the use of the resources of the Argonne Leadership Computing Facility at Argonne National Laboratory. This work was supported 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 Postdoctoral Fellowship. We are grateful to Rob Latham, Rob Ross and Phil Carns for the application benchmarks and related discussions. We acknowledge Joseph Insley, Kamil Iskra, Goerge Jordan, Randy Hudson, Chris Daley, Pat Marion, Berk Geveci, Ken Jansen, Michel Rasquin, Bill Allcock, Phil Carns, Tisha Stacey, Ray Loy, Kevin Harms, Susan Coghlan, Andrew Cherry, and the ALCF team for discussions and help related to the paper.

REFERENCES

[1] H. Abbasi, M. Wolf, G. Eisenhauer, S. Klasky, K. Schwan, and F. Zheng. Datastager: Scalable data staging services for petascale applications. In *HPDC*, pages 39–48, 2009.

[2] ADIOS. <http://www.nccs.gov/user-support/center-projects/adios/>.

[3] C. Docan, M. Parashar, and S. Klasky. Enabling high speed asynchronous data extraction and transfer using DART. *Proceedings of the*

17th International Symposium on High-Performance Distributed Computing (HPDC). IEEE Computer Society Press, Boston, MA, 2008.

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

[5] FLASH user guide. <http://flash.uchicago.edu/website/>.

[6] 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.

[7] Kitware. ParaView. <http://www.paraview.org/>.

[8] J. Lofstead, F. Zheng, S. Klasky, and K. Schwan. Adaptable, metadata rich io methods for portable high performance io. In *In Proceedings of IPDPS'09, May 25-29, Rome, Italy, 2009*.

[9] O. Sahni, M. Zhou, M. Shephard, and K. Jansen. Scalable Implicit Finite Element Solver for Massively Parallel Processing With Demonstration to 160K Cores. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–12, Portland, Oregon, 2009. ACM.

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

[11] T. Tu, H. Yu, J. Bielak, O. Ghattas, J. C. López, K.-L. Ma, D. R. O’Hallaron, L. Ramírez-Guzmán, N. Stone, R. Taborda-Rios, and J. Urbanic. Analytics challenge - remote runtime steering of integrated terascale simulation and visualization. In *SC*, page 297, 2006.

[12] Visit. Visit. <http://wci.llnl.gov/codes/visit/>.

The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory (“Argonne”). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.