SAND2017-12459C

Faodail: Enabling In Situ Analytics for Next-Generation Systems

Craig Ulmer, Shyamali Mukherjee, Gary Templet
Sandia National Laboratories
Livermore, California
cdulmer|smukher|gjtempl@sandia.gov

Scott Levy, Jay Lofstead, Patrick Widener
Sandia National Laboratories
Albuquerque, New Mexico
sllevy|gflofst|pwidene@sandia.gov

Todd Kordenbrock
DXC Technology
thkorde@sandia.gov

Margaret Lawson*
Sandia National Laboratories
Albuquerque, New Mexico
mlawso@sandia.gov

ABSTRACT

Existing approaches for in situ analysis and visualization (ISAV) assume that scientific simulations are written in a bulk synchronous parallel (BSP) execution model. However, because of the projected increase in heterogeneity on future systems, alternative execution models, e.g., asynchronous many-task (AMT), have been proposed. In an AMT environment, application data is no longer fixed to a particular location which makes in situ processing more difficult. One solution to this problem is to enhance the data management services used by AMT runtimes to make their data accessible to both AMT applications and non-AMT ISAV tools. Decoupling these data management services from the AMT runtime provides an opportunity to support ISAV tools that can interact with either AMT or BSP applications.

In this paper, we introduce a new data management layer called *Faodail*[1] that can support the diverse needs of multiple communities on modern platforms. While Faodail is designed to serve as a native data management service for Sandia's DARMA AMT framework, it also provides a flexible means of integrating applications with ISAV tools. We explore this idea with an example that connects ISAV tools to a particle-in-cell plasma simulation.

KEYWORDS

Asynchronous Many-Task, Data Staging, in situ, analytics, visualization

ACM Reference Format:
Craig Ulmer, Shyamali Mukherjee, Gary Templet, Scott Levy, Jay Lofstead, Patrick Widener, Todd Kordenbrock, and Margaret Lawson. 2017. Faodail: Enabling In Situ Analytics for Next-Generation Systems. In *Proceedings of In Situ Infrastructures for Enabling Extreme-scale Analysis and Visualization workshop, Denver, Colorado USA, November 2017 (ISAV 2017),* 5 pages.
https://doi.org/10.475/123_4

*Also with Dartmouth College.
[1]"Lucky Find" in Scots-Gaelic

ISAV 2017, November 2017, Denver, Colorado USA

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

1 INTRODUCTION

Traditionally, the integration point between scientific simulations and analysis/visualization routines has been the parallel storage system. With data sizes growing faster than storage bandwidth and the increasing availability of nonvolatile memory (NVM) within the compute area, in situ techniques are becoming both more necessary and popular. Existing bulk synchronous parallel (BSP) in situ tools (*cf.* [5, 23] for [15, 21] respectively) eliminate the need for the parallel file system to stage data between steps, but must share a programming model to be integrated closely.

While asynchronous many-task (AMT) frameworks are not currently widely deployed, they are an important investigation area for future computing systems. The BSP model broadly assumes that the performance of computational resources is uniform and time-invariant. When these assumptions do not hold, the burden of managing heterogeneity falls to the application programmer. However, current projections suggest [4] that resource heterogeneity (e.g., GPUs, manycore processors) and performance variability (e.g., due to power caps or thermal throttling) will increase significantly on next-generation systems. As a result, alternative execution models, e.g. AMTs, are growing in importance. AMT runtimes (e.g., Charm++ [11], Legion [6], Uintah [10]) are designed to explicitly manage the complexities that arise due to variations in computational performance. A key characteristic of AMTs is their dynamic task scheduling and data placement based on processing dependencies. While this addresses performance variability, it eliminates guarantees about data locality requiring tightly integrated in situ analytics to also be rewritten and incorporated into the task graph, *cf.* [19].

Online analytics requires a data management approach that works both for the hosting simulation as well as the analytics routines themselves. Existing approaches to online analytics include: pausing the simulation to run analytics on the data in place, making a local copy of the data and running the analytics along side the simulation, and staging data into separate nodes for off-node, but still online (in situ, for a broad defintion of the term) analysis. In each of these cases, the simulation and the analytics have to agree on a processing increment and data placement to work together. Faodail offers general data management facilities that support in situ analytics for both BSP and AMT programming models. Unlike a data management service dedicated to an AMT framework, Faodail has been generalized to also support BSP programming models.

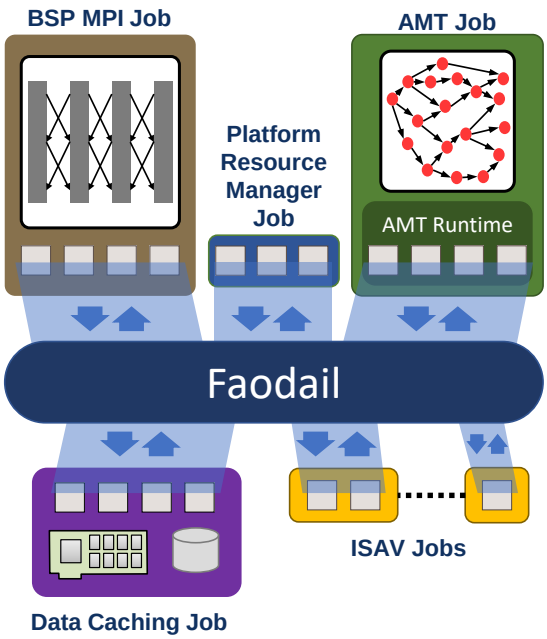

Figure 1: High-level diagram showing the relationship of Faodail to an AMT application and *in situ* analysis and visualization. The large colored shapes encompass the resources used by each entity. The small squares represent computational resources (e.g., compute nodes, processors, or threads). The cylinders represent storage resources.

This frees AMT applications from having to use ported analytics frameworks by integrating at the data management level.

In the remainder of this paper, we provide a discussion of the operating and design challenges addressed by Faodail in (§2) and a case study in (§3). A short evaluation is presented next (§4). We then examine related work (§5) and conclude (§6).

2 FAODAIL

In response to the need for a more flexible means of connecting applications, ISAV tools, and other runtime services, we are developing a collection of data management services called Faodail. These services provide performant, job-to-job communication and are architected to fit the needs of multiple communities. This section describes the application environment emerging in HPC to motivate key design principles needed for a flexible data management service. A discussion of implementation details provides information about how our initial prototype meets these goals.

2.1 Application Environment Requirements

The design of Faodail is largely driven based on requirements derived from the operating environment of today's HPC platforms. As depicted in Figure 1, Faodail provides a means of connecting several different jobs that run concurrently on a platform. First, BSP or AMT parallel simulation jobs run and produce data objects that are either stored in internal resources or published to other resources. While coupling and workflow scenarios may use Faodail to pass data between components, this more complex use case has not been fully explored yet. Second, Faodail may use distributed memory and NVM to absorb bursts of data from the application or replay results from simulations to requestors. Third, ISAV tools use Faodail to retrieve and analyze data. Finally, coordination applications provide a means of helping the different jobs in the environment locate and connect with the resources of other jobs.

An examination of the application environment motivated three fundamental requirements for Faodail design. First, Faodail must provide basic primitives for users to reason about and decompose their datasets, but at the same time the API must be as agnostic as possible about how developers manage their data. Rather than force users to design algorithms around a data store's indexing and migration policies, it is better to provide mechanisms for users to express how the system should manage their data. By offering mechanisms to control data epoch visibility and a simple key/blob interface, Faodail offers an approach that can serve many kinds of clients. Second, to aid scalability, separating application fates (i.e., the simulation from the analytics) also offers independent scalability through loose coupling. This requires using a communication layer that offers efficient data transfers between jobs while at the same time not breaking the communication libraries used within jobs (e.g., MPI). As such, Faodail cannot simply rely on sockets or splitting an MPI communicator and must instead use a low-level Remote DMA (RDMA) communication layer. This layer is an evolution of the long proven NNTI layer from the Nessie RPC library [17]. Finally, Faodail must provide a way of migrating data objects from memory to higher-capacity resources, such as burst buffers or the parallel filesystem (PFS). This requirement implies Faodail must transition in-memory objects to systems with vendor-proprietary or file-based APIs.

2.2 Design and Operation

Faodail provides a *key/blob* abstraction to facilitate flexible data exchange between different executables (e.g., simulation application and applications for visualization and analysis). A *key* is a programmer-defined text string that allows the programmer to attach semantic significance to the associated data, a *blob*. Although a key attaches programmer-cognizable meaning (and possibly structure) to a blob, Faodail is entirely ignorant of any meaning attached to a key or its associated blob. An example key might encode the application name, run number, iteration number, variable name, and some information about what part of that globally distributed array this blob represents. Separate processes can exchange data via Faodail by exchanging key information. Key exchange can be explicit or implicit (i.e., keys can be constructed in a well-known way).

A simple example of how Faodail might be used to facilitate in situ analysis is depicted in Figure 1. This diagram depicts a composed workload comprised of a scientific simulation application within an AMT runtime, an in situ analysis and visualization code, and Faodail. Three resource sets are required. First are the application (simulation) resources. Second are the analysis/visualization resources. Third are Faodail resources that are used for both the application and analysis routines. In simple cases, all three may be

from a single job allocation. Long-term, we recommend that they all have separate lifetimes to better support decoupling resilience domains, scalability, and runtimes. The AMT runtime uses Faodail to facilitate migration of application variables as tasks are scheduled for execution. Faodail also enables the in situ analysis code to efficiently collect a subset of these variables for analysis.

2.3 Managing Data for ISAV, AMT, and BSP

The AMT data management approach offers new integration opportunities that can support both ISAV and BSP applications. The key thing to think about with AMT and BSP applications is that while the processing model is different, both decompose data in a similar, if not identical, way. For example, a large 3-D array is split across many processes (tasks) and computation cannot proceed on an element unless all of the dependent elements are at an equal simulation progression. For BSP applications, this is the whole data set at once. For AMT, each task has dependencies that determine when processing can occur. This allows some overlap of processing the next timestep during the end of processing the previous timestep.

Since the data decomposition is essentially the same, Faodail can store data from both an AMT and a BSP application with only the BSP application changing to write to Faodail rather than using an IO library like HDF-5 [1] or ADIOS [14]. Ideally, a new transport layer, such as an ADIOS transport or an HDF-5 Virtual Object Layer plug-in could be provided eliminating the need to change any application code. With an interface focusing on the data description from a metadata standpoint, such as a hyperslab, it is possible for the ISAV to query into Faodail and extract whatever data it needs.

2.4 A Different Approach

Typical data management solutions for in situ analytics rely on just processing local data or leveraging knowledge of neighbors or other data locality to access necessary data. Offline techniques common for BSP applications typically use file stored in the parallel file system and only activate the analytics when there is confidence that the data set has been written completely. Given system buffering, this may be as long as waiting for the next output to start or mistreating the parallel file system by constantly executing 'ls -l' commands waiting for a file to stablize at a particular size (typically this requires querying each storage target that has part of the file's data to see how much data it currently has for each invocation).

By using a key/blob approach using descriptive, predictable keys, we can bring the direct access typical for a file-based approach to an in-compute-area solution required for an in situ approach. This decouples the data management from the AMT (or BSP) application and the analytics and instead focuses on data set availability. A transactional technique like D^2T [12, 13] can offer scalable transactions to control consistency and visibility.

3 CASE STUDY: SIMPLEPIC

Particle in Cell (PIC) methods are a class of well-established computational techniques for simulating plasmas. Plasmas are comprised of charged particles in gaseous form interacting with each other and the surrounding environment. PIC methods simulate plasmas by computing: 1) *electromagnetic (EM) fields* created by charged particles in the plasma, 2) *chemical reactions* of plasma particles with the environment, and 3) *motion of particles* due to forces exerted by the EM fields. This last step generates currents and charges which further drive the EM field. Sandia is developing a new, PIC code named EMPIRE that is being architected to scale to next-generation computing platforms. While EMPIRE is currently implemented as a BSP-style MPI code, developers are evaluating whether better load balancing can be achieved through a DARMA-based AMT implementation. In order to allow researchers to better explore AMT tradeoffs, the EMPIRE developers have constructed a reference application called SimplePIC that isolates the particle *move* phase. This phase is especially appealing for an AMT environment because the move routinely causes imbalances in the way the data is distributed across compute nodes.

Whether in a BSP or AMT setting, PIC simulations generate a substantial amount of data that can make analysis challenging for two reasons. First, the volume of data managed by a simulation is large enough that writing to disk and performing post processing is infeasible. This characteristic drives the need for ISAV tools that can summarize current conditions in a way that is meaninful for users. Second, efficient PIC applications periodically redistribute their datasets to achieve better load balancing as particles disperse over time in the simulation. While this load balancing is performed manually in the BSP case and automatically in the AMT case, the end result is that downstream ISAV applications need a mechanism for locating and retrieving particles. Faodail performs this function in both cases. Examples of ISAV applications include tasks such as quantifying how many particles are close to regions of interest in the mesh, identifying how many particles exceed a threshold velocity, and rendering images to help developers verify the simulation is modeling an environment correctly.

In order to provide a portable way for connecting PIC and ISAV applications through Faodail, we have constructed a *Particle DIM (Data Interface Module)* that is usable by both producers and consumers of PIC data. From the producer perspective, an application task periodically generates a *patch* of particle data that is injected into Faodail through the task's Particle DIM. The Particle DIM serializes the patch data into one or more key/blob pairs that are then published into Faodail's resources. The unique key names for patch data are also published as metadata in Faodail to provide a way for downstream applications to locate data. From the consumer's perspective, an application uses the Particle DIM to query metadata and retrieve relevant particle patches.

From the programmer's perspective, there are multiple advantages to using a DIM to interface with Faodail. First, a DIM establishes a contract between producers and consumers about how data is exchanged, but does not dictate how those transfers are implemented. This property allows DIM developers to write multiple implementations that decompose datasets in different ways if needed. Second, a DIM enables simulations to be decoupled from ISAV applications while retaining the ability to leave data objects in place if needed. The Particle DIM can be configured to store data objects in the application or to other distributed resources that are part of Faodail. Finally, a DIM provides a mechanism for implementing indexing that's right for the application. While the current Particle DIM performs basic indexing to locate items, it can easily be extended to allow consumers to make advanced queries. Our approach in this work is to have data producers compute statistics

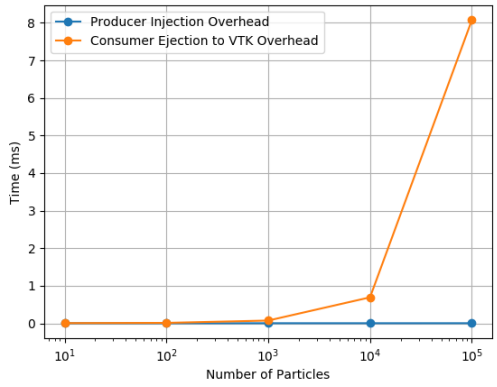

Figure 2: Preliminary data demonstrating the costs of moving data into and out of Faodail

on data as it is inserted and store the information as additional metadata in Faodail.

4 EVALUATION

To demonstrate the feasilbility of Faodail, we implemented a simple visualization example using SimplePIC and VTK [2]. In this example, particle data from SimplePIC are inserted into Faodail. The visualization application then retrieves data from Faodail and uses it to construct VTK objects for the purpose of generating a visual representation of the data.

The results of this experiment are shown in Figure 2. This preliminary evaluation is intended to demonstrate functionality and feasibility rather than performance. This figure shows the time required to move groups of particle data as a function of the number of particles being moved. The blue line shows the cost of inserting a block of particle data into Faodail. Because Faodail is designed to exchange data using shallow copies of reference-counted data structures, the time required to insert a block of particle data is independent of the number of particles for which data are being inserted.

The orange line in Figure 2 shows the cost of reading particle data from Faodail and generating VTK objects from its contents. In this case, the time required increases with the size of the data being manipulated. This is due to the fact that using particle data to create VTK objects currently requires a deep copy of the data read from Faodail.

5 RELATED WORK

There are multiple efforts related to this work that provide a way to share data between different application jobs. DataSpaces [8] has a long development history with increasing capabilities over time for coupling BSP applications in HPC via RDMA primitives. It offers a similar idea, but does not offer any management facilities for handling slow data epochs, such as those from AMT application. Catalyst [5] is a library for ISAV that uses adapters in an application's compute nodes to convert data into VTK structures that on-node and off-node (via non-RDMA transfers) visualization pipelines can process. The libsim [23] package provides similar capabilities, but for the VisIt environment. Both of these libraries are the kinds of analytics frameworks Faodail seeks to support. GLEAN [22] is an I/O service for the IBM Blue Gene platform that connected analytics and I/O services to applications. It offers data aggregation services and hooks for analytics, but does not address the slow data epochs nor integration for AMTs because the pNetCDF and HDF integration is difficult to handle efficiently for AMT codes. ADIOS is a popular I/O library for working with persistent datasets that can be coupled with DataSpaces for ISAV work. ADIOS suffers from far fewer, but similar synchronization points as HDF5 and pNetCDF, but offers integration via custom transport methods (similiar to HDF Virtual Object Layer plugins). Conduit [7] and Bredala [9] focus on the data type management tasks for integrating codes. Neither of these tools offers the extensive data management support including AMT-style slow data epochs.

Additional related work is taking place in the AMT communities. Pebay et al. [19] argues for implementing ISAV applications as task DAGs themselves in the AMT frameworks. This approach is appealing from a systems perspective because the runtime can then schedule the ISAV tasks alongside the applications tasks and manage data handoffs. Earlier work in a similar style to this integration is all of the Data Staging efforts [3, 16, 18, 20, 24]. These efforts were more focused on moving data to a new area rather than attempting to address AMT. Some focused on incorporating processing, *see e.g.*, [24], but not full analytics tools and did not effectively support the slow data epochs in AMT codes.

6 CONCLUSION

In this paper we have describe that an AMT data management service, such as our Faodail tool, is capable of supporting ISAV in addition to the primary functionality of supporting AMT-based calculations. While our initial evaluation simply demonstrates feasibility, the performance is still reasonable for analysis and visualization tasks.

Future work includes optimizing the performance and scaling for exascale sized workloads. Additional efforts to demonstrate full applications for both the simulation and the ISAV are also desired.

ACKNOWLEDGEMENTS

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

This work was supported under the U.S. Department of Energy National Nuclear Security Agency ATDM project funding. This work was also supported by the U.S. Department of Energy Office of Science, under the SSIO grant series, SIRIUS project and the Data Management grant series, Decaf project, program manager Lucy Nowell.

REFERENCES

[1] [n. d.]. The HDF Group. http://www.hdfgroup.org/. ([n. d.]).
[2] [n. d.]. VTK - The Visualization Toolkit. ([n. d.]). http://www.vtk.org

[3] Hasan Abbasi, Jay Lofstead, Fang Zheng, Scott Klasky, Karsten Schwan, and Matthew Wolf. 2009. Extending I/O Through High Performance Data Services. In *Cluster Computing*. IEEE International, Luoisiana, LA.

[4] Sean Ahern, Arie Shoshani, Kwan-Liu Ma, Alok Choudhary, Terence Critchlow, Scott Klasky, Valerio Pascucci, Jim Ahrens, E Wes Bethel, Hank Childs, et al. 2011. Scientific discovery at the exascale. In *Report from the DOE ASCR 2011 Workshop on Exascale Data Management*. US DOE, Washington, DC, 20.

[5] Utkarsh Ayachit, Andrew Bauer, Berk Geveci, Patrick O'Leary, Kenneth Moreland, Nathan Fabian, and Jeffrey Mauldin. 2015. Paraview catalyst: Enabling in situ data analysis and visualization. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*. ACM, ACM, New York, 25–29.

[6] Michael Bauer, Sean Treichler, Elliott Slaughter, and Alex Aiken. 2012. Legion: Expressing locality and independence with logical regions. In *Proceedings of the international conference on high performance computing, networking, storage and analysis*. IEEE Computer Society Press, 66.

[7] Adam Kunen Cyrus Harrison, Brian Ryujin. 2015. Conduit. (2015). https://github.com/LLNL/conduit

[8] Ciprian Docan, Manish Parashar, and Scott Klasky. 2012. Dataspaces: an interaction and coordination framework for coupled simulation workflows. *Cluster Computing* 15, 2 (2012), 163–181.

[9] Matthieu Dreher and Tom Peterka. 2016. Bredala: semantic data redistribution for in situ applications. In *Cluster Computing (CLUSTER), 2016 IEEE International Conference on*. IEEE, IEEE, New York, 279–288.

[10] J Davison de St Germain, John McCorquodale, Steven G Parker, and Christopher R Johnson. 2000. Uintah: A massively parallel problem solving environment. In *High-Performance Distributed Computing, 2000. Proceedings. The Ninth International Symposium on*. IEEE, IEEE, New York, 33–41.

[11] Laxmikant V Kale and Sanjeev Krishnan. 1993. CHARM++: a portable concurrent object oriented system based on C++. In *ACM Sigplan Notices*, Vol. 28. ACM, ACM, New York, 91–108.

[12] J. Lofstead, J. Dayal, I. Jimenez, and C. Maltzahn. 2014. Efficient, Failure Resilient Transactions for Parallel and Distributed Computing. In *Data Intensive Scalable Computing Systems (DISCS), 2014 International Workshop on*. 17–24. https://doi.org/10.1109/DISCS.2014.13

[13] Jay Lofstead, Jai Dayal, Karsten Schwan, and Ron Oldfield. 2012. D2T: Doubly Distributed Transactions for High Performance and Distributed Computing. In *IEEE Cluster Conference*. Beijing, China.

[14] Jay Lofstead, Fang Zheng, Scott Klasky, and Karsten Schwan. 2009. Adaptable, metadata rich IO methods for portable high performance IO. In *Proceedings of the International Parallel and Distributed Processing Symposium*. Rome, Italy.

[15] K Moreland, D Lepage, D Koller, and G Humphreys. 2008. Remote rendering for ultrascale data. *Journal of Physics: Conference Series* 125, 1 (2008), 012096. http://stacks.iop.org/1742-6596/125/i=1/a=012096

[16] Arifa Nisar, Wei-keng Liao, and Alok Choudhary. 2008. Scaling Parallel I/O Performance Through I/O Delegate and Caching System. In *SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*. IEEE Press, Piscataway, NJ, USA, 1–12. https://doi.org/10.1145/1413370.1413380

[17] Ron A. Oldfield, Patrick Widener, Arthur B. Maccabe, Lee Ward, and Todd Kordenbrock. 2006. Efficient Data-Movement for Lightweight I/O. In *Proceedings of the 2006 International Workshop on High Performance I/O Techniques and Deployment of Very Large Scale I/O Systems*. Barcelona, Spain.

[18] Ron A. Oldfield, David E. Womble, and Curtis C. Ober. 1998. Efficient Parallel I/O in Seismic Imaging. *International Journal of High Performance Computing Applications* 12, 3 (Fall 1998), 333–344. ftp://ftp.cs.dartmouth.edu/pub/raoldfi/salvo/salvoIO.ps.gz

[19] Philippe Pébaÿ, Janine C Bennett, David Hollman, Sean Treichler, Patrick S McCormick, Christine M Sweeney, Hemanth Kolla, and Alex Aiken. 2016. Towards asynchronous many-task in situ data analysis using legion. In *Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International*. IEEE, IEEE, New York, 1033–1037.

[20] Charles Reiss, Gerald Lofstead, and Ron Oldfield. 2008. *Implementation and evaluation of a staging proxy for checkpoint I/O.* Technical Report. Sandia National Laboratories, Albuquerque, NM.

[21] M. Riedel, T. Eickermann, S. Habbinga, W. Frings, P. Gibbon, D. Mallmann, F. Wolf, A. Streit, T. Lippert, W. Schiffmann, A. Ernst, R. Spurzem, and W.E. Nagel. 2007. Computational Steering and Online Visualization of Scientific Applications on Large-Scale HPC Systems within e-Science Infrastructures. In *e-Science and Grid Computing, IEEE International Conference on*. IEEE, New York, 483 –490. https://doi.org/10.1109/E-SCIENCE.2007.21

[22] Venkatram Vishwanath, Mark Hereld, Michael E Papka, Randy Hudson, G Cal Jordan IV, and Christopher Daley. 2011. In situ data analysis and I/O acceleration of FLASH astrophysics simulation on leadership-class system using GLEAN. In *Proc. SciDAC, Journal of Physics: Conference Series*. US DOE, Washington, DC.

[23] Brad Whitlock, Jean M. Favre, and Jeremy S. Meredith. 2011. Parallel in Situ Coupling of Simulation with a Fully Featured Visualization System. In *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization (EGPGV '11)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 101–109. https://doi.org/10.2312/EGPGV/EGPGV11/101-109

[24] Fang Zheng, Hasan Abbasi, Ciprian Docan, Jay Lofstead, Scott Klasky, Qing Liu, Manish Parashar, Norbert Podhorszki, Karsten Schwan, and Matthew Wolf. 2010. PreDatA - Preparatory Data Analytics on Peta-Scale Machines. In *In Proceedings of 24th IEEE International Parallel and Distributed Processing Symposium, April, Atlanta, Georgia*.