Towards Scalable Visualization Plugins for Data Staging Workflows

David Pugmire
Oak Ridge National Laboratory
Oak Ridge, Tennessee

James Kress
University of Oregon
Eugene, Oregon

Jeremy Meredith, Norbert Podhorszki, Jong Choi, Scott Klasky
Oak Ridge National Laboratory
Oak Ridge, Tennessee

***Abstract*—The scientific data that are being generated today, and in the near future, will quickly outpace our ability to process and understand it. The data generated are growing in multiple ways, including the size of the data, the rate at which it arrives, and the varying types of data. Additionally, data are available from multiple sources, including for example, computational simulations and sensor data extracted from experiments. In such situations, workflows for the movement and management of data, as well as the analysis and visualization while the data are in transit, become even more critical, and increasingly challenging. In this paper we discuss some early work focused on the development of in-transit visualization techniques that are deployed within a large data management system.**

I. Introduction

The cost of data movement is quickly becoming a bottleneck to computation [2]. Recently deployed super computers, and those planned for the future provide far more computational capacity than I/O bandwidth. Traditionally, visualization is performed as a post-processing task, where simulation outputs are read from disk, into the memory of a parallel tool where analysis and visualization are performed. Visualization is generally I/O bound [3], and as the relative I/O bandwidth continues to decrease, the challenges of visualization of increasingly larger data will become more problematic. In the case of traditional visualization, the I/O bottleneck is exacerbated as data is first written to disk by the simulation, and then read back from disk by the visualization routine.

Additionally, the parallelism on compute nodes is dramatically increasing, especially with the growing trend to heterogenous systems. At the same time, the memory is not increasing at the same rate as the computational capability [2]. These pressures place further constraints on analysis and visualization algorithms to operate in a variety of computational environments and memory footprints.

Managing these complexities requires careful handling of data movement and operations on data that can be performed while the data are being transfered through the system. To better understand these complexities, we have designed and tested a system which reads and performs visualization operations on simulation data as it is being moved within the workflow. This system is built on top of two different research projects, ADIOS [7], which is a middleware layer designed to efficiently handle the movement of large amounts of data, and EAVL [9] which is a library for efficiently representing and operating on large data in heterogeneous environments.

In this paper, we describe and explore a workflow system we have designed, built and tested which explores light-weight EAVL visualization operations that have been deployed as plugins in the ADIOS data staging framework. We are particularly interested in the usability, performance and scaling of these operations in a production environment, and how this system can scale up to larger and larger workflows.

II. Contributions

The major contribution of this work is the integration and use of light-weight visualization plugins implemented in EAVL within the ADIOS middleware framework, and the initial exploration of using this in a workflow for a production scientific application. We also experiment with using the advanced data model of EAVL to more efficiently and compactly represent scientific data, and compute the results. We also experiment with using the execution environment of EAVL and compare two different methods for optimization of the parallel rendering pipeline.

As the I/O bottleneck has increased over time, a number of strategies have been proposed to address the I/O bottleneck. Two such examples are clever and more efficient I/O strategies, and the reduction of the size of data to write to disk (e.g. through compression). These strategies will help, but ultimately will not suffice in preventing the widening gap between computational throughput and I/O [2].

One active area of research in addressing the I/O bottleneck is to process the data while it is in memory and only write a subset of the simulation data, or derived data, to disk. These techniques, known as in situ processing, are typically either tightly-coupled or loosely-coupled. In a tightly-coupled scenario, the visualization is performed in the memory space of the running application. Additional work has been done in integrating in situ visualization into parallel visualization tools [14] [6]. In tightly-coupled in situ, the processing is done on the same data being used by the simulation, and the simulation must often pause while the processing is taking place.

The work here is done in a loosely-coupled framework, where the simulation data is de-coupled and staged to a secondary memory location where processing can take place. In a loosely-coupled in situ environment, the application performs a data write (to the staged secondary memory location), and then can continue computation. Once the data have arrived at the secondary memory location, the data processing can be performed. This de-coupling from the simulation code has the advantage of being more portable and modular, not interfering with the simulation code or compute resources, and the ability

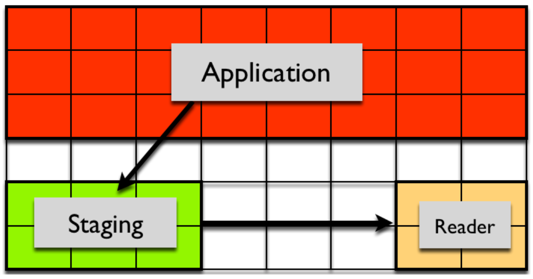

Fig. 1: Data staging in ADIOS. A simulation is launched with an allocation that is divided into three groups, application, staging, and the reader. The application writes data to the staging nodes, and the reader reads data from the staging nodes.

to move the data to a different set of appropriate visualization and analysis nodes. Such nodes might, for example, contain more main memory.

Recently, there has been research into visualization frameworks for next generation hardware. The current de facto standard (VTK) has made a significant impact on the visualization community over the past decade, but it is ill-equipped to run on nodes with massive parallelism (including heterogeneous nodes), or to represent certain types of scientific data. The EAVL [10], DAX [12] and PISTON [8] frameworks have been developed to address these shortcomings. PISTON has focused on efficient algorithms for GPUs, while DAX has focused on an execution engine for heterogenous compute nodes. We have used EAVL, which has an advanced data model for efficiently using memory to represent scientific data, and an execution engine which operates on the data model on hetereogenous compute nodes. We explore the use of EAVL to model a production scientific simulation and peform visualization operations in a data staging workflow environment.

Finally, as in-situ and in-transit methods lessen the impact of the I/O bottleneck, analysis and visualization operations become increasingly compute- and memory-bound. Additionally, as the amount and frequency of data increase, it will become important for analysis and visualization operations to be able to quickly perform their work, and then be prepared to process the next simulation step.

III. System

A. Overview

The system we are developing is architected as shown in Figure 1. The application, in this case is the plasma fusion simulation code XGC1, is running on a larger allocation within the compute cluster. A smaller allocation of nodes are dedicated to data staging where the application transmits data over the high speed network. A visualization service is running on another set of allocations which reads the data directly from the staging server, performs visualization operations, and then saves an image for each simulation time step. The only data that are written to disk are the images of visualization operations, which are orders of magnitude smaller than the original simulation data.

Our system is designed to utilize ADIOS and EAVL to perform the data staging and visualization operations. We briefly give an overview of these two technologies here, then describe in detail how these are used in our system in Section IV.

B. ADIOS

The Adaptable I/O System (ADIOS) [7], is a componentization of the I/O layer used by high end simulations and/or for high end scientific data management, providing an easy-to-use programming interface, which can be as simple as Fortran file I/O statements. ADIOS abstracts the API away from implementation, allowing users to compose their applications without detailed knowledge of the underlying software and hardware stack. ADIOS framework has been designed with a dual purpose: to increase the I/O throughput of simulations using well known optimization techniques, and also to serve as the platform for introducing novel data management solutions for production use without extensive modifications of the target applications.

ADIOS is used by a variety of mission critical applications running at DOE and NSF facilities, including combustion, materials science, fusion, seismology, and others. At the same time, ADIOS offers the community a framework for developing next generation I/O and data analytics techniques [15][4].

To address the growing imbalance between computational capability and I/O performance, ADIOS introduced the concept of data staging, where rather than writing data directly to shared backend storage devices, a staging pipeline moves data to a transient location, on separate physical nodes and/or on memory resources on the same node where data is generated. Once on the *staging* nodes, data can be aggregated, processed, indexed, filtered, and eventually written out to persistent storage. A key outcome of staging has been dramatic reductions in the total volume of data to be stored through the use of *in-situ* and *in-transit* data analytics. There are several staging transports in ADIOS, one of them being Dataspaces [5], which provides methods for dynamic interaction patterns between distributed scientific application processes. Providing a semantically specialized shared-space abstraction to distributed processes using staging nodes, ADIOS/DataSpaces has been deployed with about 50 recent downloads, and is currently being used by production coupled scientific simulation workflow on large-scale NSF and DOE resources.

C. EAVL

The Extreme-scale Analysis and Visualization Library (EAVL) [9] [10] was developed to address three primary objectives: update the traditional data model to handle modern simulation codes; investigate the effiency of I/O, computation and memory on an updated data and execution model; and explore visualization algorithms on next-generation architectures.

EAVL defines more flexible mesh, and data structures which more efficiently supports the traditional types of data supported by de-facto standards like VTK, but also allows for efficient representations of non-traditional data. Examples of non-traditional data includes graphs, mixed data types (e.g.

molecular data, high order field data, unique mesh topologies (e.g. unstructured adaptive mesh refinement and quad-trees)).

EAVL uses a functor concept in the execution model to allow users to write operations that are applied to data. The functor concept in EAVL has been abstracted to allow for execution on either the CPU or GPU, and the execution model manages the movement of data to the particular execution hardware.

IV. Evaluation

A. Experiments

The goal of these experiments was to begin studying the performance of light-weight visualization plugins in a staging environment with an HPC application. We are interested in both performance, scalability, and ease-of-use of these plugins.

One of the configuration parameters for XGC1 is the number planes in the mesh in the torrodial direction. This parameter defines how the large the mesh, and the associated field data are. For this experiment, we ran XGC1 with 32, 64, and 128 planes. The visualization plugin uses a spatial decomposition of the mesh for parallelization. This spatial decomposition is defined by the number of planes, and the maximum number spatial domains is equal to the number of planes in the XGC1 mesh.

For this experiment, we focused on using a single dataspaces server and large number of visualization plugin options. For the visualization plugin, we ran them at different concurency levels and node layouts to study the performance and scalability, and we evaluate whether the visualization operations are efficient enough for in situ analysis scenarios where the data is changing very quickly.

The experiments were run on *sith*, an OLCF 40 node cluster where each node has four 2.3 GHz 8 core AMD Opteron 6134 processors, and 64 GB of memory. We also ran the 128 plane simulations on *rhea*, an OLCF 512 node cluster where each node has dual 8 core Intel Xeon E5-2650 CPUs and 64 GB of memory.

B. Data staging in ADIOS

Since the computational mesh in XGC1 doesn't change throughout the simulation, it is written once at start up. XGC1 uses an ADIOS configuration file to specify which variables are written, and where they are written, either to disk or to a dataspaces server. In our test configuration, we specify that the common field variables used for monitoring and analysis are written to the dataspaces server after every simulation time step of the application code. XGC1 was run with 32 planes and 384 cores, with 64 planes and 768 cores, and finally with 128 planes and 1536 cores.

C. Visualization operations in EAVL

The visualization plugin was developed in EAVL, and consists of a parallel data reader and a visualization pipeline engine, and is shown in Figure 2. For this particular experiment, we used a visualization pipeline consisting of an isosurface operation, rendering in parallel using the Mesa 3D graphics library [1], and parallel image compositing. For

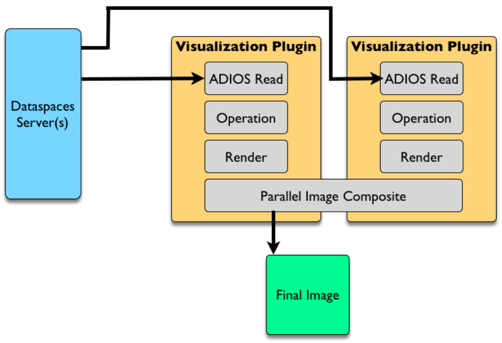

Fig. 2: Overview of visualization plugins. Each plugin performs a read operation, operates on the data followed by rendering, and then a parallel image composite to produce the final image.

parallel image compositing we experimented with two different techniques. The first was a basic depth-based compositor that uses MPI to perform the reduction operations, and the second was with IceT [11]. We configured IceT to composite a single tile using the ICET_STRATEGY_SEQUENTIAL as the compositing strategy. This strategy results in the fastest composite possible when rendering a single tile, as it avoids certain MPI overhead associated with other IceT strategies.

We ran the plugin at a variety of concurrencies, from 1 process up to 128. In these experiments the visualization plugin will read each simulation step from the dataspaces server, perform the operation, and then write an image to disk.

V. Results

Our experiments demonstrate the viability of running light-weight visualization plugins with a production run of XGC1. The ease of use of this system is highlighted with the fact that no changes to the application were required, as the modificatons to data movement are accomplished by only a change to the ADIOS configuration file. All that is required is then to launch the dataspaces servers and the visualization plugins. The data management and movent is handled by the dataspaces servers and read and processed by the visualization plug-ins as the data become available. At each simulation step read by the visualization plugin, an isosurface is extracted and rendered in parallel, and the resulting image is saved to disk. These images can easily be used for monitoring of the simulation, and for post run analysis.

While visualization done in situ or with data staging does not always eliminate the need for post-processing of simulation data, it is able to automatically perform simulation monitoring and perform a priori visualization and analysis operations. This eliminates the ever-increasing I/O bottleneck for these types of very common operations.

With the reduction of the I/O bottleneck, visualization operations become compute- and memory-bound. In these environments, efficient representations of data in memory, and efficient computation become a limiting factor.

We have implemented the representation of the XGC1 mesh using the advanced data model capabilities in EAVL. The

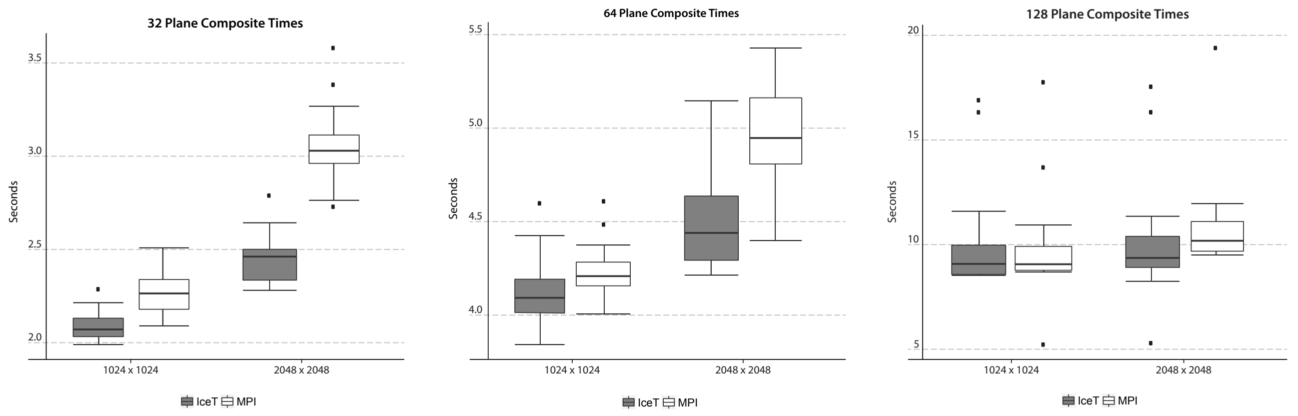

Fig. 3: Composite times for concurrency and node layouts that were performed on the 32, 64, and 128 plane data sets.

flexibility in the data model allows us to represent the XGC1 mesh between 32X and 128X more efficently than traditional models like VTK. The unstructured mesh in XGC1 is such that a single set of nodes in the plane are rotated around the torus to produce the entire mesh. In VTK, these nodes and cells must be explicitly enumerated, whereas in EAVL a single plane is fully represented, and a sequence of angles specifies where each subsequent plane should be positioned. The particular nodes and cells are then algorithmically determined on the fly. Additionally, since each cell in an XGC1 mesh is a wedge, instead of enumerating each cell type (as required in VTK), a specialized unstructured cell type is used. This reduces the memory requirement for the cell type information from linear in the number of cells to a single constant.

The advanced computational model in EAVL provides the flexibility to run on both CPUs or GPUs. While we did not study the performance of GPUs in this experiment, they are well documented in previous work [10]. In this set of experiments we focused on the parallel rendering aspect of visualization plugins. In our preliminary studies, the parallel depth-compositing step was a signficant cost in rendering. We experimented with an MPI-only implementation which used a global reduction to perform Z-buffer based depth compositing of images, and the IceT library. We experimented with a variety of data configurations (32, 64, and 128 planes), different levels of concurrency and layouts of visualization plugins, and image size (1024x1024 and 2048x2048 pixels).

The plots in Figure 3 show a summarization for all the concurrency and node layouts tests that were performed on the 32, 64, and 128 plane data sets. The box plots show the first to third quartile of the data, the whiskers represent data within the 1.5 IQR of the upper and lower quartiles. The dots above and below the whiskers identify any outliers. These plots provide a convenient way to capture the entire essence of the runs across each of the three data sets. While the performance advantages of IceT are fairly small for the small image size, it becomes more significant for the larger image size. Overall, for the smaller 1024x1024 images, IceT out performed the basic MPI implementation by 5%, and by 14% for the larger 2048x2048 images.

In Figure 4 we show a representative scaling study of the

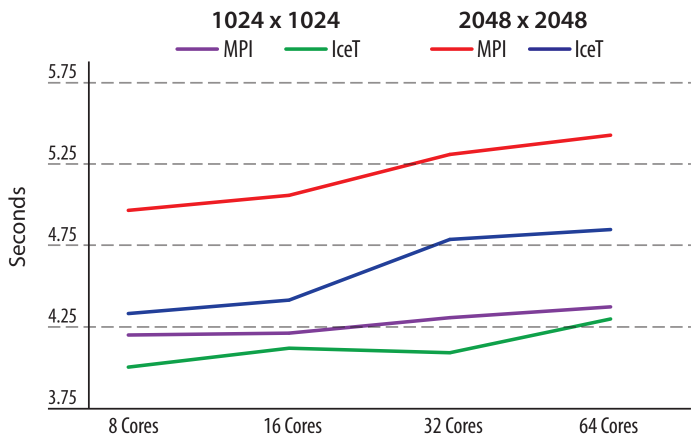

Fig. 4: Scalability of parallel rendering of the 64 plane data set run on 8 nodes with varying levels of concurrency.

64 plane data set running on 8 nodes with varying levels of concurrency. The advantages of IceT as concurrency grows is clearly seen, especially for larger image sizes.

VI. Conclusions and Future Work

Light-weight plugins developed in EAVL provide a viable method for the analysis and visualization of large scientific data within the ADIOS data staging framework. As the I/O bottleneck is significantly reduced by data staging techniques, visualization and analysis operations become compute- and memory-bound. It is critical that careful attention is paid to the efficiency of each step in a visualization pipeline. This need will only increase as the volume and velocity of data increases in the future.

The parallelization and general scalability of the plugins we have experimented with give us confidence that these methods will be effective as a tools for scientists to understand and analyze data. We plan to continue working with larger simulation runs, including runs of XGC1 on the Oak Ridge

Leadership Computing Facility's Titan supercomputer. We also plan on working with the particle data in XGC1, which is significantly larger than the field data on the mesh.

We also plan on making use of the ADIOS visualization schema [13] in the plugins. The ADIOS visualization schema makes the data streams much more self-describing and allows for automatic data format detection, aids in feature detection, and assists in the generation of specific analyses and visualizations. We also plan to handle more complex, multi-stage workflows, including code coupling, and comparative visualization.

VII. Acknowledgements

This work was supported by the SciDAC Institute of Scalable Data Management, Analysis and Visualization (SDAV) funded by the DOE Office of Science through the Office of Advanced Scientific Computing Research (Contract No. 12-015215). Computational resources were made available by the Oak Ridge Leadership Computing Facility.

References

[1] *The Mesa 3D Graphics Library*. http://www.mesa3d.org.

[2] S. Ahern, A. Shoshani, K.-L. Ma, A. Choudhary, T. Critchlow, S. Klasky, V. Pascucci, J. Ahrens, E. Bethel, H. Childs, et al. Scientific discovery at the exascale. report from the doe ascr 2011 workshop on exascale data management. *Analysis, and Visualization*, 2, 2011.

[3] H. Childs, D. Pugmire, S. Ahern, B. Whitlock, M. Howison, Prabhat, G. H. Weber, and E. W. Bethel. Extreme scaling of production visualization software on diverse architectures. *IEEE Comput. Graph. Appl.*, 30(3):22–31, May 2010.

[4] J. Dayal, D. Bratcher, G. Eisenhauer, K. Schwan, M. Wolf, X. Zhang, H. Abbasi, S. Klasky, and N. Podhorszki. Flexpath: Type-based publish/subscribe system for large-scale science analytics. In *14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing(CCGRID '14)*, 2014.

[5] C. Docan, M. Parashar, and S. Klasky. Dataspaces: an interaction and coordination framework for coupled simulation workflows. *Cluster Computing*, 15(2):163–181, 2012.

[6] N. D. Fabian, A. C. Bauer, N. Podhorszki, R. A. Oldfield, and U. Ayachit. In-situ visualization with catalyst. 01 2012.

[7] Q. Liu, J. Logan, Y. Tian, H. Abbasi, N. Podhorszki, J. Y. Choi, S. Klasky, R. Tchoua, J. Lofstead, R. Oldfield, M. Parashar, N. Samatova, K. Schwan, A. Shoshani, M. Wolf, K. Wu, and W. Yu. Hello adios: the challenges and lessons of developing leadership class i/o frameworks. *Concurrency and Computation: Practice and Experience*, 26(7):1453–1473, 2014.

[8] L.-t. Lo, C. Sewell, and J. P. Ahrens. Piston: A portable cross-platform framework for data-parallel visualization operators. In *EGPGV*, pages 11–20, 2012.

[9] J. S. Meredith, S. Ahern, D. Pugmire, and R. Sisneros. EAVL: the extreme-scale analysis and visualization library. In *Eurographics Symposium on Parallel Graphics and Visualization*, pages 21–30. The Eurographics Association, 2012.

[10] J. S. Meredith, R. Sisneros, D. Pugmire, and S. Ahern. A distributed data-parallel framework for analysis and visualization algorithm development. In *Proceedings of the 5th Annual Workshop on General Purpose Processing with Graphics Processing Units*, GPGPU-5, pages 11–19, New York, NY, USA, 2012. ACM.

[11] K. Moreland. Icet users' guide and reference. Technical Report 2011-5011, Sandia National Laboratory, 2011.

[12] K. Moreland, U. Ayachit, B. Geveci, and K.-L. Ma. Dax toolkit: A proposed framework for data analysis and visualization at extreme scale. In *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*, pages 97–104, Oct 2011.

[13] R. Tchoua, J. Choi, S. Klasky, Q. Liu, J. Logan, K. Moreland, J. Mu, M. Parashar, N. Podhorszki, D. Pugmire, et al. Adios visualization schema: A first step towards improving interdisciplinary collaboration in high performance computing. In *eScience (eScience), 2013 IEEE 9th International Conference on*, pages 27–34. IEEE, 2013.

[14] B. Whitlock, J. M. Favre, and J. S. Meredith. Parallel in situ coupling of simulation with a fully featured visualization system. In *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization*, EG PGV'11, pages 101–109, Aire-la-Ville, Switzerland, Switzerland, 2011. Eurographics Association.

[15] F. Zhang, C. Docan, M. Parashar, S. Klasky, N. Podhorszki, and H. Abbasi. Enabling in-situ execution of coupled scientific workflows. In *Proceedings for the 26th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2012), Shanghai, China*, 2012.