DOI: 10.14529/jsfi160404

Preparing for In Situ Processing on Upcoming Leading-edge Supercomputers

James Kress[1], *Randy Michael Churchill*[2], *Scott Klasky*[3], *Mark Kim*[3], *Hank Childs*[4], *David Pugmire*[3]

© The Authors 2016. This paper is published with open access at SuperFri.org

High performance computing applications are producing increasingly large amounts of data and placing enormous stress on current capabilities for traditional post-hoc visualization techniques. Because of the growing compute and I/O imbalance, data reductions, including in situ visualization, are required. These reduced data are used for analysis and visualization in a variety of different ways. Many of the visualization and analysis requirements are known a priori, but when they are not, scientists are dependent on the reduced data to accurately represent the simulation in post hoc analysis. The contributions of this paper is a description of the directions we are pursuing to assist a large scale fusion simulation code succeed on the next generation of supercomputers. These directions include the role of in situ processing for performing data reductions, as well as the tradeoffs between data size and data integrity within the context of complex operations in a typical scientific workflow.

Keywords: Scientific Visualization, In Situ Methods, Data Staging Methods, Data Reductions, High Performance Computing.

Introduction

As leading-edge supercomputers get increasingly powerful, scientific simulations running on these machines are generating ever larger volumes of data. However, the increasing cost of data movement, in particular moving data to disk, is increasingly limiting the ability to process, analyze, and fully comprehend simulation results [1], hampering knowledge extraction. Specifically, while I/O bandwidths regularly increase with each new supercomputer, these increases are well below corresponding increases in computational ability and data generated. Further, this trend is predicted to persist for the foreseeable future.

Given this reality, many large-scale simulation codes are attempting to bypass the I/O bottleneck by using in situ visualization and analysis, i.e., processing simulation data when it is generated. A key question for in situ analysis is whether there is a priori knowledge of which visualizations and analyses to produce. If this a priori knowledge exists, then in situ processing is often superior to post hoc processing on leading-edge supercomputers, since it avoids disk usage. However, it is not guaranteed that the required visualizations and analyses are known a priori. That is, a domain scientist may need to explore the data in an interactive fashion, or unanticipated analysis may be required. With this research, we consider the model where in situ processing is used to reduce the data to a form small enough that it can be saved to disk and later read back for post hoc exploration.

Data reductions under this model could take on many different forms. A few examples of data reduction types could be compression (lossy or lossless), summary data (vector fields, min/max in a region, etc.), and reduced precision data formats. Importantly, data reductions should consider the types of analysis that will be done in the future, as to not introduce errors or artifacts that are not expected. This means that data reductions should be done within a set of known error bounds that is acceptable to the researchers doing the post hoc analysis.

[1]University of Oregon, Eugene, USA & Oak Ridge National Laboratory, Oak Ridge, USA; kressjm@ornl.gov
[2]Princeton Plasma Physics Laboratory, Princeton, USA; rchurchi@pppl.gov
[3]Oak Ridge National Laboratory, Oak Ridge, USA; {klasky, kimmb, pugmire}@ornl.gov
[4]University of Oregon, Eugene, USA & Lawrence Berkeley National Laboratory, Berkley, USA; hank@cs.uoregon.edu

With this work, we consider in situ data reduction in the context of a cutting-edge fusion simulation code, XGC1. We collaborate closely with XGC1 domain scientists and have been considering which reductions will be appropriate for this code as their ability to store data further and further decreases. We consider two distinct approaches, both of which we believe will be necessary for successfully maintaining meaningful analysis and visualization as XGC1 simulations are run on the next generation of supercomputers. In both cases, the techniques we consider keep in mind the balance between reduction and integrity. The contribution of the paper, then, is the description of the directions we pursue for XGC1 to succeed on the next generation of supercomputers, and our results to date in these directions. XGC1 contains two different types of data — particle data and mesh-based field data — and we consider techniques for both. For particle data, we consider sub-selection of particles and how to carry this out in a meaningful fashion. For mesh-based field data, we consider reduced precision and its effects.

In the remainder of this paper we discuss related work in Section 1, briefly discuss XGC1 in Section 2, present two example problems motivating in situ data reductions in Sections 3 & 4, and conclude with a discussion of areas of continuing and future research.

1. Related Work

Our work builds off of the momentum from the in situ movement, as well as past visualization work within XGC1. We present the related work for in situ data reduction in three sub-categories: (1) in situ visualization, (2) XGC1 visualization, and (3) HPC data compression.

1.1. In Situ Visualization

Visualization algorithms are particularly sensitive to I/O bandwidth [3, 4], causing the community to turn to in situ techniques to alleviate this growing problem. There has been significant work and successes with the in situ visualization paradigm. For instance, ParaView Catalyst coprocessing [7] and VisIt LibSim [33] are frameworks that are tightly-coupled to the simulation, *i.e*, the visualization runs at scale with the simulation. Visualization and analytics can be performed during the transport of the simulation data to the I/O layer as well. Three examples of this loosely coupled approach are Nessie [26], GLEAN [32], and ADIOS [18]. For a more thorough overview of the three loosely-coupled in situ visualization frameworks, we refer the reader to [23].

1.2. XGC1 Visualization

Early work on production visualization for XGC1 mainly focused on addressing the immediate data needs of scientists during the course of a simulation run. One example of this was an online dashboard that was developed for XGC1 simulation monitoring called eSimon [30]. This dashboard was launched in conjunction with each simulation run, and was responsible for performing common visualization and analysis tasks in XGC1. First, the dashboard was responsible for creating and updating plots of approximately 150 different variables every 30 seconds and plotting 65 different planes from the live simulation. At the conclusion of a run, the dashboard would automatically output movies of each of these plots of interest for quick review. In addition this dashboard catalogued simulation output, allowing users to search for and retrieve data of interest, without having to locate and search through simulation output files. Finally, this dashboard was available to scientists anywhere in the world through their internet browsers, making the data quickly and readily available.

More recent work has focused on expanding the visualization capabilities and opportunities for XGC1 through the utilization of in situ methods. For example, they utilized the features of ADIOS and EAVL [28], and demonstrated the effectiveness of loosely coupled in situ visualization for large scale simulation codes using a workflow consisting of ADIOS, data staging, and EAVL. In that work, they focused on the performance, scalability, and ease of use of visualization plugins that were used on the output of the XGC1 simulation code. One component of this study looked at optimizing the parallel rendering pipeline in situ, and gave insight into getting high performing renderings in continuing studies.

Following this effort, work shifted towards researching in transit visualization opportunities for XGC1 on the wide area network [27]. This research looked at data coupling and near-real-time analysis and visualization between two geographically separated sites using ADIOS and its ICEE [5] transport method. This capability is important when considering future use cases for visualization and analysis, when both simulation and experimental data need to be used and streamed together.

1.3. HPC Data Compression

Data compression strategies for HPC data are typically split into two categories, lossless and lossy compression. Typical lossless compression techniques include Huffman encoding [10], LZ77 [34], Gzip [8] and Fpzip [15]. Huffman encoding is an entropy-based compression method that identifies common occurrences in the data and assigns them unique codes. The LZ77 algorithm uses a sliding window to search for repeated sequences and then replaces them with a single copy that occurred previously in the data. Gzip is a general purpose compression technique for compressing any type of data, and makes use of both Huffman encoding and the LZ77 algorithm. Fpzip is a lossless compression method that is focused on floating point data. To improve the compression ratio, a variety of lossy compression methods have been developed over the years, including ISABELA [13] and ZFP [14]. ISABELA applies a pre-conditioner to the data, and then fits the data with B-spline interpolation to achieve very impressive compression rates for floating point data. ZFP is a fixed-rate compression scheme that encodes structured scientific data into $4 \times 4 \times 4$ blocks using a lifting scheme. Although designed for encoding scientific codes, ZFP was used for lossy disk storage compression for full-3D seismic waveform tomography, and reduced the strain-field disk storage by at least an order of magnitude [16].

Work in mesh decimation and simplification is a well researched topic, and Luebke [19] provides an overview of the various strategies including simplification and view-dependent methods. Also included are the approaches for each strategy, including sampling, decimation, vertex merging, topology driven, and error minimization techniques.

2. XGC1 Overview

XGC1 [2] is a 5D gyrokinetic ion-electron particle in cell (PIC) code used to study fusion of magnetically confined burning plasmas. XGC1 is used in particular to study turbulence in the outer region of the plasma called the *edge*. The simulation proceeds by computing the interactions of a very large number of particles (ions and electrons), and depositing them onto a finite element mesh at each time step. The mesh consists of a number of 2D planes positioned uniformly around the toroidal shape of the tokamak, as shown in Figure 1. At each time step, the particles, which interact within the toroidal space of the mesh, are statistically deposited as scalar fields onto the mesh. This deposition step provides a statistical view of simulation, and also helps optimize the simulation runtime.

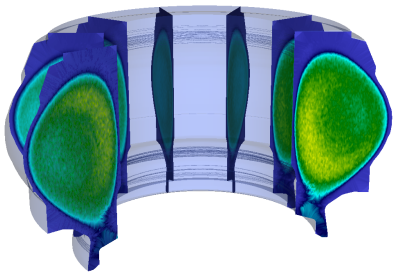


Figure 1. Example of a mesh in XGC1. Its planes are equally spaced around the central axis of the tokamak

3. Data Reduction in Staging

In situ methods allow access to all of a simulations output at each simulation time step. This is a powerful technique that is enabling new types of analysis and finer spatiotemporal resolutions than ever before. However, this newfound access to all of the simulation's data brings with it challenges in how to efficiently process that much data. With our workflow, we will show that in situ data summaries of a very large number of particles is possible using a data staging environment, enabling a new type of analysis for the simulation scientists.

This portion of our study used the particle data from XGC1. Particle data is the largest output dataset from this simulation. This data can be very large, generally ranging from over 900 GB to nearly 20 TB per time step. Furthermore, in order to get good bulk particle velocity vector fields as the particles move around the tokomak, we were required to access the particles at each time step of the simulation. Since this analysis routine is still being developed however, we did not run the simulation at full scale, instead we used smaller test scale run on 256 processors with 100,000 particles per core. Our final particle files were 4 GB. We do however relate our experience from this smaller run to how larger in situ runs will need to be handled in our conclusion.

3.1. Analysis Workflow

Our workflow consists of three primary elements: (1) the simulation code, (2) a data transfer system to move data from the simulation to the visualization nodes, and (3) an efficient parallel visualization library. This study was conducted on the Sith cluster at the Oak Ridge Leadership Computing Facility. Each of the 39 nodes in Sith contains four 2.3 GHz 8 core AMD Opteron processors and 64 GB of memory, configured with an 86 TB Lustre file system for scratch space. The components of the workflow that interact with XGC1 are described below, followed by a description of how the components of the workflow interact.

3.1.1. ADIOS Data Staging

The Adaptable I/O System (ADIOS) [17] is a componentization of the I/O layer accessible via a posix-style interface. The ADIOS API abstracts the operation away from implementation, allowing users to compose their applications independent of the underlying software and hardware. This capability, along with the functionality of DataSpaces [6], allows this same API to support read and write operations to and from the memory space of visualization staging nodes.

The loosely coupled paradigm in ADIOS and DataSpaces provides for a clean interface and separation from XGC1 that provides ease of use and fault tolerance. Also, this method allows the resource requirements for the visualization and staging tasks to be tailored for specific purposes.

For this study, we utilized two nodes on Sith to launch eight staging servers to handle the data connections from both the simulation and visualization nodes. Staging servers are used to store in memory the data coming from the simulation until the visualization routines call for it. This server configuration gave the best performance, and is easily scaled as more simulation or visualization nodes are added to the workflow.

3.1.2. Visualization Library

We designed our visualization routines as flexible, light weight plugins. Our plugins are based on the Extreme-scale Analysis and Visualization Library (EAVL) [20]. EAVL was developed to address three primary objectives: update the traditional data model to handle modern simulation codes, investigate the efficiency 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.

For this study, we ran our visualization routines on four nodes, with four processes per node. The number of visualization nodes can readily be scaled up as the data size increases so as not in slow down the simulation as each time step is processed.

3.1.3. Workflow Composition

Our visualization and analysis workflow combines three separate elements: the XGC1 simulation, the staging servers, and the visualization libraries. The workflow is launched as three separate binaries, with resources for each partitioned as follows: (1) 256 XGC1 processors; (2) 8 Staging servers; and (3) 16 visualization processors. Data flows through the workflow for every XGC1 time step as follows:

- When a new time step is ready from XGC1, it is immediately sent to our staging servers and subsequently consumed by the visualization routine. The visualization routine does a parallel read of the restart file, with each process taking $1/nProcs$ of the data. As consecutive time steps become available they are consumed.
- Next, the visualization routine statically maps particle ID's based on the visualization rank.
- Once each rank has the correct particles, a vector is computed between time step n and $n+1$. These vectors are then averaged onto an unstructured grid. Since the XGC1 mesh is very finely resolved, we use a coarser version of the mesh in this depositing step.

The resulting vector field is then written to disk for further analysis. It is important to note that this step is a major data reduction. As shown in Table 1, by performing the vector computation in situ, we are reducing the amount of data written to disk by between 95 and 476,190 times. This reduction factor is based on the output particle size of our Test-Scale Run, and the output size of a Large-Scale Run.

3.2. Results

Using the workflow setup described in the previous section, we are able to successfully create effective bulk plasma particle velocity vector fields for XGC1. We are in the early stages of the analysis

Table 1. Showing the output particle size in relation to the actual data size being written to disk at each simulation time step by performing the particle vector analysis in situ

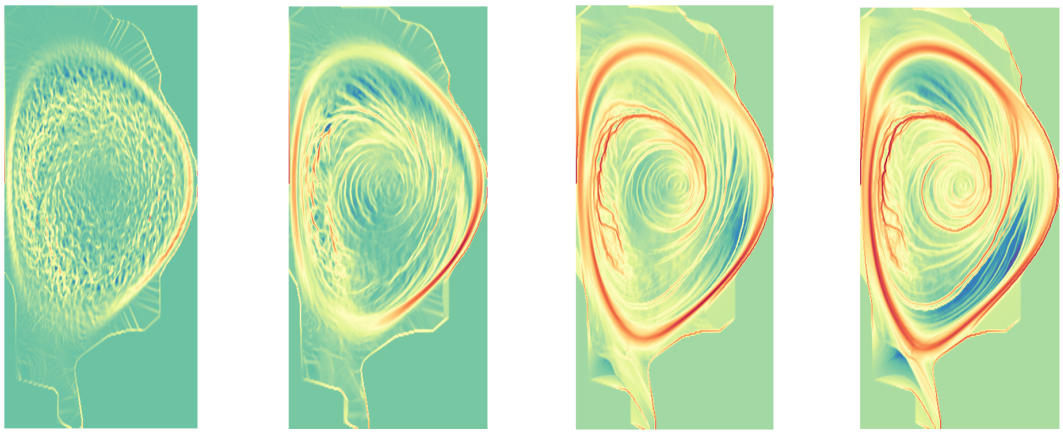
XGC1 Sim Run Size	Particle Size	Mesh Size	Reduction
Test-Scale Run	4 GB	42 MB	95x
Large-Scale Run	20 TB	42 MB	476,190x

of these new fields, so our main highlighted results are that this analysis is now feasible, as well as the timings that we have gathered from our different analysis runs. A few of the analysis routines that we plan to explore with the physicists using this data include streamlines, FTLE (Figure 2), Poincaré plots [29], and more.

From the runs we performed to create the effective bulk plasma particle velocity vector fields, we measured the amount of time used by the visualization routine, XGC1, and staging. Timings are an important metric for this simulation, as additional time impacts to the simulation are carefully scrutinized for production runs. From the timings that we gathered, we found that the impacts of the analysis routine on the XGC1 simulation are minimal. The only impact to the simulation occurs during the particle write at each time step. We are currently writing at each time step in order to achieve maximum temporal resolution with our vectors, but this number can be tuned with future runs as we perform further analysis.

We performed timings of the particle write step of XGC1 using the two currently possible mechanisms for accessing the particles for vector analysis: staging and file based. As shown in Table 2, we found that by staging the particles, we were able to get a 3.2x time reduction with staging versus writing the particles to disk. This meant that the simulation was impacted less, as it was paused for a shorter period of time with staging versus the file based method. It is important to note that the time to write the file to disk will exponentially increase as the run is scaled up to production sizes, while we believe that staging times will rise at a much lower rate.

Another important data point that we found is that we are able to easily complete the analysis portion of the pipeline during the time that XGC1 uses to compute a new time step, using only on average 35%



(a). Advection time 0.10 **(b).** Advection time 1.0 **(c).** Advection time 5.0 **(d).** Advection time 10.0

Figure 2. A slice of an FTLE plot using the effective bulk plasma particle velocity vector field. The shorter advection times demonstrate smaller scale features (plots a and b), while longer advection times bring larger scale features to light (plots c and d). The red values correspond to areas where the flow tends to separate, and blue is where the flow stays together

Table 2. Runtime of a single XGC time step using our test-scale run size when either sending data to staging or writing it to a file. The time variance is due to staging being much faster than the file based method, impacting the simulation less

XGC1 Simulation Time step Time (sec)		**XGC1 Simulation Particle Write Time (sec)**		
Staging	*File*	*Staging*	*File*	*Reduction*
96.6	102.8	1.8	5.8	3.2x

of that time. This is good news as we progress towards using production runs, as we should be able to scale the visualization nodes in order to keep up with each simulation time step, and even add other analysis tasks to the workflow to fill in idle analysis periods.

4. Data Reduction Through Reduced Precision

In this example, we motivate and explore the implications of using reduced representations of data for analysis, and the impacts on preservation of information, and the associated errors.

The errors associated with data reduction techniques, where f is the original data, and $\tilde{f}$ is the reduced data, are typically defined as follows:

$$E = |f(x,t) - \tilde{f}(x,t)| \tag{1}$$

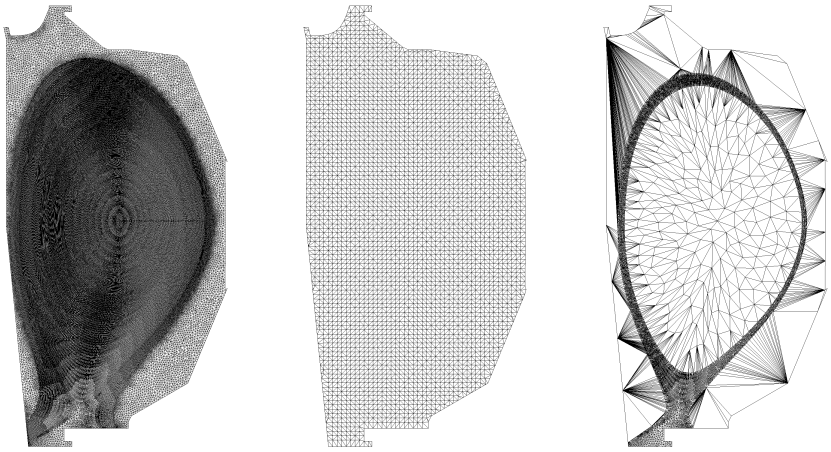
However, scientists will typically require derived quantities or operations, $G(f)$, on their data, and these derived quantities are not always known a priori. And so the errors that scientists care about becomes:

$$E = |G(f(x,t)) - G(\tilde{f}(x,t))| \tag{2}$$

These post-simulation analyses span a large space of possibilities. The simplest cases involve analysis and visualization of additional simulation variables, additional slice planes, subsets of data, or isovalues for contouring, as examples. More complicated examples involve transformations of the data, for example, changes of coordinate systems, different mappings, or analysis in different spaces such as the Fourier Transform. Derived variables can involve simple mathematical operations such as sum, difference, product, or division, or more complicated operations such as gradient, curl, divergence, and norms. More complex operations can include things such as feature detection and tracking, or particle advection for streamlines, pathlines, Poincaré plots, and Lagrangian Coherent Structures.

While there are a number of ways to reduce the size of scientific data, in this motivating example we focus on two lossy data reduction methods. The first is the floating point precision of the variable data, and the second is the spatial resolution of the underlying mesh. We also consider a combination of these two methods. In order to understand these methods in practice, we have applied these methods to primary variables from a simulation, derived variable calculations, feature detection, and more complex analysis operations.

Our work with reduced precision arose from our collaborations with the SIRIUS [11] project. The SIRIUS project is researching methods for the management and layout of large scientific data across the memory hierarchy of an HPC system. This layout might include breaking the data up into separate, dependent pieces. For example, 3 digits of precision in fast memory, and the rest of the precision in lower levels of the memory hierarchy. If needed, for particular operations, the extra precision can be combined with the lower precision representation.



(a). Full mesh.

(b). Uniform decimation of the full mesh.

(c). Adaptive decimation of the full mesh, preserving detail along the edge of the plasma.

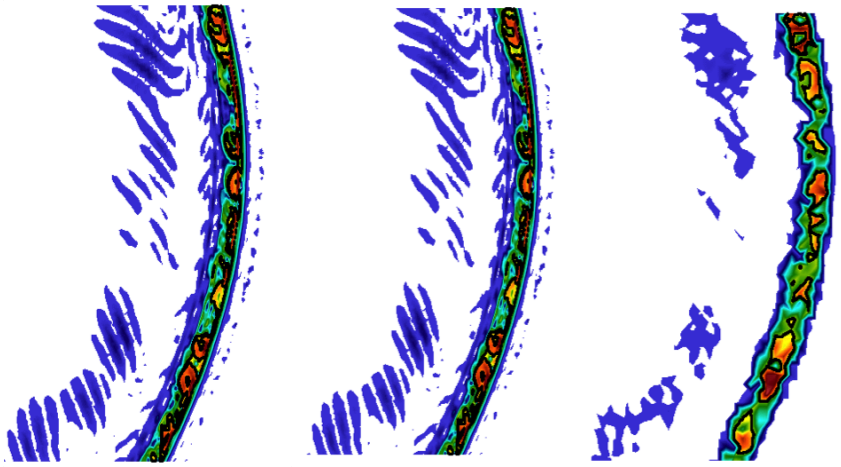
Figure 3. Different types of decimation for the simulation mesh in XGC1

Scientific simulations are typically done using double precision values, which are represented with 64 bits. While this level of precision is required for solving the equations in a simulation, it is typically not required for basic analysis and visualization operations. For example, when mapping the values of a variable through a color map, the resolution of the color map is often quite small compared to the range of the floating point values. The resulting pixels are then a quantized representation of the actual values. Because of this, very similar, or, in some cases, identical images can be produced from greatly reduced data. However, as stated previously, one under-explored area is the implications of these reduced precision data when more complex operations are applied to the data.

The spatial resolutions of the meshes for scientific simulations are determined by the convergence requirements of the underlying mathematics. This resolution may be appropriate for downstream processing of analysis and visualization operations, or it may be overkill. In this paper we explore two different types of mesh decimation: uniform and adaptive (see Figure 3). Adaptive decimation reduces the resolution of the mesh in a manner consistent with the underlying science. For example, in Figure 3(c), which is from an XGC1 fusion simulation, the plasma profiles near the edge region have sharp gradients, and so the resolution is higher in the edge region of the mesh to capture the fine-scale physics in this region. On the other hand, using uniform decimation, the mesh is reduced without this underlying knowledge. To adaptively decimate the mesh, we use a scheme based on quadric decimation [9]. For the error metric that drives the decimation, we use the proximity of mesh points to the edge region of the tokamak.

4.1. Results

To explore these ideas, we have been working with the XGC1 fusion simulation code. In this work, we are focusing on the field variables on the unstructured mesh in XGC1. The *primary* variables we are examining include the scalar potential (ϕ), and the magnetic field (B). We are interested in examining *derived* variables computed using mathematical expressions on these *primary* variables, as well as particle tracing through the magnetic field.



(a). Full data with 48 features

(b). Reduced precision data with same 48 features

(c). Reduced precision and mesh decimation with 13 features

Figure 4. Feature detection of reduced representations of a primary variable

Figure 4 shows an example of feature detection on the scalar potential (ϕ) in XGC1. In Figure 4(a), an edge detection algorithm has detected a set of 48 features from the full data set. In Figure 4(b), the same set of 48 features are identified using a 3 digit precision (5X data reduction) version of ϕ.

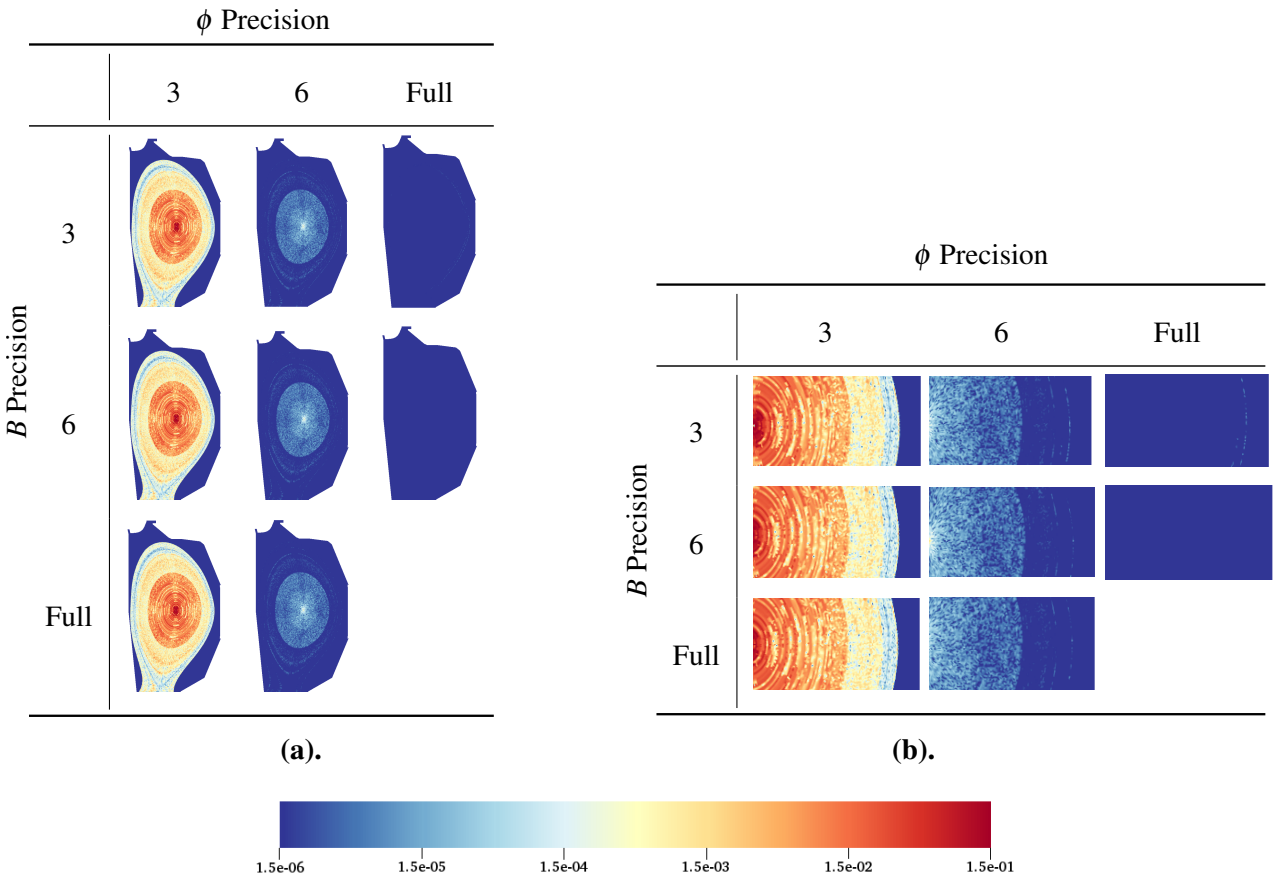


Figure 5. Relative error for fluid velocity using differing amounts of precision. Full cross section is shown in (a), and a zoomed in section in shown in (b)

In Figure 4(c), 3 digits of precision, and a $20X$ adaptively reduced mesh results in 13 features. The 3 digits of precision ($5X$ reduction) and $20X$ reduced mesh results in a total data reduction of $100X$. The question of whether or not the reduced data in Figure 4(c) provides an adequate representation of the underlying physics is something that we are in the process of evaluating with our collaborators in the XGC1 application team.

As an example of a derived variable, periodically scientists want to examine the perpendicular velocity of the plasma driven by the electric field (V_F), which is defined as follows:

$$V_F = \frac{\nabla\phi \times B}{|B|} \tag{3}$$

The relative error in the derived quantity of varying levels of precision in the primary variables is shown in Figure 5. Note that in this particular case, the error is clearly dependent on the precision of the variable ϕ. The precision of B has a much smaller impact on the errors in the derived variable. This leads to important insights about the relationship between errors and data reductions on variables, and can result in more efficient use of very limited data resources in an HPC system.

Relying solely on preserving scientific features in the primary variables can be problematic, as illustrated in Figure 6. In this example, by looking only at the reduced precision and decimated meshes for primary variables, ϕ and B, the main features appear to be preserved. However, when computing derived variables, in this case the fluid velocity, V_F, the features are *not* preserved using a uniform decimation scheme. This highlights the imperative to understand the implications of reduced data representations

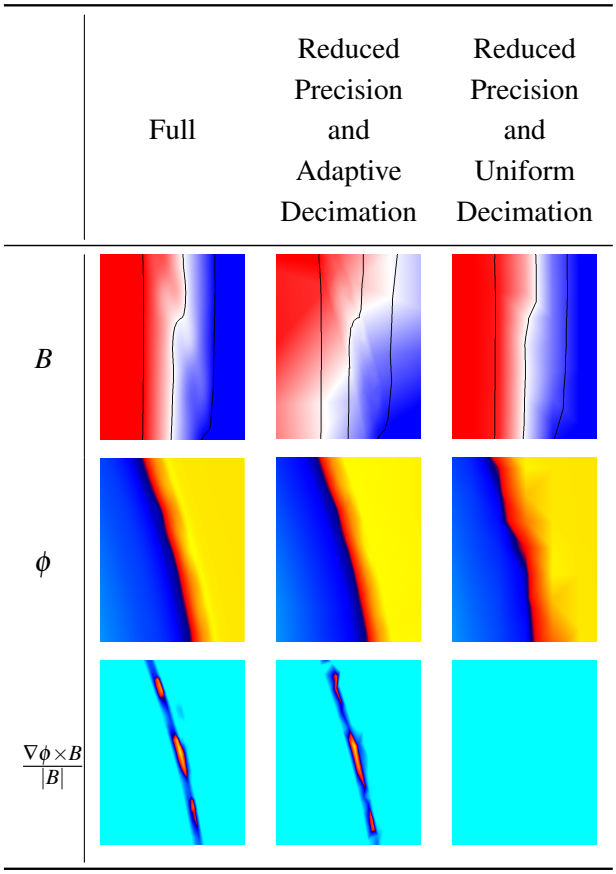


Figure 6. An example where features are preserved in the primary variables across different types of data reduction, but are lost when the derived variable is computed

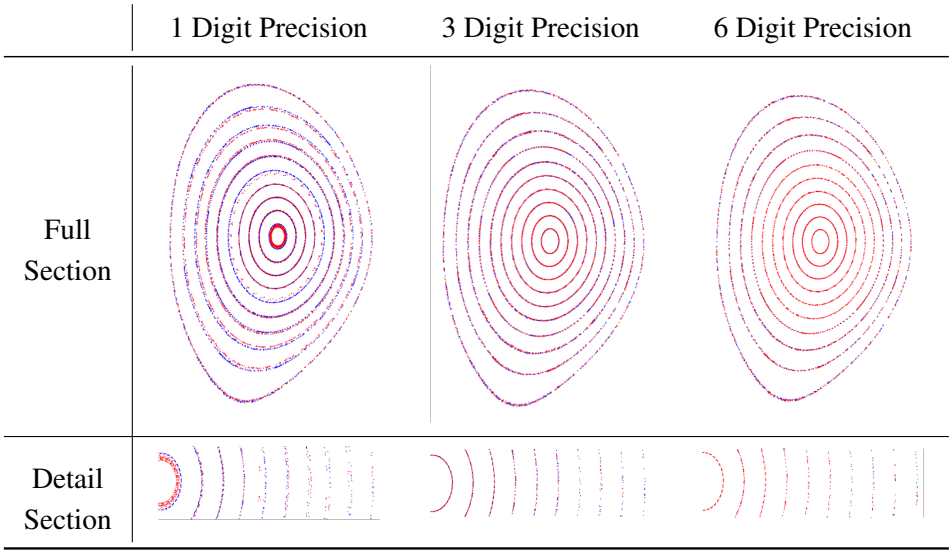


Figure 7. Poincaré plots using differing levels of precision in the magnetic field. The original data are shown in blue, and the reduced data are shown in red

on down stream analysis and visualization operations, and the unanticipated implications of reduced on data.

The magnetic field, B, is a fundamental driving mechanism in a fusion device. There are a number of ways to analyze and visualize the magnetic field in a fusion simulation, but one of the most common is by using particle advection based techniques. In particle advection methods, a set of massless particles are traced through the mesh by the vector field. As the magnetic field in a tokamak is cyclical, one common analysis technique is the Poincaré, or puncture plot. Each particle is traced through the vector field, and each time it intersects a plane perpendicular to the field, the intersection point is marked. Since each particle travels along a magnetic surface, after the particle has circulated enough, a 2D representation of the magnetic surface can be seen.

The images in Figure 7 show Poincaré plots of several reduced versions of the magnetic field. The full data are shown in blue, and the reduced data are shown in red. The first row shows the entire Poincaré plot, and the second row shows a zoomed in section. As shown in the far left column, a single digit of precision produces results with significant errors, particularly near the center of the plasma. However, using only three digits of precision produces results that are very good visual approximations of the full data.

Conclusions and Future Work

The growing disparity between compute and I/O on HPC systems will require simulation codes to radically alter how results are output and analyzed, and how scientific information is obtained. This transformation will require simulations to compute and output analysis and visualizations that are greatly reduced in size and complexity. These reduced outputs include a wide variety of data, including results that are more easily analyzed like images, movies, plots and graphs, as well as outputs that require post-hoc processing to understand, such as data summaries, and mesh-based data. At times, additional downstream processing is required in order to fully understand output results.

In situ and in transit processing methods will play a critical role for both categories of outputs mentioned above. For situations where a priori knowledge is available, these outputs should be computed in situ and output to disk. For situations where a priori knowledge is not available, and post-hoc analysis,

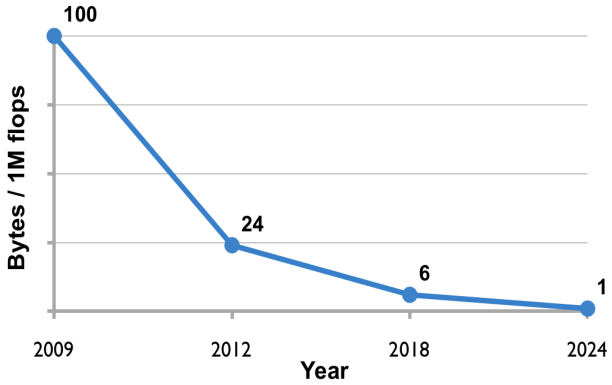


Figure 8. Current and expected I/O in bytes per 1 Million FLOPS. Data derived from forecasts on past, current and future HPC systems [12, 25, 31]

visualization, and processing are required, in situ and in transit methods will play a crucial role in producing integrity preserving reduced data that can be analyzed later. Because all the data are available, in situ methods play a crucial role in performing data reductions with specified error bounds. These error bounds should be valid for the simulation variables which are saved, as well as for post-hoc operations on data that are anticipated on the scientific workflows associated with the simulation.

While the growing disparity between compute and I/O is a reality, there is still significant I/O capability in current and future HPC systems, and efforts should be taken to ensure that a maximum amount of information is saved. The challenge is determining the proper set of data to be saved. As shown above in Figure 8, the expected trend for I/O is one byte per one million floating points operations on an exascale computer. The challenge then becomes to make sure that each byte written accurately represents the information produced from the one million floating point operations. In situ methods will play a critical role in determining the proper byte to be output. In transit methods, where data staging is used, will also play an important role in providing asynchronous computational capability. Additionally, because data staging nodes use a separate set of resources, analysis and visualization algorithms that require communication can operate on reduced data much more efficiently, and with a limited impact on the simulation.

Because of the breadth of issues involved, solutions to these problems will require collaborative research between a number of disciplines, including applied mathematics, analysis, visualization and middleware.

From a visualization perspective, a better understanding is required of the errors bars associated with operations on data, and how these errors propagate through visualization and analysis workflows. The data models for visualization tools needs to be expressive enough to represent data in new and unique forms. For example, native operations on data streams with compressed, or reduced precision data, efficient operations for variables that are on different meshes.

To address the widening of the memory hierarchy on HPC systems, projects like SIRIUS, in conjunction with middleware systems like ADIOS, are working to optimize the placement of data. SIRIUS aims to place the most valuable data in memory locations that are easily accessible, and data that are less important are pushed down the memory hierarchy, and eventually to long term storage systems, such as tape. Collaborations with this type of system would involve reduced, approximated representations in faster memory and the ability to pull up increasingly more accurate representations of the data as needed. Analysis and visualization processes should work seamlessly in these types of environments to operate on, and meet the error bars required by scientists. As such, developing the appropriate interfaces between these different layers is an important research direction.

We have demonstrated the advantage of these techniques through our work with the particle and mesh data in the XGC1 fusion code. We have focused on techniques for reducing the data in ways that preserve information for the scientists. For the particles we have shown how in transit processing can be used to compute the bulk velocity field, which is a reduced representation of the particle data. For mesh variables we have shown that care must be taken to reduce data in ways that preserve scientific information. We have focused on eddy's and features in the potential and fluid velocity field, and magnetic field analysis using streamlines and Poincaré methods.

In Section 3 we have shown how data staging can be used to perform data reductions on particle data in XGC1. Data staging proves useful in two distinct ways. First, the non-trivial data reduction task can be performed asynchronously, and not interfere with the simulation. Second, the communication required to derive the bulk velocity field can be done much more efficiently on a smaller set of nodes. We have developed these techniques using small runs of XGC1 on a moderately sized cluster. These small runs allowed us to operate on all of the simulation particles. A full production run of XGC1 on an HPC system would produce significantly more particles than could processed in a staging environment. For a full production run a subset of particles would be required that is statistically equivalent to the entire set of particles. In particular, given a total of N simulation particles, we need a query that will return a set of M (where $M << N$) particles uniformly distributed within the spatial extents of the simulation. Because the particles move at each time step, the representative set of particles must be periodically recomputed. As a result we will need methods for determining the quality of the particle subset, and deciding when it is required to recompute. We are currently collaborating with the ADIOS project to develop such particle queries in order to facilitate production runs on HPC systems. We are also working to determine the appropriate interface between the visualization tools and the middleware for these types of operations.

In Section 4 we have shown how reduced representations of XGC1 data can be used in visualization, and the unforseen issues that arise on derived quantities. Resolution of these issues will require a significantly better understanding of data reductions, and the propagation of errors. To formalize these ideas, we must understand the mathematical implications of these reductions, and be able to provide error bounds on the use of these reduced data under a variety of operations. These include simple operations like sum, difference, product and division, as well more complex operations like cross product, gradient, and curl. With a better understanding of the relationship between errors on input data, and operations in complex workflows, scientists can accurately specify the requirements for their analyses, and be confident in the results derived using reduced data.

From an analysis and visualization perspective, we need data models that provide the flexibility to operate on reduced data in a zero-copy paradigm. Tradeoffs exist between converting reduced data streams to floating point data and performing on-the-fly conversions as data are used. We are currently exploring how the data model in VTK-m [21] [22] [24] can be used to address these issues and explore the tradeoffs of reduced data size and efficiency of computations, particularly as it relates to computations on accelerators.

We have shown the clear benefits to using an adaptive decimation technique for XGC1. We are actively exploring techniques for other codes, as well as more generalized methods. When performing operations with variables on different meshes, collaborative research with applied mathematics is needed on how best to calculate the derived quantities in a way that minimizes the error. Solutions might be to interpolate from one mesh to another, or to derive a new mesh that meets the error requirements.

Finally, all of this work takes place in context of a middleware system that manages and coordinate the movement of data within the HPC system. We are actively collaborating with the ADIOS and SIRIUS

projects that provide methods to manage data across the the memory hierarchy, as well as the data staging capabilities in order to provide asynchronous processing. We are also collaborating with these projects to explore the proper interfaces between analysis and visualization components, and the middleware system components.

Acknowledgements

This work was supported by the Scalable Data Management, Analysis, and Visualization (SDAV) which is funded by the DOE Office of Science through the Office of Advanced Scientific Computing Research (Contract No. 12-015215).

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Sean Ahern, Arie Shoshani, Kwan-Liu Ma, Alok Choudhary, Terence Critchlow, Scott Klasky, Valerio Pascucci, J Ahrens, EW Bethel, H Childs, et al. Scientific discovery at the exascale. *Report from the DOE ASCR 2011 Workshop on Exascale Data Management*, 2011.

2. CS Chang, S Ku, PH Diamond, Z Lin, S Parker, TS Hahm, and N Samatova. Compressed ion temperature gradient turbulence in diverted tokamak edgea). *Physics of Plasmas (1994-present)*, 16(5):056108, 2009.

3. Hank Childs, David Pugmire, Sean Ahern, Brad Whitlock, Mark Howison, Prabhat, Gunther H. Weber, and E. Wes Bethel. Extreme scaling of production visualization software on diverse architectures. *IEEE Comput. Graph. Appl.*, 30(3):22–31, May 2010.

4. Hank Childs, David Pugmire, Sean Ahern, Brad Whitlock, Mark Howison, Prabhat, Gunther H. Weber, and E. Wes Bethel. Visualization at extreme scale concurrency. In E. Wes Bethel, Hank Childs, and Charles Hansen, editors, *High Performance Visualization: Enabling Extreme-Scale Scientific Insight*. CRC Press, Boca Raton, FL, 2012.

5. Jong Y Choi, Kesheng Wu, Jacky C Wu, Alex Sim, Qing G Liu, Matthew Wolf, C Chang, and Scott Klasky. Icee: Wide-area in transit data processing framework for near real-time scientific applications. In *4th SC Workshop on Petascale (Big) Data Analytics: Challenges and Opportunities in conjunction with SC13*, 2013.

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

7. Nathan Fabian, Kenneth Moreland, David Thompson, Andrew Bauer, Pat Marion, Berk Geveci, Michel Rasquin, and Kenneth Jansen. The paraview coprocessing library: A scalable, general purpose in situ visualization library. In *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*, pages 89–96. IEEE, 2011.

8. Gzip compression. `http://www.gzip.org`.

9. Hugues Hoppe. New quadric metric for simplifying meshes with appearance attributes. In *Proceedings of the 10th IEEE Visualization 1999 Conference (VIS '99)*, VISUALIZATION '99, pages –, Washington, DC, USA, 1999. IEEE Computer Society.

10. D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, Sept 1952.

11. Scott Klasky, Hasan Abbasi, Mark Ainsworth, Qing Liu, Jay Lofstead, CArlos Maltzahn, Manish Parashar, and Feyi Wang. Sirius: Science-driven data management for multi-tiered storage. *Proceedings of the XXVII IUPAP Conference on Computational Physics*, December 2015.

12. Kalyan Kumaran. Introduction to Mira. `https://www.alcf.anl.gov/files/bgq-perfengr.pdf`. Visited June 20, 2016.

13. Sriram Lakshminarasimhan, Neil Shah, Stephane Ethier, Scott Klasky, Rob Latham, Rob Ross, and Nagiza F. Samatova. Compressing the incompressible with isabela: In-situ reduction of spatio-temporal data. In *Proceedings of the 17th International Conference on Parallel Processing - Volume Part I*, Euro-Par'11, pages 366–379, Berlin, Heidelberg, 2011. Springer-Verlag.

14. P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, Dec 2014.

15. P. Lindstrom and M. Isenburg. Fast and efficient compression of floating-point data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1245–1250, Sept 2006.

16. Peter Lindstrom, Po Chen, and En-Jui Lee. Reducing disk storage of full-3d seismic waveform tomography (f3dt) through lossy online compression. *Computers & Geosciences*, 93:45 – 54, 2016.

17. Qing Liu, Jeremy Logan, Yuan Tian, Hasan Abbasi, Norbert Podhorszki, Jong Youl Choi, Scott Klasky, Roselyne Tchoua, Jay Lofstead, Ron Oldfield, Manish Parashar, Nagiza Samatova, Karsten Schwan, Arie Shoshani, Matthew Wolf, Kesheng Wu, and Weikuan 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.

18. Jay F. Lofstead, Scott Klasky, Karsten Schwan, Norbert Podhorszki, and Chen Jin. Flexible io and integration for scientific codes through the adaptable io system (adios). In *Proceedings of the 6th international workshop on Challenges of large applications in distributed environments*, CLADE '08, pages 15–24, New York, NY, USA, 2008. ACM.

19. David P. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Comput. Graph. Appl.*, 21(3):24–35, May 2001.

20. Jeremy S Meredith, Sean Ahern, Dave Pugmire, and Robert Sisneros. EAVL: the extreme-scale analysis and visualization library. In *Eurographics Symposium on Parallel Graphics and Visualization*, pages 21–30. The Eurographics Association, 2012.

21. Jeremy S Meredith, Sean Ahern, Dave Pugmire, and Robert Sisneros. EAVL: the extreme-scale analysis and visualization library. In *Eurographics Symposium on Parallel Graphics and Visualization*, pages 21–30. The Eurographics Association, 2012.

22. Jeremy S. Meredith, Robert Sisneros, David Pugmire, and Sean 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.

23. Kenneth Moreland, Ron Oldfield, Pat Marion, Sebastien Jourdain, Norbert Podhorszki, Venkatram Vishwanath, Nathan Fabian, Ciprian Docan, Manish Parashar, Mark Hereld, et al. Examples of in transit visualization. In *Proceedings of the 2nd international workshop on Petascale data analytics: challenges and opportunities*, pages 1–6. ACM, 2011.

24. Kenneth Moreland, Christopher Sewell, William Usher, Lita Lo, Jeremy Meredith, David Pugmire, James Kress, Hendrik Schroots, Kwan-Liu Ma, Hank Childs, Matthew Larsen, Chun-Ming Chen, Robert Maynard, and Berk Geveci. VTK-m: Accelerating the Visualization Toolkit for Massively Threaded Architectures. *IEEE Computer Graphics and Applications (CG&A)*, 36(3):48–58, May/June 2016.

25. Lucy Nowell. Science at extreme scale: Architectural challenges and opportunities, 2014. `http://www.mcs.anl.gov/~hereld/doecgf2014/slides/ScienceAtExtremeScale_DOECGF_Nowell_140424v2.pdf`.

26. R. A. Oldfield, P. Widener, A. B. Maccabe, L. Ward, and T. Kordenbrock. Efficient data-movement for lightweight i/o. In *2006 IEEE International Conference on Cluster Computing*, pages 1–9, Sept 2006.

27. David Pugmire, James Kress, Hank Childs, Matthew Wolf, Greg Eisenhauer, Randy Churchill, Tahsin Kurc, Jong Choi, Scott Klasky, Kesheng Wu, Alex Sim, and Junmin Gu. Visualization and analysis for near-real-time decision making in distributed workflows. In *High Performance Data Analysis and Visualization (HPDAV) 2016 held in conjuction with IPDPS 2016*, May 2016.

28. David Pugmire, James Kress, Jeremy Meredith, Norbert Podhorszki, Jong Choi, and Scott Klasky. Towards scalable visualization plugins for data staging workflows. In *Big Data Analytics: Challenges and Opportunities (BDAC-14) Workshop at Supercomputing Conference*, November 2014.

29. Allen Sanderson, Guoning Chen, Xavier Tricoche, David Pugmire, Scott Kruger, and Joshua Breslau. Analysis of recurrent patterns in toroidal magnetic fields. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1431–1440, 2010.

30. Roselyne Tchoua, Jong Choi, Scott Klasky, Qing Liu, Jeremy Logan, Kenneth Moreland, Jingqing Mu, Manish Parashar, Norbert Podhorszki, David 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.

31. Patrick Thibodeau. Coming by 2023, an exascale supercomputer in the U.S. `http://goo.gl/qKTa8q`. Visited June 20, 2016.

32. V. Vishwanath, M. Hereld, and M.E. Papka. Toward simulation-time data analysis and i/o acceleration on leadership-class systems. In *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*, pages 9–14, 2011.

33. Brad Whitlock, Jean M Favre, and Jeremy 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*, pages 101–109. Eurographics Association, 2011.

34. J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, May 1977.