Exacution: Enhancing Scientific Data Management for Exascale

Scott Klasky*†‡, Eric Suchyta*, Mark Ainsworth§*, Qing Liu¶, Ben Whitney§, Matthew Wolf*, Jong Choi*, Ian Foster||, Mark Kim*, Jeremy Logan†, Kshitij Mehta*, Todd Munson||, George Ostrouchov*, Manish Parashar***, Norbert Podhorszki*, David Pugmire*, Lipeng Wan*

* Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
† Department of Electrical Engineering and Computer Science, The University of Tennessee Knoxville, Knoxville, TN 37996, USA
‡ School of Computer Science, Georgia Institute of Technology, Atlanta, GA 30332, USA
§ Division of Applied Mathematics, Brown University, Providence, RI 02912, USA
¶ Department of Electrical and Computer Engineering, New Jersey Institute of Technology, University Heights Newark, NJ 07102, USA
** Rutgers Discovery Informatics Institute, Rutgers University, Piscataway, NJ 08854, USA
|| Argonne National Laboratory, Argonne, IL 60439, USA

***Abstract*—As we continue toward exascale, scientific data volume is continuing to scale and becoming more burdensome to manage. In this paper, we lay out opportunities to enhance state of the art data management techniques. We emphasize well-principled data compression, and using it to achieve progressive refinement. This can both accelerate I/O and afford the user increased flexibility when she interacts with the data. The formulation naturally maps onto enabling partitioning of the progressively improving-quality representations of a data quantity into different media-type destinations, to keep the highest priority information as close as possible to the computation, and take advantage of deepening memory/storage hierarchies in ways not previously possible. Careful monitoring is requisite to our vision, not only to verify that compression has not eliminated salient features in the data, but also to better understand the performance of massively parallel scientific applications. Increased mathematical rigor would be ideal, to help bring compression on a better-understood theoretical footing, closer to the relevant scientific theory, more aware of constraints imposed by the science, and more tightly error-controlled. Throughout, we highlight pathfinding research we have begun exploring related these topics, and comment toward future work that will be needed.**

I. Introduction

As scientific projects continue to get larger, more complex, and more data-intensive, high performance scientific data management is becoming more important to a larger community. However, it is a challenging endeavor, presenting a number of difficulties derived from both the domain science and computer science aspects of the problem. Supercomputing environments are ever-changing, and solutions should be mindful of this, to benefit from the latest technologies and to harness the full potential of current-to-future leadership computing facilities. We begin with a discussion of ongoing data management challenges, noting hardware trends, then present ideas for new techniques to address the problems.

Not to be overlooked is that scientific data is rich and diverse; it comes in many forms, where very different physical scales, subtle sensitivities to various operators, or correlations among variables may be crucial. Related, the *utility* of different pieces of data, driven by their intended uses and lifetimes, is starkly varied. For example, quantities needed for statistical inference over the experiment's variables may be needed at high precision for the duration of entire campaigns, while other data for near real-time diagnostics or visualization might only be needed at low quality for short times, but at high frequencies.

Scientific data management, particularly in high performance simulations, entails storing substantial sized datasets as intermediate outputs before analyzing them for insight. Complicating storage is the *data-compute gap*, shown in Fig. 1. The current trend among leadership class machines is that increases in the bandwidth to storage systems are comparatively much smaller than the growth in the computational throughput (e.g. [1]–[3] and references therein), and this trend is expected to continue[1]. Thus, there is a growing disparity between how much can be computed and how much can be saved. Increased FLOPS enable scientists to calculate previously unstudied, more computational demanding components of the problem at hand. However, if I/O is not to become more of a bottleneck, a reduced ratio in the number of bits/FLOP that can be saved means that more of the data generated during computation would be lost, possibly preventing diagnostic output and degrading the sensitivity of the scientific analyses.

Recently, high-throughput, lossy compression has been gaining traction as a means to bridge the data-compute gap.

[1]The following is an outlook from Lucy Nowell, the Department of Enerergy's Advanced Scientfic Computing Research program manager: http://www.mcs.anl.gov/~hereld/doecgf2014/slides/ScienceAtExtremeScale_DOECGF_Nowell_140424v2.pdf.

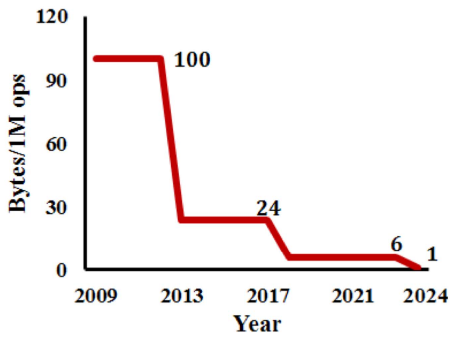

Fig. 1. Data-compute gap. Plotted is how the total filesystem throughput of leadership class facilities compares to the total number of floating point operations per second that can be perfomed. The I/O throughput is not scaling as quickly as computational speed.

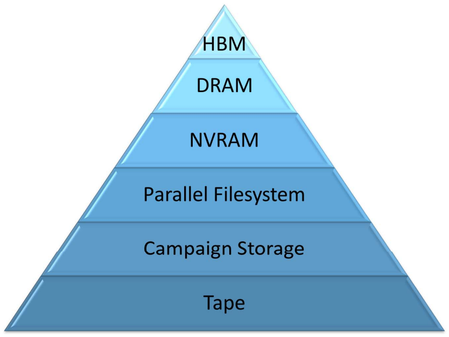

Fig. 2. Example memory/storage layout for future computing systems. Throughput increases toward the top and capacity increases toward the bottom. (HBM = high bandwidth memory, DRAM = dynamic random access memory, NVRAM = non-volatile random access memory)

Indeed, several compression algorithms have generated interest for potential use within high performance applications, including ISABELA [4], ZFP [5], SZ [6], and parallel tensor decomposition [7], among others. The idea is to write fewer bits, but retain adequate *information* content not to impact analysis. However, given the breadth of scientific data, with very different salient features necessary to retain in different applications, guaranteeing adequate quality for derived quantities in the general case is no easy matter.

Looking forward, an important direction to consider is that memory and storage hierarchies are deepening, offering more levels of progression of speed-size tradeoff in computing environments. Fig. 2 shows a simple illustration: with faster layers toward the top, and the slower, larger capacities ones toward the bottom. High bandwidth memory (HBM) is emerging with GPU devices, offering higher throughput and lower power consumption than more traditional dynamic random-access memory (DRAM) for CPUs. (See e.g. the work of Lee et al. [8] as an example of recent progress.) Furthermore, non-volatile random-access memory (NVRAM) is now common on personal computers, and becoming more available at scale: e.g. the burst buffer at NERSC on Cori[2] or the one planned for Summit[3] at the Oak Ridge Leadership Computing Facility. These solid state devices (SSDs) are intended to accelerate high-priority I/O operations, and offer performance advantages over the usual parallel filesystems (e.g. Lustre [9], GPFS [10]) alone. Finally, campaign storage or tape (HPSS [11]) can accommodate the most data and still factors in when archiving or backing up data for very long periods of time.

To follow, we lay out a vision for scientific data management, which is forward-looking toward leveraging deepening storage hierarchies to emphasize data utility level. It shifts away from the paradigm of thinking of data quantities in terms of a single type (usually 64-bit floating point numbers), in favor of offering multiple "views" of the data, available at different precision and latency. Compression plays an important role, as a means to help overcome the data-compute gap. However, it needs to be monitored closely and developed cooperatively with scientific applications, to make sure not to inadvertently reduce-away vital features. More generally, we advocate an increased emphasis on mathematical awareness and rigor as the field of scientific data compression continues to gain further traction. Overall, our goal is to facilitate a faster, easier, more flexible path to insight, which fully capitalizes on forthcoming new technologies.

The document is organized as follows. Section II further exemplifies current-to-next generation high performance scientific applications. To this point, the discussion has been rather general, and it is beneficial to add more concrete, real-world details. Section III describes the idea of organizing scientific data in a progressively refined way, with examples from preliminary work we have undertaken. Section IV focuses on achieving robust, efficient compression in scientific data, again highlighting pathfinding work. Section V concludes.

II. Motivating Examples

To elucidate some of the needs of current-to-future generation high performance scientific applications, we offer two illustrative examples: one from plasma physics and the other from combustion science, both relevant within the context of the Exascale Computing Project[4] (ECP). Most CPU-hours on leadership class machines are devoted to large-scale simulations, and both workflows include simulations. However, experimental data is also important, and the second touches on this category as well. The two examples help motivate the visions for scientific data management that we will describe in Section III and Section IV, and data products described in Section II-A are used throughout the paper to exemplify relevant work.

A. High-Fidelity Tokamak Whole Device Modeling

ITER[5] is currently under construction, whose intent is to demonstrate the capability to harness energy from nuclear

[2]http://www.nersc.gov/users/computational-systems/cori/burst-buffer/

[3]https://www.olcf.ornl.gov/kb_articles/summitdev-quickstart/#System_Overview

[4]https://exascaleproject.org/

[5]https://www.iter.org/

fusion, with a ten-fold power yield compared to the input operating power. ITER and next-step machines will operate under regimes never previously achieved, and predictive numerical simulations are needed to help design future experiments. Tokamaks like ITER have toroidal designs, where the plasma is confined by a strong magnetic field, whose direction is nearly aligned with the toroidal axis. However, turbulence driven by micro-instabilities leads to disruptions in the plasma that dissipate the confinement. Tokamak simulations allow us to better understand the turbulent behavior.

Different phenomena present modeling challenges in either the inner core or the outer edge of the tokamak, and current-day simulations usually only accurately predict one of the two regimes. ECP's whole device modeling project[6] targets the first ever high-fidelity full-tokamak simulation framework, by self-consistently coupling two individually developed applications: the Gyrokinetic Plasma Turbulence Code (GENE) [12]–[14], a continuum code that has been optimized for the core, and X-point included Guiding Center (XGC) [15], [16], a particle-in-cell code that has been designed to study the outer edge. Both solve the same underlying gyrokinetic equations, but using quite different strategies. Developing a mathematically robust, stable coupling approach is a challenge itself, but here we focus on the data management aspects of the problem.

A run at modern-day leadership scale ($\gtrsim$ 1M CPU-hours) would consist of GENE and XGC running simultaneously, sharing one or more quantities (e.g. the plasma distribution function) back and forth in an overlap region between the core and the edge. The exchange will be frequent, possibly needing to occur as often as *every* time step ($\sim$ seconds walltime), and must be fast enough not to significantly impact performance. Furthermore, both codes must periodically save large checkpoint/restart files to disk, $\sim$10 TB/hour in the larger-size case of XGC. Not all this data can be kept on the parallel filesystem and needs to be migrated elsewhere. Additionally, other diagnostic quantities (such as electromagnetic potentials) must be saved to yield any insight from the run, as they are the data products used for downstream scientific studies. These have a much smaller data volume per time step, but are written at a higher frequency. As much output as possible is best, as it allows either more precise versions of existing analyses or enables otherwise impossible ones, dependent on a previously unwritten variable. Ideally, in situ analysis/visualization would be enabled as well, for near real-time monitoring of the run, especially throughout early stages of the coupling development, while identification of the most robust mathematical approach to the coupling is still under investigation. From a scientific data management perspective, it is a challenge with both high volume and high frequency I/O operations, and the problem will continue to compound as computational power continues to outgrow I/O throughput.

[6]https://exascaleproject.org/2016/11/05/high-fidelity-whole-device-model-of-magnetically-confined-fusion-plasma/

B. Experimental+Simulation Combustion Science

The typical workflow in the combustion community is highly collaborative. They often consists of one or more groups of experimentalists, experimental data analysts, modelers, computationalists, and project coordinators. These teams are typically composed of specialized groups built to solve the specific problem at hand, and they disband and re-form with different participants as new projects arise. The scientific goals of such teams are to develop models of the combustor physics that scientist can understand then use to make predictions, and often to construct physics-based engineering tools. The work cycle to develop these models/tools requires detailed quantitative measurements of the combustor physics, many of which can be experimentally measured, but some of which cannot. The physics that cannot be experimentally measured is computed in high fidelity simulations, which rely on experimental data for boundary conditions and validation/refinement.

Cutting-edge combustion programs largely focus on research and development of reduced-emissions combustion systems. The work relies heavily on fluid dynamics measurements in combustors, using state of the art laser-based diagnostics, such as particle image velocimetry (PIV). The PIV measurement consists of seeding a combustor flow with small tracer particles, illuminating the small particles with short-duration, rapid bursts of a planar laser light sheet, and then photographing the illuminated particles at a high rate. A single experiment in a campaign might have 5-10 cameras capturing full-resolution images at >10,000 frames per second. These series of photographs are then analyzed by an algorithm that uses the space-time history of the tracer particles to compute velocity vector fields within the illumination plane, potentially using multi-camera viewpoints to deduce 3D flow information. The PIV algorithm is computationally expensive, and its successful implementation relies on many critical tuning parameters as well as very high quality photographs.

In order for this work cycle to proceed to its scientific goals, the experimental campaign must be completed and validated across all the various cameras and sensors, and its results must be reduced and efficiently distributed. This creates a highly varied data problem, with several different types of files (images, simulation checkpoints, analysis output) to manage, as well as timeliness concerns. There are also significant difficulties in terms of understanding data cleanliness; on the experimental side, data may suffer from a number of possible failures, such as misalignment of lasers or failure of the particle injector to get a uniform distribution. However, all this complexity is necessary to enable the intended collaborative science workflow, which is to allow the leadership-scale simulation results to help interpret, analyze, and inform the experimental campaigns, as well as vice versa.

Here, we identify another complication for future scientific data management. The length and time scales that are accessible with modern high speed cameras are still almost impossibly long by simulation time frames. Cutting-edge research is studying how to best calculate observable properties from the millisecond-scale time frame of the experiments that are relatable to properties that can be evaluated in the 10's to 100's of nanoseconds that exist in full simulation runs. For a direct numerical simulation (DNS) chemistry code like S3D [17], [18], this means calculating temporal and spatial autocorrelation functions, as well as computational geometries and more, over the 10's and 100's of TBs of data that come from the multiple simulation outputs.

III. Progressively Refined Data Organization

As exemplified above, production-scale scientific datasets are already large. Exascale datasets will only continue to grow in size. Reading in such data is potentially time consuming and wasteful, especially if more than is actually required is returned. In this section, we describe new strategies for refactoring scientific data to attempt to make the read process more seamless, highlighting preliminary work we have undertaken, discussing several challenges presented, and citing examples for context.

A. Motivation

One familiar read scenario is two-dimensional images. When trying to visualize this data, one does not necessarily need information from every bit in every pixel to reconstruct an image good enough for human consumption. If *quality of service* is too low, and constructing the full resolution version will take too long, users could benefit from the option of relaxing the full quality demand, and instead returning a degradation of the original in a shorter time, where the degradation may not even be visually perceptible. This is the main advantage of the JPEG 2000 standard [19] compared to the original standard [20]; JPEG 2000 uses decimation and a wavelet-based compression method, such that truncation of the codestream at any position still yields a signal that can be decoded into an image resembling the original, at the penalty of reduced resolution for increased truncation. Developed at a time when network bandwidth was much lower than today, this was a natural enhancement to consider.

In our case, we are targeting scientific data instead of JPEG images, and it is filesystem bandwidth that can be at a premium. Contention with other jobs causes variation in the quality of service delivered by the filesystem, and it would be ideal to be able to access different quality views of the data depending on this contention, analogous to how services like Netflix are able to deliver reduced-quality streams when there are bandwidth interruptions from the internet service provider, instead of imposing long buffering times.

What we seek is *progressive refinement* of the data – a scheme presenting variable-fidelity versions of the data, at different quality resolutions. ViSUS [21] and related projects [22], [23] in the visualization community have undertaken efforts to define one such scheme for scientific data on grids, using hierarchical Z-ordering [24] to to efficiently subsample the data to grids courser than the full original. We are currently developing our own progressive organization schemes. We are interested in techniques beyond grid decimation alone to enable more flexibility in how to define which information has priority. Eventually, we will need to support several methods, as different scientific application will have different requirements. We will make related comments about this topic in Section IV. We are also mindful that the data's reorganization should be aware of the possibility of multi-tiered storage to fully support current and future storage systems.

B. Refactoring Progress

Initial work we have begun exploring includes two refactoring techniques: a "precision-based" approach that utilizes adjustable floating point compression for each data value, and "feature-based" approach that organizes the data into bins according to some calculation over the grid values. Fig. 3 and Fig. 4 show examples using data from XGC, one of the ECP fusion applications we discussed in Section II-A. Choosing a single time step, we plot deviations from background in the electric potential, one of the many grid quantities calculated by XGC. Rotating the plane about $x = 0$ would generate the shape of the torus.

The precision-based approach is a modification of ZFP, a high-throughput lossy compression algorithm (with variance-bounded errors) for floating-point arrays, whose encoder applies a highly-optimized, nearly-orthogonal transformation, analogous to the discrete cosine transform [5]. A full treatment of the technical details of our modifications is beyond the scope of this paper, and will be included as part of a publication in preparation. Here, we summarize conceptually. We have extended ZFP such that the encoded signal can be split into N progressive streams, which are all saved to different files. Reading additional files beyond the first "appends" more accuracy, further reducing the error upon decompression. Importantly, the compression kernel only runs a single time during the encoding process, and no duplicate data is written multiple times. Fig. 3 shows an example using an initial stream with 4 bits/value precision, and a second stream adding 8 more bits/value. A user now has access to progressive levels of compression, without having to run ZFP independently for each level.

Our feature-based approach calculates the data vector's gradient over the grid, and specifies that cells whose gradient values are large enough in magnitude will be grouped into a priority level, with the values over that portion of the grid saved exactly. The rest of the grid is decimated. Different levels of decimation and/or gradient magnitudes form the progressive-quality versions of the data. Reusing the same dataset as Fig. 3, the middle panel of Fig. 4 shows the XGC data from the left panel with $|\nabla(\Delta\phi)| > 0.003$ (where $\Delta\phi$ was normalized between -1 and 1), decimating to a factor of 4 compression. The right panel only decimates to a factor of 2 compression. The gradient threshold is such that a little more than one percent of the area has been retained exactly. Here, the gradient serves as an example placeholder operator to identify and retain high variation regions in the data as important, though more generally one could define other

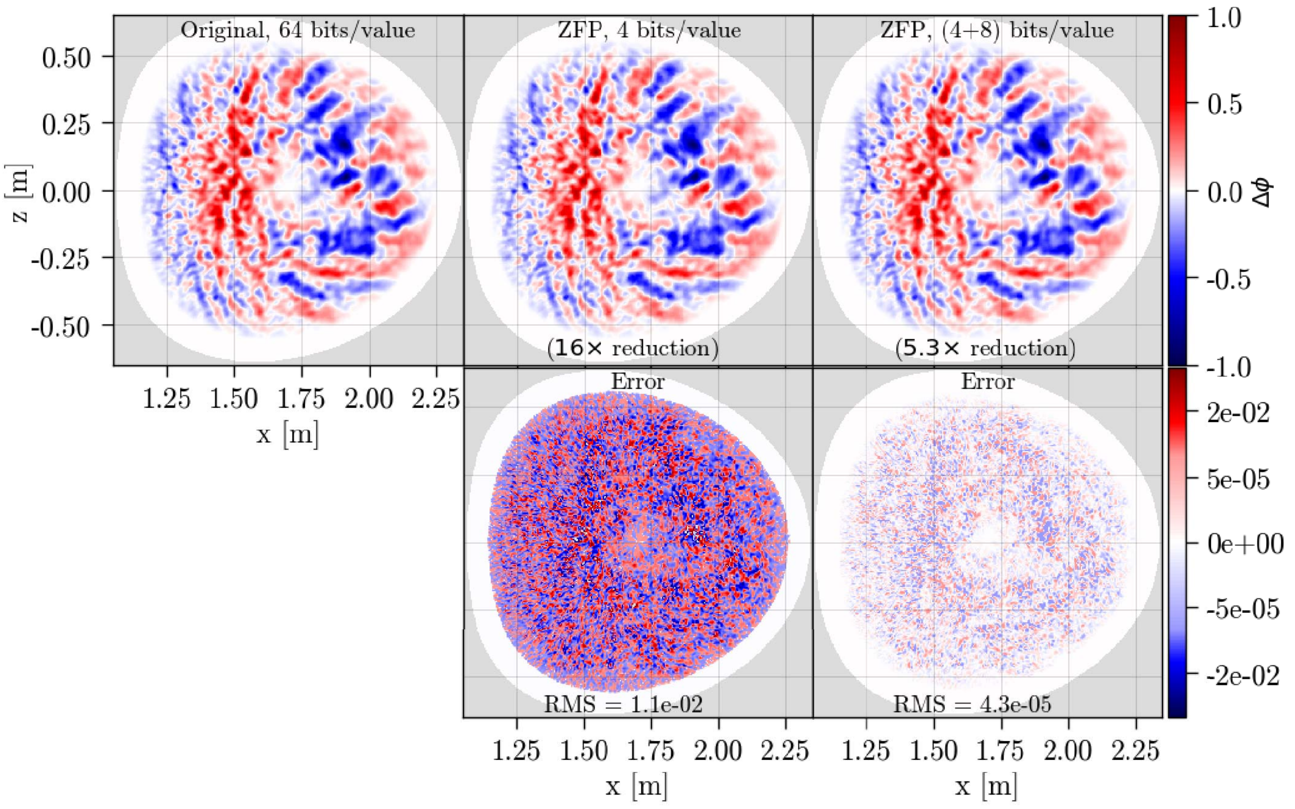

Fig. 3. Progressive error-level "views" of XGC simulation data, using ZFP compression. In the **top row**, the color scale shows how the scalar potential (ϕ) deviates from the nominal background level, normalizing by the largest magnitude deviation in the first panel. Each panel is the same planar-slice through the torus, at the same time step, but reduced differently. We have included three precision levels: (**left**) full precision of 64 bits/value, (**middle**) 4 bits/value, and (**right**) 12 bits/value. The 12 bit version reads the first 4 bits from the same file as the middle panel, extending by an additional 8 bits from a separate file, possibly saved to a different storage type than the first file. In the **bottom row**, the colors encode the ZFP-compression error levels; note, the scale is not linear.

features and tag those instead. (Again, a full discussion of the technical implementation of the approach described here is beyond the scope of this paper, but will be further documented in our subsequent publication.)

Importantly, the different levels in both our data organization schemes can be mapped to different partitions of the storage hierarchy. This facilitates storing the most important subset of the much larger total dataset as close to the processors as possible, accelerating access to it. For example, to facilitate in situ visualization or analysis, one might direct relatively few bits to an SSD, because all the data would be too large for the SSD's storage quota. The middle panel in Fig. 3 could be saved to the SSD, and then the extra bits added in right panel saved to the parallel file system. We are now beginning to undertake performance impact evaluations under such scenarios, testing against different partitions with different latency and bandwidth for different compression resolution progressions, and will present our findings in a forthcoming publication.

C. *Challenges, Research, and Development*

The primary technical challenges facing the methodology we have presented so far originate from how the data model is not the way scientific data has typically been framed. Current parallel I/O packages were not designed conceiving of data as multiple related, yet distinct units to be selected in different read contexts. Considerable infrastructure development has been needed, and will continue to be needed in the future as progressive refinement methodology matures We have used the Adaptable I/O System (ADIOS) [25] as the basis for our work, as its framework does provide some support for adding extensions of research methods to the code base. However, "splitting" the output buffer to write to disk breaks expectations of what is natively expected in the code, so extra workarounds were needed. ADIOS-2[7], an ECP-funded project to rewrite ADIOS in C++ and much further enhance user-defined customization, including the kinds of transports we have described in this section, will assist in exploring new organization schemes in years to come.

[7]https://github.com/ornladios/ADIOS2

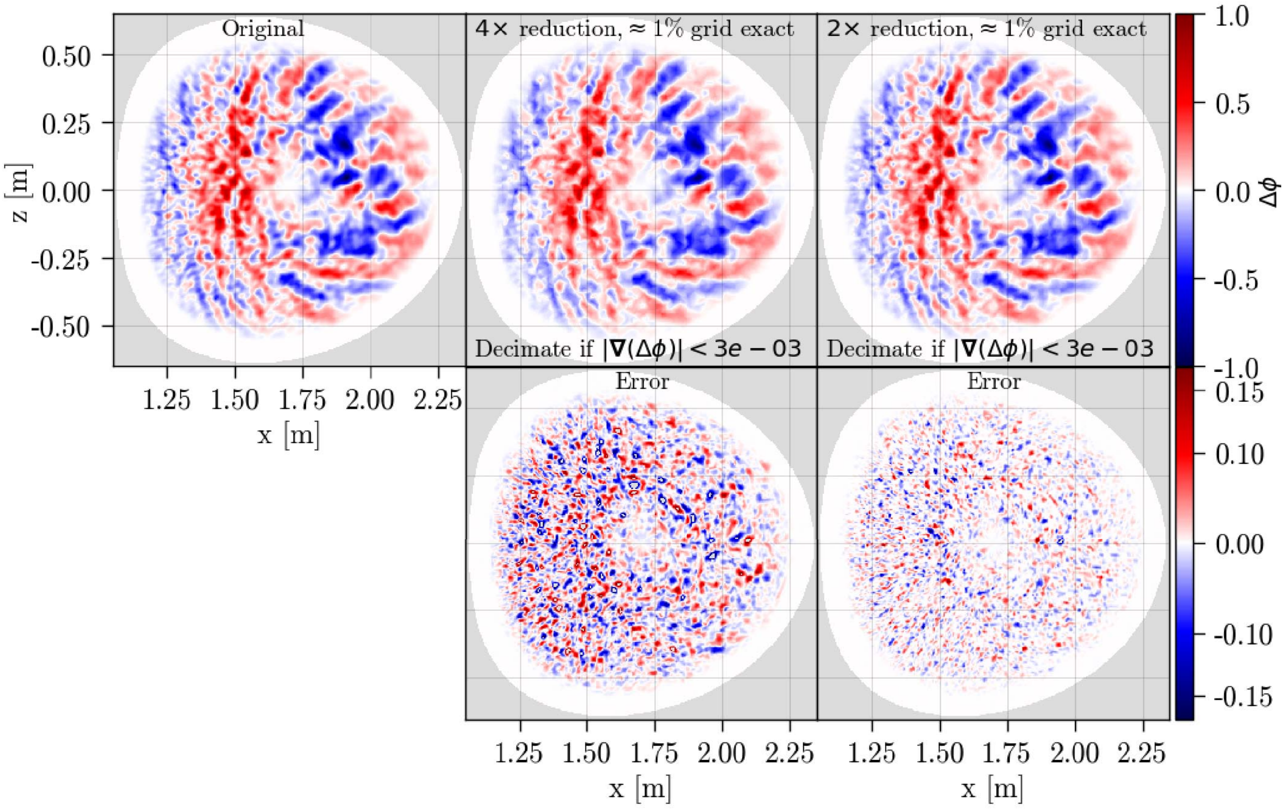

Fig. 4. Feature-driven“views” of the same XGC simulation data as Fig. 3. Here, features are defined with a threshold in gradient magnitude, and grid cells exceeding that threshold are saved with full precision. The remainder of the grid is decimated: to either (**middle**) a compression factor of 4 or (**right**) a compression factor of 2.

Longer term, we envision a full software environment (SIRIUS [26]), capable of managing data organized in the ways we have described, which would optimize and automate several features, which we will highlight to come. For example, a user could enter a maximum time constraint, and the system would return the best-quality data available within that time. One might instead prefer to retrieve a version of the data, with at least 95 percent precision compared to the original. We would like to support numerous access options.

A system like SIRIUS presents opportunities for data placement optimizations. Where in the filesystem (SSD, parallel filesystem, etc.) to save each level of the data is currently a decision for the user, and this can have a significant impact on read performance, (see e.g. the work of Jin et al. [27] studying data placement in staging scenarios for scientific simulations). If the data is being accessed frequently, it would be convenient to automatically move it into faster storage. Similarly, if it is saved to a scratch parallel filesystem and about to be purged, it would be natural to back up to tape. Automated migration in multi-device filesystem operations is not a novel idea; features of this kind existed in Unitree [28] years ago, but the software suffered from high latency. Modern research filesystems, such as Sirocco [29] or Ceph [30], offer appealing technology to explore, but an end-to-end system of the kind we have described still needs further development.

Complicating the placement problem, users typically only need to read a set of variables at once, not all of them. Moving the entire file, instead of only the portion of it that is relevant, could be non-ideal and wasteful. This motivates deconstructing the files into objects (or even sub-objects for portions of a variable), and SIRIUS will maintain an object store to manage the data, leveraging technology such as Ceph's Controlled Replication Under Scalable Hashing (CRUSH [31]) for scalable, predictable performance.

One final longer term consideration that we highlight here is designing the metadata service. When refactoring datasets, one challenge is incorporating enough information in the storage system so future clients are able to recreate the data structures, but not too much to bloat the metadata overhead. Also, different kinds of storage media usually have independent namespaces, so distributing related data across multiple devices makes locating it more difficult.

IV. Robust Compression

In Section III we emphasized read scenarios. Compression was primarily framed as a means to help achieve progressive refinement. However, lossy compression for scientific data beckons further discussion, which we will pursue in this section.

We reiterate that compression's potential impact on the write side should not be overlooked. Checkpoint/restart data for simulations is typically written infrequently, because it tends to be large. If one were willing to accept lossy checkpoints, more time steps could be saved. However, since many scientists would be opposed to lossy restart data, we will downplay any importance there. Many other quantities (in simulations or experiments – which do not have checkpoint/restart data in the first place) are saved at higher frequency, and these are the ones that drive the analyses that ultimately extract new insight. Likely not all these data products need to be saved at full 64-bit precision. One could write out reduced precision, and budget the time saved to perform more frequent output of the current variables or to enable dumps of new quantities that were previously uncalculated/unsaved. Though the methods described in Section III-B were both defined in terms of spatial quantities, one can also envision beginning to think of different time steps in the data as a dimension that one can compress over for further potential gain.

Ultimately, the scientific applications are what drives what is required of the compression techniques, and what determines if compression is robust enough to make it into production. Many questions arise. Is the compression-accuracy precise enough? Is it fast enough? Does it preserve important relationships? Is it clear how it will affect downstream analytics? Keeping all these in mind, we emphasize the need for careful quality monitoring, both error- and performance-wise. We are also interested in exploring how to best utilize hardware technologies for acceleration. It is beneficial to closely scrutinize any compression technique, even common ones such as decimation, and pursue an increased level of rigor in the methods.

Overall, we advocate increased *co-design* in scientific data management. This is a shift away from designing applications and general-purpose software completely decoupled from each other, and realizing that there is benefit for them to co-evolve; hardware must factor in to the evolution as well. As we continue toward exascale, the need for more reduction and other in situ processing is anticipated among many computer scientists, and co-design can widely help facilitate solutions, capitalizing on the interrelatedness between the domain sciences, computer science, and vendor product development.

A. Science-driven Constraints

Section III-B alluded to how compression in scientific data management will need to support several schemes, because different applications are sensitive to different types of operations. It is unlikely there will ever be a one-size-fits-all solution. Already, several different classes of lossy floating point compressors exist. Four examples are included in those listed in Section I alone: ISABELA is effectively a smoothing kernel [4], ZFP is a spectral method [5], SZ is a form of curve-fitting [6], and parallel tensor decomposition is a statistical reduction of several quantities simultaneously [7].

We assert, there is almost certainly a need for non-uniform/adaptive techniques, which produce different resolutions in different regions of the domain; this is why we included the feature-based approach among our initial two in Section III-B. Science is inherently multi-scaled, where much more variation can occur in very small regions compared to the total system size, and resolving such features is critical. This is the motivation for adaptive mesh refinement (AMR, e.g. [32]), which has numerous applications throughout science: galaxy formation simulations in astrophysics and fracture/fragmentation studies in material sciences to name two.

Typically, compression in computer science has been formulated in a quantity-agnostic way, meaning no distinction is made about what is being reduced. For a compressor which operates on one-dimensional arrays, it is not particularly important whether a collection of energy values, momenta components, or anything else is given, only the floating-point values are significant. There is no notion of relationships between variables. We caution that this may not be most appropriate when working with scientific data. Conservation laws are pervasive throughout science, and mathematical constraints must be satisfied. Quantity-agnostic compression could interfere here. For instance, certain problems might require momentum conservation, but the chosen compression kernel suffers correlated errors when operating over the dimensions separately, such that globally momentum is no longer conserved after the compression. Concerns of these kind contribute to why it is important to better understand the mathematics of compression algorithms, and the explicit or implicit assumptions they make, related to the discussion in Section IV-D.

B. Careful Monitoring

To further exemplify possible pitfalls of compression, we return to the case of tokamak simulations. One relevant derived quantity is the outward velocity driven in the plasma by the electric field:

$$\mathbf{v}_\perp = \nabla\phi \times \hat{\mathbf{B}}, \tag{1}$$

where ϕ is the electric scalar potential, $\hat{\mathbf{B}}$ is the direction of the magnetic field, and ∇ is the gradient operator. Near the wall of the tokamak, electromagnetic "islands" or "blobs", small regions with significantly different potentials compared to background, are found to develop. Kress et al. [33] demonstrated that uniform mesh decimation is enough to make features in $\mathbf{v}_\perp$ driven by these blobs completely disappear from XGC data. One needs sufficient resolution in the outer regions to resolve the derivatives in the gradient, and uniform decimation breaks this condition.

This example reinforces a need for quality monitoring in future online systems. Scientists need to be able to verify new reduction methods, and having a common harness through which do to this would be ideal. Here, several authors are beginning work in the Co-Design Center for Online Data

Analysis and Reduction at the Exascale (CODAR) project[8]. We stress the importance of meaningful metrics to evaluate the impact of the errors. Like in the XGC blob-feature example, point-wise differences, or statistical calculations over them, do not necessarily directly translate to what is most critical. Once the mapping in errors from fundamental data products to derived data products is understood, progressive reduction along the lines outlined in Section III becomes more attractive, because scientists can proceed with analyses using a smaller-size view for performance, but maintain firm uncertainty bounds. At any rate, as reduction becomes more common in high performance applications, the application scientists need to provide feedback to the computer scientist concerning relevant quality checks to implement.

In addition to error-performance monitoring, future simulation data management systems should also include built-in time-performance tracking. This is useful not only for benchmarking the reduction, but for diagnosing other possible bottlenecks as well. Those, in turn, could become the subject of acceleration/optimization exercises. The challenge with performance monitoring is not to incur significant overhead in collecting the statistics compared to the application's usual run time. Saving too much information can be burdensome to write to disk, and effective reduction of the performance data is a topic of research. The work of Nataraj et al. [34] is one example using TAU [35], (one of several popular performance monitoring tools), and three statistical filtering schemes to reduce the output to different verbosity levels. Unlike the scientific portions of the data, the peformance data is "non-physical", so we are interested in researching machine learning reduction techniques for use within future monitoring systems.

C. Acceleration

Something that we have largely glossed over to this point is that compression/decompression does not come for free; it takes time. It is only worth compressing if the extra time to compress and decompress is not prohibitive and/or gain is to be had by reducing the data volume that will need to be written. Accordingly, compression kernels are a natural target for hardware acceleration.

Focusing the scope to GPU acceleration, compression on GPUs is one direction of interest. For example, O'Neil and Burtscher developed GFC [36], improving upon the CPU-based performance of the FPC family of lossless floating-point compressors [37], [38] by factors of several. Similarly motivated, we have begun investigating a GPU implementation of ZFP. GPUs factor in to system design as we build toward exascale, and other GPU ports should be pursued as well.

A little longer term, it also worthwhile to consider other hardware acceleration options. For instance, we are undertaking a project with Mellanox to study offloading the compression into the FPGA environment embedded into the networking hardware. It would also be interesting to investigate disk storage technology that has some kind of compression-aware hardware embedded. Research along these lines is a largely unexplored topic, but offers a unique opportunity to leverage technology advances in other areas to possibly accelerate I/O.

Complicating hardware acceleration with compression for big scientific data at leadership computing facilities, not all hardware technologies will be available everywhere. Some compression algorithms may only be performant enough if run on the appropriate hardware, which again emphasizes the need for careful performance monitoring. Different hardware options could also put more onus on the user to understand what is available and what are the possible consequences of the different choices, so where possible, automating the "best" implementation would be ideal.

Representation is another matter to consider. Typically codes decompress a compressed quantity before working with it, so the same routines can be used with the original or reduced quantities. However, the possibility of working directly with the reduced representation is an alternative that skips decompression altogether, which could offer another level of optimization. ZFP, for example, builds with code to allow arithmetic over the compressed arrays themselves from C++. As compression enters into more scientific workflows, thought should be spent if the workflows could benefit from computing in non-standard representations, such as that of ZFP or unum 2 [39]. This is almost never done today; the IEEE 754 floating-point standard [40] is ubiquitous to say the least. However, the original specification [41] was designed over 30 years ago, when computers were much different than they are today. Newer ideas are not without merit.

D. (Lossless) Decimation Mathematics

A commonly adopted approach to data reduction that has been mentioned several times throughout this paper is decimation. To conclude this section, we highlight early work toward better understanding and controlling error effects of decimation. It is an example of attempting to approach compression from a well-principled standpoint, and not overlooking implicit assumptions.

Consider data $\{u_j\}$ produced by a computational simulation at a sequence of times $\{t_j\}$. Decimation consists of retaining say, every 10-th datum and discarding the remainder. This decimation clearly gives a guaranteed 10 times reduction of the data and is easy to apply. However, the procedure also has drawbacks, most notably that 90 percent of the data is simply discarded and lost irretrievably.

Fundamentally, discarding 90 percent of the data seems unacceptable. As such, it is perhaps rather surprising that decimation is commonly used. However, this view fails to recognize the fact that many simulations are performed at resolutions or time step sizes that are dictated by the need to maintain stability of the underlying computational algorithm rather than by the nature of the underlying data per se. Indeed, many simulations model localized phenomena for which the data is smooth in large portions of the (space or time) domain. As a result, the full dataset can often be well-represented by the decimated data, with the implicit assumption that if a

[8]https://exascaleproject.org/2016/11/11/ecp_co-design_centers/

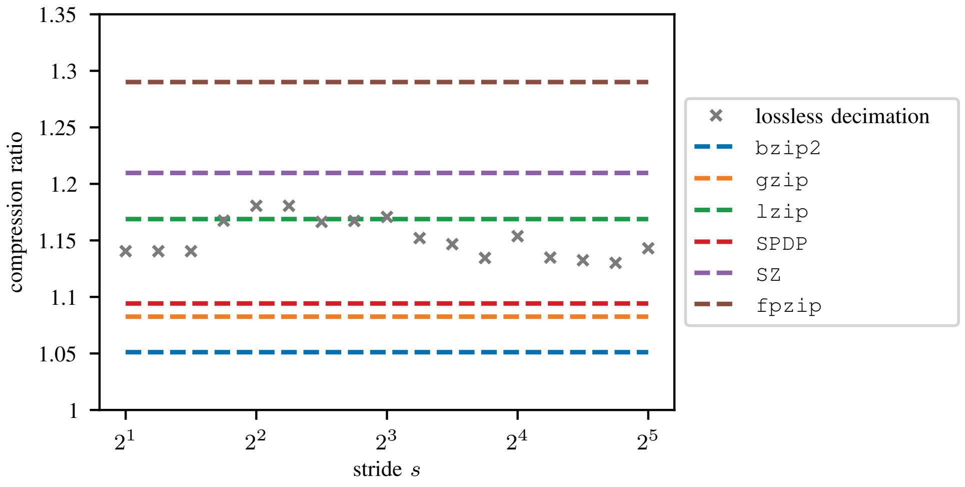

Fig. 5. Comparison of lossless decimation and a selection of general-purpose and specialized compressors. The data used is a $1024 \times 256 \times 768$ timeslab of single-precision pressure values (768 MB in total) taken from a simulation of flow in a turbulent channel [42]–[44]. For each stride, the data is decimated, linear interpolation is used to calculate a surrogate for each discarded value, and the residuals are compressed with `lzip` [45]. For this data, lossless decimation achieves a larger compression factor than `bzip2` [46], `gzip` [47], and `SPDP` [48], and is comparable to that of `lzip`. The specialized floating point compressors `SZ` [6] (run in near-lossless mode by requiring that the absolute error be at most 2^{-149}) and `fpzip` [49] (run in lossless 1D mode) outperform lossless decimation in compression ratio. A unique advantage of lossless decimation is that the decimated data, which is stored uncompressed in the output, can be interpolated to generate a surrogate dataset, without the need to decompress the entire dataset when full reproduction is not required.

discarded datum is needed, then a surrogate or replacement can be regenerated by carrying out linear interpolation of the retained values.

For the moment, let us assume that the above argument is valid, such that the replacement data obtained by linear interpolation provides an acceptable surrogate for the discarded data. This means that the difference between the original discarded data and the surrogate data is "small". More precisely, one would expect the information or entropy of these differences to be small, which means that the differences could, if necessary, be compressed effectively. Accordingly, one could augment the decimated dataset with the encoded differences and thereby try to retain benefits of the decimation procedure without losing 90 percent of the data – in essence, employing a "lossless decimation" technique.

Now we return to the more realistic setting in which the reconstructed values might be an acceptable surrogate in certain parts of the data, but in regions of localized features or where interesting physics is occurring, the reconstructed values are inadequate. (Of course, if the latter condition holds for the entire dataset, one must accept that decimation is not an appropriate tool and seek other, more sophisticated means by which to reduce the data. However, this worst-case scenario is generally not the norm in cases where practitioners currently employ decimation.) While the differences between the reconstructed values and the discarded values will be large in regions where interesting phenomena are occurring, they will be small over large portions of the original data points, again meaning that the differences are amenable to compression through entropy encoding, and that the lossless decimation procedure can be anticipated to be an effective compressor.

This lossless decimation is considered in the recent work of Ainsworth et al. [50]; expected compression rates are quantified in terms of smoothness measures of the data. For instance, the smoothness of deterministic data might be measured in terms of the size of the derivatives or difference quotients of the underlying function representing the data. Alternatively, non-deterministic data or data otherwise subject to large noise or random fluctuations might be measured in terms of statistics of the data such as the variance. Fig. 5 is a plot of compression rates obtained using the lossless decimation procedure compared with other compressors; the lossless decimation results are comparable to those obtained using the alternatives.

Of course, lossless decimation is not and never will be a panacea. Nevertheless, given the widespread use of decimation, it does offer a practical, simple, and competitive alternative to other compression techniques that are currently available. The decimated data is stored uncompressed in the output, allowing one to to bypass decompressing the entire dataset in certain scenarios where full reproduction is not necessary and/or interpolation suffices. Also, the difference encoding explicitly acknowledges that decimation discards data, which is commonly overlooked with decimation.

V. Conclusion

Building upon motivation of concrete current-to-next generation high performance scientific workflows in Section II, we have described several directions that we believe will increase in importance as scientific data management continues toward exascale, and presented ideas for how to improve state of the art techniques. These were grouped into two broad, overlapping categories:

1) *Data organization*: Storage hierarchies are deepening. Users should to be able to seamlessly distribute various pieces of their data across the hierarchy, with accelerated access to the most important data from storage levels closer to the processors and automatic migration across levels according to the data's usage and life cycle. Progressive refinement – presenting users with multiple views of their data, at various resolutions – affords increased flexibility when filesystem bandwidth can be at a premium.
2) *Compression*: Smaller fractions of a job's total data volume can be saved compared to before. Compression can reduce the I/O load and mitigate the imbalance, but must be robust and error-controlled. Compression for scientific data should be closely monitored to safeguard against the many potential pitfalls, including feature loss and performance degradation. Compression kernels need to be performant and are a natural target for hardware acceleration.

Early progress has been made toward these goals, and throughout the paper, we highlighted initial work we have undertaken related to these two categories including:

- Two progressive compression techniques, exemplified with simulation data in Fig. 3 and Fig. 4.
- The SIRIUS system, for managing multi-level, mulitview data (Section III-C).
- A co-design initiative dedicated to online reduction, analysis, and monitoring (Section IV-B).
- GPU and FPGA studies for compression kernel acceleration (Section IV-C).
- An error-aware approach for "lossless decimation" (Section IV-D).

Of course, unforeseen obstacles are likely to occur as we approach closer to exascale computing, and considerations beyond what we have addressed here may very well be essential to tackle. However, we believe that ideas that we have laid out in this paper would help facilitate significant enhancements to the science output of future high performance workflows.

Acknowledgment

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, in support of the nations exascale computing imperative.

Work was also funded by research grants from the Advanced Scientific Research Office (ASCR) at the Department of Energy, under contracts SDAV: ERKJ200/KJ0403000, RSVP: ERKJU60/KJ0402000, and SIRIUS: ERKJ311/KJ0402000.

We would like to thank two DOE ASCR program managers, Drs. Lucy Nowell and Ceren Susut-Bennett. In particular we wish to point out that much insight into these problems was articulated by Dr. Nowell, which allowed our team to push several software artifacts described in this paper into several large scale scientifc applications.

References

[1] J. Ahrens and T. M. Rhyne, "Increasing scientific data insights about exascale class simulations under power and storage constraints," *IEEE Computer Graphics and Applications*, vol. 35, no. 2, pp. 8–11, Mar 2015.

[2] K. Moreland, "The tensions of in situ visualization," *IEEE Computer Graphics and Applications*, vol. 36, no. 2, pp. 5–9, Mar 2016.

[3] A. C. Bauer, H. Abbasi, J. Ahrens, H. Childs, B. Geveci, S. Klasky, K. Moreland, P. O'Leary, V. Vishwanath, B. Whitlock *et al.*, "In situ methods, infrastructures, and applications on high performance computing platforms," in *Computer Graphics Forum*, vol. 35, no. 3. Wiley Online Library, 2016, pp. 577–597.

[4] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, *Compressing the Incompressible with ISABELA: In-situ Reduction of Spatio-temporal Data*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 366–379. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23400-2{_}34

[5] P. Lindstrom, "Fixed-Rate Compressed Floating-Point Arrays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2674–2683, dec 2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6876024

[6] S. Di and F. Cappello, "Fast Error-Bounded Lossy HPC Data Compression with SZ," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, may 2016, pp. 730–739. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7516069

[7] W. Austin, G. Ballard, and T. G. Kolda, "Parallel Tensor Compression for Large-Scale Scientific Data," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, may 2016, pp. 912–922. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7516088

[8] D. U. Lee, K. W. Kim, K. W. Kim, K. S. Lee, S. J. Byeon, J. H. Kim, J. H. Cho, J. Lee, and J. H. Chun, "A 1.2 v 8 gb 8-channel 128 gb/s high-bandwidth memory (hbm) stacked dram with effective i/o test circuits," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 191–203, 2015.

[9] J. Peter, "The lustre storage architecture," *Cluster File Systems, Inc*, 2004.

[10] F. B. Schmuck and R. L. Haskin, "Gpfs: A shared-disk file system for large computing clusters." in *FAST*, vol. 2, no. 19, 2002.

[11] R. W. Watson and R. A. Coyne, "The parallel i/o architecture of the high-performance storage system (hpss)," in *Mass Storage Systems, 1995.'Storage-At the Forefront of Information Infrastructures', Proceedings of the Fourteenth IEEE Symposium on*. IEEE, 1995, pp. 27–44.

[12] F. Jenko, W. Dorland, M. Kotschenreuther, and B. Rogers, "Electron temperature gradient driven turbulence," *Physics of Plasmas*, vol. 7, no. 5, pp. 1904–1910, 2000.

[13] T. Görler, X. Lapillonne, S. Brunner, T. Dannert, F. Jenko, F. Merz, and D. Told, "The global version of the gyrokinetic turbulence code gene," *Journal of Computational Physics*, vol. 230, no. 18, pp. 7053–7071, 2011.

[14] F. Jenko, D. Told, T. Görler, J. Citrin, A. B. Navarro, C. Bourdelle, S. Brunner, G. Conway, T. Dannert, H. Doerk *et al.*, "Global and local gyrokinetic simulations of high-performance discharges in view of iter," *Nuclear Fusion*, vol. 53, no. 7, p. 073003, 2013.

[15] C. Chang, S. Ku, P. Diamond, Z. Lin, S. Parker, T. Hahm, and N. Samatova, "Compressed ion temperature gradient turbulence in diverted tokamak edge a," *Physics of Plasmas*, vol. 16, no. 5, p. 056108, 2009.

[16] R. Hager and C. Chang, "Gyrokinetic neoclassical study of the bootstrap current in the tokamak edge pedestal with fully non-linear coulomb collisions," *Physics of Plasmas*, vol. 23, no. 4, p. 042503, 2016.

[17] E. R. Hawkes, R. Sankaran, J. C. Sutherland, and J. H. Chen, "Direct numerical simulation of turbulent combustion: fundamental insights towards predictive models," in *Journal of Physics: Conference Series*, vol. 16, no. 1. IOP Publishing, 2005, p. 65.

[18] J. H. Chen, A. Choudhary, B. De Supinski, M. Devries, E. R. Hawkes, S. Klasky, W. K. Liao, K. L. Ma, J. Mellor-Crummey, N. Podhorszki, R. Sankaran, S. Shende, and C. S. Yoo, "Terascale direct numerical simulations of turbulent combustion using S3D," *Comput. Sci. Disc. Computational Science & Discovery Computational Science & Discovery Computational Science & Discovery*, vol. 2, no. 2, pp. 15 001–15 001, 2009. [Online]. Available: http://iopscience.iop.org/1749-4699/2/1/015001www.iop.org/journals/csd

[19] D. S. Taubman and M. W. Marcellin, "Jpeg2000: standard for interactive imaging," *Proceedings of the IEEE*, vol. 90, no. 8, pp. 1336–1357, Aug 2002.

[20] G. K. Wallace, "The jpeg still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, Feb 1992.

[21] V. Pascucci, G. Scorzelli, B. Summa, P.-T. Bremer, A. Gyulassy, C. Christensen, S. Philip, and S. Kumar, "The visus visualization framework," in *High Performance Visualization: Enabling Extreme-Scale Scientific Insight*. Chapman and Hall/CRC, 2012.

[22] S. Kumar, C. Christensen, J. A. Schmidt, P.-T. Bremer, E. Brugger, V. Vishwanath, P. Carns, H. Kolla, R. Grout, J. Chen *et al.*, "Fast multiresolution reads of massive simulation datasets," in *International Supercomputing Conference*. Springer, 2014, pp. 314–330.

[23] S. Kumar, J. Edwards, P. T. Bremer, A. Knoll, C. Christensen, V. Vishwanath, P. Carns, J. A. Schmidt, and V. Pascucci, "Efficient i/o and storage of adaptive-resolution data," in *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2014, pp. 413–423.

[24] V. Pascucci and R. J. Frank, "Global static indexing for real-time exploration of very large regular grids," in *Supercomputing, ACM/IEEE 2001 Conference*, Nov 2001, pp. 45–45.

[25] Q. Liu, J. Logan, Y. Tian, H. Abbasi, N. Podhorszki, J. Y. Choi, S. Klasky, R. Tchoua, J. Lofstead, R. Oldfield *et al.*, "Hello ADIOS: the challenges and lessons of developing leadership class I/O frameworks," *Concurrency and Computation: Practice and Experience*, vol. 26, no. 7, pp. 1453–1473, 2014.

[26] S. A. Klasky, H. Abbasi, M. Ainsworth, J. Choi, M. Curry, T. Kurc, Q. Liu, J. Lofstead, C. Maltzahn, M. Parashar, N. Podhorszki, E. Suchyta, F. Wang, M. Wolf, C. S. Chang, M. Churchill, and S. Ethier, "Exascale storage systems the sirius way," *Journal of Physics: Conference Series*, vol. 759, no. 1, p. 012095, 2016. [Online]. Available: http://stacks.iop.org/1742-6596/759/i=1/a=012095

[27] T. Jin, F. Zhang, Q. Sun, H. Bui, M. Romanus, N. Podhorszki, S. Klasky, H. Kolla, J. Chen, R. Hager, C. S. Chang, and M. Parashar, "Exploring data staging across deep memory hierarchies for coupled data intensive simulation workflows," in *2015 IEEE International Parallel and Distributed Processing Symposium*, May 2015, pp. 1033–1042.

[28] F. McClain, "Datatree and unitree: software for file and storage management," in *Tenth IEEE Symposium on Mass Storage Systems: Crisis in Mass Storage*, May 1990, pp. 126–128.

[29] M. L. Curry, H. L. Ward, and G. Danielson, "Motivation and design of the sirocco storage system, version 1.0," Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2015.

[30] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, ser. OSDI '06. Berkeley, CA, USA: USENIX Association, 2006, pp. 307–320. [Online]. Available: http://dl.acm.org/citation.cfm?id=1298455.1298485

[31] S. A. Weil, S. A. Brandt, E. L. Miller, and C. Maltzahn, "Crush: Controlled, scalable, decentralized placement of replicated data," in *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, ser. SC '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1188455.1188582

[32] M. J. Berger and P. Colella, "Local adaptive mesh refinement for shock hydrodynamics," *Journal of computational Physics*, vol. 82, no. 1, pp. 64–84, 1989.

[33] J. Kress, R. Churchill, S. Klasky, M. Kim, H. Childs, and D. Pugmire, "Preparing for in situ processing on upcoming leading-edge supercomputers," *Supercomputing frontiers and innovations*, vol. 3, no. 4, 2016. [Online]. Available: http://superfri.org/superfri/article/view/115

[34] A. Nataraj, A. D. Malony, A. Morris, D. C. Arnold, and B. P. Miller, "A framework for scalable, parallel performance monitoring," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 6, pp. 720–735, 2010.

[35] S. S. Shende and A. D. Malony, "The tau parallel performance system," *The International Journal of High Performance Computing Applications*, vol. 20, no. 2, pp. 287–311, 2006.

[36] M. A. O'Neil and M. Burtscher, "Floating-point data compression at 75 Gb/s on a GPU," in *Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units - GPGPU-4*. New York, New York, USA: ACM Press, 2011, p. 1. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1964179.1964189

[37] M. Burtscher and P. Ratanaworabhan, "FPC: A High-Speed Compressor for Double-Precision Floating-Point Data," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 18–31, jan 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4589203

[38] ——, "pfpc: A parallel compressor for floating-point data," in *Data Compression Conference, 2009. DCC'09.* IEEE, 2009, pp. 43–52.

[39] J. L. Gustafson, "A radical approach to computation with real numbers," *Supercomputing frontiers and innovations*, vol. 3, no. 2, pp. 38–53, 2016.

[40] I. C. Society, "Ieee standard for floating-point arithmetic," *IEEE Std 754-2008*, pp. 1–70, Aug 2008.

[41] ——, "Ieee standard for binary floating-point arithmetic," *ANSI/IEEE Std 754-1985*, pp. 1–19, 1985.

[42] Y. Li, E. Perlman, M. Wan, Y. Yang, C. Meneveau, R. Burns, S. Chen, A. Szalay, and G. Eyink, "A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence," *Journal of Turbulence*, vol. 9, p. N31, Jul. 2008. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/14685240802376389

[43] E. Perlman, R. Burns, Y. Li, and C. Meneveau, "Data exploration of turbulence simulations using a database cluster," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, vol. 23. Reno, NV, USA: ACM, Nov. 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1362654

[44] J. Graham, K. Kanov, X. Yang, M. Lee, N. Malaya, C. Lalescu, R. Burns, G. Eyink, A. Szalay, R. Moser, and others, "A Web services accessible database of turbulent channel flow and its use for testing a new integral wall model for LES," *Journal of Turbulence*, vol. 17, no. 2, pp. 181–215, 2016. [Online]. Available: http://dx.doi.org/10.1080/14685248.2015.1088656

[45] A. Diaz Diaz, "lzip," Jun. 2016. [Online]. Available: http://lzip.nongnu.org/lzip.html

[46] J. Seward, "bzip2," Sep. 2010. [Online]. Available: http://www.bzip.org

[47] J.-l. Gailly and M. Adler, "gzip," Jun. 2013. [Online]. Available: http://www.gzip.org/

[48] M. Burtscher and S. Claggett, "SPDP v1.0," 2016. [Online]. Available: http://cs.txstate.edu/~burtscher/research/SPDPcompressor/

[49] P. Lindstrom and M. Isenburg, "Fast and Efficient Compression of Floating-Point Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1245–1250, Sep. 2006.

[50] M. Ainsworth, S. Klasky, and B. Whitney, "Compression using lossless decimation: analysis and application," *SIAM J. Sci. Comp*, (In review), 2017.