# STORAGE SYSTEMS AND I/O: ORGANIZING, STORING, AND ACCESSING DATA FOR SCIENTIFIC DISCOVERY

## REPORT FOR THE DOE ASCR WORKSHOP ON STORAGE SYSTEMS AND I/O

Gaithersburg, Maryland
September 19–20, 2018

U.S. DEPARTMENT OF ENERGY | Office of Science

Sponsored by the Office of Advanced Scientific Computing Research

# Storage Systems and I/O 2018: Organizing, Storing, and Accessing Data for Scientific Discovery

**Gaithersburg, Maryland**
**September 19–20, 2018**

**Meeting Organizers**
Robert Ross (Argonne National Laboratory) (lead organizer)
Lee Ward (Sandia National Laboratories) (co-lead)
Gary Grider (Los Alamos National Laboratory)
Scott Klasky (Oak Ridge National Laboratory)
Glenn K. Lockwood (Lawrence Berkeley National Laboratory)
Kathryn Mohror (Lawrence Livermore National Laboratory)
Bradley Settlemyer (Los Alamos National Laboratory)

**Workshop Document Contributors**
Philip Carns (Argonne National Laboratory)
Quincey Koziol (Lawrence Berkeley National Laboratory)
Matthew Wolf (Oak Ridge National Laboratory)

**ASCR Program Contact**
Lucy Nowell
Laura Biven

DOI: 10.2172/1491994

## Table of Contents

## Figures

# 1 Executive Summary

Storage systems are a foundational component of computational, experimental, and observational science today. The success of U.S. Department of Energy (DOE) activities in these areas is inextricably tied to the usability, performance, and reliability of storage systems and input/output (SSIO) technologies.

SSIO technologies use the organization and placement of data to enhance computation and discovery. Functionally, these tools focus on storing, moving, and accessing data in ways that allow researchers to maintain their conceptual representations of scientific data, while simultaneously optimizing for performance and productivity across a variety of platforms.

Fundamentally new challenges for effective SSIO architectures arise from architectural changes in future high-performance computing (HPC) platforms, applications that generate extreme-scale data, and use cases that require simultaneous analysis of data from a variety of sources, often in real time. In order to make effective use of diverse but finite resources, SSIO technologies will need to integrate more data management functions such as data reduction, determinations about data retention times and access needs, and even value judgments informed by downstream analysis needs. To cope with this complexity, a large variety of memory and storage devices are deployable in the SSIO context, but this creates its own challenges in terms of leveraging these resources without overburdening the application researchers.

Attendees recognized that change is being driven in two ways. First, new approaches to scientific discovery and activities in new science domains are driving the rapid growth of scientific dataset sizes, increasing the importance of support for streaming data, and resulting in a greater emphasis on learning applications. Second, new storage, computing, and sensor technologies require a rethinking of traditional software models to better leverage these technologies or adapt to new scientific methods enabled by them.

In September, 2018, the Department of Energy, Office of Science, Advanced Scientific Computing Research Program convened a workshop to identify key challenges and define research directions that will advance the field of storage systems and I/O over the next 5–7 years. The workshop concluded that addressing these combined challenges and opportunities requires tools and techniques that greatly extend traditional approaches and require new research directions. Key research opportunities emerging from this workshop include:

- Enabling science understandability and reproducibility through rich data formats, metadata, and provenance (Section 4.2)
- Accelerating scientific discovery through support of *in situ* and streaming data analysis (Sections 4.3 and 4.6)

- Enhancing SSIO usability, performance, and resilience through monitoring, prediction, and automation (Section 4.5)
- Improving efficiency and integrity of data movement and storage through architecture of systems and services (Sections 4.1 and 4.4)

## 2 Introduction

Computation and simulation advance knowledge in science, energy, and national security. The United States has been a leader in high-performance computing (HPC) for decades, and U.S. researchers greatly benefit from open access to advanced computing facilities, software, and programming tools. As HPC systems become more capable, computational scientists are now turning their attention to new challenges, including the certification of complex engineered systems such as new reactor designs and the analysis of climate mitigation alternatives such as carbon sequestration approaches. Problems of this type demonstrate a need for computing power 1,000 times greater than we have today; and the solution is exascale computing, the next milestone in HPC capability.

At the same time, high-performance computing is playing an increasingly critical role in understanding experimental and observational data (EOD) from platforms such as the Large Hadron Collider, which is a key tool in better understanding fundamental questions in physics, and the upcoming Large Synoptic Survey Telescope, which, when deployed, will provide greater insight into the structure of the universe. Learning applications, too, are beginning to employ high-performance computing. Achieving the power efficiency, reliability, and programmability goals for exascale HPC, EOD, and learning applications will have dramatic impacts on computing at all scales, from personal computers (PCs) to mid-range computing and beyond; the broader application of exascale computing can provide tremendous advantages for fundamental science and industrial competitiveness.

Storage systems and input/output (SSIO) research and development is a cornerstone of data-intensive computing tasks of all types, including simulations at scale, experimental/observational science, and learning applications. SSIO technologies include a range of hardware and software, from the low-level parallel file system (e.g., Lustre, general parallel file system [GPFS]) and archival storage (e.g., high-performance storage system [HPSS]) up to libraries that serve as the interfaces to applications and provide format interoperability, as well as software that monitors and reports on the utilization of the storage system. Advances in SSIO improve the capability, scalability, and robustness of storage solutions, enabling larger volumes of data to be stored and accessed and their integrity to be maintained. Improvements to SSIO software systems improve the productivity of U.S. Department of Energy (DOE) scientists by facilitating the discovery of and access to their data, and such improvements reduce the cost to operate storage systems by increasing our understanding of storage system behavior and enabling adaptation in these systems.

Technological improvements are occurring at a rapid pace in the areas of memory, storage, and input/output (I/O). New, nonvolatile memory technologies

(e.g., 3D XPoint®[1] and Intel Optane®[2] dual in-line memory modules [DIMMs]), drive technologies (e.g., shingled disks, nonvolatile memory express [NVMe]), and changes in system architectures, such as the proliferation of cores on node and shifts to more cost-effective networks, all necessitate research to understand how these technologies are best adapted for use in service of DOE science. At the same time, an explosion of new types of DOE science are appearing on DOE systems — machine learning (ML) applications and the analysis of experimental and observational data being two examples. These new applications break many of the assumptions made in the past by developers of SSIO system software, and in many cases, new designs are needed to account for these new behaviors. This combination of rapid technological change and a dramatic influx of new application classes drive the prioritization of essential new research activities in the SSIO area.

To address the changing landscape and requirements for storage systems and I/O, The Department of Energy, Office of Science, Advanced Scientific Computing Program Office (ASCR) convened a workshop in September, 2018 with the following charge:

*As HPC architecture becomes more complex, the lines between what operating and runtime systems experts call memory and the emerging off-system storage hierarchy that includes solid state devices blur. These changes result in increased complexity for application developers and increased difficulty in managing the entire process for input and output. A combination of rapid change in memory and storage technology and meeting the related requirements for the range of application classes using high performance computing (HPC) must drive the prioritization of essential new research activities in the SSIO area.*

*The goal of this two day workshop is to identify key challenges and define research directions that will advance the field over the next 5–7 years.*

This workshop report captures the outcomes of the 2018 Storage Systems and I/O Workshop[3]. This document includes a summary of mission drivers, assesses the state of the art and challenges in key SSIO research areas, summarizes the workshop discussion related to six key topic areas, and identifies research directions in each of these areas. The research opportunities listed above are a high-level summary of the research directions identified throughout this report.

## 2.1 Workshop Panels and Presentations

The workshop brought together a number of experts in related fields to present their views on requirements and the impact of new technologies on SSIO going forward. These presentations and panels spanned the two days of the workshop, with the first morning session focusing on how application trends are creating new requirements

[1] 3D XPoint is a trademark of Intel Corporation.

[2] Optane is a registered trademark of Intel Corporation.

[3] https://www.orau.gov/ssioworkshop2018/

for SSIO, or application pull. Highlights from this morning session follow in Section 2.1.1.

### 2.1.1 Background: Application Pull

Wes Bethel of Lawrence Berkeley National Laboratory (LBNL) presented his take on this trend from the perspective of the DOE Office of Advanced Scientific Computing Research's (ASCR's) 2015 Experimental and Observational Data (EOD) Workshop [Bethel2015], reminding the attendees of the key findings from this workshop and mapping these into possible gaps in current SSIO capabilities. He noted that both experimental and observational science are increasingly impeded by data lifecycle challenges: massive volumes of data being produced at unprecedented rates and, in some cases, with real-time constraints on analysis; the lack of adequate provenance and other metadata to assist in interpretation and reusability; and issues related to how to make this data available to a broad audience over potentially long timelines (e.g., years or decades). He further noted the close connection between data management and workflow in this context and a need for better coordination between these systems, and he suggested that supervised learning and multi-modal data analysis will play increasingly important roles in these science endeavors.

Graham Heyes of the Thomas Jefferson National Accelerator Facility presented a view of current and future streaming data science, expanding upon the information from the prior presentation. He provided a summary of the current state-of-the-art workflows in these fields and summarized some shortcomings related to SSIO: the current methodology operates with close synchronization, missing opportunities for overlap and performing poorly in the face of work imbalance; the use of files as an intermediate form — and the batch processing of these files — forces inefficiencies in the analysis phase; and a lack of indexing and subsetting leads analysis codes to examine more data than necessary. The end result is that these workflows can take years to complete. Moving to a streaming data model is compelling in that it could allow for more flexible design of the analysis workflow, dramatically improve the parallelism of analysis, and decouple relative rates of detectors and analysis. However, in order to realize this vision, new SSIO capabilities are required. First, the storage infrastructure needs to support a more stream-based model: the rapid ingest of small records, strictly ordered, and easily referenced; guarantees on rates of ingest that ensure that data are not lost; and an ability to remove records over time and thus avoid performance degradation resulting from fragmentation. In addition, research into multidimensional data stores, streaming data stores with strong performance guarantees, and methods to closely connect with high-performance transports are all important for the success of streaming data science going forward.

Next, Tom Peterka of Argonne National Laboratory (Argonne) presented a summary of the Workshop on the Future of Scientific Workflows [Deelman2015], reminding the audience of nomenclature and a number of canonical workflow use cases from a variety of science fields. Summarizing the workshop's findings as they relate to SSIO, he noted that (1) lagging I/O bandwidth is motivating an increased use of *in situ*

workflows; (2) new storage technology and heterogeneity are seen as driving new SSIO work in the same way they drive new workflow advances; (3) the provisioning of storage resources and services needs revisiting; (4) research is needed on how storage relates to communication in data flow, or is used in that role; (5) enhancements are needed in our ability to store and retrieve provenance, including performance data; and (6) performance models for prediction are needed to assist in workflow scheduling.

Jack Wells of the Oak Ridge Leadership Computing Facility (OLCF) presented a summary of data-intensive science requirements at the leadership computing facilities, with an OLCF focus. He first noted that there has been a shift since 2013 in the priorities of their users away from floating-point operations per second (FLOP/S) and toward memory bandwidth, marking an increased attention on data. Data sharing, archival capacity, and disk bandwidth were also on the radar for users, and the facilities are adjusting their systems in response. He summarized the capabilities of OLCF's new Summit system and noted successful applications in the simulation, data analysis, and artificial intelligence (AI) areas. He noted that data is emerging as a common theme across many DOE offices, and that a capable storage system and scalable I/O are fundamental to success in many fields. He recognized that, while modeling and simulation are still a significant driver for storage system deployments, applications with read-heavy workloads, small writes, and heavy metadata use are a more significant portion of the portfolio than in the past. These changes drive a need for new solutions that perform better in read-optimized and random access modes, which are not the strengths of current solutions. Furthermore, continued work is needed to retain the model of a center-wide storage resource and to develop solutions that can provide a coherent view of SSIO resources even across facilities.

Kevin Harms of the Argonne Leadership Computing Facility (ALCF) presented a summary of drivers for storage from the ALCF perspective. ALCF is similarly supporting a broad base of applications employing simulation, data, and learning approaches. New services such as Jupyter Lab and Petrel data sharing are indicative of changing requirements from applications. These tools are facilitating new methods of interfacing to the system and allowing new methods of data sharing outside the facility. ALCF is also moving in the direction of a global storage system to hold data for extended periods, and one emphasis in design of this system is strong support for legacy interfaces (e.g., portable operating system interface [POSIX]). This support would be coupled with dedicated storage for specific platforms that is performance focused. ALCF is attentive to advances in the object storage model as well, and is collaborating with Intel on the distributed application object service (DAOS) activity, one candidate solution for dedicated storage in future systems.

Following these presentations, a panel discussed these and other topics. Participants included the speakers as well as Evan Felix (Pacific Northwest National Laboratory [PNNL]) and Kristy Kallback-Rose (National Energy Research Scientific Computing Center [NERSC]). Some important points that emerged from this discussion were the desire for alternatives to the file model including search, a recognition of the difficulty

in presenting a deep storage hierarchy in a usable manner, a request for more capabilities in the area of data lifecycle management, and an admission that we are no longer primarily supporting checkpoint/restart as our primary workload at some sites.

### 2.1.2 Background: Technology Push

The second morning focused on how technology developments are driving research into new directions (technology push).

Lucy Nowell of DOE's ASCR summarized the outcomes of the 2018 ASCR Extreme Heterogeneity Workshop. This workshop was designed to identify the new algorithms and software tools needed from basic research to enable facilities to support work on the SC program's grand challenge problems. Extreme heterogeneity is defined to capture massively increasing parallelism, the heterogeneity of data movement and increasing levels of hierarchy, the heterogeneity of performance, the increasing diversity of memory and storage technologies, and the increasing diversity of applications and user requirements. In this context, the participants recognized (1) that mapping workflows onto complex hardware/software storage resources will be a challenge, (2) that a greater degree of autonomous operation will be needed to make best use of storage and I/O resources in the face of rapidly changing workloads and operating conditions, and (3) that new data organizations are called for in order to meet the needs of future analytic workflows, including streaming workflows.

Gary Grider of Los Alamos National Laboratory surveyed storage and data trends likely to drive changes in SSIO solutions. He emphasized the role that economics plays in the architecture of storage systems, influencing the technologies that are used to provide bandwidth, capacity, and resilience in what today are typically hybrid solutions. He noted the shrinking gap between solid-state and hard disk storage and considered whether this trend would lead to another high-level shift in architectural models. He further noted that the shift in emphasis from being primarily simulation focused to being inclusive of experimental and observational data (EOD) may lead us to architectural adjustments as well, for example in response to rapid reprocessing workloads. He also suggested a need to revisit the problem of resilience, and noted that traditional single-tier approaches may not provide the guarantees needed in a multitier context, but that solutions are being developed both inside and outside of our community. As methods to address the growing volume and multitier nature of our storage systems, Gary suggested another look at name spaces, search, and access methods. Finally, he noted that the NVMe push is one of the fastest growing sectors that we have seen, and that push is bringing with it new interconnect-storage capabilities, performant user-space access methods, tight coupling to compute, and other changes that provide opportunities for the success of approaches that have not been viable in the landscape to date.

Dan Ernst of Cray, Inc., looked at the blurring lines between storage and memory in current and future systems, bringing a memory/node architecture perspective to the

meeting. He discussed a number of memory technologies and considered viable configurations using multiple technologies for a simulation-focused platform, recognizing that power, bandwidth, and access density all have a significant impact on the viability of specific point solutions. He also emphasized the increasing importance of global visibility of data (across the platform) in our current era of diverse applications and coupled workflows, and noted that this naturally draws the interconnect into the picture as well, both in terms of the ability to directly access particular technologies but also the rates at which the network must communicate in specific configurations. This model leads to a possible future where new global storage resources supplant the traditional role of the parallel file system as the next layer of storage beyond application memory.

Following these presentations, a panel consisting of Gary Grider (Los Alamos National Laboratory [LANL]), Kevin Harms (ALCF), Eric Pouyoul (ESnet), Dan Ernst (Cray), and Lance Evans (Cray) discussed the implications of technological change in greater depth. Alternatives to block-based storage were of great interest to the panelists and audience, with no obvious consensus on whether to replace block or what to replace it with — this appeared to be an area ripe for new ideas. The topic of offloading functionality into the network interface card (NIC) was also discussed as a reflection of hardware specialization that could benefit SSIO.

## 2.2 Report Organization

The remainder of the report is organized as follows. Section 3 summarizes the DOE mission requirements for SSIO. Section 4 includes in-depth discussion of the state of the art and challenges to be addressed to advance SSIO technologies in support of DOE missions. Section 5 addresses some supporting activities that were identified as helpful in enabling successful research and development (R&D) in the SSIO area.

# 3 Mission Drivers

For the 2014 SSIO workshop series, the workshop organizers invited input from a group of distinguished scientists developing application codes for next-generation and future exascale DOE platforms. These scientists are involved in a wide range of DOE Office of Science (SC) and National Nuclear Security Administration (NNSA) mission-critical applications. The scientists reported on anticipated scientific challenges and how SSIO capabilities might enable them to meet these challenges. Many of the discussions were based on work from the NNSA Advanced Simulation and Computing (ASC) code teams, as well as the Scientific Discovery through Advanced Computing (SciDAC) co-design centers [CoDesign2016] that were charged to ensure that future architectures are well suited for DOE target applications.

Since then, numerous workshops and requirements reviews have considered technology changes and storage and I/O requirements for DOE-SC science teams using large-scale computing resources, including:

- The 2015 Workshop on Management, Analysis, and Visualization of Experimental and Observational Data [Bethel2015].
- The 2015 Workshop on Integrated Simulations for Magnetic Fusion Energy Sciences [Bonoli2015].
- The 2016 Streaming Requirements, Experience, Applications and Middleware Workshop (STREAM2016) [Fox2016].
- The 2015–2016 Exascale Requirements Reviews for High-Energy Physics (HEP) [Habib2015], Basic Energy Sciences (BES) [Windus2015], Fusion Energy Sciences (FES) [Chang2016], Biological and Environmental Research (BER) [Arkin2016], Nuclear Physics (NP) [Carlson2016], and ASCR [Vetter2016], which are summarized in the 2017 Crosscut Report [Hack2017].

These workshops and requirements reviews contain the combined expertise of vendors, hardware architects, system software developers, domain scientists, computer scientists, and applied mathematicians who are at the forefront of anticipating features and trade-offs in exascale hardware, software, and underlying algorithms.

This section builds on the material from the 2014 workshop series, augments that material with findings from these subsequent events, and also incorporates extensive knowledge from the organizing committee on numerous applications from fusion energy, materials science, climate science, accelerator physics, and other domains.

## 3.1 Overview

Historically, the DOE-SC computing facilities have served science teams employing simulation codes to better understand phenomena of interest to the U.S. Department of Energy. However, more recently, teams have begun to investigate the efficacy of using these resources for new computational tasks, including analysis of large datasets from scientific instruments and the training for machine and deep learning (Figure 3.1). In fact, *all* of these computational tasks can be considered "big data" science relevant to DOE missions, in that they have significant data requirements in addition to significant computational needs. In the subsequent discussion, we refer to this as *data-intensive science*. In this section, we give a quick rundown of their characteristics in terms of the common nomenclature of the five Vs: volume, velocity, variety, veracity, and value.

**Volume.** Much of the total volume from HPC applications comes from checkpoint files. Most of this information is written once, and almost never read in. With the inclusion of "burst buffers" in modern systems, simulation applications are demonstrating an ability to write large volumes of checkpoint data efficiently. However, many scientists are expressing a growing need to understand more of the "physics" in their simulations; simple data reduction techniques are becoming insufficient for proper analysis. Users of codes such as the XGC1 [Ku2006] simulation, one of largest users of leadership-class facilities (over 300 million hours at Argonne, NERSC, and Oak Ridge National Laboratory [ORNL] in 2015), have launched a series of simulations that need to write out 100 petabytes (PB) of data in order to capture all of the turbulence data for runs on the Titan system [ORNL2015], the prior OLCF platform. Limitations in available space and off-system bandwidth can cause scientists to miss important artifacts and opportunities for discovery. Large Hadron Collider (LHC) data output is expected to grow from approximately 10 PB/year to 150 PB/year by 2025, with an additional 600 PB/year of derived data produced by the community [Habib2015].


Figure 3.1: The DOE-SC computing portfolio has diversified in recent years, beyond the traditional simulation focus to include significant numbers of activities employing big data analysis and machine and deep learning techniques to accomplish mission goals. Image credit: B. Helland (ASCR).

**Velocity**. Higher velocities (i.e., rates of data generation) will be seen from many leading applications because of the nature of compute accelerators and nodes with high core counts on certain next-generation systems. Simulations such as the QMC code [QMCPACK2015], which use quantum Monte Carlo techniques to understand material properties are already investing in *in situ* data reduction and analysis given that they are generating more than 2 terabytes (TB) of data from their simulations every 10 seconds (s). Similarly, next-generation experiments such as ITER [Lister2003] will begin to generate data at over 2 PB/day. More recently, the FES community has identified near-real-time data analysis as a key enabler of decision-making; this approach depends both on effective management of high-velocity data and also on the training of machine learning models on large scientific datasets [Bonoli2015]. Similar requirements are expected in EOD-based science. For example, the Large Synoptic Survey Telescope (LSST) is slated to come online in 2019, eventually generating 3.2 Gpixel images on a 20-second cadence, with a requirement to provide initial analysis of these images within a minute of image capture [LSST2009].

**Variety.** Simulations are producing a wide range of variables in their output that they later need to correlate and understand together. We see this situation from climate simulations, among other leading-edge simulations. Generally, on each process each variable is "small," but there are often hundreds of variables, which create many new challenges when they are generated from high levels of concurrency. One implication for the SSIO community is that metadata will continue to increase as the variety of data increases, and the management of large amounts of small data will become increasingly important, including the ability to organize or find and quickly access this large variety of data.

**Veracity.** Data integrity has become a critical part of the simulation workflow, and many application teams are focusing on some aspect of uncertainty quantification (UQ) [Carey2014, Najm2003, Reagana2003]. These simulations are using either intrusive UQ techniques (e.g., in combustion) that could potentially generate zettabytes of data, or they are employing non-intrusive techniques (used in many of the NNSA applications) and creating new I/O and storage use-cases (described in Section 3.2). Data need to be moved and processed with this integrity information in hand for subsequent analysis. In the case of stockpile modernization efforts, quantification of uncertainty in simulations is *essential* as experimental verification and validation are no longer available.

**Value.** As we reach the age where simulations cannot output as much data as they would like (e.g., the XGC1 simulation, see "Volume"), many choices must be made to understand which data products will have later value. Among other characteristics, the value of the data is also affected by how much it can be reduced for fast post-processing. Historically, one of the common themes voiced by application scientists is that once data go to archival storage, they are rarely read again because of the time required to access those data. However, this is not universally true, and new storage tiers, such as object-based tiers [Inman2017], have the potential to keep more data

readily available, while new research is investigating how to retain the provenance of the data and understand what the different variables may contain.

## 3.2 Scientific Workflows

In Flynn's taxonomy for classifying computer programming paradigms [Flynn2011], traditional monolithic simulation codes are known as single instruction, multiple data (SIMD) programs. The common alternative to SIMD is the multiple instruction, multiple data (MIMD) paradigm, where a number of different tasks are executed at the same time in a large parallel job. These tasks may be expressed as separate executables and may each be handling a different aspect of a complex model, such as chemistry and fluid dynamics in a turbulent combustion model. Alternatively, the different tasks may form a simulation program plus a number of different *in situ* analysis programs, simulations coupled with analysis routines that compare the outputs from the simulations, or simulations being compared with experimental observations. In the context of EOD analysis and in learning applications, these tasks may be different phases of data triage and training or analysis. Another way to describe these MIMD applications is as a *scientific workflow*, also described as an *in situ* workflow [Deelman2018]. A high-level view of these is provided here, and more detail is included in Section 4.3.

In many of these cases, there is a stringent demand on communicating a large amount of data from one task to another, often accomplished via the storage and I/O system [APEX2015]. Some of these tasks produce a large amount of data to be stored persistently. In addition, some workflows may also read a large amount of input data, such as the initial condition required at the start of a simulation, the boundary conditions needed at every step of the simulation, a large corpus of training data, EOD for comparison against simulation results, or for other purposes. Application scientists identify a number of characteristics of these scientific workflows; we briefly highlight three:

- **Homogeneous tasks.** An important class of scientific workflow is one made up primarily of a large number of homogeneous tasks. One example is a set of independent tasks in a UQ run, where each task is using the same executable but with different input parameters. Another common example is an ensemble run of climate models where each instance of the ensemble uses a different model. Of course, and alternatively, UQ jobs and climate modeling runs could easily be composed of different executables that each perform a different set of operations.
- **Long-running services.** Certain tasks in a scientific workflow may need to be run as persistent services. For example, a number of simulation programs involve complex materials, and the large number of chemical reactions and information about these chemistry processes could best be captured in an equation-of-state service, or more generally in a computation caching service (e.g., [Jenkins2017]). Other workflows may benefit from data services such as multidimensional or key-value data stores (e.g., [Greenberg2015],

[Zhang2017]). However, existing large-scale systems typically execute in batch mode, where all executables are terminated when the batch job terminates, and persistent services need to last beyond the end of any single batch job. Consequently, supporting long-running service tasks will require supercomputer centers to change the prevailing mode of operations. Such a change would also benefit long-running data analysis services.

- **Composition.** The approach of connecting different tasks into a larger structure is used extensively for large-scale distributed data analysis and learning applications, and is beginning to be employed in parallel simulations. Considerable work will be needed to develop and refine workflow composition, scheduling, and execution tools for use on future HPC platforms. A large workflow is likely to produce and consume data in a variety of ways. It may also utilize the I/O system to carry information among the workflow components and therefore impose strong performance requirements on the SSIO systems. These workflows will almost certainly have a new type of I/O where different nodes write large amounts of data from one component often at the same time as other components, which can increase the I/O variability.

### 3.2.1 Simulation Example: Adjoint-based Sensitivity Analysis

To examine the I/O operations in more detail, we next consider a UQ workflow, a combustion simulation program from the Center for Exascale Simulation of Combustion in Turbulence (ExaCT), one of the SciDAC co-design centers. It employs a UQ approach known as adjoint-based sensitivity analysis, an optimal approach for the direct numerical simulation in combustion [Carey2014]. A key challenge of the adjoint workflow for time-dependent applications is the storage and I/O requirements for saving the application state. During the time-reversal portion of the workflow, the forward state is required in last-in-first-out order. To avoid storing all the states, the co-design team developed an approach of regenerating the states from checkpoints.

This approach dramatically reduces the total volume of stored data, allows the caching of state in the regeneration window in memory and on local solid state disks (SSDs), may accelerate the application execution by reducing output frequency, and reduces the power overhead from I/O. For example, a number of checkpoints that are hundreds of time steps apart may be stored on disk. During the time-reversal phase, the application uses the checkpoints on disk to restart the computations, generates the intermediate states, and stores those intermediate states on local SSDs. Because the intermediate time steps are not written back to global storage, this approach reduces the I/O time. Because it does not recompute all the time steps, this approach also reduces the amount of computation. The researchers in this project are particularly concerned with the cost of data recomputation as compared with the cost of storage (e.g., write the data, and then read the data a little later because of the limited memory on the system). This is a specific example of the more general trade-off between storing code and recomputing (where possible) versus storing data; and

as FLOPs become cheaper, this ratio of cost of recomputation vs. write/read will change.

In this use case, the application scientists are also using two techniques to reduce space requirements, and these techniques also affect the I/O operations. The first technique is to replace the simple uniform mesh used in earlier simulations with an AMR (adaptive mesh refinement) mesh [Berger1989]. The AMR mesh is dynamically adjusted to place more mesh points in regions in the simulation domain where the quantities of interest are varying quickly. This approach allows more mesh points to be used in regions that need a higher resolution and can reduce the overall number of mesh points used in the simulation. However, the simulated quantities are stored in more complex structures as compared with the original uniform mesh. The second technique used by application scientists is to concentrate on "regions of influence" for sensitivity analysis instead of computing on the entire simulation domain. This strategy again reduces the amount of computation performed during sensitivity analysis; however, because the regions of influence can be of arbitrary shape, additional data structures are needed in order to keep track of the domain of sensitivity analysis computations. Both techniques have implications for how SSIO technologies can best support science data storage.

### 3.2.2 EOD Example: The Compact Muon Solenoid (CMS) Experiment

The CMS experiment [CMS n.d.] at the Large Hadron Collider [CERN2019a] is an international collaboration focusing on the Higgs boson and new physics beyond the standard model. The CMS detector is a general purpose particle detector that captures particle collisions within the LHC at a rate of up to 40 million raw detector readouts per second; these initial readouts are filtered at the detector down to a stream at 600 Hz. The design of the detector enables scientists to reconstruct the paths of particles passing through the detector with great precision.

Storage and I/O requirements for CMS are detailed in [Bockelman2018]. The detector has generated 90 PB of raw data to date; however, simulations and analysis of these data have resulted in a volume of additional derived datasets that is twice as large as the raw data. Simulations are used to understand properties of the detector and also to assess the quality of reconstruction algorithms: in the latter case, synthetic events are generated and the output of the detector simulated, then the reconstruction algorithm is employed: output of the reconstruction algorithm is then compared to the known "truth" generated by simulation. In the case of detector analysis, a simpler multiphase workflow reconstructs "tracks" (i.e., paths of particles) from detector data and aggregates results from many such reconstructions, and later workflows fit models to the aggregated data.

The first dataset required by most workflows is the physics software itself: this is a complex software package often deployed using FUSE-based technologies (i.e., CVMFS [Blomer2015]) and can be as large as 10 Gigabytes for a release. Raw detector data and derived data products are typically managed using the ROOT

software package [Brun1997]. ROOT files store a sparse tabular structure of events, each with a set of associated objects. An event is typically described with a few hundred objects, each with a few thousand attributes. Additional information on the run, statistics, and provenance is also typically included to allow for reconstruction of the analysis steps. This complex structure has proven very useful in providing the flexibility necessary for scientists to perform a wide variety of tasks using the same underlying data model. ROOT files are typically stored directly on a file system (e.g., a parallel file system in the HPC context), and many such files might be independently analyzed as part of a large analysis workflow. For performance reasons, output files are often merged in memory before writing. Due to the size of the data managed as part of CMS activities, compression is critical to make the best use of hardware investments: this is further facilitated by merging.

It is estimated that the High Luminosity LHC (HL-LHC) [CERN2019b] upgrade, to begin taking data in 2026, will generate between 5 and 30 times more data than has been currently produced, and it will take significantly more time to run reconstruction of events [Bockelman2018, Albrecht2017].

### 3.2.3 Learning Example: Low-Level Whole-Detector Data Analysis at the LHC

Researchers are also exploring the application of learning approaches to data analysis for the LHC. In [Bhimji2017], researchers apply deep neural networks directly to detector data with the goal of identifying new massive supersymmetric ('RPV-Susy') particles in multi-jet final states. Prior analysis methodology and results are described in [ATLAS2017]. In this work, the researchers use the Pythia event generator [Sjostrand2008] and Delphes detector simulator [DeFavereau2014] to generate event data for training, using a training sample of approximately 400,000 events and finding the approach compares favorably to traditional approaches.

The availability of tools such as Pythia and Delphes allow for the generation of sufficient training data, and commodity learning packages and libraries facilitate rapid development of classifiers. In this case, Tensorflow [Abadi2016] was employed on the Cori system [NERSC2015a] for training. While just one example of the application of learning algorithms to DOE mission science, it is indicative of a growing interest in using large-scale computing systems in conjunction with learning algorithms as a tool for understanding complex datasets.

### 3.2.4 Code Coupling Example: Fusion Whole Device Modeling

Tokamak physics is an important area in fusion plasmas which attempts to model across many orders of magnitude in space and time. In this scenario, a code that models core transport in tokamak cores is coupled with a code that models wall physics and edge plasma–wall interactions. The coupling of core and edge simulations is necessary for more accurate whole device modeling. As Figure 3.2 shows, this scenario illustrates the new and growing practice of integrating, executing, and

instrumenting a variety of tasks in a workflow for on-line analysis and coupling. In addition to exchanging data through code-coupling software, each simulation code is coupled to a variety of analysis, visualization, and data reduction codes. The feature extraction and visualization codes are used to detect and monitor simulation output. Because these simulations can require a long time to execute, the coupling with analysis and visualization codes step needs to be performed dynamically. That is, scientists should be able to dynamically start running a group of analysis and visualization codes and attach them to output from the simulations.

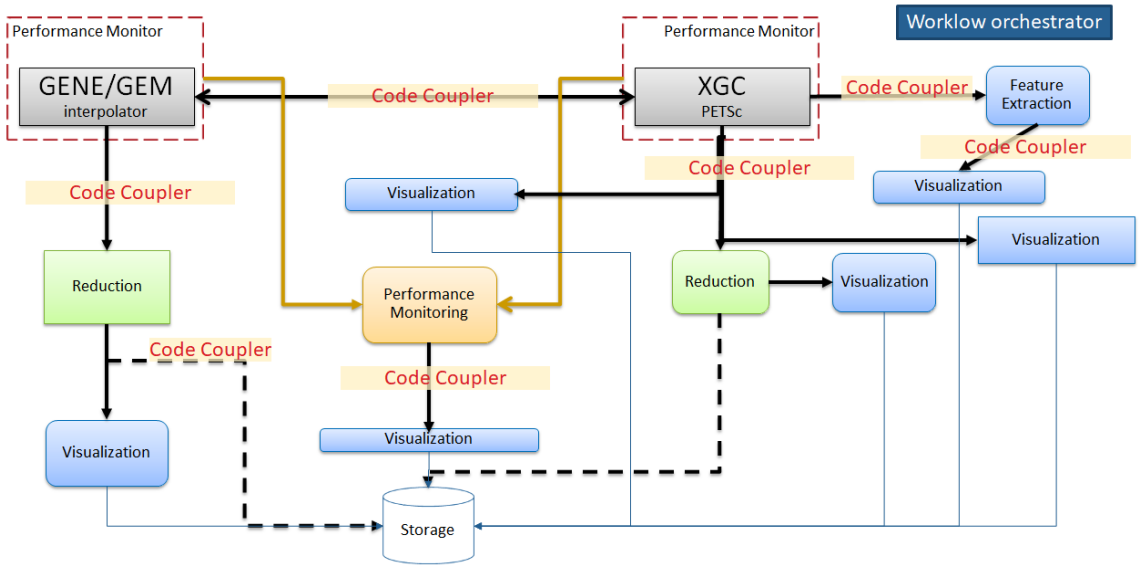


Figure 3.2: The schematic illustrates a complex set of processes to couple two fusion codes on an LCF, along with reducing, analyzing, and visualizing the results.

The amount of data generated by these simulations on extreme-scale machines often exceeds the amount of space allocated to a scientific campaign. Moreover, analyzing all of the data at the highest resolution can require significant computational power and render near-real-time or interactive rates infeasible. Hence, data exchanged between simulations and analysis codes as well as output directed to the storage system may be coupled to reduction methods to reduce data volumes.

This type of online coupling also illustrates the need to monitor performance to make adaptive changes in resource allocation and data management strategy when running a complex workflow. As these simulations are executed over long periods of time, the computational workload of individual simulations may change over time. In addition, dynamic coupling and decoupling of analysis codes and visualizations introduce variations in the workload.

## 3.3 Input/Output Characteristics

To summarize the I/O characteristics of the representative applications, we first consider common use cases involving file systems. We then describe the more advanced uses involving deep memory hierarchies, *in situ* data exchanges, and selective access to data.

### 3.3.1 Simulations

Most of these applications require a modest amount of input data at the beginning of a simulation run, along with data that may be read in when they continue from a previous run. The input data typically contain parameters defining the simulation's initial conditions to be used in the differential equations that represent the evolution of the variables being simulated. In such cases, this input data may be shared among the processors. Having immutable storage specifically for such input files could reduce the I/O operation overhead and improve the overall application's performance. In addition, as many simulations begin to validate their solutions against experimental or observational results, data must be read in from the different experiments in order to ensure that the simulation is "realistic" for the given conditions. For example, in many fusion experiments, data from the many diagnostics on fusion devices are ingested at the beginning of a simulation. As time progresses, the fusion reactors continue to grow in size and more diagnostic instruments are built into the reactors, where each instrument is capable of collecting data more quickly than before. Together, the data collected from the experiments increases, and the data passed to the simulations will also grow.

Simulations produce many different types of output data. We generally categorize them into two types: *defensive output* for error recovery and *productive output* for scientific objectives. A typical defensive output is a global checkpoint file (or set of files), where a globally consistent state of the simulation is written to persistent storage to assist in restarting the application should it terminate before completion. A productive output can be just the output of the current state from a fusion experiment, which is derived from the magnetic field vector from the simulation. In many cases, the defensive output files are also used as productive output, because all of the data (e.g., in a combustion simulation such as S3D) are necessary for full data analytics.

Because checkpoint files contain all the information necessary to regenerate the entire state of the simulation, whereas the productive analysis output needs only to summarize key features of the simulation, defensive output is generally larger than productive output. For example, many fusion scientists using particle-in-cell (PIC) techniques [Dawson1983] might write out the cell data frequently in order to understand the "fluid" effects of the physics. The kinetic (particle) information is much larger and is often written infrequently because of the size of the data.

In applications that use checkpoint files for analysis, current codes can generate petabytes from a single run, and future runs may produce a large number of such files, cumulatively totaling exabytes in size. For those whose checkpoint data are productive as well as defensive, application users often adjust the frequency of checkpointing based on the expected analysis needs, rather than based on error recovery needs. They frequently produce more checkpoint files than the "optimal" rate recommended for error recovery [Daly2006]. Application scientists also adjust the frequency of checkpointing to limit the I/O time to a relatively small fraction of the total execution time. Most existing simulation codes perform their checkpointing operations by directly writing data to files instead of using a checkpointing library.

As parallel computers grow in size, there is considerable interest in moving away from writing checkpoints to a global, parallel file system (Figure 3.3). Hybrid checkpointing schemes, such as the Scalable Checkpoint/Restart (SCR) library [Moody2010] and the Fault Tolerant Interface (FTI) [Bautista-Gomez2011], are gaining acceptance among application scientists.

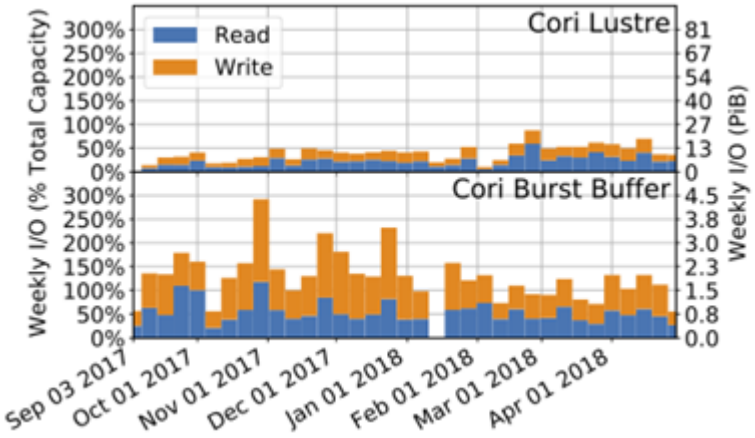


Figure 3.3: While traditionally, we have expected HPC storage systems to be dominated by checkpointing writes, some centers project that they will have a more balanced workload. Data from Cori supercomputer at NERSC, both for the Lustre filesystem (top) and the DataWarp burst buffer (bottom) [Lockwood2018].

So far, the discussion on I/O operation has touched only on bulk data operations. Alongside these operations are common operations involving metadata, such as provenance retrieval. In most cases, such metadata operations involve a relatively small volume of data and do not take a significant amount of time. However, a complex simulation may generate a large amount of metadata, especially when the simulation consists of a large ensemble of relatively small tasks. Metadata are discussed further in Section 4.2.

### 3.3.2 Data, Learning, and Hybrid Applications

DOE also supports a number of important applications that exhibit different data access patterns from the more traditional pattern described above. For example, earth system models frequently assimilate observational data into their simulations, and high-energy physics collision simulations often incorporate calibration data of

accelerators. In a number of other use cases, data analysis operations fuse simulation data and experimental observations, and this data analysis also requires reading a large amount of experimental data while the simulation is progressing. Data and learning applications are often very read-intensive, in many cases operating on datasets consisting of many modest-sized files. The topic of streaming data is further discussed in Section 4.6.

### 3.3.3 Initial Impact of Nonvolatile Storage

Modern machines have nonvolatile random access memory (NVRAM) that can be used for storing checkpoint files and frequently used input data (e.g., through systems like DataWarp [Henseler2016]). NVRAM allows the checkpoint data to be written more quickly. When the checkpoint data can be discarded, this approach clearly reduces the traffic to the relatively slow disk storage systems and significantly improves the I/O time.

When multiple tasks in a parallel job share data, the data transfer may be conducted through in-memory mechanisms instead of through the parallel file systems. One realization of this is through *in situ* data analysis systems, which will be discussed in more detail in Section 3.4. Another common issue is that the analysis may require only a portion of the data instead of the whole dataset—for example, only those data records in the region of influence mentioned in Section 3.2.1. These selective data accesses could be made more efficient through techniques such as indexing. However, most existing checkpoint files or checkpointing libraries do not yet support indexing. The topic of nonvolatile storage is further discussed in Section 4.4.2.

## 3.4 Implications of *In Situ* Analysis on the SSIO Community

Many large-scale scientific simulations routinely write out immense amounts of data on today's HPC systems, such as in the case of XGC1 writing 100 PB of data per run on the Titan platform. Such "big data" impose steadily increasing pressure on the SSIO systems. In fact, I/O is now widely recognized as a severe performance bottleneck for both simulation and data analysis, and this bottleneck is expected to worsen with an order of magnitude increase in the disparity between computation and I/O capacity on future exascale machines.

In order to mitigate the I/O bottleneck, leadership scientific applications (e.g., XGC1, QMCPACK, S3D, hardware/hybrid accelerated cosmology code [HACC]) have begun to use *in situ* data analytics, in which analytics are deployed on the same platform where the simulation runs, with simulation output data processed online while they are being generated. Compared with conventional post-processing methods that first write data to storage and then read it back for analysis, *in situ* analytics can reduce on-machine data movement and disk I/O volume and can deliver faster insights from raw data [Klasky2011, Ovsyannikov2016].

Incorporating *in situ* analysis and visualization poses many challenges for applications. Arguably, however, essentially *all* application scientists already use their home-grown *in situ* analysis in codes. Scientists routinely create derived variables from a combination of their fundamental variables and then perform different analyses (Fourier, feature finding routines, addition of Lagrangian particles to understand flows in simulations, etc.). The question that is generally posed to scientists—"What will you do when you can't write as much as you want, because of architectural changes?"—has existed since the advent of supercomputing. The fundamental change that applications are now seeing is not the inclusion of *in situ* analysis but rather the inclusion of "computer science codes" developed outside the application team for use during *in situ* analysis. Scientists are wary of including other code in their simulations for good reasons.

Today, several *in situ* visualization and analysis services are being used in applications. The Adaptable I/O System (ADIOS) [Lofstead2008] is an I/O framework that allows applications to use I/O staging (on-node, off-node, off-machine) and run different executables. GLEAN [Vishwanath2011] uses a similar methodology in order to execute analysis pipelines. Catalyst allows users to embed analysis routines into their simulations, which then call VTK/Paraview [Henderson2004] code, which is similar to LibSim [Whitlock2011]. Each of these frameworks has trade-offs, and more research is necessary to understand how to best provide needed data services in support of exascale science.

## 3.5 Data Organization and Archiving

Many application programs running on the current generation of supercomputers still write data in custom formats. However, many data files being shared by large scientific projects are using popular file formats such as ADIOS BP [Lofstead2008], Hierarchical Data Format 5 (HDF5) [Folk1999], and netCDF [Rew1990] (Figure 3.4). Some applications teams use popular file formats because of their convenience and portability, while others use a custom data format for performance reasons or to minimize code dependencies. Other data organizations, such as databases and key-value stores, are being investigated in response to different scientific models and use cases (e.g., [Docan2012, Greenberg2015]).

This diversity of data organizations and needs creates challenges for our community that must be addressed for future architectures. One of the biggest challenges is how to integrate new solutions into many of the leading DOE applications. In particular, how do we take current I/O solutions and improve the performance for common I/O tasks, without having to customize them for each application? Alternatively, how can we streamline the process of service customization?

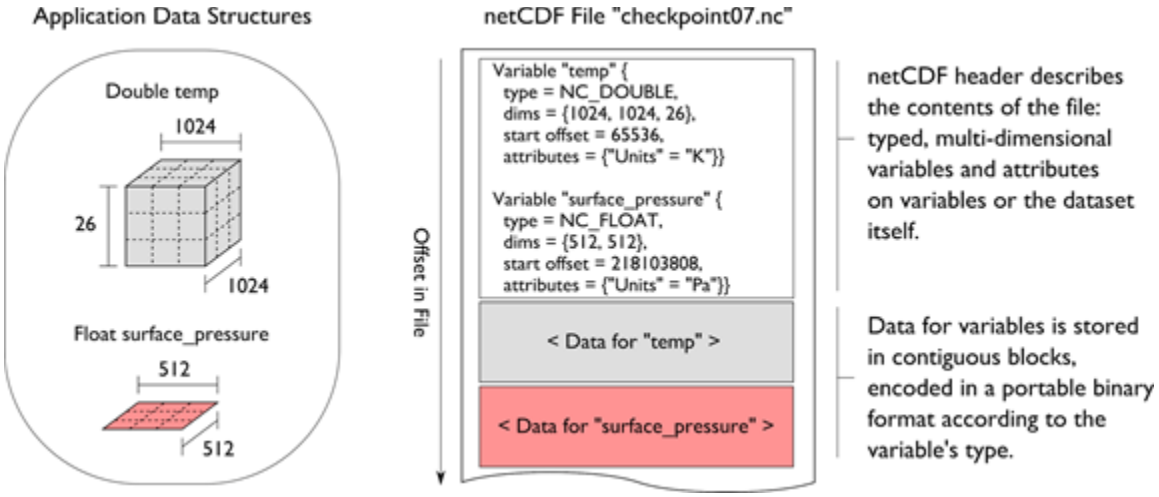


Figure 3.4: Self-describing file formats, such as the netCDF format shown here, capture not only array data but also structural information such as names, units, types, and dimensions.

The application teams commonly argue for application-specific forms of compression as part of the I/O routines. Asynchronous *in situ* techniques are being explored to decouple the I/O application performance from the storage system. I/O variability is also an important phenomenon that greatly affects the applications' ability to write effectively, deterministically, to the file system.

Using a well-supported, high-level I/O library facilitates the sharing of data among a large community of scientists. Professional software development efforts could be directed to build high-quality data analysis tools using such I/O libraries. For example, the climate community is using a large set of data analysis tools on petabytes of netCDF files [Williams1997]; the high-energy physics community is using a highly effective data analysis environment based on ROOT files [Brun1997]; and many researchers in the fusion community have used ADIOS-BP to exchange many petabytes of simulation data [Lofstead2008]. These shared I/O libraries are also making it easier for applications to read and write a large amount of data in parallel. Challenges exist to ensure that the "schemas" from different communities remain standard, so that data can be easily converted among the common file formats. Such standardization will reduce the need to develop customized data readers for data analysis and visualization.

Because high-quality, efficient I/O libraries reduce the amount of programming effort needed to handle I/O operations and facilitate exchanges of data in large user communities, they will be an essential component of any exascale software stack, and three exascale software activities are targeting this challenge [ADIOS n.d., DataLib n.d., ExaHDF5 n.d.]. For these libraries to be adopted effectively in upcoming high-performance computers, the following issues must be considered:

- **Performance.** The I/O system must be highly efficient in a wide variety of use cases: uniform, semi-structured, and unstructured meshes; particles; and events. These records could be organized in a variety of ways (e.g., arrays, trees, networks). Furthermore, the system must have efficient read and write operations for all of these use cases, not just one.
- **Scalability.** A successful I/O library for exascale computers must be efficient at supporting different job sizes, ranging from a few nodes on the machine to the whole machine. The library needs to make efficient use of the different architectural variations available within the federated machine. These different architectures either place more resources (e.g., memory, computational power, NVRAM) on each node (scale-up) or utilize more nodes with fewer resources on each node (scale-out). Each option has its own SSIO challenges.
- **Resilience.** Given that persistent data files are the key results of many important activities, the integrity of these data files must be unimpeachable. This requirement plays an important role in the adoption of new file formats. As the data sizes increase, files must still be readable even when a portion of the data is not reliable. Having methods to ensure that data stored in these formats can withstand failures is critical for future adoption, to the degree that file systems fall short in providing this guarantee. This protective stance is the strategy taken by many Monte Carlo simulations, such as QMCPACK and in some PIC simulations.
- **Compression.** Some forms of compression are already supported by the current generation of I/O libraries and can be effective in reducing the output size as well as reducing the I/O time. Both lossless and lossy compression methods could be used to reduce the I/O cost. When lossy compression is used, it is highly desirable to provide ways to quantify the loss introduced by the compression. However, because the impact of compression typically depends on the analysis operations to be performed, it is challenging to be able to quantify the impact without knowing the analysis to be performed after the data files are produced. Compression techniques must also be very fast in order to keep up with the high data velocities being presented. For example, in the QMCPack application, 2 TB of data are produced on 8K nodes every 10 seconds, which means that 256 megabytes (MB)/node must be compressed and written to the storage system every 10 seconds. Because generally the data at this scale may overflow the performance tier storage devices, compression must be very fast in order to greatly reduce the I/O overhead, and it must be significant in cost savings (e.g., 10 times less data) in order for it to be relevant to application science. Data reduction can also be achieved by selectively reducing the spatial, temporal, and numerical resolution of the data saved for later analysis, often without compromising the value of the data.
- **Function shipping.** As more analysis operations are added to a simulation, some analysis tasks may need to be deferred or sent to another computer. In such cases, the I/O systems may need to record the analysis operations and execute these operations at a later time or place. In addition, the storage

system may need to present a notion of locality, so that other software can co-locate analysis with data.

Outside of demands on I/O libraries themselves, many of the large scientific projects keep only relatively recent data on disk, while keeping older data records on tertiary storage systems such as HPSS [Watson1995]. Data stored on disk are often considered to be online because the disks can be accessed with common I/O libraries, whereas data in tertiary storage are considered offline because an extensive data transfer process must be undertaken before the data are usable by a data analysis program. Typically, online data are available in milliseconds, whereas offline data may require hours to become available. Such a gap is a tremendous barrier for user access to data in tertiary storage and thus a challenge to analyzing data stored in this way.

A number of application scientists have expressed the desire for a near-line storage system with latencies much lower than their expectations for the offline storage. Such a system would increase scientific productivity by allowing the scientists to access a larger amount of data for a longer period of time even though the access might be somewhat slower than true online storage. Early examples of systems such as this are deployed today (e.g., MarFS [Inman2017]). This feature might be particularly useful for large scientific experiments with highly valuable data and large user communities. The challenge for the SSIO community is to understand how best to organize data across the many tiers of storage.

### 3.6 Metadata and Provenance

Metadata is commonly divided into two broad categories: *structural metadata*, which concerns the design and specification of the data structure, and *descriptive metadata*, which comprises all other associated information such as creator, meaning, intended uses, provenance, associations, and context. Historically, such metadata was captured in handwritten entries in laboratory notebooks. Many attempts have been made to capture metadata automatically. So far, the most well-known success stories in HPC are the self-describing file formats used to capture the bulk of scientific data. These contain not only the arrays of raw data but also the structural information about the arrays, such as their names, data types, and array dimensions. Because of the diversity of descriptive metadata, however, attempts to capture this information automatically have not produced widely adopted tools.

Agencies have begun to require the data displayed in research publications, such as data used to create a graph, to be accessible in order to support validation of results [SC2016]. This policy has been translated into a number of efforts to automatically capture provenance information, which describes the origin and history of a data object or a dataset. Provenance information could also make the validation of results more likely. However, certain important details of the data generation process, such as compiler optimization flags used to generate the executables or floating-point rounding properties, are frequently omitted from the provenance information, thus

frustrating potential replication efforts. In certain highly regulated computing environments, it may be possible to require that all such information be documented precisely and all executables run under the same workflow management system; however, in general, it is not possible to force all users to develop and run their programs in the same programming environment. Automatic capturing of provenance information about a MIMD program and its runtime environment remains an open research topic.

Because metadata can grow to be extremely large, we must understand what needs to be captured and what can be discarded, so that resource constraints can still be met (Figure 3.5). For example, scientists might wish to capture the different types of algorithms used for analysis to help them understand accuracy vs. power trade-offs, leading to huge amounts of performance information captured on each process. In the combustion UQ use case, capturing the regions of interest that then help identify regions of influence is critical for a complete understanding for the final behavior of the system [Carey2014]. Metadata also grows when hundreds of variables are involved, as in the XGC1 case, and researchers prefer to keep information at the granularity of message passing interface (MPI) processes.

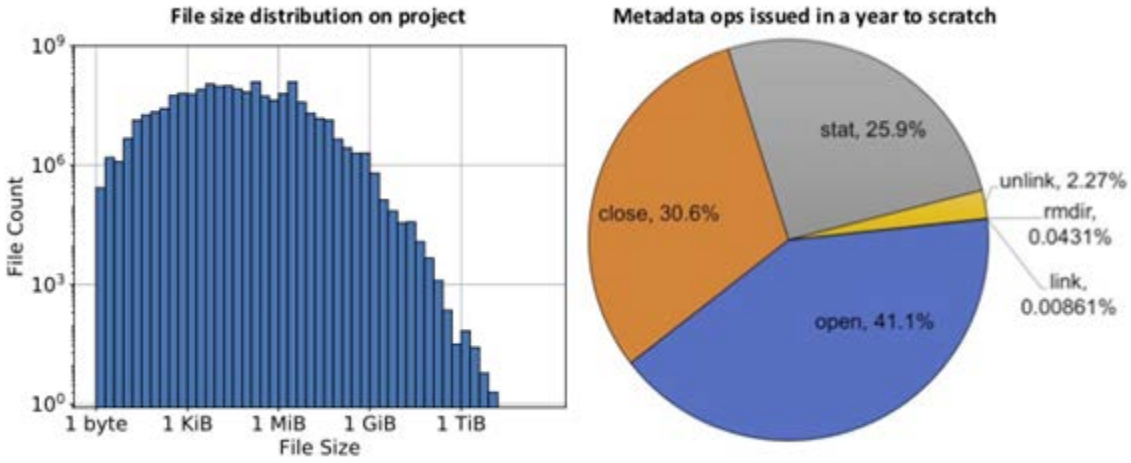


Figure 3.5: Distribution of file sizes and metadata operation counts on NERSC Edison file systems from June 2016 to June 2017. The presence of a massive number of modest-sized files drives an increased focus on metadata operations and performance [Lockwood2017b].

Scientists generally want full control over their data, and this desire also extends to metadata. As a result, many applications scientists have developed their own ways of capturing and encoding their metadata. However, the pressure to produce verifiable provenance information may lead many more of these application scientists to use automated tools. In order for such a tool to be adopted by scientists, it must be easy to use and allow sufficient flexibility for users to specify exactly what information to capture and store. In addition, the following features are strongly desired:

- The provenance capturing system needs a durable way of associating the metadata with the data. Under current data management systems, when data files are moved, the associated metadata is often lost.
- The system should accurately capture information about the programming environment and the runtime environment.
- The information captured must be easily searchable or otherwise accessible.
- The system must provide useful feedback about errors and faults. Furthermore, such feedback should be instructive in helping users recover from the errors.

## 3.7 Summary

Mission scientists see the SSIO community as facing a number of exciting challenges, summarized here in terms of the five Vs presented at the outset of this chapter in Section 3.1:

1. Fast data access is essential to large data-intensive applications. High-level, self-describing data formats are critical to allowing concerted efforts to improve the SSIO system and to best use burst-buffer technologies and deeper I/O hierarchies (volume, velocity, and variety).
2. Effective metadata management is critical in allowing vast amount of high-velocity data from different sources to be used effectively together to generate meaningful science results (variety, velocity, and volume).
3. Provenance capture is essential. As application workflows grow in complexity and variety, capturing provenance becomes critical for future understanding of what occurred throughout all phases of data generation and analysis (veracity and value).
4. Workflow services/frameworks are needed that can co-locate *in situ* tasks (on node, off-node, and on external resources) and can be specialized for specific applications (velocity, volume, variety, and value).

# 4 Computer Science Challenges

A number of contributors to the overarching challenge of providing high-bandwidth, low-latency, storage access across multiple tiers are discussed. Together, the sub-sections below are intended to capture the breadth of the problem, with each detailing the specifics and magnitude of particular, classified regions. Each subsection begins with materials provided to the attendees that were intended to seed the discussion during the workshop. At the end of each subsection the workshop discussion themes are captured and a set of research directions are distilled.

## 4.1 Storage System Architectures

Architectural changes in exascale and post-exascale HPC platforms raise new challenges for effective SSIO architectures. The generation, analysis, and management of multiple extreme-scale datasets will place increasing stress on the storage capabilities of existing HPC software, platforms, and facilities. Increases in working set sizes, simultaneous analysis of simulated and experimental data, and an increased understanding of data retention times are enough to motivate a need for improved hardware and software architectures for storing and accessing scientific data. When combined with the emergence of new storage and networking technologies, it is clear that fundamental changes in HPC storage architectures offer the opportunity to significantly reduce time to scientific insight.

The successful deployment of NAND Flash storage media within HPC platforms and the resulting performance improvements in common scientific workflow elements such as checkpoint/restart demonstrate how focused community research efforts can affect broad swaths of the simulation science community. Several years of research efforts [Bent2012, Liu2012a, Barton2014] led to the successful deployment of Flash media into HPC platforms at multiple DOE facilities [NERSC2015a, LANL2015, LLNL2017, ORNL2018]. Further transformational opportunities likely exist in leveraging emerging media types within new and existing scientific workflows and re-architecting storage subsystems and interfaces to better match existing media characteristics.

In order to better understand the challenges and opportunities that exist within hardware and storage architecture we have divided the research and development space into six coarse but essential areas: storage media and interfaces, network technologies, accelerators and active storage, resilience, advanced systems management, and security. Crosscutting research across these areas may be required for some projects. For example, emerging types of storage media exhibit different resilience characteristics in the forms of wear rates, bit error rates (BER), and data retention times. This organization is provided to systematically explore areas within software and hardware architectures and describe emerging opportunities and challenges.

### 4.1.1 Storage Media and Interfaces

New storage media typically present new performance levels, new performance asymmetries (e.g., the difference between random access and sequential access) and new economies of scale that accelerate time to scientific insight when deployed effectively into HPC platforms and facilities. New storage interfaces instead require the revisiting of past assumptions about how best to access storage devices and ensure that each layer's assumptions correctly match the characteristics of the underlying media. The combination of both techniques presents opportunities to transform HPC storage systems and greatly accelerate time to scientific insight.

**State of the Art**

The use of emerging storage media technologies throughout the data center is an area of immediate interest. Evolutionary changes include the changing asymmetry between read and write track widths in some modern hard disk drives [Feldman2013, Aghayev2015], the emergence of energy-assisted magnetic recording (EAMR), the increasing densities of 3D NAND Flash, new Flash architectures [Samsung2017b], and the use of volatile memory as the backing store for reliable storage systems. Specialized file systems targeting hard disk head-size asymmetries [Kadekodi2015, Manzanares2016, Aghayev2017], flash read/write/erase asymmetries [Gal05], and distributed in-memory stores [Rumble2014, Rambus2018] have demonstrated that significant performance improvements can be achieved by leveraging the underlying device characteristics carefully. Changing economies of scale also motivate the use of Flash media in long-term storage tiers [Gupta2014].

Disruptive changes in the storage media landscape include the availability of fast, non-volatile memory devices such as 3D XPoint® and the availability of ultra-dense, deoxyribonucleic acid (DNA)-based storage solutions. Byte-addressable persistent storage, such as 3D XPoint® [Optane2018, QuantX2018] and Resistive RAM [Lu2016], has been an area of recent research interest and specialized file systems [Xu2016, Wu2011]; and data structures [Coburn2011, Venkataraman2011, Zhang2016] have been developed to exploit performance that greatly exceeds existing solid-state storage media. DNA storage, while further from commercialization, offers transformative opportunities in archival storage with low-cost media, thousands of years of data durability, and nearly free data duplication [Bornholt2016, Organick2018, Milenkovic2018]. Similarly, advances in light-based media offer promise as a dense storage media replacement for traditional spinning disks [Anderson2018].

Previous efforts to improve storage media interfaces resulted in development of the object storage paradigm, notable for demonstrating the fundamental performance improvements achievable by allowing storage devices to make device-aware space allocation/deallocation decisions [Mesnier2003] and used in multiple HPC storage systems [Braam2004, Nagle2004]. Block-based interfaces such as SCSI Block Commands [SCSI2018], NVM Express (NVMe) [NVME2018a], and the current Linux DAX implementation [DAX2018] continue to be the dominant mechanisms for

reading and writing data to and from storage media. Emerging storage interfaces that take advantage of byte-addressable media characteristics such as the SNIA NVM Programming Model [SNIA2017, PMDK2018] or the translation and logging logic within Flash devices [Samsung2017a] are becoming more commonplace.

**Seeding Workshop Discussion**

Rapid changes in storage media and interfaces result in several challenges in creating scalable storage systems for both high-performance systems and archives:

- Challenges exist with incorporating byte-addressable media into distributed HPC storage systems given that network-based interfaces to persistent memories may not be able to leverage the byte-addressability advantages effectively.
- The extent of the read and write asymmetries for emerging media such as 3D XPoint® and DNA stores are not widely understood, and techniques to exploit those asymmetries are not well studied.
- New media types introduce new economic models for HPC platforms and facilities that present challenges and opportunities in designing HPC storage tiering and design.
- New media interfaces provide opportunities to eliminate the inefficiencies of accessing block interfaces at sub-block sizes or non-block alignments, but leveraging new interfaces may require revisiting assumptions throughout the storage stack.

### 4.1.2 Networking Technologies

HPC networks continue to improve dramatically in performance and pose significant challenges for systems designers attempting to provide storage performance capable of saturating local network interfaces. The ability to aggregate tens of GB/s of network bandwidth into a single storage node requires a matching amount of storage bandwidth to produce an economically efficient storage system. In addition, storage systems typically require network abstraction layers that allow remote access across a variety of platform interconnection networks and dedicated storage networks. Multiple approaches to leveraging the capabilities in modern interconnection networks and providing broad network compatibility exist within HPC research efforts.

**State of the Art**

Modern HPC interconnects currently provide approximate 1µs latencies and throughputs of 100–200 Gbps [Alverson2012, BXI2017, HDR2018, OPA2018], with most technologies planned to double performance in the next several years. Commodity Ethernet provides similar bandwidths and similar throughput improvement plans but with moderately worse multi-hop latencies. HPC topologies are typically dominated by efficient topologies such as Dragonfly [Kim2008] and high-radix tori/Clos [Scott2006], although I/O backbone networks that connect

parallel file systems to multiple HPC platforms are more commonly implemented with fat tree topologies (possibly oversubscribed).

One area of recent innovation has been the low-level networking libraries used to leverage network devices from user-space [MacArthur2017, Shamis2015, Goodell2015, GenZ2018]. These libraries are typically used below a network abstraction layer provided by MPI or by other high-level networking libraries [Soumagne2013]. While HPC interconnects typically provide fewer management capabilities as compared to modern Ethernet infrastructure, these libraries are able to leverage common HPC networking features of potential benefit to SSIO, such as support for remote memory access (RMA), atomic remote memory operations (AMO), asynchronous progress engines (APEs), virtual lanes, and quality-of-service (QoS) mechanisms. New storage-specific network operations to accelerate data caching [Jin2017] and key-value storage [Li2017] have also been explored recently.

Finally, networked storage protocols including SRP [SCSI2018], iSER [Kim2003], and nonvolatile memory express over fabric (NVMEoF) [NVME2018b] have been used within HPC facilities to provide redundant copies of important datasets (e.g., mirroring PFS metadata) and export block interfaces from existing and emerging storage enclosures.

**Seeding Workshop Discussion**

Future HPC systems will incorporate multiple levels of storage distributed across one or more networks with potentially complex topologies and thousands of storage end points. Networks on these systems will likely present significant new capabilities including the following:

- QoS via throttling, performance isolation, and co-scheduling with preemption;
- Advances in RMA and AMO operations, potentially end-to-end from compute memory to the storage device;
- Support for asynchronous operations and independent progress of communication;
- Collective communication support and the ability to embed computation for data reduction or reorganization within networking end points and the switching hierarchy; and
- Leveraging direct access to storage devices efficiently and effectively within HPC platforms.

Although many of these capabilities will be researched outside the context of SSIO, the SSIO community should be ready to incorporate or leverage these advances where appropriate. SSIO-specific R&D should be encouraged, allowing co-design of network and storage technologies as appropriate.

### 4.1.3 Active Storage and Storage Accelerators

As local and distributed storage systems have been augmented with advanced features such as compression, distributed erasure codes, and checksums, the limited spare compute and memory resources available on compute and storage nodes indicate that accelerators may be essential in achieving high levels of storage performance. Interest in executing additional user-specified tasks near the data (data reductions, statistical summaries, etc.) continues to motivate research into active storage.

**State of the Art**

Active storage aims to expose computation elements within the storage infrastructure for general-purpose computation on data. Active storage has been motivated by the increasing computational capabilities within storage devices and the ability to reduce data movement (filtering) and storage requirements (ephemeral views) by embedding computations in the storage device. Active storage research has been pursued for more than a decade [Riedel1997, Amiri2000, Son2010]. The current state of the art includes the extension of active storage concepts to the device level (T10 OSD [object-based storage]) [Mesnier2003] and HPC parallel file systems [Felix2006, Piernas2007]. Programming models include streaming [Acharya1998, Felix2006, Qin2006, Piernas2007], remote procedure calls [Riedel1997], and object-oriented models. While general-purpose computation has been explored in active storage, more limited forms of computation have also been investigated, including ephemeral views [Ma2003] and filtering [Riedel1997]. More recent work has examined mechanisms enabling the user to run predefined computations [Felix2006] that are of a more general-purpose nature but with well-known computational characteristics similar to stored procedures in databases, as well as extending the MPI-IO interface for analytics shipping [Son2010]. Still other research has proposed applying active concepts in the context of Flash devices [Boboila2012, Samsung2018].

**Seeding Workshop Discussion**

A number of significant challenges exist in active storage, particularly as it relates to HPC environments:

- Embedding computation within an HPC storage infrastructure brings about challenges in data and programming models for these environments, including security issues (e.g., how to control access of embedded computation) and resource management challenges (e.g., how to balance service between active and passive operations).
- Changes in central processing unit (CPU) capabilities, networking, and storage device speeds may fundamentally alter the types of computation that can be effectively amortized while processing a data stream (e.g., fundamental operations may be effectively offloaded into storage while more complicated

analysis tasks may only be appropriate for a single active storage architecture).

- Methods to expose acceleration and active storage capabilities with flexible programming interfaces that integrate well with existing HPC I/O middleware are poorly understood.
- The use of storage and networking devices that include acceleration technologies is one promising technique for providing scalable, generally useful active storage capabilities — however, the efficacy of this approach is still not well researched at scale.

### 4.1.4 Resilience

The differentiation of data retention times for the phases of scientific workflows has demonstrated the need for multiple resiliency models within HPC storage infrastructure [APEX2015]. Temporary and intermediate datasets may leverage different protection schemes than those utilized by the highly processed output data used for scientific visualization. Similarly, storage systems maintaining data for days or weeks typically use different data protection schemes than storage systems maintaining data for months or years.

**State of the Art**

Resiliency in SSIO has been an active area of research, spanning techniques to provide resilience to individual component failures [Patterson1989, Rizzo1997] up to the application [Zhao2004, Chang2008] of generalized algorithms [Birman2007, Lamport2001, Elnozahy2002] for fault detection and recovery. Numerous strategies have also been employed for data availability, including network redundant array of independent disks (RAID)/erasure encoding, consensus/quorum protocols [Ongaro2014], and multiple forms of data replication [Cidon2013]. While significant work has focused on resiliency of the underlying storage server infrastructure, some efforts have also focused on end-to-end data integrity [Zhang2010], although only limited work has been done in this area specifically for HPC environments. Above the storage system level, some work has explored fault-tolerant runtimes and application-level resiliency strategies [Hargrove2006, Sankaran2005], but the scalability of such techniques remains in question. Fault-tolerant programming models, such as MapReduce [Dean2008] and Legion [Bauer2012], and the application of the consistency, availability, and partition tolerance (CAP) theorem and peer-to-peer systems principles are also gaining momentum and adoption within the scientific HPC community.

New stacked dynamic random access memory (DRAM) technologies provide extremely high levels of memory bandwidth and have the opportunity to provide high-performance buffer operations in HPC storage systems; they also come with new reliability models that require new data corruption protection schemes [Jian2016, Gupta2018]. For example, the HDF5 middleware library provides checksums for files [HDF2018], but not the end-to-end data protection that may be necessary for these new memory technologies.

**Seeding Workshop Discussion**
Next-generation HPC systems will raise significant resiliency challenges for the SSIO community. While some resiliency challenges will crosscut with the broader resource management, networking, application, and parallel programming environment communities, SSIO-specific resiliency challenges will remain.

- New storage media, tiered storage organizations, increasing storage capacities, and tighter margins in component designs (silent data corruption) will necessitate significant R&D in SSIO resiliency to account for device and data properties.
- Another challenge is to leverage new memory technologies inside of storage systems without memory protection strategies like ChipKill.
- Another is to provide appropriate levels of performance and resiliency for persistent data within the storage hierarchy.

### 4.1.5 Advanced Storage Systems Management

Bursty I/O requirements and bulk-synchronous, write-dominant HPC workloads have traditionally motivated a research interest in advanced storage management infrastructure for HPC platforms. Efficient purging toolsets, statistical summaries, and data layout tuning are some examples of the advanced management infrastructure common within HPC facilities. The increasing popularity of machine learning as a tool within the systems community introduces new possibilities in the application of statistical models to storage systems management. Autonomous storage systems capable of responding to cyclic workload demands, policy-driven toolsets capable of managing resource sharing, dashboards for monitoring the intersection of scientific workflows and storage systems, and new capabilities for enabling rapid data reorganization could all contribute to more effective use of current and emerging storage hardware.

**State of the Art**
*Autonomics* refers to the ability of a system to adapt to a changing environment, such as tuning for higher performance in response to a change in workload or redistributing work in response to a faulty component. In SSIO, autonomic approaches are potentially useful for management, monitoring, and optimization in response to user behavior. Multiple efforts have identified models that provide forward predictions for storage layout decisions [Behzad2014a, Behzad2015, Liu2017]. There have also been efforts to build storage policy engines that enable dynamic policy switching based on model predictions [Sevilla2015]. Low-level storage management and monitoring protocols, such as SMART [Allen2004] and Swordfish [SNIA2018], are integrated and leveraged at many HPC facilities but are not widely used for policy management.

Significant efforts exist within storage systems management to accelerate common scientific workflow tasks. Parallel data movers exist for both wide area networks [Hanushevsky2002, Allcock2005, Settlemyer2011] and data movement within data

centers' storage tiers [LaFon2012, Wang2016, Pftool2018]. Similarly, several external indexing and policy engines exist for implementing common HPC policies, such as storage accounting and purging [Declerck2014, Bonnie2018].

Finally, systems for providing quality-of-service policies for networking and storage have been an area of frequent research interest. Efforts for quality-of-service guarantees for both storage devices [Wachs2009] and distributed storage systems [Gu2011] have been proposed but not widely adopted by HPC facilities. Modern software-defined networking controllers [ONF2018a, ONF2018b] provide a substantial set of user-programmable, quality-of-service guarantees for Ethernet-switched networks and are beginning to see deployment in HPC facilities.

**Seeding Workshop Discussion**

The increasing complexity of SSIO systems introduces multiple challenges in the space of possible configurations and complicated interactions created by diverse sets of scientific workflows:

- Although toolsets for implementing policies and managing data are well-suited for the systems and problems for which they were defined, they are typically not modular or composable in ways that address complex tasks.
- Interactions between storage system components and tiers within storage systems are subtle, complex, and often lead to unexpected behaviors.
- Policy engines, autonomic or otherwise, are needed that are able to explore the system configuration space and measurably improve reliability, performance, and/or TCO for storage systems.

### 4.1.6 Security

Security continues to be a pressing concern within HPC platforms and facilities. As security monitoring techniques have trended toward deep packet inspection and continuous security audits, the lack of end-to-end security within HPC storage systems continues to be an apparent shortcoming. Although HPC facilities have traditionally addressed the issues of balancing authentication, secure access control, secure data transmission, and efficient performance, research into this area is still needed.

**State of the Art**

Security for storage systems in HPC is typically implemented by using traditional UNIX users or groups and access control lists. Specifically, this security is implemented via trusted software running in the kernel on storage clients in conjunction with one or more trusted servers. Data are not typically encrypted at rest or over the wire.

Numerous, more advanced security approaches have been investigated in the context of HPC but not productized. These include a technique for fine-grained encryption of large datasets [Li2013] and methods for aggregating security operations

[Leung2007]: authorizing multiple client-file pairs in a single operation and allowing a representative client to act on behalf of a large group (e.g., the processes in a parallel application). Scalable methods for security in large-scale HPC storage systems were also investigated as part of the light-weight file system (LWFS) project [Oldfield2007]. Security partitioning for secure and efficient search using Bloom filters has been explored [Parker-Wood2010], as well as using a keyed hash tree [Li2013] and scalable authorization mechanisms [Leung2007].

**Seeding Workshop Discussion**
Emerging storage system architectures create a number of challenges in applying the current state-of-the-art security strategies:

- Additional layers in the storage hierarchy (i.e., nonvolatile storage layers, "campaign" or "data lake" storage layers between the parallel file system and archive) mean that the security system will need to control access to multiple tiers potentially provided by multiple vendors.
- The dependence on node operating system (OS) or network hardware for enforcement of security needs to be relaxed: there must be ways of preventing information leakage from nonvolatile storage located within the compute fabric or between jobs running in the system without reliance on the kernel for enforcement, given that the kernel itself is outside application control yet increasingly subject to compromise.
- Security must be supported at a range of granularities that may leverage knowledge of file layout (e.g., HDF5 or netCDF).
- New security solutions must be decentralized and allow fast paths for common operations; security needs to be as performance-transparent as possible.
- Security solutions should integrate with resource management to deter denial of service (e.g., consumption of available storage space or bandwidth).

### 4.1.7 Discussion Themes

A number of themes emerged during workshop discussion, summarized below.

**Storage Media and Interfaces:** Attendees noted that changes in both storage media types and interfaces to storage media present new challenges in developing HPC-centric system software. The participants identified challenges in determining classes of interfaces that belong in hardware versus software, in particular with respect to emerging key-value devices and media with highly asymmetric access characteristics, such as shingled disks. The participants agreed that economic factors will largely govern the hardware interfaces presented, but that research into leveraging these interfaces offers opportunities to greatly improve storage software efficiency. The discussion then moved to the challenges in improving SSIO software efficiency to leverage fast storage devices. The discussion concluded that storage software efficiencies and load-balancing strategies need to be radically improved to match the performance of the storage devices aggregated into modern storage servers.

**Networks:** Networking speeds are continuing to advance rapidly, and the discussion focused on topics for leveraging new networking technologies inside of SSIO software stacks. Challenges related to leveraging networks efficiently in software were discussed, and challenges associated with fabric-attached storage were a frequent topic of discussion. Discussions included the description of open research questions surrounding: how do users perform connection management for fabric-attached storage devices, device management at scale, and flow control at scale? Further, are existing consistency models appropriate for new high-bandwidth, low-latency networks providing NVMEoF access to storage devices? Finally, several participants noted the difficulties associated with building high-performance SSIO software for HPC platforms using lightweight cores.

**Active Storage and Storage Accelerators:** The workshop attendees discussed the challenges with developing more capable storage software stacks including both active storage and the offload of storage systems software onto accelerators. The initial discussion focused on the challenges related to understanding the structure of data at all levels of the storage hierarchy. Solving this problem is a key factor in offloading advanced data analysis capabilities into the storage system. After discussing multiple possible applications for acceleration, including machine learning, attendees agreed that the primary challenges appeared to be in co-designing approaches to active storage and active networks such that domain-specific offloads can be developed and deployed.

**Resilience:** During the resilience discussion, participants noted that resilience in science workflows is multifaceted, and the trade-off space is not well-supported by existing modeling efforts. The trade-offs related to the rates of data loss, performance, and response time are not straightforward for scientific workflows, where some percentage of the output data can also be recreated at a later time. The need for better models and metrics for understanding these trade-offs was mentioned multiple times. The participants further noted that improved models may incorporate emerging media types or may carefully fit complex scientific workflows, and that both types of efforts could improve the understanding of resiliency for SSIO software architectures.

**Advanced Storage Management:** The workshop participants discussed the many crosscutting concerns related to advanced storage management. Challenges exist in managing data as they move through hierarchies of storage, in particular in allowing scientists to express goals and in SSIO management software achieving those goals efficiently. Many participants felt that automated approaches to storage management, including autonomics and system self-regulation, are important research areas that merit additional study — in particular with advances in the ML for systems communities. The participants identified opportunities for greatly improved system efficiencies if heterogeneous systems could be extended beyond traditional hierarchies and include heterogeneity within storage tiers. Systems approaches for coping with extremely diverse media types (including different drive models and network technologies within a single storage tier) present extreme usability

challenges but also opportunities for radically improving the economics and value proposition of future storage systems.

**Security:** During the security discussion, the participants focused on two separate challenge areas within the security domain. First, what threat model exists for data stored and accessed from HPC storage systems? Second, how do SSIO software systems provide both efficient access to data and adequate protection against threats? With respect to the threat model, participants emphasized the need for a rigorous, formalized threat model that addresses provenance tampering, metadata tampering, and data tampering. In addition, difficulties in developing a threat model include the long data retention life times and the parallel access to data. With respect to developing efficient techniques for accessing data, the participants noted that recent security vulnerabilities such as Spectre and Meltdown reduce I/O system performance by 10 to 50%. The participants concluded that this area is changing rapidly, and the scope of both problems and solutions are not well understood today.

**Composability:** In this new topic emphasized by the workshop participants, the discussion focused on how to *deliver* new software and hardware storage technologies to scientists. In particular, the discussion centered on the difficulties that science teams experience when trying to leverage disparate storage system software effectively within their domain science codebases and workflows. The community noted the difficulties that exist in composing domain-specific storage services, including the following questions: how does the SSIO community guide multidisciplinary teams to create efficient, effective data services; how are these services deployed across multiple facilities; and how can new storage services effectively adapt for deployment into a new environment? The difficulties associated with composing new storage services is such that the design space may require a relatively thorough exploration to identify new composition techniques. Finally, the difficulties in composing systems dovetails with difficulties in understanding SSIO software systems, and the combination of techniques described in Section 4.5 may be required to guide the composition of storage services.

### 4.1.8 Research Directions

Participants concluded that new software architectures are needed to improve access to scientific data and ensure the proper level of durability for scientific datasets. The following priorities emerged from the discussion.

**Improving storage system efficiency and capabilities.**

- Leverage emerging storage media, networks, and accelerators; and
- Create efficient storage systems components to address the disparity between user-level performance and hardware capabilities.

Emerging storage hardware continues to improve the performance, capacity, and durability of the media available for new software architectures. High-performance

storage class memories provide a faster, byte-addressable alternative to existing solid state storage; and emerging molecular information storage offers promise in addressing limitations in capacity and data retention time for existing magnetic media. When combined with fast networks and accelerators capable of improving storage system efficiency, future storage architectures will have a multitude of underlying technologies to integrate to accomplish scientific data management tasks. These future storage systems must improve the performance delivered to actual scientific applications. While current storage systems typically provide excellent performance in benchmarking conditions, real scientific applications continue to see a disparity between the potential performance and the actual performance experienced by users.

**Composing advanced storage services.**

- Improve capabilities for the composition of domain-specific storage systems; and
- Enable autonomics and the orchestration of science data management tasks across multiple, simultaneous storage systems.

As scientific workflows grow more complex and incorporate simulation and experimental datasets simultaneously, it becomes more important to rapidly construct storage services capable of efficiently supporting domain-specific data management tasks. Orchestrating activities across a variety of storage systems is time-consuming for users and requires robust storage systems and toolsets for executing tasks efficiently. New software building blocks are required to enable complex data management tasks at scale.

## 4.2 Metadata, Name Spaces, and Provenance

As the complexity and scale of systems, applications, and data continue to grow, there is an increasing need to develop robust capabilities that enable both systems and users to extract, search, and track lineage for the massive volumes of data generated for scientific purposes. While some of these capabilities exist today, they are typically deployed through a set of ad hoc tools that are not designed for the scale or complexity anticipated for large scientific datasets (e.g., scripts that use UNIX grep, find, and awk). In general, managing large datasets on existing HPC systems requires a level of discipline and organization by users that is extremely time consuming, if done well, and error prone if not. In addition, requirements for repeatability, application workflow management, and data curation (among others) are driving the need for robust and integrated tools for provenance capture and management. These provenance capabilities must allow exploration of the provenance information for debugging, anomaly detection, visualization, and other purposes.

As of October 1, 2015, the DOE policy [DOE2018] on data reproducibility requires that all funded research have an associated data management plan (DMP). Those DMPs must "describe whether and how data generated in the course of the proposed

research will be shared and preserved," and DOE offices must assess the long-term needs for data sharing "about three years after this policy goes into effect." We are now (as of early 2019) in that timeframe, and the SSIO community should be an active participant in ensuring that long-term science data sharing is possible in a way that enables the greatest value to science communities. The solutions to the metadata, namespace, and provenance challenges described below should support those goals, as well as the needs of the scientists producing the original data.

### 4.2.1 Metadata

Metadata, in this context, refers generally to the information about data. It may include traditional user-visible file system metadata (e.g., file names, permissions, and access times), internal storage system constructs (e.g., data layout information), and extended metadata in support of features such as provenance or user-defined attributes. Metadata access is often characterized by small, latency-bound operations that present a significant challenge for SSIO systems that are optimized for large, bandwidth-intensive transfers. Other challenging aspects of metadata management are the interdependencies among metadata items, consistency requirements of the information about the data, and volume and diversity of metadata workloads.

**State of the Art**

Scalability of metadata services has seen significant attention in the context of file and and object storage systems. Distributed metadata servers with strong consistency semantics, such as Ceph's MetaData Server (MDS) [Weil2004, Weil2007] and GIGA+ [Patil2011], are either focusing on ease of load balancing using hashing (GIGA+) or aiming for improved locality by dynamic subtree partitioning (Ceph's MDS). More recent object-storage systems scale metadata management across a large set of storage devices [Aviles-Gonzalez2014]. Others manage parts of the metadata in the clients to achieve scalability [Zheng2014, Weil2006, Ren2014] but rely on relaxed consistency semantics to achieve performance.

While most HPC file systems support some notion of extended attributes for files [Braam2002, Welch2008, Weil2006], this type of support is insufficient to capture the desired requirements to establish relationships between distributed datasets, files, and databases; attribute additional complex metadata based on provenance information; and support the mining and analysis of data. Some research systems provide explicit support for searching the file system name space based on attributes [Aviles-Gonzalez2014, Leung2009], but most of these systems rely on effective indexing, which has its own scalability and data-consistency challenges [Chou2011].

Investigations have been made into the use of integrated databases for metadata storage [Johnson2014], but this technique has not been applied in an HPC storage system. Metadata-rich science formats such as HDF5, netCDF, ROOT, and ADIOS have been integrated into science data servers, such as SciServer [SciServer2018] and science community "web-portals" [ESS-DIVE2018, DES2018] that may provide

higher-level semantic metadata that enables science communities to publish, search, and share datasets more productively.

Frameworks for harvesting metadata from science data have been produced in academia [Gupta2010, Parker-Wood2013, Devarakonda2010] but have not been adopted by broad science communities. Machine learning techniques are beginning to be applied to large science archives [Orphus2018] but are still in their infancy. Techniques such as these may enable DOE DMP goals to be achieved in an automated and scalable fashion and with minimal effort by scientists.

**Seeding Workshop Discussion**

A number of nontraditional use cases for the metadata management system have emerged as key to DOE missions. These include multiple views of the metadata to support, for example, different views at different levels of the name space hierarchy and different views for different users' purposes; user-defined metadata; provenance of the metadata; and the ability to define relationships between metadata from different experiments (e.g., to support the provenance use case).

As the collection of metadata expands, it is important to ensure that all metadata associated with a dataset remains with the data. Metadata storage at different storage tiers, storage and recovery of metadata from archive, and the transfer of datasets to different storage systems are all important use cases to consider.

Hashing a namespace balances the load but does not account for locality. Fixed namespace partitioning accounts for locality but creates a load imbalance. Approaches are needed that provide the best of both of these known techniques. Further, these approaches must scale across exascale-sized metadata services that maximize caching, wear- or power-leveling, and other key properties.

Currently, the end user explicitly enters a large portion of metadata. As workflows grow in size and metadata becomes more complex, it is highly desirable to automate the capture of most metadata about the workflow and provenance. A number of attempts have been made at the fairly coarse level [Schissel2014]; however, as parallel jobs on a supercomputer become MIMD (multiple instruction multiple data) or composite workflows, there is a need to capture the complex dependencies within a single parallel job. Because a job on an exascale machine may have million-way concurrency, the metadata associated with a simple parallel write of a checkpoint file could be large and complex, not to mention the dependencies and interactions among the different components of a million interrelated parallel tasks. The volume and velocity of the metadata associated with such a fine-grained metadata could present a serious challenge to manage.

### 4.2.2 Namespaces

A namespace is a view or perception of data to the user. The subject includes a broad range of topics, including discussion of data-model specific namespaces, time-

oriented naming schemes, consistency of naming across systems and storage hierarchies, and search and discovery in large namespaces.

**State of the Art**

Previous work has focused on improving the scalability of access to traditional, POSIX-based namespace hierarchies [Weil2004, Weil2006, Patil2011, Moore2011]. More recent efforts have investigated how to manage scientific data in the context of object-oriented namespaces [Barton2013, Goodell2012]. The grid computing community has also made significant practical contributions to the problem of federating namespaces across facilities [Baru1998].

Composing data dynamically into "views" has been performed by traditional databases for decades and is being explored for large science data as well [Kryza2015]. Approaches that index metadata for datasets and present a scalable search mechanism to return results (see earlier science data servers/web-portal citations in Section 4.2.1, i.e., ESS-DIVE2018, DES2018) are also promising.

**Seeding Workshop Discussion**

The existing work generally is hierarchical and focused on file systems. A number of researchers, however, have argued that such hierarchical namespaces impose inherent limitations on concurrency and usability. Eliminating these limitations with object storage systems or higher-level systems could be the fundamental breakthrough needed to scale namespaces to million-way concurrency and to enable new and more productive interaction modalities.

### 4.2.3 Provenance

Provenance is broadly defined as metadata that describes the lineage of data. In simple terms, provenance contains details on how a particular file was generated; these details can be used to reproduce scientific results. For large-scale computational problems, the information can include the origin of data (sometimes experimental data); algorithms, libraries, and associated parameters and versions of software used for processing and transforming the data; details of the systems used for these transformations such as memory requirements, number of resources, and system software; and perhaps even ownership or user attribution for the various steps performed.

**State of the Art**

Most of today's scientific datasets have little to no provenance information at all. Provenance information that does exist is collected and managed in an ad hoc way through custom-developed scripting tools (e.g., Perl or Python) with no direct support for managing this data in the storage system. In these cases, the quality of the provenance data is directly related to the discipline and management skills of the data owner.

An important aspect of provenance information is how it relates to application workflows. Work on automatic extraction, management, or analysis of provenance data has begun in isolated research groups [Schissel2014, Davidson2008, Muniswamy-Reddy2006] and is available in many workflow automation tools such as Kepler/Komadu [Indiana2014], VisTrails [Callahan2006], and Pegasus [Mandal2007]; however, such tools are not in wide use and are often not deployed on HPC systems. Most of these tools rely on third-party databases and custom-designed tools [Davidson2008]. While this approach is effective for managing workflows in a single environment, the ability to encapsulate entire datasets and associated provenance for archival purposes is problematic. In addition, no effective way exists to integrate provenance information for workflows that span multiple systems.

High-performance computing facilities are also using tools such as an automatic library tracking database (ALTD) [Fahey2010] to collect provenance information for more traditional applications as well. As is the case in workflow systems, however, these tools are not integrated with the SSIO storage ecosystem.

Preliminary work has investigated the use of graph data structures [Ames2011, Dai2014] for provenance storage, but these concepts have not been fully realized at scale.

Although further opinions have been published [Hills2015, Mikdadi2017] emphasizing the importance of provenance tracking for science data and bemoaning how much effort goes into cleaning up previously published data [Delgado2016], very little effort appears to have been made in this area during the last few years.

**Seeding Workshop Discussion**

Existing workflow tools manage all provenance data internally. There is no storage-system support that enables the association of provenance-related metadata with scientific datasets. As datasets increase in size and complexity, the ability for tools to manage the files, databases, and other storage by-products will become a significant challenge without implicit storage-system and operating-system/library support.

The size and complexity of mining and analyzing provenance data could become an extreme-scale computing problem itself. Use cases for mining and analysis include debugging, anomaly detection, and visualization. One challenge is the need to identify all datasets derived from an application that used a particular version of a (known-buggy) library so they could be removed from the archive and rerun.

Workflows that span multiple systems also merit attention. For example, consider a workflow consisting of preprocessing a large collection of data from a scientific device. The results are consumed by a large HPC simulation, and the results of the simulation get transformed into a graph and analyzed on a graph-analytics system. The provenance information should include the complete description of these steps; but there is no formal way to construct, capture, and manage this type of data in an interoperable manner.

A complete description of steps may not suffice to reproduce a simulation. The setup of a simulation depends on the particular environment, which continually changes, often irreversibly (e.g., after the application of security patches). Capturing and understanding the impact of changes to the computing environment are important aspects of using provenance data for reproducibility.

### 4.2.4 Discussion Themes

Workshop attendees expressed concern over the willingness and ability of both system developers and scientists to understand and embrace many aspects of this topic area. Metadata, provenance, and namespaces cover an extremely broad range of semantic scope and are a rich area for both novel research and standardization.

**Metadata:** Workshop attendees discussed the need for research into lightweight methods to gather metadata from science teams, in order to improve the quantity and quality of the metadata stored with science data. However, attendees also noted that science domains have important domain-specific information that may be difficult to gather in an automated way. Research into methods to gather, express, and store both general and domain-specific information was considered valuable by attendees.

Attendees noted that the generally small and heterogeneous nature of metadata was a poor fit for most parallel file systems but could be a good match for database systems, if they could scale to the volume and performance needed. Alternatively, attendees noted, parallel file systems could broaden their scope to encompass efficient, scalable metadata storage for applications. Attendees speculated that decoupling file system development from metadata storage may be valuable to this direction of inquiry and allow metadata storage and file systems to advance independently of one another.

**Namespaces:** Attendees found that namespaces encompass a broad range in scope, from data structures in applications down to bytes in a file. The need for namespaces that are hierarchical, extensible, inheritable, and abstractable — while being accessible to both domain scientists (either directly or programmatically) and to algorithmic consumption — extended throughout the discussion. Attendees reiterated that current HPC namespaces are too large, and methods to search them and dynamically display more focused subsets of information would greatly benefit scientific analysis and discovery. Attendees brought up the need for domain-specific namespace views that reflect the "scientists' view of the data" and could incorporate information within structured files such as HDF5/netCDF/ADIOS, as well as across many files.

**Provenance:** The most discussed use case for provenance information was to support re-use of data for validation of published results, since the Office of Science Statement on Digital Data Management now requires projects to provide access to data for this purpose, but the group also discussed numerous other use cases such as

understanding of performance, system, and software variance; certification of results; and forensic analysis useful for debugging, auditing, and security.

Attendees mentioned that tracking provenance is a well-explored aspect of many other fields (art history, digital library science, etc.) and that effort should be made to apply the best practices from those fields to our challenges, rather than reinventing them. Attendees extensively discussed the high value of provenance in science reproducibility, error detection and correction in stored data, software fault detection, and I/O performance improvement of current and future systems.

Attendees also discussed the need for research into how much provenance information to store, for how long, in what level of detail, and how to ensure that the provenance information was immutable and trustworthy. The value of using provenance beyond strictly validating science data itself was brought up; attendees pointed out that provenance information can be used to train new staff as well as help to retain and propagate institutional knowledge of data gathering processes and procedures.

### 4.2.5 Research Directions

Workshop attendees reached consensus about several research directions to pursue:

**Storing science metadata in a scalable manner.**

- Develop methods to store science **metadata** in a scalable manner, and in a standardized and productive way. Of particular note, mechanisms to decouple metadata storage from the underlying file system, so that they may evolve independently, were desired by workshop participants.

**Enhancing the process of arriving at science insights by improving namespace query and display capabilities.**

- Enhance the process of arriving at science insights by improving **namespace** query and display capabilities to be more dynamic, programmable, and extensible. Methods that support data lifecycle management capabilities and incorporate views across and within science data are highly desired.

**Enhancing provenance information.**

- Enhance **provenance** information stored for science data to ensure that data reproducibility is guaranteed over a very long time period, but in a way that is easily accessible to domain scientists and their workflows. Generalizing the use of provenance information beyond science reproducibility and validation are also valuable.

A crosscutting theme in this area was the need to enable and exploit domain-specificity across all of these research areas, allowing scientists to "speak their own language" while exploiting as much semantic commonality as possible.

## 4.3 Integrating with and Supporting Science Workflows

In the exascale era, there has been a significant growth of applications that incorporate some form of workflow technology in their daily routine of running a scientific campaign. Scientists need to be methodical about how they design their computational experiments when running their large-scale experiments as they are limited by the amount of time they can compute and are then left to make choices about how they spend their time computing. Typically, they make choices about the potential cost of I/O to storage, and which analysis and visualization routines they integrate into their code for *in situ* analysis and visualization.

In this present age of computing, computational science is coupled with experiments/observations to validate computational models and predict and control scientific experiments. This intricate process of "validation" consists of a mixture of interrelated online analysis workflows along with offline post-simulation processing workflows. These online and offline workflows are coupled together in a scientific campaign. An illustrated example is shown in Figure 3.2.

Research and development in SSIO is critical to enabling scientists to (1) express their online and offline workflows (**composition**) and allow the workflows to be captured and saved (**provenance**) with their scientific campaign (Programming Model Integration); (2) integrate and execute sub-routines for **coupling** their main computational code with their other codes, such as visualization software and data analytics codes implemented in a mixture of languages, all in a **resilient and predictable way;** (3) orchestrate their workflows in a **dynamic, resilient, and predictable fashion**; and (4) describe and place the generated data for later post-processing during the data's lifetime in the scientific campaign, **across the multiple tiers of storage and memory hierarchies**.

### 4.3.1 Workflow Composition

The capacity and associated bandwidth of today's file systems can be strained by the ephemeral intermediate data in workflows, since applications such as SPECFEM_3D_GLOBE [Komatitsch2015] have been writing more than 1.5 PB on small-scale runs, and starting to produce more than 15 PB in six hours. The ongoing integration of SSD devices into compute infrastructures, both as burst buffers and as extended memory, offers opportunities to enhance science workflow productivity. Interfaces to enable the creation and orchestration of workflows over complex SSIO protocols require new tools, whether as programmatic interfaces, new libraries, schedulers, or other approaches. In addition, interfaces to extended memory hierarchies will be required that allow for the discovery of system characteristics and the integration of data from multiple storage and compute systems in a workflow.

**State of the Art**

While significant research has been performed in the area of programming models for HPC [Draper1999, Chamberlain2007, Charles2005], relatively little research outside of MPI IO [MPI2019], POSIX extensions [Vilayannur2008], and UPC IO [UPCIO2004] has focused on providing better programming model support for HPC SSIO and on imparting more information from the programming model to lower-level storage system interfaces. Other communities outside HPC that have conducted R&D (e.g., MapReduce [Dean2008]) to better integrate storage within programming models have seen widespread success. Further studies have sought to better understand these programming models and their connections to HPC and MPI [e.g., Plimpton2011, Hoefler2009, Ekanayake2008], including research to better understand their relationship to parallel file systems [Tantisiriroj2011]. Additional work has looked at how to map scientific data into non-POSIX storage [Goodell2012, Lofstead2016].

Some research has been performed on extending the programming model to incorporate processing capabilities within the storage system. Active Disks [Acharya1998, Riedel2001] investigated programming models, interfaces, and runtime support for pushing computations near disks. As discussed in Section 4.1.3, active storage has been investigated in the context of HPC applications [Son2010] and more generally in the context of object-based storage [Qin2006]. Recently, this concept was taken further [Jin2013] when looking at methods for passing information from applications to the runtime (via the programming model) so that trade-offs in the performance, power, and resilience space can be effectively evaluated and decisions made.

Researchers also have been investigating the applicability of task-based programming models such as Legion [Bauer2012] for use in HPC systems and have begun exploring how to manage a deepening memory hierarchy, including understanding how to incorporate support for SSIO [Watkins2015].

Scientific workflow tools exist for desktop (Kepler [Altintas2004]) and grid scheduling systems (Pegasus [Deelman2002]). Some workflow engines are built closely with schedulers such as DAGMan and HTCondor. They have also been branching out into collaborative efforts such as MyExperiment.org [DeRoure2008] powered by workflow engines such as Taverna [Wolstencroft2013]. These systems are very high level and are designed to abstract concepts of computation and data movement into nodes in a graph for specific scientific applications.

Some efforts have been made to extend work on scientific workflows to the storage layer [Bugra2008]. For example, Swift [Wozniak2014] is a popular big data workflow engine gaining some traction in HPC-related areas. Python-based engines such as Dispel4Py [Filguiera2014] and FireWorks [FireWorks2013] are becoming predominant because of the ease they enable for doing data-type discovery. Workflow performance optimization also requires information about the status and

availability of resources, at near-real time, in order to optimize execution of workflows [Wieczorek2009].

**Seeding Workshop Discussion**

Future HPC systems will incorporate multiple levels of memory and storage, including high-bandwidth designs, NVRAM, DRAM, disk, and tape. This structure will provide opportunities for speeding up data-intensive workflows significantly but will introduce a more complex storage system. Programming models and interfaces that do not expose any information about the storage hierarchy to the programmer and rely solely on the OS and hardware to transparently manage the hierarchy will lead to sub-optimal performance gains from such hierarchical storage configurations. Programming model support is needed that would provide programming interfaces and abstractions that allow for query of the characteristics, state, and workload of a storage level and placement of data at a specific level and potentially on a specific storage unit within a level. Such interfaces and abstractions would help better coordinate activities across the storage hierarchy, rather than forcing programmers to decode the behavior of the local memory and storage hierarchy and the layout of global storage resources.

In order to achieve this functionality, a successful programming model and its underlying infrastructure need to consider the execution of user functions at a variety of locations within the system, including within the storage system, to support their execution near data. Further support also is required to provide complex data mappings across the various levels of the storage hierarchy. A key challenge will be to find a right balance of how much storage detail and complexity are exposed through abstractions so that the application programmer is not overburdened yet is able to take advantage of the storage and performance characteristics of different storage layers. Successful programming models will likely have a layered approach to interfaces and abstractions that will enable implementation by experienced developers and the community of domain and data type-specific storage optimizations, will integrate such optimizations in the programming model through lower-level interfaces, and will allow for application developers to take advantage of such optimizations through higher-level abstractions and interfaces. Via appropriate layers of abstraction, the potentially complex mappings of computation and data performed by the programming model may be hidden from the user, simplifying development.

In any discussion of future HPC systems, the issue of fault tolerance arises. The current model of data resilience in HPC systems is simplistic, and the level of protection provided by the system is not visible to the user. Richer capabilities to express an application's persistence requirements for resilience are needed.

In addition to opportunities in workflow-aware storage (e.g., [Vairavanathan2012]), there remain gaps between high-level scientific workflow tools and the heterogeneous storage environment in HPC centers. Identifying middleware (e.g., [Lofstead2008]) and messaging [Subramoni2008] layers that appropriately

abstract the storage hierarchy and perform reasonably well is the first step to enabling scientific workflow tool users to best leverage HPC systems with burst buffers or other emerging storage architectures. Significant challenges exist in the area of wide-area data transfers in support of site-spanning and data-streaming workflows.

### 4.3.2 Workflows (Engine) - Provision and Placement

Workflow systems are an increasingly relevant software system to be considered in conjunction with scalable storage and I/O for HPC.

In the context of HPC, the Swift [Zhao2007] activity has shown the potential for high-throughput workflow on HPC systems and, in conjunction with the Hercules store [Duro2014], has shown the potential for exploiting data locality in task placement [Duro2014]. Similarly, the ADIOS [Lofstead2008, Liu2014] activity is taking advantage of the DataSpaces [Docan2012] in-memory store to optimize task coupling in HPC systems. Research enabling the MapReduce programming model in HPC systems (see previous section) is also relevant to this area. However, no general production capability for supporting workflows in HPC systems, nor a methodology for exposing locality from HPC storage to workflow systems, is available at this time.

The combination of workflow engines and Linux containers is an area of interest for many simulation code teams and HPC facilities. Multiple HPC containerization efforts [Canon2016, Kurtzer2017, Priedhorsky2017] are improving the ease of deploying scientific software and running complex software stacks on HPC platforms. Scientific container orchestration, similar to the service orchestration provided by Kubernetes [Brewer2015], is currently in a nascent state, but may come to dominate how scientific code is executed on future HPC platforms.

**Seeding Workshop Discussion**
A production workflow capability clearly is needed for use on future HPC systems. Specific to SSIO, effective workflow execution on future platforms will require efficient communication of data between tasks. Research on reducing data exchange overheads between tasks in a workflow has generally focused on memory to memory communications of running tasks, homogeneous systems, and wide area networks. Dynamically configured workflows cannot be realized until complex storage hierarchies can be fully exploited and stronger linkages to resource management systems are defined that enable more dynamic allocation of resources. While some research has been done on partitioning tasks into *in situ* and in-transit components [Bennett2012], substantial work remains before these hybrid workflows are developed. Success will require co-design with programming models.

In addition, workflows capture a great deal of relevant provenance information. This information is important for validation of results, but no mature method for passing this information to the SSIO system is available at this time. Solutions are needed that link with workflow and resource management systems. Containerized scientific

applications may simplify some provenance collection tasks but also generate significant difficulties in efficiently launching large (multi-Gigabyte) binary images across thousands of processes simultaneously, as well as challenges in providing secure and efficient I/O within containers.

### 4.3.3 I/O Middleware and Libraries (Connectivity) - both online and offline

The trend in I/O middleware is increasingly toward software that provides flexible data management capabilities addressing both data-at-rest and data-in-motion states. These libraries must support a wide array of connectivity between components (services) running on-node, off-node, or between physically separated hardware across the wide-area network, as well as handling more traditional I/O needs across local storage hierarchies. These libraries must also facilitate the use of a wide range of hardware components including different network fabrics, and heterogeneous storage components. Such middleware must gracefully manage failures in individual components by allowing connected components to survive such failures and offering control plane mechanisms to allow restarted components to reconnect with existing workflows. These control mechanisms must also allow runtime adjustment of resource allocation to allow workflow optimization in response to human controls, changing needs of an application, or performance impacts from other applications running on shared platforms. For many applications, the ability of middleware to provide QOS guarantees for data movement costs is essential.

**State of the Art**

A variety of middleware libraries are addressing these needs to various degrees. ADIOS [Liu2014] and associated data-staging mechanisms sustainable staging transport (SST), SST-2, Dataspaces [Docan2012], and Flexpath [Dayal2014] represent rapidly maturing set of middleware tools in use in HPC applications. Other libraries aimed at this space include Conduit, Damaris [Dorier2012], Intel's Distributed Asynchronous Object Storage (DAOS), and Decaf.

The libraries are primarily linked into application codes and perform data transformation and optimization on the nodes on which the computation is running. Other codes, such as analysis and visualization routines, can also be linked with middleware libraries to accomplish memory-to-memory data sharing, avoiding the expense of file system I/O.

**Seeding Workshop Discussion**

One significant challenge will be incorporating more runtime flexibility into middleware libraries. Current approaches for connecting application components lack resilience, with single failures often breaking other components. Furthermore, it is often not possible to dynamically reconfigure connections made by current libraries. It will be necessary to extend current approaches to support these sorts of runtime flexibility.

Another challenge involves the need to make I/O middleware work well across a wide range of different platforms. Middleware frameworks have to be designed to be extensible and carefully modularized so that support and optimizations for a constantly changing set of hardware can easily be integrated to keep middleware up-to-date and working efficiently. This includes changes in remote direct memory access (RDMA) mechanisms and network fabrics as well as a range of accelerator technologies, which may have separate memory and interconnect concerns. State-of-the-art network abstractions (Section 4.1.2) help address networking portability but not accelerator portability.

Data access patterns will also provide a challenge for these libraries. Many are tuned for particular classes of application and the data access patterns associated with those classes. Existing software has not been optimized to address some increasingly relevant data access patterns. Ensemble applications, for example, represent scenarios in which multiple relevant applications are executed to optimize performance and/or accuracy of application output. A high-fidelity simulation, for instance, can feed data into multiple instances of lower-fidelity simulations that can run faster so that longer time periods can be simulated. Output from these lower-fidelity simulations is then merged and returned to the higher-fidelity simulation to correct for errors and simulate shorter time periods at much higher accuracy. Other application scenarios include deep learning methods, in which large training datasets are read multiple times in the training phase, and task-based, high-throughput applications, such as applications that analyze large collections of sensor readings. Efficiently handlings these types of situations will require additional effort.

### 4.3.4 Data Abstractions and Representation

Within multideveloper, monolithic code bases, it is not unusual to have policy-enforced standards for the layouts of particular data structures. However, as users move toward assembling scientific workflows from both bespoke and generic component elements, the need for self-describing and/or semantic data management approaches becomes more apparent. The techniques may exploit a common data mark-up approach, a shared third-party data representation, or an I/O library with hierarchical attributes.

**State of the Art**

As mentioned in the I/O middleware section, the dominant data model supported by HPC I/O storage software (beyond simple POSIX) is dense, multidimensional arrays. Although some libraries support more complex data structures, such as geodesic grid data structures for climate [Palmer2011] and particle data [Adelmann2005], these are typically implemented atop a dense, multidimensional array model. A second model that is supported in systems such as DataSpaces [Docan2012] and Hercules [Duro2014] is a tuple representation. Tuple representations provide a flexible method for users to define their own organizations. The Damsel project [Northwestern2014] investigated methods for storing unstructured arrays, but did not produce a final product.

Compression of floating-point data has been studied as a method for concise representation of scientific data [Lindstrom2006], including methods that represent data as a function and capture error [Lakshminarasimhan2011].

Various methods of indexing scientific data have also been investigated. For example, the FastBit project [Wu2009] produced an indexing tool that has been used in a number of scientific activities, and hybrid compressing/indexing of data along the data path has been researched as well [Jenkins2012].

Numerous approaches for organizing data in storage have been investigated, and some implemented. For example, chunking approaches to data storage have been studied as part of the work in the HDF5 project [Folk1999] and in the Panda project [Seamons1994]. The use of algorithmic distributions of data is common in parallel file systems such as PanFS [Welch2008], Lustre [Braam2004], and the parallel virtual file system (PVFS) [Carns2000], with PVFS and Lustre providing mechanisms for the definition of new layouts and application of these on a per-file basis. The Scientific Data Services framework [Dong2013] manages partial replicas of frequently used data in locality-friendly organizations.

Log-based approaches to storage of scientific data have been investigated in the parallel log-structured file system (PLFS) [Bent2009] and ADIOS [Lofstead2008] projects, both available on systems today, and as a method of writing data through the MPI-IO interface [Kimpe2007].

Storage of adaptive, multidimensional data structures has received some attention as well, for example, in the Chombo project [Colella2000] and in the FLASH astrophysics project [Ross2001]. However, packages specifically targeting storage of multiresolution data have not emerged.

**Seeding Workshop Discussion**

The complex data structures used by scientific codes to organize their data are not well supported by current SSIO products. Methods for specializing general data abstractions to support specific activities would improve the productivity of application teams, as well as new abstractions optimized to support data models present in HPC codes and analysis tools. Furthermore, these abstractions should efficiently map to, and enable the use of, emerging architectural solutions such as burst buffers.

In the context of expected deep memory hierarchies, many orders of magnitude of variance can be present in the time to access data on the basis of its location. Expectations of cost of data access would be helpful in order for workflow systems, programming models, and users to effectively schedule operations. Similarly, passing additional information on the future use of data *to* the storage system could allow for optimizations that are otherwise not possible or effective.

Initial work has explored abstractions and runtime mechanisms for application-driven data management across deep memory hierarchies [Jin2015], and associated energy/performance trade-offs have been explored [Gamell2013]; however, this work only serves to highlight the potential of further work in this area.

Relationships between data are not represented well in current SSIO approaches. In models such as HDF, although data can be grouped and organized in a hierarchy within a file, relationships across files are not readily captured. Overall, a richer method for expressing relationships between data items would be an important building block for provenance solutions and help in understanding data as a whole.

Another challenge is to better understand how indexing techniques and different organization approaches can further facilitate analysis and to develop new techniques that target specific HPC and EOD concerns. Early work in indexing and reorganization in transit has shown promise for this approach.

### 4.3.5 Data Refactoring and Compression

The compression of scientific data is becoming an increasingly important aspect of working with scientific data because of the difficulty of transmitting, storing, analyzing, and understanding the drastically increasing quantity of data being produced. Though compression is typically separated from analysis as a distinct step of the scientific workflow, the two share a common goal: to extract from a mass of raw data the essential structure and key features of the phenomenon under study while ignoring or discarding the noise and simulation errors that have little or no impact on the quantities of interest. So that the compression does not compromise the results of the analysis, it is important to understand how compression methods affect the specific quantities of interest used in the analysis.

**State of the Art**

There are two major classes of data reduction techniques being researched and developed in the community: lossless and lossy. Both of these areas provide important tools that allow scientific workflows to reduce the volumes of data being moved and processed. Compression is a class of reduction techniques that can reduce the total amount of data that needs to be transferred and stored; alas, the challenge of speed has quite often remained. Luckily, new algorithms along with faster processors have helped the community move forward in this direction.

Lossless compression has a challenge to achieve both good compression rates, and good compression and decompression speeds. Blosc [Alted2010] is a high-performance compressor optimized for binary data. It has been designed to transmit data to the processor cache faster than traditional, uncompressed, direct memory fetch. FPC [Burtscher2009] is a fast lossless compression algorithm for linear streams of 64-bit floating-point data. FPC works well on hard-to-compress scientific datasets and meets the throughput demands of high-performance systems.

Lossy compression, such as JPEG [Wallace1992], is widely used in the computer graphics and digital image community, leveraging the fact that the human eye's perception falls well below machine precision. Moreover, excessive lossiness in such applications generally has few, if any, serious consequences. In stark contrast, the consequences of excessive lossiness in scientific applications are often severe and may even render the value of the reduced data meaningless. This has led users to be leery of lossy compression algorithms for scientific data. Nevertheless, it is widely accepted that lossy compression is inevitable if one is to attain the compression rates needed to handle data produced from future HPC applications. Scientific data are primarily composed of high dimensional floating point values for which a number of data compression procedures have begun to emerge, including ISABELA [Lakshminarasimhan2011], ZFP [Lindstrom2006], SZ [Di2016], MGARD [Ainsworth2017], and tensor-based procedures [Sears2009].

A key feature required of a lossy compression procedure is that the user be provided with a reliable estimate on the level of lossiness. Without a reliable estimate, the user will generally err on the side of caution — meaning that the level of lossiness in the data is much less than what would be needed in order to maintain scientific integrity, resulting in a much lower level of compression being achieved. An equally important feature, though less widely appreciated, is that the user should be able to prescribe the norm or metric in which the loss is measured. Typically, lossy compression procedures will measure the loss pointwise or in a least square sense via the peak signal to noise ratio (PSNR). Although such measures are widely used, it is often the case that neither is appropriate for a given application: pointwise loss control is often too stringent (resulting in sub-optimal compression), whereas PSNR control can often result in smearing or even complete loss of local features.

Some existing lossy compression procedures (e.g., MGARD [Ainsworth2017]) are able to provide this kind of functionality; nevertheless, many open questions remain, and there is considerable scope for further development. One aspect which should not be underestimated is the need to bring these developments to the attention of the wider community and raise awareness of how these techniques can be leveraged by applications.

**Seeding Workshop Discussion**

Although lossy reduction offers the most potential to mitigate the growing storage and I/O cost, there is a lack of understanding of how to effectively employ lossy compression from a user perspective, for example, which compressor should be used for a particular dataset and what level of reduction ratio should be expected. The community also needs to address questions regarding: what data features are indicative of compressibility? How does the error bound influence the compression ratio? Which compressor (or technique) can benefit the most from relaxing the error bound? How does the compression implementation influence compression throughput? What is the relationship between the compression ratio and throughput? What is the impact of lossy compression on data fidelity and complex scientific data analytics? How can data features be extracted and used to accurately

predict the compression ratios of various compressors? By answering these questions, the community can gain the ability to help HPC end users better understand what to expect from lossy compressors.

### 4.3.6 Storage Hierarchy for Campaign Knowledge Management of Scientific Workflows

Scientific data analysis has become an increasingly complex process involving large volumes and numbers of datasets and large numbers of data analysis workflows. In most cases, a scientific study goes through multiple phases: (1) pre-experiment/simulation phase; (2) experiment/simulation phase; and (3) post-experiment/post-simulation phase. These phases can be executed multiple times in an iterative process of designing new experiments/simulations (phase 1), monitoring and online analysis of data as they are generated (phase 2) and analyzing data from experiments/simulations to refine experiments/simulations and publish results (phase 3). A large number of derived datasets may be generated in these iterations during a large scientific workflow. In some cases these datasets are stored in files that may reach hundreds of terabytes in size, as well as large numbers of very small files. Moreover, data access requirements and patterns will vary across these phases and may change dynamically and rapidly within a phase. In phase 1, for example, the science team may carry out exploratory analyses on a broad set of datasets to design simulations/experiments. Phase 2 involves analysis that require near-real-time or interactive response rates. In phase 3, the science team performs deeper analyses on select sets of data from phase 2 to refine an experiment or a numerical model and collect results for publication.

One of the major challenges is that users not only have to take care of when to write the data, but also where to place the data and how to move the data among different storage levels in an efficient manner. In the current HPC storage hierarchies, parallel file systems are used for temporary storage most of the time. Scientists need to explicitly move their data from the parallel file system to a longer-term storage level before their datasets are deleted. This challenge becomes significant when a study involves multiple workflows, large volumes of data, and large numbers of data files.

**State of the Art**

Several research projects and tools have investigated mechanisms for aggregating state information across diverse resources [Czajkowski2001, Ripeanu2002] in distributed environments. The Integrated Rule-Oriented Data System (iRODS) [Rajasekar2010] provides a unified virtual collection model across heterogeneous data resources. There is also work on rule-based data management frameworks and rule languages [Paschke2005, Horrocks2004]. More research is needed to integrate rule-based mechanisms in SSIO systems so that (1) storage and retention policies for each storage level can be described, (2) users can express their data usage intentions with respect to storage and retention policies, and (3) system components can efficiently stage and place datasets across the storage hierarchy to meet policy requirements and users' data intentions while enabling fast access to data subsets.

**Seeding Workshop Discussion**

Scientists are in need of tools to manage and track data that are produced and consumed throughout the data lifecycle of a scientific study. This capability should take into account different data access patterns and data processing intentions so that desired data subsets in a phase can be located and accessed efficiently. Management of datasets and resources will require that SSIO systems be aware of the intentions of resource consumers and producers. Data can be viewed as both a resource and a consumer of other resources (e.g., storage, network, computation).

Interfaces and software services are needed so that scientists can express their data usage intentions for specific datasets and the underlying SSIO middleware could move and track datasets based on these intentions while complying with the storage and retention policies of different storage levels. For example, scientists should be able to specify that "datasets of type A will be repeatedly accessed in the next three months" or "dataset X can be archived after it has been analyzed and feature set F has been computed." Expression and execution of such data lifecycle intentions might be enabled via rule-based systems and integration of rule-based decision-making capabilities in SSIO systems.

### 4.3.7 Discussion Themes

A number of themes emerged during workshop discussion, summarized below.

**Workflows Composition and Provenance Capture**: Attendees agreed that a new programming model that can do the following — incorporate user intentions, take advantage of multitiered storage systems, and enable workflows to move the most important information first — is important for the composition of scientific workflows. In addition, building serializable, reproducible, and containerized scientific workflows is interesting. Workshop attendees noted that there might be concepts in orchestration systems used in several DOE facilities (e.g., Jefferson Lab, SLAC) that could be adapted. Attendees also discussed the challenges of optimizing the workflow by leveraging some domain knowledge possessed by application developers. For example, it might be difficult to share information across some workflow boundaries.

There was also discussion about the need to capture provenance to allow the science performed at scale to be reproducible. They agreed that detailed and useful metadata of every workflow step should be captured, which requires new tools for indexing/querying. This step becomes essential when workflows start to capture information across entire scientific campaigns, as well as being of use for repurposing data captured from the experiment/simulation and using it in another context. Several discussions covered the possible growth of metadata coming from provenance capture and resulting challenges in the SSIO layer. In particular, there was discussion about what to capture, but it was understood that much of the provenance needs to be retained for very long periods of time given that the data can be reused; and without sufficient capturing of this information, much of the

knowledge within the original data can remain hidden. There was also discussion about separating metadata storage from the bulk storage because much of this access can be random, facilitating the possible need for different types of storage media for this class of data.

**Resilient and Predictable Code Coupling**: Attendees agreed that building resilient scientific workflows is an unsolved problem, and code coupling makes it even harder. One of the major challenges mentioned by attendees is how to make scientific workflows adapt not only to the diversifying hardware and increasing concurrency, but also the different data generation and access patterns. Based on the discussion, several important pieces are needed: (1) modularity of support for workflow composition, orchestration, and control; (2) semantic modeling that can assist the workflow composition; (3) tools for annotating and tracing scientific workflows; (4) tools for indexing/cataloging/querying the workflow data; and (5) an automated way to reproduce workflows and verify the results. Research directions discussed included human-in-the-loop during the workflows and the ability to support synchronous and asynchronous coupling, along with the ability to dynamically change workflows and provide methods that help ensure that failures in one part of the workflow do not kill the entire workflow. Another topic discussed by the participants is the total quantity of data that needs to be captured during the workflow. As pointed out in the audience, code coupling has often been carried out in the storage layers. With the increase of new memory technologies, there is renewed interest in using these new storage layers to improve on the capabilities of code coupling. The issue discussed in particular was that the total amount of data that can be generated by even one single application can be very large. Although data reduction and data compression were discussed as possible methods to reduce the load, there was also discussion of whether scientists would actually use lossy data compression methods. Some discussion centered around the idea that scientists always reduce the data by way of time decimation, and data reduction must be brought into the SSIO layer not only to reduce the total payload brought to the storage system but also to help prioritize which parts of the datasets get to be placed in the higher storage tiers of the storage hierarchy.

**Dynamic Workflow Management**: Attendees pointed out that the dynamisms of scientific workflows come from both the resource and workflow levels, and that semantic modeling directly addresses these dynamisms by leveraging the metadata. Attendees also pointed out that leveraging data reduction techniques in a dynamic workflow is important and the data reduction has to balance the needed fidelity and cost for each workflow step. Especially on multitiered storage systems, where different storage tiers have different performance/capacity trade-offs, the data reduction strategies should vary. Attendees also discussed the support that job schedulers can provide for managing the dynamic workflow, such as I/O-aware scheduling. There was a short discussion of the possibility for the workflow scheduling to work with the batch job schedulers to allow many concurrent workflows to schedule around each other; this idea was further explored in the storage hierarchy session (see Section 4.4).

**Campaign Data and Metadata Management**: Attendees agreed that naming and locating data, which are critical for coupling workflows, continue to be challenging. Furthermore, fundamental research is needed to support data reproducibility through better metadata management and access. Attendees also pointed out that the lifetime of scientific data is critical for determining data management strategies for scientific campaigns. There was also discussion about placing more important information in the higher storage tiers, and having users express their intents somewhere during the workflows so that that information can be accessed in a timely fashion.

### 4.3.8 Research Directions

Through the workshop discussion, three research directions emerged. In all cases, these directions involved the ability for scientists to include more "science-based intentions" to their workflows/data that would allow the storage system and I/O system to more effectively manage their data. One example of a science-based intention is the ability for the scientist to describe how long they will use certain large subsets of their data, so intelligent data migration policies can automatically move parts of data to tertiary storage when data becomes less valuable over time.

The most promising research direction concerned enhancing the community's ability to reduce the total amount of data and even to identify which parts of the data are more important. It was discussed that metadata is critical for the overall datasets, so that the data can not only be found in the storage layers, but it can also be understood by having enough provenance.

**Improving scientific exploration by exposing intent.**

- Utilize abstractions that express the scientific intentions of the data. For example, users could express which variables need to stay in fast storage for the most rapid access.
- Employ machine learning algorithms that account for scientific intent to facilitate location and movement across storage tiers.

**Composing, capturing, moving and querying provenance from scientific workflows to the storage system.**

- Route provenance data so that it flows from workflows to the storage system, and "link" provenance data to the scientific data for validation, verification, and accelerating scientific knowledge discovery.

**Refactoring, or reducing scientific data through advanced infrastructures and algorithms.**

- Develop reduction and compression algorithms that enable the user to specify quantities of interest in the data that should be preserved by the compression procedure (e.g., conservation of mass).
- Develop adaptive, scalable methods to refactor data in flight, using domain-specific content, machine load, or predicted system resources.
- Develop cost models that include scientific intentions, performance/accuracy trade-offs, and compute load and resources in order to understand how to optimize access, bandwidth, and throughput.
- Develop reduction and compression algorithms that can provide near-optimal compression while providing quantitative control of the loss to a user-prescribed tolerance.
- Develop new compression algorithms that are capable of dealing with unstructured data (e.g., arising from AMR simulations) and particle data (e.g., arising from fusion simulations), as well as typical unstructured datasets from many large-scale engineering codes.

## 4.4 Deepening Storage Hierarchies

SSIO systems are becoming increasingly complex and hierarchical. Much like on-node memory hierarchies containing registers, multiple levels of cache, main memory, and swap space, storage systems now consist of multiple levels including node-local storage, storage on I/O nodes, parallel file systems, campaign storage, and archival storage. The storage levels differ in capacity, access speed (latency and bandwidth), and data lifetimes (both with respect to resilience and expected duration of allowed data persistence) with node-local storage delivering the fastest access speed, smallest capacity, and shortest lifespan; and archival storage providing the slowest access speed, largest capacity, and nearly infinite lifetimes for data. To further complicate matters, there is a growing need to understand how to incorporate storage-class memory devices into the storage hierarchy. Up until recently, users explicitly managed placement and movement of their data across the storage devices in the hierarchy. However, given increasing complexity, this practice is no longer a viable, long-term solution.

To address this problem, there is a need for SSIO infrastructure that facilitates appropriate movement of users' data in the storage hierarchy. This infrastructure needs to include not only data movement within a single compute cluster, but also across storage systems in a center. There is a need for application and tool interfaces to allow SSIO systems to work with resource managers and schedulers, monitoring systems, and workflow systems. The interfaces need to provide a mechanism for the tool or the user to inform the system of the needs of its data (e.g., lifetime requirements and use by other application components), as well as to query the

devices in the storage hierarchy for their characteristics (e.g., capacity and bandwidth).

### 4.4.1 Storage Hierarchy Levels

As discussed in the introduction to this section, SSIO hierarchies are becoming increasingly complex. The storage levels in the hierarchy differ in capacity, access speed, and data lifetime (see Figure 4.1).

At the very top of the hierarchy are the storage devices that exist on compute nodes themselves, typically NVRAM such as SSD; in the near future, however, these are expected to include storage-class memory (SCM) devices such as 3D XPoint® memory or Z-NAND flash memory. Node-local storage devices offer the fastest access times, with SCM devices being several times faster than SSD but also having only the smallest capacities and shortest data lifetimes, typically a job lifetime. At the next level of the hierarchy is I/O node storage (e.g., Cray DataWarp). Here, the storage devices are located on specialized nodes in the compute cluster for faster access than when retrieving from the parallel file system, typically with capacities large enough for most HPC jobs' input and output and data lifetimes equal to that of the job, although persistent stores may be allocated. The next level of the hierarchy is the familiar parallel file system, with an order of magnitude or more capacity than the previous tier and data lifetimes on the order of months. Another level in the storage hierarchy is campaign storage designed to support long-running application campaigns that can last months or longer. While slower than the parallel file system, campaign storage data lifetimes last the length of the application campaign. At the bottom of the storage hierarchy is archival storage, with relatively infinite capacity and data lifetimes but requiring extremely long lead times to secure access.

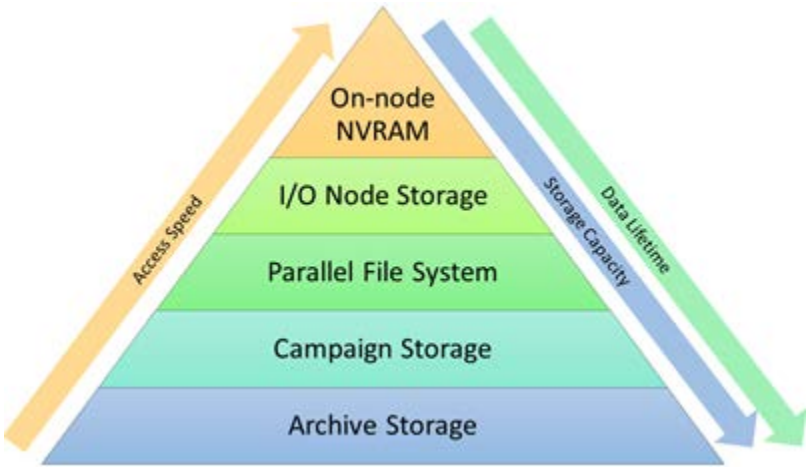


Figure 4.1: The storage hierarchy on HPC systems is growing more complex. An important consideration is how to best integrate on-node, storage-class memory devices into the traditional storage infrastructure.

**State of the Art**

Until recently, when storage hierarchies mainly consisted of the parallel file system and archival storage and possibly node-local NVRAM as SSD, users of HPC systems managed data movement explicitly either in their code or job scripts. However, client-side caching and extensions to remote compute and ION caching have been widely explored [Thakur1999, Liao2007, Isaila2011, Lofstead2008, Abbasi2010, Qin2009]. With the advent of burst buffer devices on systems, vendors are providing application programming interfaces (APIs) for transferring data to/from burst buffers and parallel file systems (e.g., Cray DataWarp API [Cray DataWarp n.d.] and IBM BBAPI [IBM CAST n.d.]). With these APIs, users can manage data movement asynchronously with asynchronous transfer or stage in/out commands. A number of research efforts [Ni2012, Rajachandrasekar2013, Barton2014] have begun exploring the use node-local burst buffers within HPC systems. In addition, DDN IME [DDN IME n.d.] provides a single namespace for applications across NVRAM and other storage levels in the system.

The current state of the art for hierarchical storage management (HSM) in HPC is a set of tools, APIs, and I/O libraries that help users manage the storage hierarchy. A few researchers [Barton2014, Goodell2012, Brinkmann2014, Jones2017] have investigated how best to manage and expose this deepening storage hierarchy in the general case. Other researchers have focused on specific use cases. SCR [Moody2010], FTI [Bautista-Gomez2011], and VeloC [UChicago2018] are checkpoint/restart libraries that capitalize on the short lifetimes of checkpoint files, which are needed only until a more recent replacement checkpoint is written. These libraries optimize checkpointing time by transparently utilizing node-local storage to cache checkpoint data and move selected checkpoints to the parallel file system. Recently, SCR added an API to handle automated movement of general output files in addition to checkpoints. I/O libraries that support automated movement of files in the storage hierarchy include HIO [HIO n.d.], HDF5 with Data Elevator [Dong2016], and PDC [Tang2018].

**Seeding Workshop Discussion**

Significant challenges in this area exist, given the deepening of the storage hierarchy and performance characteristics of next-generation storage technologies. A better understanding is needed of how data and programming models expose and interact with this deep hierarchy, how resource management can be coordinated across this diverse set of devices, and what capabilities are needed from interfaces in order to support science needs. In addition, facility teams need tools to inform procurement decisions to match the overall hierarchy to their projected workloads when they develop specifications for a new system or an upgrade.

An additional difficulty in designing application support for future hierarchical storage is understanding the workload requirements of exascale applications. For example, will bulk synchrony at the application level survive? If so, the entire storage stack may not need radical changes. However, if applications become increasingly

asynchronous, they may require new mechanisms in order to manage consistent views of distributed data in the hierarchy.

### 4.4.2 The Role of Nonvolatile Memory in SSIO

Nonvolatile RAM or NVRAM currently exists in HPC systems as simple node-local SSD storage or as a burst buffer with system software support (Figure 4.2). In the near future, HPC systems are additionally expected to include storage-class memory (SCM) devices [Freitas2008], such as 3D XPoint® memory or Z-NAND flash memory. Next-generation systems devices such as phase change memory or memristor may play a role as well (see Section 4.1.1 for more discussion of emerging storage hardware). While the addition of NVRAM devices in systems has and is expected to continue to improve I/O performance, their presence increases complexity in the storage hierarchy.

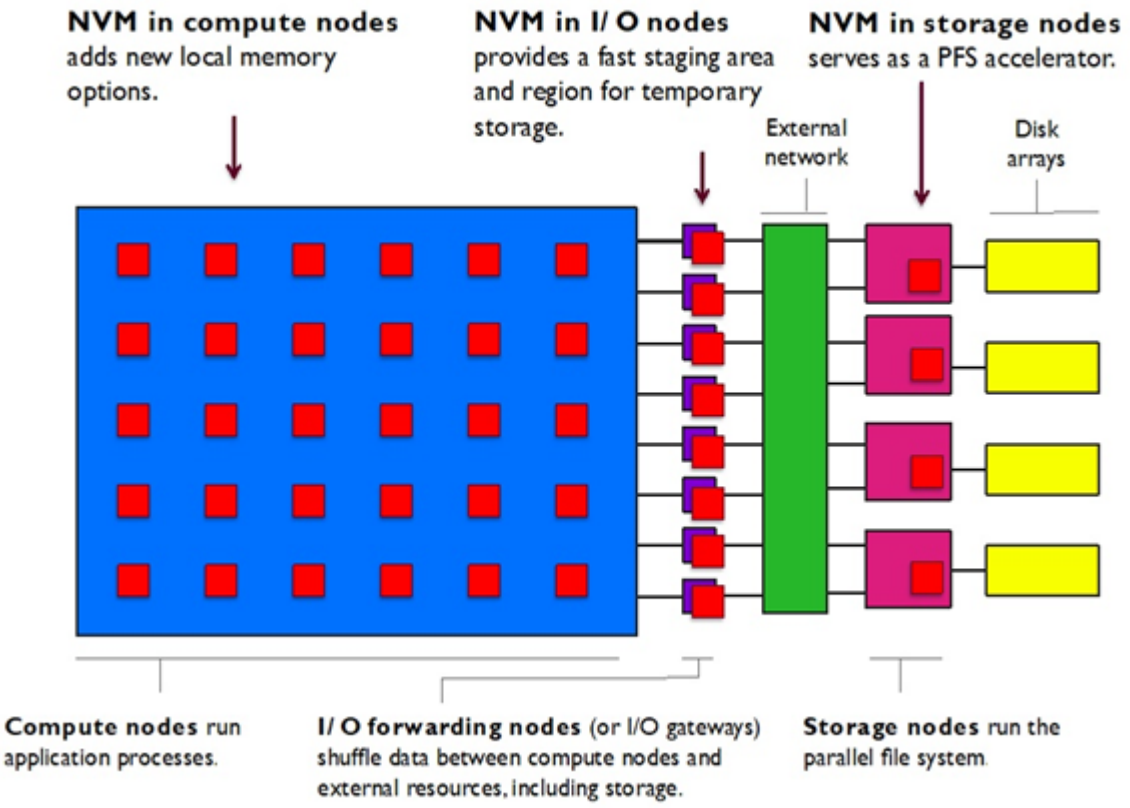


Figure 4.2: Nonvolatile memory (NVM) is an important component of today's and future SSIO architectures. Placement of NVM in the system influences the utility of the NVM for specific use cases, and additional research is needed to best understand where and how to integrate this technology to provide the greatest value.

**State of the Art**

Next-generation NVRAM technologies such as 3D XPoint® [Optane2018, QuantX2018], Z-NAND, and phase change memory will present yet another layer within the hierarchy with semantics akin to traditional DRAM in some cases. Research has begun in this area to assess their impact on file system design [Miller2001, Wang2002, Condit2009, Jung2010, Wu2011, Xu2016] and their use as caches [Liu2012b, Kannan2011a, VanEssen2012, TXu2016] or for staging and checkpointing [Kannan2011b, Kannan2013]; and current systems employ SSD-based NVRAM into their design (e.g., Cori and Trinity with DataWarp, and Summit and Sierra with node-local burst buffers).

**Seeding Workshop Discussion**

The addition of NVRAM in HPC storage systems increases complexity, arising not only from determining the best placement of NVRAM in the system, ranging from compute node to I/O node to the parallel file system (see Figure 4.2), but also from understanding its role in the storage hierarchy from an application viewpoint. NVRAM devices add to storage system costs, and so there is a need to evaluate the cost/performance trade-offs for placement of NVRAM in different locations in the storage hierarchy. In addition, for NVRAM placed on compute nodes, understanding whether the devices are best employed as extensions of memory or as fast storage devices in the hierarchy will be important. Considerations for this choice include application I/O workload, as well as read and write asymmetries and wear characteristics of the device.

### 4.4.3 Scheduling and Resource Management

Integration of I/O needs with scheduling and resource managers is becoming increasingly important to effectively use and manage hierarchical storage systems that can include NVRAM, burst buffer, parallel file system, campaign storage, and archival storage. Better coordination between the storage system and the scheduler can help ensure that less contention occurs at the storage system, which can result in improved job runtimes.

**State of the Art**

Batch scheduling has been used for supercomputers for some time, and current systems have integrated burst buffer devices into scheduling and resource management decisions. For example, with Cray DataWarp, users can request DataWarp allocations in their job request; with IBM burst buffers on Sierra and Summit, users can stage data in and out of the node-local burst buffers via their job script. Work involving multi-resource scheduling, including moving data and scheduling jobs, has been performed for grid computing [Schopf2002]. Several efforts have shown ways to map jobs onto compute nodes [Mubarak2017] and to integrate storage with an HPC batch scheduler [Bent2004, Gainaru2015, Herbein2016] to reduce I/O contention and improve application throughput. Other projects have investigated predicting storage system performance for applications [Xie2017], identifying the root causes of I/O interference [Yildiz2016], and scheduling

application I/O phases to avoid contention and for power management [Thapaliya2016, Savoie2016].

**Seeding Workshop Discussion**

There are major challenges associated with scheduling storage resources and I/O activities on HPC systems. For example, pre-staging data onto compute node-local burst buffer devices can prevent efficient backfilling of jobs when unexpected resources are available (e.g., when a job crashes and the allocation is terminated early). Here, the choice must be made to either continue or abandon the pre-staging progress that has been made on the original allocation, potentially increasing the waiting time of the user in the job queue. In addition, dynamic and multi-resource scheduling (as opposed to static scheduling simply for compute nodes) adds significant complexity to scheduling algorithms, potentially leading to less than optimally scheduled queues.

Another challenge is scheduling I/O activities of running jobs to reduce contention. While many HPC applications tend to have regular behavior and thus relatively regular I/O patterns, it is a complex problem to accurately predict I/O phases of regular applications in the presence of other active jobs that may interfere with or change the performance of the predicted application pattern. In addition, the emergence of I/O workloads from experimental/observational data sources and learning applications complicates I/O pattern prediction. To reduce inter-job I/O contention, the community will need a wealth of information on historical and running application I/O characteristics, storage system load, and the ability to feed these data back to tools and job schedulers.

Workflow management systems present a multidimensional resource provisioning challenge for the SSIO system. As suggested in Section 4.3.8, workflow systems could, however, become part of the solution, in that they can provide *a priori* and runtime information about the workflow components to schedulers and resource managers or can adapt their execution to work within the available resources. The scheduler will need to coordinate workflow or job capacity and bandwidth needs with the burst buffer and factor in stages and drains to the disk and/or archival subsystem. This effort will require elastic provisioning of storage and bandwidth across storage tiers in order to satisfy dynamic workflow needs.

### 4.4.4 Campaign and Archival Storage

Campaign and archival storage represent the slowest tiers in the storage hierarchy, but provide storage space for long-lived data. Campaign storage supports long-running application campaigns that consist of many jobs (e.g., to explore the application parameter space) that can last for months or longer. Archival storage has long been synonymous with tape storage, with relatively infinite capacity and data lifetimes, but extremely slow access times. Archival storage is intended for storing data that may need to be retrieved years or decades later.

**State of the Art**

Campaign storage, sometimes implemented as a center-wide file system, has recently been named as a tier in the storage hierarchy (e.g., on Trinity at LANL). Its purpose is to provide relatively fast access compared to archival storage to colder application data via a POSIX or near-POSIX file system interface [Braam2017]. Previous implementations of campaign storage used disk-based object stores that sit between the parallel file system and archival storage [Bent2016]. More recent versions are built on tiered-parity systems (e.g., 10+2 distributed erasure across 17+3 RAID groups) using chunk servers that will provide data protection even in severe device failure scenarios [Inman2017].

Facilities have been using tape for archives for some time. Hierarchical storage management systems such as HPSS [Watson1995] use a disk front-end to speed the time to first byte (TTFB) for small and recently accessed files. Work has been performed to connect namespaces across file systems and archives [Lustre2010, Degremont2013] and to understand how archives are currently used in HPC [Adams2012]. Archives today are predominantly centralized, stand-alone services that are used as a long-term storage where read access is often slow. Recent research has investigated the viability of replacing tape with disk drives in archival storage [Inman2014].

**Seeding Workshop Discussion**

For campaign storage, there is a need for scalable mechanisms for handling extremely large datasets with good performance. Aspects of this need include: investigation into the POSIX/near-POSIX requirements for this tier of storage, efficient metadata handling, dataset semantics for efficient grouping, management, and movement of large sets of application output.

We anticipate a real need for both the campaign and archive tiers to integrate with the higher levels in the storage hierarchy. Given that we expect a larger fraction of jobs to employ burst buffers as in-transit storage or for data analysis, we need to devise scalable, explicit, and automated HSM approaches for moving data between the tiers. We expect that the architecture and design of archival storage will need to be able to adapt to changes in the storage hierarchy and provide access mechanisms to support active-archive processing, cross-site data movement, and rich metadata services. This adaptation may necessitate the use of alternative archive technologies beyond tape, such as power-managed disk or optical storage.

Challenges of particular concern for archival storage are (1) effective verification of the integrity of archive files over time and continuous technology migration, (2) the cost and practicality of retrieving and searching data that reside on slow TTFB devices, and (3) investigation of new storage media for archive,

### 4.4.5 Discussion Themes

Discussion at the workshop largely followed the structure of the pre-workshop document.

**Hierarchy and Data Management**

The attendees first discussed the issue of the number of levels needed in a storage hierarchy, because establishing more levels increases complexity for users as well as for storage system software. From the discussion, there was agreement that there is a need for system designers to have tools to explore the trade-offs of different devices in the layers of the storage hierarchy. The designers need to be able to understand where money is best spent, for example, whether on the fastest, most expensive tiers or on more robust intermediate tiers and a fast network between tiers.

Determining best practices for data placement and management in the storage hierarchy is a challenge. The attendees noted that to address this challenge, research is needed on communication mechanisms, such as APIs or domain-specific language support, for applications and workflows to indicate their I/O needs either to middleware (e.g., an I/O library) or to the system (e.g., the scheduler). The communication mechanisms need to address the full requirements of current and expected workflows, as well as to support the requirements of I/O middleware that assists the workflow components. Examples of these requirements could include quality of service, performance, data lifetime, and resilience. Another approach to data management discussed by attendees was the possibility of utilizing machine learning approaches to learn and predict application I/O patterns to inform data management software. The attendees noted that for this approach to be successful, a large amount of data for training will be needed that does not yet exist.

The attendees agreed that the added complexity in storage hierarchies presents challenges for locating users' data. A primary reason is that the community does not yet have efficient mechanisms for representing and querying the metadata of users' data in a storage hierarchy. Given the bottlenecks that already exist in metadata operations for simple parallel file systems, there is a strong research need to explore how to efficiently support metadata in hierarchical storage systems. A promising direction of research could be to allow users to tag and name their data to facilitate locating the data in the future. The appropriate tagging and naming schemes need investigation and could include information about the data contents to facilitate locating particular datasets, as well as to communicate I/O requirements for the data (e.g., data lifetime or resilience).

Overall, facility representatives noted seeking to provide the best experience for users of the storage hierarchy in terms of ease of use and performance. On the one hand, providing a single namespace across the storage hierarchy is easiest for users; on the other hand, however, it might not provide the best performance. To provide a simple and efficient I/O abstraction, the attendees agreed that research to explore the needs of applications and middleware, as well as the performance implications of

proposed approaches, must be pursued. In addition, the community needs to explore the possibilities and performance implications associated with presenting a single namespace for applications across the entire storage hierarchy in contrast to explicit namespaces for each level of the hierarchy.

**The Role of Nonvolatile Memory in Storage Systems**

Attendees agreed that storage-class memory represents a challenge because it can potentially be accessed as a memory device (load/store of variables) or a storage device (read/write operations). Much discussion was devoted to how SCM might be used, after which it was agreed that we need to understand the potential use cases for applications and their benefits if SCM is used as extended memory instead of storage, given that it adds complexity to the hierarchy. Here, research in the area of programming models is a promising avenue to provide abstractions to applications that are independent of the access mode of the underlying storage device.

The attendees also noted that we expect SCM to have different persistence characteristics than traditional storage devices (e.g., it may or may not be persistent across power loss). We need to develop an understanding of the intersection of SCM use cases, performance, and resilience characteristics to determine whether SCM is best implemented as persistent through power loss (i.e., understand the use cases for the nonvolatility of SCM).

**Scheduling and Resource Management**

The attendees agreed that scheduling for HPC systems is already challenging and that adding the constraints of storage resources and I/O activities further exacerbates the challenge. To address this topic, attendees decided that we need research into efficient multi-resource and hierarchical scheduling for workflows. The scheduling research will need to account for mitigating or avoiding I/O contention between competing co-scheduled applications and will need to determine optimal cost models for encouraging "good" behavior by users with respect to their I/O activities and storage needs. Scheduling decisions will require detailed I/O behavior information from applications and systems, and machine learning approaches may be of benefit for predicting application behavior to avoid contention.

Another aspect noted by the attendees was that the resource manager may also have a role in determining optimal data placement and movement policies through the storage hierarchy. For example, the resource manager can determine whether a particular storage resource is overutilized and then redirect I/O activity to another resource, or it may determine that the network is under heavy utilization from messaging traffic and pause or lower the priority of I/O activity. Giving the resource manager this capability will require research in system monitoring and adaptive I/O scheduling.

The discussion led to the idea that integration with workflow managers will be important because workflow managers have knowledge of the data needs of the workflow components. In an HPC workflow, it is typical for one component in the

workflow to produce data that will be used by another component in the workflow. By integrating workflow managers into the allocation and scheduling activities of HPC systems, we can optimize data movement (e.g., have data moved close to the compute storage resources in advance of job start) and decide when data can safely be moved to archival storage if it will not be needed again in the near future.

**Campaign and Archival Storage**

The discussion by the attendees concluded that we need to explore best practices for placing and moving data between hierarchy layers, especially between the faster storage tiers and archival stores because retrieval from archival stores can take very long times (e.g., hours and more). They noted that automated movement of data is a promising approach because it would free up valuable space on the faster tiers. However, it is unclear whether the movement to slower tiers of storage should be user-directed (e.g., through an API or by tagging the data with movement instructions) or automated. For automated movement, we need models, possibly developed via machine learning, to help us understand the trade-offs of moving the data to find the most optimal approaches. We also need mechanisms for keeping track of data versions to keep globally consistent snapshots in the event that we need to roll back to a previous version of data (e.g., if a transfer is corrupted or fails).

Attendees noted that as a further layer in the storage hierarchy, some storage locations are geographically disjointed (i.e., external to the HPC center), so we need to understand how to best move data between distributed storage locations. For example, would it be better to drive a truck full of disks between locations or is the network able to handle the request in a reasonable amount of time? We also need to explore the significant challenges that geographically distributed storage presents for metadata.

### 4.4.6 Research Directions

A number of themes emerged during workshop discussion, summarized below.

**Allocating and scheduling resources and I/O activity for better center-wide performance.**

- Optimize resource usage and data movement across jobs and realize HPC workflow integration.
- Pursue multi-resource and hierarchical scheduling, contention avoidance, cost models for I/O, and storage allocation and scheduling

The primary goal associated with this challenge is to optimize the overall utilization of HPC systems. To achieve this goal, we need to understand how best to schedule and allocate the storage resources in the hierarchy and to move data across the storage levels. We need to investigate integration of workflow managers with storage system software to optimize data movement. We also need robust support from schedulers and resource managers for I/O activities.

**Placing, moving, and locating data in the storage hierarchy to meet application I/O workload needs.**

- Develop mechanisms for applications to communicate needs to middleware and/or systems; system support for middleware management of data; efficient metadata management for hierarchical storage.
- Pursue ease of use for application developers/users; single namespace vs. explicit hierarchical namespace.

The complexity of the storage hierarchy makes decisions regarding placement and movement of data very challenging for both the users of systems and the system software and I/O middleware. In addition, because data can hypothetically be stored anywhere in the hierarchy, locating data is a challenge for users. We need research into methods for applications and workflows to communicate I/O needs to storage system software. We also need to investigate metadata support for storage hierarchies.

**Capitalizing on diverse media characteristics to design efficient storage hierarchies.**

- Understand storage hierarchy layers and interactions, automated movement between fast storage and archive, data models for campaign and archive, and version management.
- Manage the exposing of tiers in the storage hierarchy and the blurring of lines between memory devices accessed via variable references (load/store) and storage devices typically accessed by software function calls (read/write).
- Address geographically distributed storage and tools for understanding storage system design trade-offs.

The different characteristics of devices at each level of the hierarchy add complexity to decisions related to designing efficient storage systems. We need to explore best practices for moving data between hierarchy layers, especially between the faster storage tiers and the archival stores, where retrieval from archival stores can require very long waiting times. We need to explore the potential benefits and best uses for SCM in the storage hierarchy. In addition, because some storage locations are external to the HPC center, we need to understand how best to move data between geographically distributed storage locations.

## 4.5 Understanding Storage Systems and I/O

Measurement, modeling, and understanding of SSIO systems play critical roles in a wide variety of data-intensive scientific computing activities. Unfortunately, while there are numerous user-facing performance tools available for CPU and accelerator resources, I/O performance analysis remains underdeveloped. In addition, storage systems themselves are complex and prone to performance degradation as a result of

subtle interactions between system components and application workloads. Both challenges are exacerbated by the proliferation of extraordinary concurrency, heterogeneity, and complexity as SSIO systems strive to meet the needs of DOE science. **These challenges call for additional research into instrumentation and monitoring tools, workload modeling techniques, system modeling techniques, and methods for interpreting and applying the knowledge gleaned from those tools and techniques.** By increasing our understanding of SSIO systems, we will not only be able to improve the productivity of today's applications but also provide crucial intelligence for effective design, development, and procurement of tomorrow's applications, systems, and system software.

### 4.5.1 Instrumentation and Monitoring

As increasingly sophisticated SSIO systems are deployed, it will become more and more important to extract information about application behavior, the state of constituent devices, and the aggregate condition of the system. These three aspects of system state can be collected via instrumentation or monitoring and combined to produce an end-to-end view of data movement and storage. End-to-end instrumentation and monitoring data forms the foundation of any effort to understand a storage system, whether the purpose is to optimize an application, validate a predictive model, maximize utilization, or find the root cause of a problem.

Scalable and low-overhead collection of performance, fault, power, and usage data will allow timely or even online analytics for system managers, users, and researchers. The ability to align the captured parameters in time and space, as well as correlate these with system component characteristics, will be critical, not only to gain a correct assessment of the system state, but also to provide predictive capabilities. Performance metrics will need to be collected at every level, ranging from the application to the burst buffers to the storage system and then to the archives, and must be propagated throughout the system. Standardization of the format of the information being collected will enable rich analytics of the collected performance data and serve as the basis for research efforts in performance tuning, scheduling, and reliability.

#### State of the Art

On the application side, tools such as Darshan [Carns2011] are capable of producing lightweight instrumentation. Darshan has a notable breadth of impact because of its ability to instrument production applications without perturbing performance (Figure 4.3). Higher-fidelity tools such as TAU [Shende2006], IPM [Skinner2005], and Score-P [Knüpfer2011] are employed as needed for more in-depth analysis. Score-P serves as the common underlying instrumentation method for multiple performance analysis tools. Broader system-wide monitoring efforts have long benefitted from techniques developed in the data center, grid, and cloud computing community [Sigelman2010], but more recent efforts such as the lightweight distributed metric service (LDMS) [Agelastos2014] have created frameworks that are more directly tailored to HPC environments. LDMS covers a wide range of system metrics in

addition to I/O metrics. Hardware device (such as disk arrays) monitoring [DDN2018] and commercial software (such as parallel file system) monitoring [Cray2018] are most often instrumented by proprietary vendor tools or proprietary vendor tools in conjunction with facility frameworks. For example, [Kim2015] demonstrated how telemetry from storage arrays can be mined to understand facility-wide storage workloads.

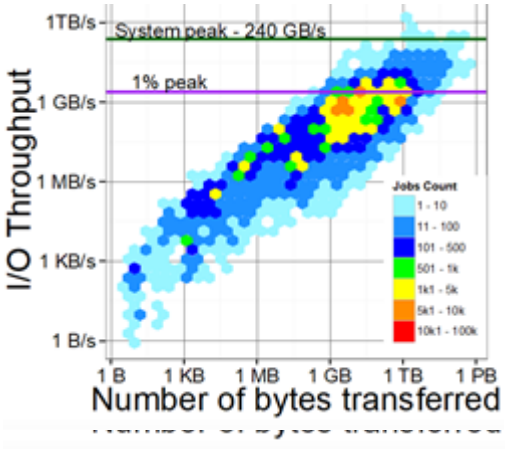


Figure 4.3: Darshan data can be used to understand the behavior of applications running on production systems. This graph shows the relationship between number of bytes transferred and effective I/O throughput for applications on the Mira Blue Gene/Q platform [Luu2015].

**Seeding Workshop Discussion**

The first challenge in instrumentation and monitoring research is dealing with the onslaught of rapidly evolving HPC technology. This onslaught includes new application models such as machine learning frameworks [Abadi2016], new hardware devices such as storage class memory [Spelman2018], and new data services such as Intel's DAOS storage system [Lofstead2016], none of which are adequately represented by today's HPC instrumentation and monitoring tools. Although tools such as Darshan, Score-P, and SIOX have made initial steps toward addressing this problem by modularizing their software architecture [Snyder2016] to permit future expansion, more work is needed.

Second, data integration and alignment are ongoing challenges, particularly as systems become more heterogeneous and diverse. This diversification is reflected in a lack of standardized formats for storage telemetry that make data sharing and common analysis difficult. It is also reflected in the limited ability to tag and trace storage events from end to end in modern systems. Ideally, it would be possible to correlate a specific line of source code in an application all the way to the level of an individual storage device access. Although early work such as [Muelder2011] has demonstrated preliminary capability, such techniques have not been widely adopted in the field.

Finally, the tension between instrumentation fidelity and runtime overhead has always been a factor in computer science, but a coherent solution to this problem in SSIO systems remains elusive. Inroads have been made in this direction with the ability to adapt polling frequencies and instrumentation methods at runtime;

however, it is largely the responsibility of administrators and individual users to make ad hoc decisions to address this trade-off.

### 4.5.2 Interpretation and Application

Instrumentation and monitoring data are only as useful as their application in the service of computer science research or scientific productivity. This usefulness can be arrived at in a number of ways, such as by combining data products and models to produce outcomes greater than the sum of their parts, integrating models into runtime systems for autonomous feedback, developing analysis tools or expert systems that turn data into actionable feedback, or incorporating data into the broader context of understanding HPC systems or collections of HPC systems.

**State of the Art**

The first task in interpreting and applying instrumentation and instrumentation data is to simply manage the large volume of data. Previous work such as [Vijayakumar2009] have explored trace compression mechanisms, and there are a number of activities underway to archive and index performance metrics in specialized databases for data mining purposes [Vazhkudai2017, Prometheus2018].

Most systems are monitored by gathering a large set of data about the system and then having a storage expert sift through the data [Gainaru2011]. Many open-source tools and vendor tools for gathering and monitoring this operational data already exist, and many of these tools are combined by using custom scripts [Miller2010]. Recent projects have made progress in creating frameworks to combine SSIO instrumentation and monitoring data sources for holistic analysis purposes [Betke2017, Lockwood2018, Frings2007]. Example studies from the TOKIO project illustrate how this type of data integration and synthesis can reveal system properties and correlations that are not otherwise visible from individual component instrumentation [Lockwood2018].

Studies have also shown how real-time data capture can be integrated into runtime systems to improve performance. Examples include runtime probing to select optimal storage resources [Son2017].

Both vendors [Vincent2018] and facilities [NERSC2015b] have pursued methods for providing actionable feedback to users, either in the form of I/O parameter optimization or visual dashboards that provide rapid performance feedback. Specialized vendors [Ellexus2018, I/ODoctors2018] have also brought to market I/O analysis tools with an increased focus on the end-user perspective.

**Seeding Workshop Discussion**

I/O performance data integration will remain a persistent, ongoing challenge without buy-in from vendors to aid in data alignment and normalization across components. The scope (and potential benefit) of data integration continues to grow as well, as understanding I/O increasingly requires not only awareness of HPC system

components but also awareness of experimental and observational data sources, wide area transfers, big data and cloud computing, and other resources that will constitute future scientific campaigns.

Despite the many advancements in I/O instrumentation and monitoring, interpretation of the data remains, by and large, an expert facility administrator activity. Only advanced application developers are likely to be able to quickly interpret I/O performance in a system, application, and historical context and assess how to improve performance, if they even have access to sufficient data to begin with.

Finally, once an I/O analysis has been performed, there are still many open questions with respect to how best to enact solutions. Possibilities include guided suggestions, automatic tuning, and automatic reconfiguration of available resources.

### 4.5.3 Modeling Workloads

In order to rigorously study an SSIO system, we must first establish the ability to accurately model the workloads that it will service. SSIO workloads include not just traffic from individual application runs, but also large-scale, multistep scientific workflows and even aggregate traffic from a facility's entire user base. It is not practical to reproduce such workloads by re-executing production applications. Re-executing full applications may consume too much CPU time, be logistically difficult to execute, contain sensitive information, or simply be impractical for a simulator or mathematical model.

The SSIO community therefore has a crosscutting need for modeling of representative workloads for storage system evaluation and performance tuning purposes. This modeling can take many forms, including proxy applications, microkernels, synthetic benchmarks, mathematical request distributions, and more. Regardless of the methodology used, our ability to accurately model SSIO workloads has a direct impact on how effective our other SSIO research activities will be in the real world.

**State of the Art**

Several research efforts have made important contributions in automatic creation of representative skeleton applications based on instrumentation or static analysis. Examples include [Logan2012, Dickson2017, Behzad2014b]. There is also promising early work in generalizing this capability so that workloads can be modeled from multiple data sources depending on available resources and desired fidelity [Snyder2015]. Many of these tools rely on instrumentation methods described earlier in Section 4.5.1, but it is also important to highlight that some researchers have developed techniques that specifically target the need for accurate workload modeling [Behzad2014a, Byna2008, Dorier2014]. Preliminary work has also shown the potential to extrapolate small-scale workloads into large-scale workloads [Luo2017].

Benchmarking is also an important part of SSIO research. Many HPC SSIO-related benchmarks exist, such as interleaved or random (IOR) [Shan2008], which focuses on bulk data performance, and MDTest [MDTest n.d.], which focuses on metadata operations such as file creation and deletion. Those two benchmarks are being merged into a single code base and are actively promoted as part of the IO500 initiative to standardize reporting and sharing of I/O benchmark results [Kunkel2017]. In addition, the recent DOE Coral 2 procurement included four I/O benchmarks: the aforementioned IOR and MDTest, as well as Simul and FTree [Coral2018]. The MACSio benchmark has made strides in representing multiple application patterns and I/O paradigms within a single benchmark tool [Miller2015]. Recent efforts have also have explored self-adaptive benchmarking, in which the benchmark adjusts itself to the capabilities of the system and also takes preliminary steps to isolate the sources of bottlenecks [Kunkel2018].

Efforts are also underway to extend the benchmarks beyond single case executions into regular regression tests that track performance over time for a system [Lockwood2017a, Palmer2015].

**Seeding Workshop Discussion**

Very little work exists in modeling *future* SSIO workloads. This task includes incorporation of emerging computational methods and their associated I/O patterns (notably, machine learning algorithms), techniques for extrapolating today's workloads to future systems, incorporation of benchmarking in co-design efforts, and modeling of larger-scale workflows that span multiple jobs and applications. As has been the case for many years, the field is in need of more microkernels that represent prevalent applications moving forward.

Current benchmarks also suffer from piecemeal methodology and lack of standardization. Most benchmarks measure low-level SSIO activities; only a few are intended for full stack understanding. Benchmarking results are also often reported inconsistently, with different levels of rigor with respect to timing methods, workload execution time, and sampling methods to account for variance. At a broader level, workload models, evaluation results, and the parameters needed to recreate them are not consistently disseminated in a standard way through the community. Facilities and researchers are not incentivized to run workload models and disseminate the results.

### 4.5.4 Modeling Systems

There is a long-standing need for accurate and easy-to-use modeling and simulation tools for SSIO systems. Modeling and simulation could be applied to experimentation with new hardware, new system architectures, new algorithms, and new software technologies without consuming costly production resources. It can also be used in real-time or near-real-time to aid in autonomous decision-making for deployed systems. As with the tools described in previous subsections, however, SSIO system

models require continuous maintenance and follow-up to ensure that they track rapidly evolving technology.

**State of the Art**

System modeling techniques can be broadly characterized into simulation-based methods (e.g., discrete event simulators) and analytical methods (e.g., machine learning or statistical models). Both can play a key role depending on the objective, use case, or availability of input data. For example, fine-grained models may provide the best approximation of behavior, while coarse-grained models may provide more rapid turnaround or results that are easier to understand. Some modeling techniques rely on intimate architectural knowledge, while others rely on large datasets for training.

The state of the art for SSIO system simulation is the CODES project, which has been used for a number of studies, including evaluation of burst buffer architectures and communication interference due to I/O traffic generated by burst buffers [Mubarak2017]. CODES has the ability to execute simulations at large scale using existing DOE computing resources and a modular software infrastructure that enables subsets of the model to be interchanged according to fidelity or modeling needs [Mubarak2017TPDS]. Other SSIO simulation frameworks have recently explored topics such as big data storage systems [Liu2015], wide area transfers [Kettimuthu2012, Settlemyer2012], and hierarchical storage [Luettgau2017]. Simulators such as SIMCAN [Núñez2010] and PIOSimHD [Kunkel2013] also have the ability to ingest HPC I/O traces and evaluate system performance at multiple levels.

Interconnect technology plays a key role in storage system simulation, and this field is well explored by not only CODES but also the SST project [Groves2016]. Both CODES and SST offer a range of modern interconnect topologies deployed in HPC systems and data centers. DiskSim remains the most widely used storage device simulator [DiskSim n.d.] in a variety of research endeavors, but others have recently been explored as well [Poremba2015].

The state of the art for SSIO system mathematical modeling applies machine learning techniques to derive predictive models based on observation and analysis of previous behavior [Madireddy2018, Xie2017]. Current work in this area focuses on how to account for variability and uncertainty in predictive models, how to adapt models in response to changes over time, and how to apply these results to future systems.

**Seeding Workshop Discussion**

A number of challenges remain in modeling of SSIO systems. The first two pertain to the previous subsections in this report: how do we incorporate instrumentation and monitoring (Section 4.5.1) into models for ongoing validation and evolution, and how do we combine workload and system modeling (Sections 4.5.3 and 4.5.4) into a coherent framework? While the CODES project has made progress on the latter issue, the former remains an open challenge. Second, the emergence of new technologies and new applications are stretching the scope of what needs to be modeled to gain a

thorough understanding of SSIO systems moving forward. At the component level, new storage technologies such as nonvolatile memory (NVM), shingled disks, and energy-assisted magnetic recording are not well-represented in today's models. At the system level, scientific campaigns will increasingly span multiple jobs, multiple system resources, and multiple datasets in order to produce a meaningful scientific result. This trend calls for storage system models to broaden their scope to include multijob workflows, wide area transfers, and other elements of the HPC ecosystem in a coherent modeling environment. At the modeling algorithm level, additional work is needed to accurately account for contention and other sources of variability in order to maximize the usefulness of SSIO models.

Other important issues with simulation tools include intellectual property limitations for industry-developed simulation tools, incomplete validations of the tools, and incomplete validation of data from simulation of SSIO systems. Validation of tools takes years of use and improvement. The community lacks the resources to promote standardization or broad use of modeling tools, and vendors are not incentivized in procurement to deliver behavioral models for their products.

### 4.5.5 Discussion Themes

A number of themes emerged during workshop discussion, summarized below.

**Instrumentation and Monitoring:** Attendees noted the tension between achieving the granularity of data capture necessary for some applications and limiting the impact of monitoring on system performance and behavior. One possible mechanism for addressing this tension is dynamic (or active) instrumentation, where triggers enable finer granularity of data capture under certain circumstances. Similarly, fine-grained data can be used for inferring the semantic requirements of applications, with the goal of relaxing semantics when possible. As part of this discussion, attendees noted that there are both online and offline use cases. For example, online monitoring can be used to feed control algorithms, whereas offline monitoring can be used for post hoc performance analysis and understanding high-level trends at the facility level. In addition, attendees noted that situations where data are directly accessible via load/store semantics require alternative methods of instrumentation.

Correlation with other storage layers and other aspects of the system was also discussed in detail. A crosscutting issue (within SSIO and beyond) that attendees noted was that placing behavioral data in context is key to understanding. This means connecting with behaviors at multiple levels of, and systems in, the storage stack (e.g., HDF5 and POSIX calls on the client, metadata and data operations on storage servers). Similarly, connecting with activities at the workflow level can provide much deeper insight into system-level behavior. Looking outside our field for methods of monitoring data organization and correlation was seen as valuable.

Finally, the group considered how instrumentation might be accomplished. One possibility is through deep integration of tracing into implementations of libraries

(e.g., built into structures), although current approaches along these lines require significant development effort. The attendees wondered whether there is a general approach or tool that could be of assistance. Similarly, there are open questions on what metrics (e.g., performance, memory utilization, power or energy) should be captured, how to assess the value of these in specific situations, and how to put metrics into appropriate context (e.g., as a fraction of some peak). Capturing and sharing best practice was recognized as valuable, and there is an understanding that coordination with compute facilities is needed to ensure that their concerns are met and that strategies employed can be reasonably deployed in production facilities.

**Interpretation and Application:** It is clear that simply capturing a wealth of data is not useful in itself, but rather that these data should aid in understanding and decision-making. Key use cases include: tracking how data elements move through a life cycle, feeding autonomic processes, identifying and addressing root causes of problems, understanding application use of specific SSIO services, and guiding future system procurements. Many open questions remain in how to present data to different users and software systems, both in online and post hoc scenarios.

Having methods for translating these data into actionable intelligence for users and then motivating and empowering users to make appropriate adjustments were seen as a key pragmatic issue standing in the way of making the best use of this class of data. Facility staff are another key consumer of these data, and enabling situational awareness was seen as a key use case, and one in which instrumentation of all codes was not seen as viable. Finally, combining (fusing) data through correlation and annotation were discussed as important enablers, and the inclusion of data sources outside the SSIO context key for specific use cases.

**Modeling Workloads:** The workshop attendees identified a number of challenges in practical HPC I/O workload modeling. The first set of challenges arises from application scale: how do we represent entire workloads concisely, limit the cost of model generation, and extrapolate them to larger configurations? The second set of challenges arises from concerns over accuracy: how do we distinguish declarative from imperative behavior, standardize models, protect anonymity while retaining key characteristics, and ensure that the workloads are reproducible?

There was a consensus that choosing an appropriate modeling technique and validation method depends on the use case for the model. Employing predictive models to improve job scheduling was given as one example of a class of use cases in which models are used for real-time control optimization. Those use cases rely on the understanding of cross-job interference, and they emphasize rapid feedback over absolute model accuracy. Resource provisioning was given as an alternative use case. That use case puts more emphasis on fine-grain fidelity to allocate appropriate resources for a job. In summary, different modeling techniques and different validation constraints may be appropriate for different use cases.

Emerging work in probabilistic and data-driven modeling is appealing as machines and their workloads become more complex. Even with these approaches, however, it is difficult to produce truly accurate predictions, although a flawed prediction can still yield welcome improvements in control capability.

**Modeling Systems:** The attendees identified several challenges in system modeling that revolve around disconnects between system modeling, monitoring, and model application. In particular, how do we incorporate system monitoring information into system models, how do we validate system models to ensure their accuracy over time, and how do we directly integrate simulation into the modeling ecosystem?

There are many potential uses for system modeling. It could be used to guide multitier provisioning; run what-if scenarios; and provide insights into cost, reliability, and performance trade-offs between candidate architectures. Some of these use cases have more mature modeling techniques established than others. There is no one single breakthrough that will make system modeling practically applicable for all use cases, but there are several aspects of system modeling that could be improved. There is a need for better, low-level component models that reflect the reality of today's systems. There is also a need for models that can be refined or revised in response to system changes in a timely, validated manner. The attendees also identified a need to reconcile known inaccuracies in our view of system state in complex systems as a prerequisite to model building.

**Facilitating Research:** Work in this area can be hindered by the lack of availability of resources on which to test and the lack of relevant data. For example, test systems are needed on which to ensure that approaches are viable; however, in addition, methods to orchestrate specific scenarios (e.g., a specific class of fault) are key enablers for understanding how monitoring systems react and how (or how well) they observe specific behaviors. In addition, large volumes of monitoring data are required for specific approaches to be viable, such as the application of learning approaches to better understanding system behavior. Guiding the deployment of monitoring tools on today's systems could enable more rapid development and adoption of next-generation, learning-based tools for understanding SSIO systems. Important aspects of utilization of large volumes of data of this nature are anonymization and normalization. Anonymization is key to opening up availability of this data to a larger audience, while normalization is important for adapting to changes in the facility (e.g., software upgrades). On top of these issues is one of ensuring that adequate context is captured so that data made public is not misinterpreted or misused.

### 4.5.6 Research Directions

Participants concluded that better understanding of SSIO systems leads to more productive applications; enhanced adaptation of running services; and more effective design, development, and procurement of tomorrow's applications, systems, and system software. The following priorities emerged from the discussion.

**Enabling real-time and post hoc analysis through instrumentation, capture, and retention of monitoring data.**

- Develop scalable and minimally intrusive methods of data gathering on application- and system-side, keeping pace with technological advancements in hardware and software.
- Stream delivery of data for real-time analysis and decision-making.

Scalable and minimally intrusive data-gathering methods enable the investigation of full-scale applications and systems rather than just experimental examples. This ensures that instrumentation data are representative of real-world behavior, and also enables real-time feedback to users. For such instrumentation to remain relevant, it must incorporate emerging technologies as they are deployed. This includes new hardware such as NVRAM, new programming models such as machine learning frameworks, and new architectures such as object storage systems. Furthermore, streaming delivery of data will make the instrumentation data applicable not just for *post-hoc* analysis but for real-time analysis and real-time control feedback.

**Predicting behavior through workload, software stack, and architectural modeling.**

- Develop analytical, ML, and reduced models for rapid decision-making.
- Develop validated models at multiple fidelities for design space exploration.
- Develop system models to aid in data integration.

Modeling of relevant systems, services, and workloads complement data gathering activities. A variety of validated, predictive models should be considered depending on the use case for the model. This challenge calls for further research in methods including analytical, machine learning, and reduced models to cover the spectrum of high-fidelity and rapid decision-making use cases. In addition, integration of SSIO models with models capturing other aspects of the system (e.g., network and communication workload, scheduler) must be accounted for when tackling questions of importance to facilities, both in terms of day-to-day operations and future procurement decisions.

**Integrating data sources and identifying correlations to improve our understanding.**

- Pursue real-time and post hoc data fusion and learning.
- Identify and obtain the large datasets required for learning applications.

The transformation of monitoring data into knowledge is a nontrivial task that naturally involves the fusion of data from multiple sources. Important real-time (e.g., autonomics) and post hoc (e.g., performance tuning) use cases exist with distinct needs. Machine learning is a promising approach to knowledge generation, but for

these approaches to be effective, the aforementioned challenges in instrumentation and capture must be addressed in order to gather the requisite data on which learning can occur.

### 4.6 Streaming Data

The storage systems designed for advanced scientific computing have historically been architected to absorb I/O generated from bulk-synchronous checkpoint and restart operations. However, the rate at which experimental and observational data facilities are improving in throughput and resolution are forcing their resulting data volumes and scales of analysis into the realm of high-performance computing as well. The result is a growing tension between the storage and I/O requirements of EOD sources and the infrastructure provided by ASCR facilities.

Broadly speaking, there are two major EOD analysis modes from which the majority of streaming data requirements emerge. The first is tightly coupled to data acquisition with the intent of reducing the latency between an experimental observation and insight to allow rapid refinement of experimental conditions. This mode acknowledges the scarcity of time that the experimental or observational instrument can be utilized and aims to maximize the productive output of an instrument in a fixed period of time. Facilities such as telescopes (which may be observing rare events [Nugent 2015]) and beamlines (where beam time is strictly allocated [Parkinson 2016]) are often the source of streaming data workloads operating in this mode.

The second mode of streaming data analysis occurs asynchronously to data acquisition. Whereas the source of the streaming data in the primary mode is an experimental instrument itself, the input data in this secondary mode is streamed from remote, nonvolatile storage. The goal of this secondary analysis is often to derive insight from large collections of experimental data where the precise quanta of data that are relevant to the scientific objective are not known a priori. As a result, this secondary mode may access parts of certain data objects, large collections of data objects in their entirety, and/or data objects that are distributed across disparate physical storage systems. This mode is most often used in large collaborative experiments such as the Compact Muon Solenoid detector [Bloom2015] at the Large Hadron Collider where throughput, not latency, of the analysis is the chief optimization point.

These two modes are by no means mutually exclusive, and a single experimental or observational facility may require that both tightly coupled acquisition and analysis and asynchronous streaming analysis be carried out at a large-scale computing facility. Furthermore, the precise definition of streaming data in both cases continues to be the subject of some debate; for example, data acquisition systems commonly generate the output to POSIX files, and analysis applications consume input from POSIX files. In such cases, it is unclear whether such a workflow is truly distinct from a workflow that involves simple remote file transfers.

### 4.6.1 Tightly Coupled Acquisition and Analysis

**State of the Art**

The relatively high cost of generating data through experimental or observational instruments results in their data output having very high value. It follows that committing these raw outputs to nonvolatile storage as quickly as possible is a chief design point for data acquisition systems (DAQs), and many DAQs follow the archetypal design shown in Figure 4.4.

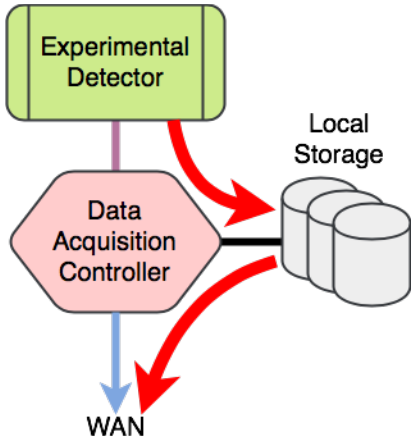


Figure 4.4: Schematic showing typical storage path for high-value data.

Because POSIX file systems have very well-defined consistency and durability semantics, DAQs attached to EOD systems often convert raw detector signals directly into POSIX files on locally attached, low-latency persistent storage. Once these file-encoded data are persisted locally, they can be transferred or streamed to a remote computing resource for analysis.

The end-to-end movement of experimental or observational data between the DAQ and computing facility can occur in several different ways, as depicted in Figure 4.5.

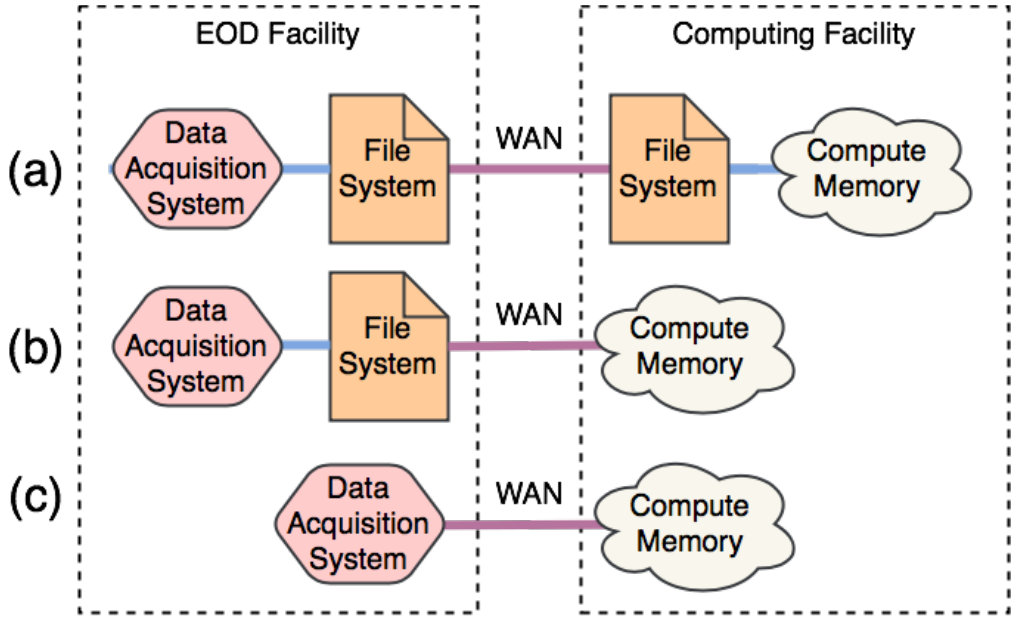


Figure 4.5: Three modes of end-to-end movement of experimental or observational data.

Mode (a) above, where a file is streamed from a local file system attached to the DAQ to a storage system to which the HPC has direct access, is most common despite the data "streaming" actually happening entirely in the form of standard POSIX file transfers via Globus, bbcp, scp, or other tools. Despite a number of significant drawbacks to scientific productivity, this is currently the prevailing mode of operation because it is fundamentally simple; file transfer tools already exist to moderate the interfacility data motion, and POSIX file systems have a very well-defined API and access semantics. As such, though, this mode is not truly streaming the data so much as it is performing near-real-time file transfer. While such near-real-time file transfers come with a set of challenges worth further discussion, these challenges are distinct from the other two modes given that file-to-file transfers are generally well understood.

A major issue with respect to file-to-file transfers is their inherently high latencies associated with persisting the transferred data in its entirety in two places before it can be processed. Even if only the first few kilobytes of a specific data object from a DAQ need to be read to determine whether the data are valid or not, the entire contents of that data unit must be transferred to the computing facility before a basic validation process can occur. Furthermore, the process by which the analysis application is informed that a datum has been successfully transferred is also often file-based; polling for a sentinel file or a complete set of data units is commonly used. These latencies of POSIX synchronization and file-based polling are antithetical to the notion of tightly coupled DAQ and analysis, leaving significant room for improvement in the responsiveness of streaming data analysis, where it may be required for applications such as experimental steering.

Mode (b) partially addresses this issue by transferring data directly from the DAQ's local storage system to the memory space from which the analysis will be computed. Because the analysis application does not have to traverse the POSIX I/O stack to access new data units as they arrive, there is potential for a significant reduction in the latency between data generation on the DAQ and analysis results from the compute. The trade-off in this case is that the data transfer between the experimental or observational facility and the computing facility is much less well defined; the analysis application is no longer receiving its input from POSIX files and instead may rely on network-based protocols (such as S3 or Swift). However, such protocols often support partial reads where required and allow analysis applications to retrieve only the data of interest over the wide-area network instead of copying entire data files.

Mode (c) minimizes the latency between data acquisition and analysis, offering the tightest coupling between the two. While this mode is the notionally optimal configuration of the three, it is also the most susceptible to data loss as a result of the data being entirely volatile from data acquisition to analysis. As previously discussed, this is not a viable solution for many EOD facility users due to their inability to tolerate data loss in the event of a failure of the wide area network or remote storage resource. Given that today's computing facilities have long served batch-oriented workloads with indeterminate latencies between job submission and execution, downtime for maintenance or long queue wait times are also commonplace. This reality further makes mode (c) untenable in the presence of networks that fail and compute facilities that do not align their maintenance windows with those of experimental facilities.

**Seeding Workshop Discussion**

As described above, the state of the art for tightly coupled data acquisition and analysis is predominantly file-based data generation, file-based data transfer, and file-based ingestion by analysis applications. This state of present practice results from a tension between the desire for low-latency, highly responsive access to data from the computational resource and the need for highly reliable data acquisition using easily implemented APIs and semantics.

A principal challenge that impedes moving from mode (a) and toward the more latency-optimized modes (b) and (c) is the lack of well-defined interfaces between DAQ systems and the compute node memory from which analysis applications ingest their data. Unfortunately, many DAQs are proprietary black-box appliances, and redefining their external interfaces compounds this challenge. Wrapping the DAQ in a data transfer agent with well-defined external interfaces would preserve the requirement for low time-to-nonvolatility while enabling external data access through APIs that are not tied to POSIX files. Other abstractions atop POSIX files, including HDF5's single writer, multiple reader (SWMR) capability, also provide a more semantically relevant API to analysis applications than what the DAQ may provide.

The tension between EOD workloads requiring highly available computing and storage resources and the shared nature of computing facilities' storage and

networking resources exposes another significant challenge. Aside from operational considerations such as maintenance windows, quality-of-service guarantees along the data path between DAQs and the analysis application would be of great benefit to streaming workloads. While most networks, transport protocols, and file systems support quality-of-service guarantees to some degree, they are not well integrated with each other and into the job control systems at computing facilities.

### 4.6.2 Asynchronous Streaming Analysis

**State of the Art**

High-value experimental datasets as described in the previous section can retain significant scientific value for years after their generation. Both large-scale experiments (such as the Compact Muon Solenoid detector [Bloom 2015] at the Large Hadron Collider) and observational data sources (such as the Sloan Digital Sky Survey [Abolfathi2018]) produce data that are consumed by thousands of researchers around the world. However, the mechanisms by which such datasets are distributed to their scientific communities are widely variable.

Owing in part to the size of these datasets, they are not often processed in their entirety; rather, researchers conduct focused analysis on subsets of the data that may be either a representative sample or a cross-section of the data types observed. If the dataset is indexed in a way that is amenable to simple subset selection, the process of transferring data is a straightforward matter of identifying the remote data resources of interest and only transferring those data to the computing facility for processing. In the cases where a suitable index is not available a priori, the process of identifying data to be analyzed can involve a significant amount of wasted data transfer to build such an index by inspecting the entire remote dataset.

The APIs and semantics of accessing these remote datasets vary. The high-energy physics community has largely standardized around the XROOTD framework [Dorigo2005], which provides the infrastructure for distributed, federated, and scalable data repositories that enable client applications to retrieve data objects from a single, remote, global namespace directly into application memory without having to stage it to a local POSIX file system. That said, XROOTD is not as semantically rich as POSIX file systems; for example, it is possible to retrieve entire data objects into application memory, but there is no straightforward mechanism to stream only a partial object (such as a data object header).

Other scientific datasets are provided in a much more manual fashion with the most rudimentary cases providing only basic HTTP access as a means to transfer remote data objects into application memory. Staging data objects to near-compute storage using Globus, bbcp, or similar file transfer tools is also prevalent for the same reason that file-to-file transfers prevail in the tightly coupled acquisition and analysis case: POSIX files provide a very well-defined API and set of access semantics, which analysis applications can easily adopt.

In all cases, asynchronous streaming analysis is distinct from tightly coupled streaming analysis in that it does not have the extreme requirement of high responsiveness and low latency between signal generation and analyzed data. Rather, asynchronous streaming workloads are driven by the desire to process as much data in as short a period of time as possible. Furthermore, such workflows are not necessarily open loops as is common with tightly coupled streaming analysis; there are cases where it would be advantageous to make multiple passes over data during analysis [Fox2015], as would be the case in algorithm development or neural network training.

**Seeding Workshop Discussion**

Because asynchronous streaming analysis often targets only subsets of large distributed datasets, a principal challenge lies in efficiently identifying and streaming only those data that are of scientific interest to its consumer. Such remote filtering and sampling tasks may map well to active storage systems, which can perform data processing and indexing without requiring wholesale data transfer to the computing facility. However, as discussed in Section 4.1.3, the programming models and APIs to perform these tasks are not well defined beyond existing relational and nonrelational database interfaces.

This lack of well-defined models and semantics can be generalized to other aspects of streaming data analysis. While the high-energy physics community has demonstrated successes with the XROOTD framework, such successes remain exceptional in all but the largest experimental and observational scientific communities. A common set of access semantics across different data sources have not been identified, and the data models used to store and represent these data are also widely varied across disciplines. While efforts such as HDF5's SWMR mode are making progress toward defining a meaningful data model and access API, it still relies on a network file system to provide a protocol for streaming data.

### 4.6.3 Discussion Themes

Because streaming data reflects a usage modality that touches many of the aforementioned computer science challenges, the discussion around streaming data at the workshop was distinctly crosscutting. Over the course of the session, the attendees expressed positions that revolved around three central themes that are relevant to a broad range of specific streaming data workflows and use cases.

**Enablement of multimodal analysis and replicated data streams.** It is rarely the case that an experimental facility generates only a single data stream with a homogenous data type. Rather, it is far more common for an experimental instrument to generate a multitude of data from different sensors as described in Section 3.3.1. Similarly, the output of large-scale simulations are often characterized by high dimensionality outputs. In both cases, the diversity of data types represent multiple types of scientific data that may be relevant to only a subset of the research

community; by extension, researchers often need to access only a subset of the data stream.

This theme of enabling researchers to access different subsets of a data stream aligns well with the more general workflow optimization technique of minimizing data motion to increase workflow throughput. It simplifies the analysis burden on domain scientists by only returning relevant data, and it additionally increases scientific throughput by eliminating the situation where different subcommunities have to line up and wait for their relevant part of the data stream to arrive.

The requirement to replicate data streams also arose during the discussion. Tasks such as neural network training (and retraining) require the consumption of large sets of data that benefit from being streamed, and workflows that employ different applications of artificial intelligence would benefit from being able to consume replicated streams.

In addition, globally streaming data sources such as those originating from very large experiments, as described in Section 4.6.2, often benefit from geographically distributed caches by reducing the need to transfer data over network paths. While streaming the same data to multiple destinations from a static data source does not require strong synchronicity as would be enabled by replicated streams, the practice of caching or buffering large datasets is likely to benefit from research applied toward enabling synchronously replicated streams.

**Flexibility to operate across diverse storage and data resources.** By definition, streaming data involves at least two different data resources: the data source and the destination computational facility. Efficiently using the storage infrastructure at the sources and destinations becomes increasingly challenging as the heterogeneity between these end points increases, and it was agreed that any tools, methods, and infrastructure designed to address the challenges surrounding streaming data must be flexible enough to accommodate this infrastructural diversity.

A number of specific infrastructural complexities were enumerated by participants:

- Many EOD facilities have local computing and storage resources, so the tools and methods for streaming data management must handle both intrafacility and interfacility streaming.
- Small-scale and large-scale computing both play a role in streaming data analysis; however, the specific computations that run on each may be different. For example, small, local computing may perform trivially parallel compression and filtering ahead of streaming to a large-scale computing facility for more complex parallel data analysis.
- Streaming data sources and destinations may possess only compute or only storage.

- In-network compute and storage may also be present and enable processing of data in transit if these resources can be accessed effectively.

The discussion also drew many parallels with other research areas' discussions at the workshop. For example, heterogeneous and tiered storage systems that may have been designed with sustained I/O workloads such as those generated by checkpoint/restart may have components that can distinctly benefit from streaming data analysis as well. Flash-based acceleration layers may also be able to support small transactional record-based I/O such as those encountered at Jefferson Laboratory. A better understanding of both storage system capabilities and the workflow applications that operate on streaming data will be required to determine the best way to leverage diverse storage and data resources without losing flexibility. It follows that there are opportunities for synergy across SSIO research areas that should be identified and exploited wherever possible.

**Resilience to interruptions that may otherwise cause experimental data to be lost.** As discussed in Section 4.6.1, there are classes of experimental data that are irreproducible and therefore require extremely reliable storage systems to ensure that rare phenomenon are correctly persisted and retained as quickly as possible. Workshop attendees noted that current storage and I/O systems at today's HPC facilities do not typically offer much flexibility or expressivity in this regard; as a result, streaming data workflows generally cannot adjust their behavior when operating over SSIO systems that offer different levels of reliability or resilience. Those data streams which are irreproducible are therefore cached to a persistent local buffer, increasing the resource consumption and time to insight.

Participants noted that the ultimate goal for streaming data would be true end-to-end QoS guarantees. The interposition of networked POSIX file systems in streaming data analysis makes this objective challenging; however, defining mechanisms by which data sources can express their resilience requirements or data destinations can advertise their resilience capabilities would enable workflows to make decisions about what resources would best match their needs. For example, if a tightly coupled acquisition and analysis workflow needs the highest possible reliability to ensure data that from an instrument are not lost, it may choose to use a highly reliable network path and remote storage resource if available. Should such resources not be available, it could choose to be sent to a local cache so as to maintain the required resilience at the cost of performance.

### 4.6.4 Research Directions

The session clearly established that streaming data from both experimental and observational data and from modeling and simulation are a new class of large-scale computational workload. Establishing flexible, foundational methods to support streaming data analysis will simplify scaling and reduce time to insight from new streaming data sources, and the following priorities emerged along the three themes described previously in Section 4.6.3.

**Providing means for multimodal analysis by enabling one data source to serve multiple, different research efforts.**

- Allow scientists to subscribe to subsets of data for different scientific objectives.
- Cache parts of data streams to support geographically distributed research efforts.

Allowing domain researchers to specify the parts of a data stream that are relevant to them would enable a much tighter coupling between experimental and observational data sources that are intrinsically complex and multimodal. Domain scientists would be able to draw only the data relevant to them without having to wait in line as other data are being delivered over the stream, reducing the latency to the first bit of valuable data during tightly coupled acquisition and analysis. Similarly, asynchronous streaming analysis would also benefit from such flexibility by reducing overall data motion during their streaming data workflows and enabling more effective use of storage and network caches available along the data path.

Enabling such multimodal data streams may benefit from the application of many new hardware and software techniques; for example, active storage could facilitate remote filtering and indexing of data to accelerate the rate at which a subset of a data stream can be identified and delivered. Different data types may also come with different output frequencies, performance optima, and resilience requirements that naturally map to different storage tiers within a computing facility or storage resources across computing facilities. The development of multidimensional data stores may also provide unique benefits to multimodal data streams by more naturally mapping the different logical components of a data stream to the storage systems ingesting those data.

**Identifying overlapping elements across different streaming data workflows to establish common interfaces through which data streams can be accessed and processed.**

- Enable in-line processing of data to and from both local and remote data and computing resources, ranging from simple compression/filtering to substantive analysis.
- Efficiently utilize storage resources at the data source, destination, and network.

Experimental workflows from different science domains often have similar technological requirements. Identifying a set of common operations, methods, and patterns would enable the creation of common tools for working with streaming data. These common interfaces would frame a single layer of abstraction that would enable domain scientists to adopt the best practices and optimal approaches for handling data streams without being exposed to the underlying hardware, middleware, and

system software that enables high performance. Such a common abstraction for streaming data may include features such as in-line processing, which can reduce the need to move all of the data between a streaming data source and the computational resources processing that data.

On a related note, using storage resources located at the data source, destination, and all stops along the data path would enable data reduction in transit to the same end. The challenges of effectively placing data as it moves draws parallels with challenges in using deep storage hierarchies; finding the optimal place for data streams or stream buffers based on the consumers of that data can have a dramatic effect on improving the overall efficiency of the workflow operating on that data. As such, there is significant overlap between the research directions outlined in Section 4.4 and streaming data analysis.

**Supporting different reliability and performance requirements for data streams.**

- Enable data sources to express reliability/performance requirements.
- Allow both end points to adapt to advertised unreliability in network and storage along data path.

Some experimental data cannot be recreated and must not be lost, whereas others can tolerate some loss if it enables faster time to science. Establishing methods and software for experiments to express these requirements and negotiate such trade-offs will be essential to allow different streaming data workflows to choose the storage and computing resources that best meet their requirements. On a related point, storage systems that can provide strong performance guarantees are expected to become essential for streaming data analysis. This expectation applies not only to traditional dimensions of performance, such as streaming I/O bandwidth, but to latency and the variation in performance as well. Meeting this expectation will require developing a better understanding of transient performance variation as well as of the longer-term performance degradation caused by time-dependent evolution of both software and physical media.

It is notable that this requirement for better introspection into storage systems' intrinsic performance and resilience capabilities was also identified as an opportunistic direction for research in the context of workflows (Section 4.3) and the storage hierarchy (Section 4.4). This commonality speaks to a greater need to allow scientific workflows, whether they depend on streaming data or not, to be able to more effectively couple with the storage system and I/O resources on which they are executing.

# 5 Supporting Activities

Although the workshop did not have a dedicated session on supporting activities for SSIO research, throughout the workshop supporting activities were called out as important for the community in order to perform the needed research. While these topics are not research activities, the workshop attendees saw them as necessary ingredients to inform, enable, and sustain research into SSIO areas.

The supporting activities that were discussed focused on two themes. The first was the availability of forward-looking (at reasonable scale) computing and storage resources (testbeds) on which realistic experiments associated with SSIO R&D topics could be performed. The second theme was the need for highly documented operational data of existing leading-edge computing systems, network systems, storage systems, and their workloads. Failure, performance, and usage-related data in an understandable, clean, and documented form were all deemed essential in order to assist SSIO researchers with deep understanding of modern SSIO problem spaces and their projections to future systems. These data have taken on a new importance in this era of machine learning.

## 5.1 Computing, Networking, and Storage Resources

Many of the challenges inherent in building storage systems and I/O capabilities to support DOE science revolve around finding solutions that scale to systems at the cutting edge of HPC and that can make best use of new technologies that have just arrived on the market. Thus, research in this domain is dependent upon the availability of relevant test platforms.

**State of the Art**

Several capabilities were identified as desirable for ensuring the availability of computing and storage resources. Cloud-oriented systems research mechanisms such as Chameleon [Chameleon2015] and Cloud Lab [CloudLab2015] were thought to have some utility in providing computing resources but are limited to R&D that can be run in a cloud-based environment. The NSF GENI [Geni2006] suite supporting research in networking was also mentioned as a potentially useful method of enabling related SSIO research, especially research involving long-distance networking. Other, more HPC-oriented resources such as the DOE/NNSA/Lawrence Livermore National Laboratory (LLNL) Hyperion [Hyperion n.d.] and the NSF/DOE/NNSA/LANL Parallel Reconfigurable Observational Environment (PRObE) [NMC2019] systems were also mentioned as useful; however, these systems are likely insufficient for future SSIO research without expanding the types of hardware and experiments they will support. Hyperion is not managed for openly competed research for extreme-scale computing, but its hardware is relatively new.

PRObE was the only recent resource that was specifically designed so researchers could have full access all the way to the hardware, and it allowed root access for

reasonably long periods of time. This ability to have full root access to the real, not virtual, testbed hardware is important. Because PRObE utilizes retired DOE hardware, however, the architectures were not well suited for future research. Unfortunately, neither the PRObE nor the Hyperion systems are still in service.

**Challenges**

The lack of modern and periodically renewed, large-scale testbed computing environments is a significant challenge, and this includes in-system and off-system storage and the ability to provide users with "bare metal" root access. PRObE and Hyperion do meet some aspects of this need area, but their accessibility levels do not extend to enough researchers and are not always modern enough to satisfy the need. The ability to try out new hardware mechanisms is also a requirement. Neither PRObE nor Hyperion is funded specifically to assist the national SSIO research community. Root access to bare metal hardware is needed in order to support reproducibility in system-level experimentation. Neither facility is rich with instrumentation tools, fault injectors, or other generally useful testing tools.

The need is really a combination of new and frequently renewed hardware in an openly competed-for resource that allows access all the way to the hardware. Also needed is coordination with efforts in advanced architecture, as well as future software stack development.

## 5.2 Availability of Highly Documented Operational Data

Relevant data informs researchers about the behavior of applications, the patterns of access in facility systems, the types of hardware failures observed, and other important details that guide the selection of research direction and improve our ability to assess solutions. With the increased adoption of machine learning techniques, these data can also be used to train models for a variety of purposes.

**State of the Art**

Several facilities have made well-documented operational data available. Among the releases are data from LANL [LANL Data n.d.] and NERSC. The LANL failure data release represented the largest operational failure data release performed in two decades when it was released in 2006. It came with FAQs, and much care was taken to clean the data well.

Storage-oriented data are provided by the large-scale Sandia trace data [Sandia Data n.d.] and the Argonne Darshan usage-related data [Carns2013, Argonne n.d.]. Not only is the Darshan data an example of well-produced data, but it also represents a usable tool by HPC sites to assist with data collection. In addition, storage system namespace statistics are provided by the DOE Petascale Data Storage Institute's FSStats effort [Felix2011, Dayal2008]. Included are data from many HPC sites, a tool for collecting the data, and even a multisite clearinghouse for making the data available. BackBlaze also released a sizable data collection that tracked hard disk failures over time [BackBlaze2015].

**Challenges**

A major challenge in this area is funding incentive for researchers to create tools to assist HPC sites to accurately produce, curate, and provide operational data (including SSIO-related data) without adversely affecting production operation. Data involving operations, failure, repair, use, and performance, for all layers of storage and for both data and metadata, are needed in order to engage the full SSIO research community. Ideally, data would span from applications, complex workflows, and all other uses of storage down to mechanisms including management of the SSIO systems themselves. This data would be provided in a consistent way over many years, and it would be periodically updated. Facilities may also need incentives to deploy tools that can capture this data, to perform necessary anonymization and documentation, and to make this data available to the community.

Funding also is needed in order to pay for collection, curation, and broad dissemination of this operational data. If the data are not well cared for and are not periodically updated to current thinking and architectures, then the full community cannot engage, and leverage will be lost.

# 6 Glossary

| | |
|---|---|
| ADIOS | Adaptive I/O System |
| ALCF | Argonne Leadership Computing Facility |
| AMO | Atomic Memory Operations |
| AMR | Adaptive Mesh Refinement |
| API | Application Programming Interface |
| Argonne | Argonne National Laboratory |
| ASCR | Office of Advanced Scientific Computing Research |
| CMS | Compact Muon Solenoid (experiment at CERN) |
| DAQ | Data Acquisition |
| DMP | Data Management Plan |
| DNA | Deoxyribonucleic Acid |
| DOE | U.S. Department of Energy |
| DRAM | Dynamic Random Access Memory |
| EOD | Experimental and Observational Data |
| FES | Fusion Energy Sciences (DOE-SC) |
| FLOPS | Floating Point Operations per Second |
| HDF | Hierarchical Data Format |
| HPC | High-Performance Computing |
| HPSS | High-Performance Storage System |
| HSM | Hierarchical Storage Management |
| I/O | Input/Output |
| IOR | Interleaved Or Random |
| LANL | Los Alamos National Laboratory |
| LDMS | Lightweight Distributed Metric Service |
| LLNL | Lawrence Livermore National Laboratory |
| MDS | MetaData Server |
| MIMD | Multiple Instruction Multiple Data |
| ML | Machine Learning |
| NERSC | National Energy Research Scientific Computing Center |
| NNSA | National Nuclear Security Administration |
| NSF | National Science Foundation |
| NVRAM | Non-Volatile Random Access Memory |
| NVMe | Non-Volatile Memory Express |
| OLCF | Oak Ridge Leadership Computing Facility |
| ORNL | Oak Ridge National Laboratory |
| OS | Operating System |
| PIC | Particle-in-Cell |
| PDSI | Petascale Data Storage Institute |
| PLFS | Parallel Log-structured File System |
| PNNL | Pacific Northwest National Laboratory |
| POSIX | Portable Operating System Interface |
| PRObE | Parallel Reconfigurable Observational Environment |
| PSNR | Peak Signal to Noise Ratio |

| | |
|---|---|
| PVFS | Parallel Virtual File System |
| QoS | Quality of Service |
| R&D | Research and Development |
| RAID | Redundant Array of Independent Disks |
| RMA | Remote Memory Access |
| SC | Office of Science (DOE) |
| SciDAC | Scientific Discovery through Advanced Computing |
| SCM | Storage-Class Memory |
| SCR | Scalable Checkpoint/Restart |
| SIMD | Single Instruction Multiple Data |
| SSD | Solid State Disk |
| SSIO | Storage System and I/O |
| SST | Structural Simulation Toolkit or Sustainable Staging Transport |
| SWMR | Single Writer, Multiple Reader |
| TTFB | Time To First Byte |
| UQ | Uncertainty Quantification |
| XDD | Command line tool for measuring I/O performance |

# 7 Workshop Attendees

| | |
|---|---|
| Mark Ainsworth | Brown University |
| George Amvrosiadis | Carnegie Mellon University |
| Michael Bender | Stony Brook University |
| John Bent | DataDirect Networks |
| Wes Bethel | Lawrence Berkeley National Laboratory |
| Laura Biven | DOE |
| David Bonnie | Los Alamos National Laboratory |
| Ron Brightwell | Sandia National Laboratories |
| Benjamin Brown | DOE |
| Ali Butt | Virginia Tech |
| Suren Byna | Lawrence Berkeley National Laboratory |
| Raghunath Chandrasekar | Amazon Web Services |
| Antonio Cortes | Barcelona Supercomputing Center |
| Matthew Curry | Sandia National Laboratories |
| Ewa Deelman | University of Southern California |
| Andreas Dilger | Whamcloud |
| Daniel Ernst | Cray, Inc. |
| Lance Evans | Cray, Inc. |
| Evan Felix | Pacific Northwest National Laboratory |
| Greg Ganger | Carnegie Mellon University |
| Ada Gavrilovska | Georgia Institute of Technology |
| Elsa Gonsiorowski | Lawrence Livermore National Laboratory |
| Gary Grider | Los Alamos National Laboratory |
| Kevin Harms | Argonne National Laboratory |
| Graham Heyes | Thomas Jefferson National Accelerator Facility |
| Dean Hildebrand | Google |
| Terry Jones | Oak Ridge National Laboratory |
| Kristy Kallback-Rose | NERSC/LBNL |
| Dimitrios Katramatos | Brookhaven National Laboratory |
| Scott Klasky | UT-Battelle, Oak Ridge National Laboratory |
| Quincey Koziol | Lawrence Berkeley National Laboratory |
| Lingda Li | Brookhaven National Laboratory |
| Glenn K. Lockwood | Lawrence Berkeley National Laboratory |
| Jay Lofstead | Sandia National Laboratories |
| Johann Lombardi | Intel Corporation |
| Darrell Long | University of California, Santa Cruz |
| Xiaosong Ma | Qatar Computing Research Institute |
| Carl Maltzahn | University of California, Santa Cruz |
| Kathryn Mohror | Lawrence Livermore National Laboratory |
| Thomas Ndousse-Fetter | DOE |
| Bogdan Nicolae | Argonne National Laboratory |
| Lucy Nowell | DOE |
| Manish Parashar | Rutgers University |

| | |
|---|---|
| Amedeo Perazzo | SLAC National Accelerator Laboratory |
| Tom Peterka | Argonne National Laboratory |
| Robinson Pino | DOE |
| Eric Pouyoul | SND/ESnet Lawrence Berkeley National Lab |
| Lavanya Ramakrishnan | Lawrence Berkeley National Laboratory |
| Robert Ross | Argonne National Laboratory |
| Sonia Sachs | DOE |
| Joel Saltz | Stony Brook University |
| Malachi Schram | Pacific Northwest National Laboratory |
| Bradley Settlemyer | Los Alamos National Laboratory |
| Michela Taufer | The University of Tennessee Knoxville |
| Deneise Terry | Oak Ridge Institute for Science and Education |
| Angie Thevenot | DOE |
| Sudharshan Vazhkudai | Oak Ridge National Laboratory |
| Lee Ward | Sandia National Laboratories |
| Jack Wells | Oak Ridge National Laboratory |
| Matthew Wolf | Oak Ridge National Laboratory |
| Weikuan Yu | Florida State University |

# 8 Workshop Agenda

The workshop was organized around six topical areas identified by the organizers as areas where gaps exist in the capabilities required for DOE science to proceed at pace. Each day began with an initial set of talks and a panel that provided context and background for the attendees, followed by rigorous moderated discussion.

| **Wednesday, September 19, 2018** | |
|---|---|
| Time | Activity |
| 8:15am – 8:35am | Welcome (Barb Helland) and opening remarks (Lucy Nowell)<br>Reminder of charge, overview of meeting, safety, etc. |
| 8:35am – 8:50am | Talk: Experimental and Observational Data<br>Wes Bethel |
| 8:50am – 9:05am | Talk: Streaming Data<br>Graham Heyes |
| 9:05am – 9:20am | Talk: Workflow Management<br>Tom Peterka |
| 9:20am – 10:00am | Talk: Science requirements for SSIO at the LCFs<br>Jack Wells & Kevin Harms |
| 10:00am – 10:30am | Break |
| 10:30am – 11:45am | Panel: Applications and Facilities Requirements (Application Pull)<br>Moderator: Kathryn Mohror<br>Participants: Wes Bethel, Graham Heyes, Jack Wells, Kevin Harms, Evan Felix, Tom Peterka, Kristy Kallback-Rose |
| 11:45am –12:35am | Lunch |
| 12:35pm – 2:05pm | Working Session 1: Integrating with Science Workflows<br>Moderator: Scott Klasky<br>Scribe: Brad Settlemyer |
| 2:05pm – 2:35pm | Break |
| 2:35pm – 4:05pm | Working Session 2: Understanding SSIO Systems<br>Moderator: Rob Ross<br>Scribe: Galen Shipman |
| 4:05pm – 4:25pm | Break |
| 4:25pm – 5:55pm | Working Session 3: Streaming Data<br>Moderator: Matt Wolf |

| | |
|---|---|
| | Scribe: Glenn Lockwood |
| **Thursday, September 20, 2018** | |
| Time | Activity |
| 8:15am – 8:30am | Talk: Extreme Heterogeneity Workshop Report<br>Lucy Nowell |
| 8:30am – 8:50am | Talk: Storage Technologies<br>Gary Grider |
| 8:50am – 9:10am | Talk: Memory Technologies; Blurring the Lines<br>Dan Ernst |
| 9:10am – 10:15am | Panel: Storage Technologies<br>(Tech Push Panel)<br>Moderator: Lee Ward<br>Participants: Gary Grider, Kevin Harms, Eric Pouyoul, Dan Ernst, Lance Evans |
| 10:15am – 10:45am | Break |
| 10:45am – 12:15pm | Working Session 4: Heterogeneous/multi-tier storage systems<br>Moderator: Kathryn Mohror<br>Scribe: Kevin Harms |
| 12:15pm – 1:15pm | Lunch |
| 1:15pm – 1:30pm | Talk: ISDM Workshop<br>Tom Peterka |
| 1:30pm – 3:00pm | Working Session 5: Metadata, Name Spaces, and Provenance<br>Moderator: Lee Ward<br>Scribe: Quincey Koziol |
| 3:00pm –- 3:25pm | Break |
| 3:25pm – 4:55pm | Working Session 6: HW/SW architectures<br>Moderator: Brad Settlemyer<br>Scribe: Rob Ross |
| 4:55pm – 5:00pm | Closing remarks and adjourn |

# 9 References


[Abadi2016] Abadi, M., et al. “Tensorflow: a system for large-scale machine learning.” *OSDI* 16 (2016) 265–283.

[Abbasi2010] Abbasi, H., et al. “Datastager: scalable data staging services for petascale applications.” *Cluster Computing* 13.3 (2010): 277–290.

[Abolfathi2018] Abolfathi, B., et al. “The Fourteenth Data Release of the Sloan Digital Sky Survey: First Spectroscopic Data from the Extended Baryon Oscillation Spectroscopic Survey and from the Second Phase of the Apache Point Observatory Galactic Evolution Experiment.” *ApJ Supp.* 235 (2018) 2.

[Acharya1998] Acharya, A., M. Uysal, and J. Saltz. “Active disks: Programming model, algorithms and evaluation.” Proc. ASPLOS'98, 1998.

[Adams2012] Adams, I.A., B.A. Madden, J.C. Frank, M.W. Storer, E.L. Miller, and G. Harano. “Usage behavior of a large-scale scientific archive.” Proc. SC12, Nov. 2012.

[Adelmann2005] Adelmann, A., R.D. Ryne, J.M. Shalf, and C. Siegerist. "H5part: A portable high performance parallel data interface for particle simulations." Proc. Particle Accelerator Conference, pp. 4129–4131, 2005.

[ADIOS n.d.] “ADIOS framework for scientific data on exascale systems,” n.d. https://www.exascaleproject.org/project/adios-framework-scientific-data-exascale-systems/, accessed Apr. 24, 2019.

[Agelastos2014] Agelastos, A., et al. “The Lightweight Distributed Metric Service: A scalable infrastructure for continuous monitoring of large scale computing systems and applications.” Proc. International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 154–165, 2014.

[Aghayev2015] Aghayev, A., and P. Desnoyers. “Skylight—A Window on Shingled Disk Operation.” Proc. 13th USENIX Conference on File and Storage Technologies, pp. 135–149, 2015.

[Aghayev2017] Aghayev, A., T. Ts’o, G. Gibson, and P. Desnoyers. “Evolving EXT4 for Shingled Disks.” Proc. 15th USENIX Conference on File and Storage Technologies, 2017.

[Ainsworth2017] Ainsworth, M., S. Klasky, and B. Whitney. “Compression using lossless decimation: Analysis and application.” *SIAM Journal on Scientific Computing*, 39.4 (2017):B732–B757.

[Albrecht2017] Albrecht, J., et al. *A Roadmap for HEP Software and Computing R&D for the 2020s*. HSF-CWP-2017-01. HEP Software Foundation. Dec. 15, 2017.

[Ali2009] Ali, N., P. Carns, K. Iskra, D. Kimpe, S. Lang, R. Latham, R. Ross, L. Ward, and P. Sadayappan. "Scalable I/O forwarding framework for high-performance computing systems." Proc. IEEE International Conference on Cluster Computing and Workshops, pp. 1–10, 2009.

[Allcock2005] Allcock, W., J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster. "The Globus striped GridFTP framework and server." Proc. of the 2005 ACM/IEEE conference on Supercomputing (SC '05): 54, 2005.

[Allen2004] Allen, B. Monitoring Hard Disks with SMART. *Linux Journal*, Jan. 1, 2004.

[Alted2010] Alted, F. "Why modern CPUs are starving and what can be done about it." *Computing in Science & Engineering* 12.2 (2010): 68–71.

[Altintas2004] Altintas, I., C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S.K. Mock. "An extensible system for design and execution of scientific workflows." Proc. 16th International Conference on Scientific and Statistical Database Management, pp. 423–424, 2004.

[Alverson2012] Alverson, B. "Cray High Speed Networking." Proc. 20th IEEE Symposium on High Performance Interconnects, 2012.

[Ames2011] Ames, S., M.B. Gokhale, and C. Maltzahn. "QMDS: A File System Metadata Management Service Supporting a Graph Data Model-Based Query Language." Proc. 6th IEEE International Conference on Networking, Architecture and Storage, pp. 268, 277, July 28–30, 2011.

[Amiri2000] Amiri, K., et al. "Dynamic function placement for data-intensive cluster computing." Proc. USENIX Annual Technical Conference, General Track, 2000.

[Anderson2000] Anderson, D.C., J.S. Chase, and A.M. Vahdat. "Interposed request routing for scalable network storage." Proc. 4th Symposium on Operating System Design & Implementation, Volume 4. USENIX Association, 2000.

[Anderson2018] Anderson, P., et al. "Glass: A New Media for a New Era?" Proc. 10th USENIX Workshop on Hot Topics in Storage and File Systems, 2018.

[APEX2015] APEX Workflows. LANL, NERSC, SNL, SAND2015-10342 O LA-UR-15-29113, Nov. 24, 2015.

[Argonne n.d.] Argonne National Laboratory. "ALCF I/O data repository," n.d. http://press3.mcs.anl.gov/darshan/data/, accessed Apr. 24, 2019.

[Arkin2016] Arkin, A., et al. Biological and Environmental Research Exascale Requirements Review, Advanced Scientific Computing Research and Biological and Environmental Research, DOE Office of Science, Rockville, Maryland, March 28–31, 2016.

[ATLAS2017] ATLAS Collaboration. *Search for R-parity-violating supersymmetric particles in multi-jet final states produced in p--p collisions at sqrt(s) = 13 TeV using the ATLAS detector at the LHC*. CERN-EP-2017-298. http://arxiv.org/abs/arXiv:1804.03568, accessed Apr. 24, 2019.

[Aviles-Gonzalez2014] Avilés-González, A., J. Piernas, and P. González-Férez. "Scalable metadata management through OSD+ devices." *International Journal of Parallel Programming* 42.1 (2014): 4–29.

[BackBlaze2015] Back Blaze (2015). https://www.backblaze.com/hard-drive-test-data.html, accessed Apr. 24, 2019.

[Barton2013] Barton, E. "Lustre* Fast forward to exascale." Proc. Lustre User Group Summit 2013, March 2013.

[Barton2014] Barton, E., J. Bent, and Q. Koziol. "Fast forward storage and IO program documents." https://wiki.hpdd.intel.com/display/DC/Resources, accessed April 24, 2019.

[Baru1998] Baru, C., R. Moore, A. Rajasekar, and M. Wan. "The SDSC storage resource broker." Proc. 1998 Conference of the Centre for Advanced Studies on Collaborative Research, p. 5, 1998.

[Bauer2012] Bauer, M., S. Treichler, E. Slaughter, and A. Aiken. "Legion: expressing locality and independence with logical regions." Proc. High Performance Computing, Networking, Storage and Analysis, p. 66, 2012.

[Bauer2014] Bauer, M. "Legion: Programming distributed heterogeneous archtectures with logical regions." Ph.D. dissertation, Stanford University, Dec. 2014.

[Bautista-Gomez2011] Bautista-Gomez, L., S. Tsuboi, D. Komatitsch, F. Cappello, N. Maruyama, and S. Matsuoka. "FTI: high performance fault tolerance interface for hybrid systems." Proc. 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, p. 32, 2011.

[Beck2002] Beck, M., T. Moore, and J.S. Plank. "An end-to-end approach to globally scalable network storage." *ACM SIGCOMM Computer Communication Review* 32 (2002): 4.

[Behzad2014a] Behzad, B., S. Byna, S. Wild, Prabat, and M. Snir. "Improving Parallel I/O Autotuning with Performance Modeling." Proc. 23rd International ACM Symposium on High Performance Distributed Computing, June 2014.

[Behzad2014b] Behzad, B., H.-V. Dang, F. Hariri, W. Zhang, and M. Snir. "Automatic generation of I/O kernels for HPC applications." Proc. Parallel Data Storage Workshop, 2014.

[Behzad2015] Behzad, B., S. Byna, S. Wild, M. Prabhat, and M. Snir. Dynamic Model-driven Parallel I/O Performance Tuning. IEEE Cluster, Sept. 2015.

[Bennett2012] Bennett, J.C., et al. "Combining in-situ and in-transit processing to enable extreme-scale scientific analysis." Proc. International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–9, 2012.

[Bent2004] Bent, J., D. Thain, A. Arpaci-Dusseau, and R. Arpaci-Dusseau. "Explicit Control in a Batch-Aware Distributed File System." Proc. First USENIX/ACM Conference on Networked Systems Design and Implementation, March 2004.

[Bent2009] Bent, J., G. Gibson, G. Grider, B. McClelland, P. Nowoczynski, J. Nunez, M. Polte, and M. Wingate. "PLFS: A checkpoint filesystem for parallel applications." Proc. Conference on High Performance Computing Networking, Storage and Analysis, p. 21, 2009.

[Bent2012] Bent, J., S. Faibish, J. Ahrens, G. Grider, J. Patchett, P. Tzelnic, and J. Woodring. "Jitter-free co-processing on a prototype exascale storage stack." Proc. 28th IEEE Symposium on Massive Storage Systems and Technologies, 2012.

[Bent2016] Bent, J., B. Settlemyer, and G. Grider. "Serving Data to the Lunatic Fringe: The Evolution of HPC Storage," *USENIX Magazine*, 41.2 (2016).

[Berger1989] Berger, M.J., and P. Colella. "Local adaptive mesh refinement for shock hydrodynamics." *Journal of Computational Physics* 82.1 (1989): 64–84.

[Bethel2015] Bethel, E., et al. *Management, Analysis, and Visualization of Experiment and Observational Data (EOD)*, DOE Office of Science, Bethesda, MD, 2015. https://science.energy.gov/~/media/ascr/pdf/programdocuments/docs/ascr-eod-workshop-2015-report_160524.pdf, accessed Apr. 24, 2019.

[Betke2017] Betke, E., and J. Kunkel (2017). "Real-Time I/O-Monitoring of HPC Applications with SIOX, Elasticsearch, Grafana and FUSE." In Kunkel, J., et al. (eds.) *High Performance Computing*. Lecture Notes in Computer Science, vol. 10524.

[Bhimji2017] Bhimji, W., et al. Deep Neural Networks for Physics Analysis on low-level whole-detector data at the LHC. *arXiv preprint:1711.03573* (2017).

[Birman2007] Birman, K. "The promise, and limitations, of gossip protocols." *ACM SIGOPS Operating Systems Review* 41.5 (2007): 8–13.

[Blomer2015] Blomer, J., et al. "The evolution of global scale filesystems for scientific software distribution." *Computing in Science & Engineering* 17.6 (2015): 61–71.

[Bloom2015] Bloom, K., et al. "Any Data, Any Time, Anywhere: Global Data Access for Science." Proc. 2015 IEEE/ACM 2nd International Symposium on Big Data Computing, pp. 85–91, 2015.

[Boboila2012] Boboila, S., Y. Kim, S.S. Vazhkudai, P. Desnoyers, and G.M. Shipman. "Active flash: Out-of-core data analytics on flash storage." Proc. IEEE 28th Symposium on Mass Storage Systems and Technologies, pp. 1–12, 2012.

[Bockelman2018] Bockelman, B. CMS IO Overview. HEP-CCE: Scalable IO for Energy and Intensity Frontier Experiments. Argonne National Laboratory, Lemont, Ill., Aug. 23, 2018.

[Bonnie2018] Bonnie, D. "Massive Scale Metadata Efforts and Solutions." Proc. 34th International Conference on Massive Storage Systems and Technology, 2018.

[Bonoli2015] Bonoli, P., and L. Curfman McInnes. *Report of the Workshop on Integrated Simulations for Magnetic Fusion Energy Sciences*. Sponsored by Office of Fusion Energy Sciences and Office of Advanced Scientific Computing Research, Rockville, MD, June 2–4, 2015.

[Bornholt2016] Bornholt, J., R. Lopez, D. Carmean, L. Ceze, G. Seelig, and K. Strauss. "A DNA-based Archival Storage System." Proc. International Conference on Architectural Support for Programming Languages and Operating Systems, 2016

[Braam2002] Braam, P.J., and P. Schwan, "Lustre: The intergalactic file system," Ottawa Linux Symposium, pp. 50, 2002.

[Braam2004] Braam, P.J. "The Lustre storage architecture." 2004.

[Braam2017] Braam, P., and D. Bonnie. "Campaign Storage," Proc. International Conference on Massive Storage Systems and Technology, 2017.

[Brewer2015] Brewer, E.A. "Kubernetes and the path to cloud native." Proc. 6th ACM Symposium on Cloud Computing, 2015.

[Brinkmann2014] Brinkmann, A., T. Cortes, H. Falter, J. Kunkel, and S. Narasimhamurthy. "E10 – Exascale IO." E10 Working Group Technical Report, 2014.

[Brun1997] Brun, R., and F. Rademakers. "ROOT—an object oriented data analysis framework." *Physics Research A* 389.1 (1997): 81–86.

[Bugra2008] Bugra G., H. Andrade, K.-L. Wu, P.S. Yu, and M. Doo. "SPADE: The System S Declarative Stream Processing Engine." Proc. 2008 ACM SIGMOD international conference on Management of data, Vancouver, Canada, June 9–12, 2008.

[Burtscher2009] Burtscher, M., and P. Ratanaworabhan. "FPC: A high-speed compressor for double-precision floating-point data." IEEE Transactions on Computers 58, no. 1 (2009): 18–31.

[BXI2017] Bull Atos Technologies. *Bull eXascale Interconnect for HPC systems*. https://atos.net/wp-content/uploads/2017/10/W-BXI-en1-web-1.pdf, accessed Aug. 1, 2018.

[Byna2008] Byna, S., Y. Chen, X.H. Sun, R. Thakur, and W. Gropp. "Parallel I/O prefetching using MPI file caching and I/O signatures." Proc. International Conference for High Performance Computing, Networking, Storage and Analysis, Austin, TX, pp. 1–12, 2008.

[Callahan2006] Callahan, S.P., J. Freire, E. Santos, C.E. Scheidegger, C.T. Silva, and H.T. Vo. "Vistrails: Visualization meets data management." Proc. 2006 ACM SIGMOD International Conference on Management of Data, pp. 745–747, 2006.

[Canon2016] Canon, S., and D. Jacobsen. "Shifter: Containers for HPC." Proc. Cray Users' Group, 2016.

[Carey2014] Carey, V., H. Abbasi, I. Rodero, and H. Kolla. "Sensitivity analysis for time dependent problems: optimal checkpoint-recompute HPC workflows." Proc. 9th Workshop on Workflows in Support of Large-Scale Science, pp. 20–30, 2014.

[Carlson2016] Carlson, J., et al. *Exascale Requirements Review for Nuclear Physics*, Advanced Scientific Computing Research and Nuclear Physics, DOE Office of Science, Gaithersburg, MD, June 15–17, 2016.

[Carns2000] Carns, P., W. Ligon, R.B. Ross, and R. Thakur. "PVFS: A parallel file system for Linux clusters." Proc. 4th annual Linux Showcase and Conference, pp. 391–430, 2000.

[Carns2011] Carns, P., K. Harms, W. Allcock, C. Bacon, S. Lang, R. Latham, and R. Ross. "Understanding and improving computational science storage access through continuous characterization." *ACM Transactions on Storage* 7.3 (2011): 8.

[Carns2013] Carns, P., et al. “Production I/O characterization on the Cray XE6.” Proc. Cray User Group, 2013.

[CERN2019a] CERN. “The Large Hadron Collider,” 2019. http://home.cern/topics/large-hadron-collider, accessed Apr. 24, 2019.

[CERN2019b] CERN. “High-Luminosity LHC,” 2019. https://home.cern/topics/high-luminosity-lhc, accessed Apr. 24, 2019.

[Chamberlain2007] Chamberlain, B.L., D. Callahan, and H.P. Zima. “Parallel programmability and the chapel language.” *International Journal of High Performance Computing Applications* 21.3 (2007): 291–312.

[Chameleon2015] Chameleon, 2015. https://www.chameleoncloud.org, accessed Apr. 24, 2019.

[Chang2008] Chang, F., et al. “Bigtable: A distributed storage system for structured data.” *ACM Transactions on Computer Systems* 26.2 (2008): 4.

[Chang2016] Chang, C.S., et al. *Fusion Energy Sciences Exascale Requirements Review,* Advanced Scientific Computing Research and Fusion Energy Sciences, DOE Office of Science, Gaithersburg, MD, Jan. 27–29, 2016.

[Charles2005] Charles, P., C. Grothoff, V. Saraswat, C. Donawa, A. Kielstra, K. Ebcioglu, C. Von Praun, and V. Sarkar. "X10: An object-oriented approach to non-uniform cluster computing." *ACM SIGPLAN Notices* 40.10 (2005): 519–538.

[Chou2011] Chou, J., et al. “Parallel index and query for large scale data analysis.” Proc. 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–30, Nov. 2011.

[Cidon2013] Cidon, A., S. Rumble, R. Stutsman, S. Katti, J. Ousterhout, and M. Rosenblum. “Copysets: Reducing the Frequency of Data Loss in Cloud Storage.” Proc. 2013 USENIX Annual Technical Conference, 2013.

[CloudLab2015] CloudLab, 2015. https://www.cloudlab.us, accessed Apr. 24, 2019.

[CMS n.d.] The Compact Muon Solenoid experiment at CERN. http://cms.cern/, accessed Apr. 24, 2019.

[Coburn2011] Coburn, J., A.M. Caulfield, A. Akel, L.M. Grupp, R.K. Gupta, R. Jhala, and S. Swanson. "NV-Heaps: making persistent objects fast and safe with next-generation, non-volatile memories." Proc. 16th international conference on Architectural support for programming languages and operating systems, 2011.

[CoDesign2016] Scientific Discovery through Advanced Computing (SciDAC) Co-Design Centers. http://science.energy.gov/ascr/research/scidac/co-design/, accessed Apr. 24, 2019.

[Colella2000] Colella, P., D.T. Graves, T.J. Ligocki, D.F. Martin, D. Modiano, D.B. Serafini, and B. Van Straalen. "Chombo software package for AMR applications-design document." 2000.

[Condit2009] Condit, J., E.B. Nightingale, C. Frost, E. Ipek, B. Lee, D. Burger, and D. Coetzee. "Better I/O through byte-addressable, persistent memory." Proc. ACM SIGOPS 22nd symposium on Operating systems principles. New York, NY, 133–146. 2009.

[Coral2018] Coral Collaboration, Coral-2 Benchmarks, 2018. https://asc.llnl.gov/coral-2-benchmarks/, accessed Apr. 24, 2019.

[Cray2018] Cray, Inc. Cray View for ClusterStor, 2018. https://www.cray.com/products/storage/clusterstor/view, accessed Apr. 24, 2019.

[Cray DataWarp n.d.] Cray DataWarp, n.d. https://pubs.cray.com/content/S-2558/CLE%206.0.UP06/xctm-series-datawarptm-user-guide, accessed Apr. 24, 2019.

[Czajkowski2001] Daynes, L., and G. Czajkowski. "High-performance, space-efficient, automated object locking." In: Data Engineering, 2001. Proceedings. 17th International Conference on, pp. 163–172. IEEE, 2001.

[Dai2014] Dai, D., R.B. Ross, P. Carns, D. Kimpe, and Y. Chen. "Using Property Graphs for Rich Metadata Management in HPC Systems." Proc. 9th Parallel Data Storage Workshop, vol. 11, 2014.

[Daly2006] Daly, J.T. "A higher order estimate of the optimum checkpoint interval for restart dumps." *Future Generation Computer Systems* 22.3 (2006): 303–312.

[DataLib n.d.] "DataLib: Data libraries and services enabling exascale science," n.d. https://www.exascaleproject.org/project/datalib-data-libraries-services-enabling-exascale-science/, accessed Apr. 24, 2019.

[Davidson2008] Davidson, S.B., and J. Freire. "Provenance and scientific workflows: challenges and opportunities." Proc. 2008 ACM SIGMOD International Conference on Management of Data, pp. 1345–1350, 2008.

[Dawson1983] Dawson, J.M. "Particle simulation of plasmas." *Reviews of Modern Physics* 55.2 (1983): 403. doi: 10.1103/RevModPhys.55.403.

[DAX2018] Direct Access for Files. https://www.kernel.org/doc/Documentation/filesystems/dax.txt, accessed Aug. 1, 2018.

[Dayal2008] Dayal, S. "Characterizing HEC storage systems at rest." Carnegie Mellon University Parallel Data Lab Technical Report CMU-PDL-08-109, 2008.

[Dayal2014] Dayal, J., D. Bratcher, G. Eisenhauer, K. Schwan, M. Wolf, X. Zhang, H. Abbasi, S. Klasky, and N. Podhorszki. "Flexpath: Type-Based Publish/Subscribe System-based publish/subscribe system for Large-Scale Science Analytics." Proc. 14th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, Chicago, IL, 2014, pp. 246–255., 2014.

[DDN2018] Data Direct Networks. "DDN Insight Data Sheet," 2018. https://www.ddn.com/products/storage-monitoring-ddn-insight, accessed Apr. 24, 2019.

[DDN IME n.d.] DDN IME, n.d. https://www.ddn.com/products/ime-flash-native-data-cache/, accessed Apr. 24, 2019.

[Dean2008] Dean, J., and S. Ghemawat. "MapReduce: Simplified data processing on large clusters." *Communications of the ACM*, 51.1 (2008): 107–113.

[Declerck2014] Declerck, T.M. "Using Robinhood to Purge Data from Lustre File Systems." Proc. Cray Users' Group, 2014.

[Deelman2002] Deelman, E., J. Blythe, Y. Gil, and C. Kesselman. "Pegasus: Planning for Execution in Grids." 2002-20. GriPhyN Technical Report, 2002.

[Deelman2015] Deelman, E., et al. "The Future of Scientific Workflows." Proc. DOE NGNS/CS Scientific Workflows Workshop, April 20–21, 2015. http://science.energy.gov/~/media/ascr/pdf/programdocuments/docs/workflows_final_report.pdf, accessed Apr. 24, 2019.

[Deelman2018] Deelman, E., et al. "The future of scientific workflows." *International Journal of High Performance Computing Applications* 32.1 (2018): 159–175.

[DeFavereau2014] De Favereau, J., et al. "DELPHES 3: a modular framework for fast simulation of a generic collider experiment." *Journal of High Energy Physics* 2014.2 (2014): 57.

[Degremont2013] Degremont, A., and T. Leibovici, 2013. http://cdn.opensfs.org/wp-content/uploads/2013/04/lug13-robinhood.pdf, accessed Apr. 24, 2019.

[Delgado2016] Delgado, R. "Why Your Data Scientist Isn't Being More Inventive," *Dataconomy*, March 15, 2016. http://dataconomy.com/2016/03/why-your-datascientist-isnt-being-more-inventive/, accessed Aug. 31, 2018.

[DeRoure2008] De Roure, D., C. Goble, and R. Stevens. "The design and realisation of the myExperiment virtual research environment for social sharing of workflows." *Future Generation Computer Systems* 25 (2009): 561–567. doi:10.1016/j.future.2008.06.010, accessed Apr. 24, 2019.

[DES2018] DES Data Management. "Dark Energy Survey Data Management (DESDM) at NCSA." https://des.ncsa.illinois.edu, accessed Aug. 31, 2018.

[Devarakonda2010] Devarakonda, R., and G. Palanisamy. "Advancements in scientific data searching, sharing and retrieval," 2010. https://arxiv.org/pdf/1101.1252.pdf, accessed Apr. 24, 2019.

[Di2016] Di, S., and F. Cappello. "Fast error-bounded lossy HPC data compression with SZ." Proc. 2016 IEEE International Parallel and Distributed Processing Symposium, pp. 730–739, 2016.

[Dickson2017] Dickson, J., S.A. Wright, S. Maheswaran, J.A. Herdman, D. Harris, M.C. Miller, and S.A. Jarvis. "Enabling portable I/O analysis of commercially sensitive HPC applications through workload replication." Proc. Cray User Group 2017, pp. 1–14.

[DiskSim n.d.] disksim. http://www.pdl.cmu.edu/DiskSim/, accessed Apr. 24, 2019.

[Docan2012] Docan, C., M. Parashar, and S. Klasky. "DataSpaces: An interaction and coordination framework for coupled simulation workflows." *Cluster Computing* 15.2 (2012): 163–181.

[DOE2018] U.S. Department of Energy. DOE Policy for Digital Research Data Management, 2018. https://www.energy.gov/datamanagement/doe-policy-digital-research-data-management, accessed Aug. 31, 2018.

[Dong2013] Dong, B., S. Byna, and J. Wu. "SDS: A Framework for Scientific Data Services." Proc. 8th Parallel Data Storage Workshop, 2013.

[Dong2016] Dong, B., S. Byna, K. Wu, Prabhat, H. Johansen, J.N. Johnson, and N. Keen. "Data Elevator: Low-Contention Data Movement in Hierarchical Storage System."

Proc. 2016 IEEE 23rd International Conference on High Performance Computing, pp. 152–161, 2016.

[Dorier2012] Dorier, M., G. Antoniu, F. Cappello, M. Snir, and L. Orf. "Damaris: How to efficiently leverage multicore parallelism to achieve scalable, jitter-free I/O." Proc. IEEE International Conference on Cluster Computing, pp. 155–163, 2012.

[Dorier2014] Dorier, M., S. Ibrahim, G. Antoniu, and R. Ross. "Omnisc'IO: A Grammar-Based Approach to Spatial and Temporal I/O Patterns Prediction." Proc. International Conference for High Performance Computing, Networking, Storage and Analysis, New Orleans, LA, pp. 623–634, 2014.

[Dorigo2005] Dorigo, A., P. Elmer, F. Furano, and A. Hanushevsky. "XROOTD—A highly scalable architecture for data access," *WSEAS Trans. Comput*. 4.4 (2005): 348.

[Draper1999] Draper, J.M., D.E. Culler, K. Yelick, E. Brooks, and K. Warren. *Introduction to UPC and language specification*. Center for Computing Sciences, Institute for Defense Analyses, 1999.

[Duro2014] Duro, F.R., J. Garcia Blas, F. Isaila, J.M. Wozniak, J. Carretero, and R. Ross. "Exploiting data locality in Swift/T workflows using Hercules." Proc. Network for Sustainable Ultrascale Computing Workshop, 2014.

[Ekanayake2008] Ekanayake, J., S. Pallickara, and G. Fox. "Mapreduce for data intensive scientific analyses." Proc. IEEE 4th International Conference on eScience, pp. 277–284, 2008.

[Ellexus2018] Ellexus. Home page, 2018. https://www.ellexus.com/, accessed Apr. 24, 2019.

[Elnozahy2002] Elnozahy, E.N., et al. "A survey of rollback-recovery protocols in message-passing systems." *ACM Computing Surveys* 34.3 (2002): 375–408.

[ESS-DIVE2018] ESS-DIVE. "Environmental Systems Science Data Infrastructure for a Virtual Ecosystem." http://ess-dive.lbl.gov, accessed Aug. 31, 2018.

[ExaHDF5 n.d.] "ExaHDF5: Delivering efficient parallel I/O on exascale computing systems," n.d. https://sdm.lbl.gov/exahdf5/, accessed Apr. 24, 2019.

[Fahey2010] Fahey, M., N. Jones, and B. Hadri. "The Automatic Library Tracking Database." Proc. Cray User Group, 2010.

[Feldman2013] Feldman, T., and G. Gibson. “Shingled Magnetic Recording: Areal Density Increase Requires New Data Management.” *USENIX*, 38(3), 2013.

[Felix2006] Felix, E.J., et al. “Active storage processing in a parallel file system.” Proc. 6th LCI International Conference on Linux Clusters: The HPC Revolution, 2006.

[Felix2011] Felix, E. “Environmental molecular sciences laboratory: Static survey of file system statistics,” 2011.

[Filguiera2014] Filguiera, R., I. Klampanos, A. Krause, M. David, A. Moreno, and M. Atkinson. “dispel4py: A Python framework for data-intensive scientific computing.” Proc. 2014 International Workshop on Data Intensive Scalable Computing Systems, pp. 9–16, 2014.

[FireWorks2013] FireWorks Workflow software, 2013. https://materialsproject.github.io/fireworks/, accessed April 24, 2019.

[Flynn2011] Flynn, M. “Flynn’s taxonomy.” In *Encyclopedia of Parallel Computing*, pp. 689–697, Springer, 2011.

[Folk1999] Folk, M., A. Cheng, and K. Yates. “HDF5: A file format and I/O library for high performance computing applications.” Proc. Supercomputing, vol. 99, pp. 5–33, 1999.

[Fox2015] Fox, G.C., S. Jha, and L. Ramakrishnan, eds., “STREAM2016: Streaming Requirements, Experience, Applications, and Middleware Workshop,” 2015.

[Fox2016] Fox, G., et al. *STREAM2016: Streaming Requirements, Experience, Applications and Middleware Workshop Final Report*. Office of Advanced Scientific Computing Research, DOE Office of Science, Tysons, VA, 2016.

[Freitas2008] Freitas, R.F., and W.W. Wilcke. “Storage-class memory: The next storage system technology." *IBM Journal of Research and Development* 52.4.5 (July 2008): 439–447.

[Frings2007] Frings, W., and M. Riedel. “LLview: User-level monitoring in computational grids and e-Science infrastructures.” Proc. German e-Science Conference, 2007.

[Gainaru2011] Gainaru, A., F. Cappello, S. Trausan-Matu, and B. Kramer. “Event log mining tool for large scale HPC systems.” *Euro-Par 2011 Parallel Processing*, pp. 52–64, Springer, 2011.

[Gainaru2015] Gainaru, A., G. Aupy, A. Benoit, F. Cappello, Y. Robert, and M. Snir. "Scheduling the I/O of HPC Applications Under Congestion," Proc. 2015 IEEE International Parallel and Distributed Processing Symposium, Hyderabad, India, pp. 1013–1022, 2015.

[Gal05] Gal, E., and S. Toledo. "Algorithms and data structures for flash memories." *ACM Computing Surveys* 37(2), 2005.

[Gamell2013] Gamell, M., I. Rodero, M. Parashar, and S. Poole. "Exploring energy and performance behaviors of data-intensive scientific workflows on systems with deep memory hierarchies." Proc. 20th Annual International Conference on High Performance Computing, Hyderabad, India, Dec. 2013.

[GenZ2018] GenZ Consortium. https://genzconsortium.org/, accessed Aug. 1, 2018.

[Geni2006] GENI, NSF. *Global environment for network innovations*, 2006. http://www.geni.net, accessed Apr. 24, 2019.

[Goodell2012] Goodell, D., S.J. Kim, R. Latham, M. Kandemir, and R. Ross. "An evolutionary path to object storage access." Proc. High Performance Computing, Networking, Storage and Analysis, pp. 36–41, 2012.

[Goodell2015] Goodell, D., P. Grun, S. Hefty, H. Pritchard, B. Russell, J. Squyres, and S. Sur. "A Brief Introduction to Openfabrics' Interfaces libfabrics." Proc. 23rd IEEE Symposium on High Performance Interconnects, 2015.

[Greenberg2015] Greenberg, H., J. Bent, and G. Grider. "MDHIM: A Parallel Key/Value Framework for HPC." *HotStorage*, 2015.

[Groves2016] Groves, T, R.E. Grant, S. Hemmer, S. Hammond, M. Levenhagen, and D.C. Arnold. "(SAI) Stalled, Active and Idle: Characterizing Power and Performance of Large-Scale Dragonfly Networks. Proc. 2016 IEEE International Conference on Cluster Computing, Taipei, pp. 50–59, 2016.

[Gu2011] Gu, J., D. Katramatos, X. Liu, A. Shoshani, A. Sim, D. Yu, S. Bradley, and S. McKee. "StorNet: Co-Scheduling of End-to-End Bandwidth Reservation on Storage and Network Systems for High-Performance Data Transfers." Proc. IEEE INFOCOM 2011 High-Speed Networks Workshop, 2011.

[Gupta2010] Gupta, C., and M. Govindaraju. "Framework for Efficient Indexing and Searching of Scientific Metadata." Proc. 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Melbourne, pp. 553–556, 2010. doi: 10.1109/CCGRID.2010.120.

[Gupta2014] Gupta, P., A. Wildani, D. Rosenthal, E.L. Miller, I. Adams, C. Strong, and A. Hospodor. "An Economic Perspective of Disk vs. Flash Media in Archival Storage." Proc. 22th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2014.

[Gupta2018] Gupta, M., et al., "Reliability-Aware Data Placement for Heterogeneous Memory Architecture." Proc. 2018 IEEE International Symposium on High Performance Computer Architecture, Vienna, pp. 583–595, 2018.

[Habib2015] Habib, S., et al. *ASCR/HEP Exascale Requirements Review Report*. DOE Office of Science, Bethesda, MD, June 10–12, 2015.

[Hack2017] Hack, J., et al. *Crosscut Report: Exascale Requirements Reviews*. DOE Office of Science, Tysons Corner, VA, March 2017.

[Hanushevsky2002] Hanushevsky, A.B. "Peer-to-peer computing for secure high performance data copying." Technical Report SLAC-PUB-9173, TRN US0206057, 2002.

[Hargrove2006] Hargrove, P.H., and J.C. Duell. "Berkeley Lab Checkpoint/Restart (BLCR) for Linux clusters." *Journal of Physics: Conference Series* 46.1, 2006.

[HDF2018] The HDF Group. "Hierarchical Data Format, version 5, 1997–2015." http://www.hdfgroup.org/HDF5/, accessed Aug. 1, 2018.

[HDR2018] Mellanox Technologies. "Introducing 200G HDR InfiniBand Solutions," 2018. http://www.mellanox.com/related-docs/whitepapers/WP_Introducing_200G_HDR_InfiniBand_Solutions.pdf, accessed Apr. 24, 2019.

[Henderson2004] Henderson, A., J. Ahrens, and C. Law. *The ParaView Guide*. Clifton Park, NY: Kitware, 2004.

[Henseler2016] Henseler, D., B. Landsteiner, D. Petesch, C. Wright, and N.J. Wright, (2016). Architecture and Design of Cray DataWarp. In Proceedings of the 2016 Cray User Group, London, 2016.

[Herbein2016] Herbein, S., D.H. Ahn, D. Lipari, T.R.W. Scogland, M. Stearman, M. Grondona, J. Garlick, B. Springmeyer, and M. Taufer. "Scalable I/O-Aware Job Scheduling for Burst Buffer Enabled HPC Clusters." Proc. 25th ACM International Symposium on High-Performance Parallel and Distributed Computing, New York, NY, pp. 69–80, 2016.

[Hills2015] Hills, D.J., R.R. Downs, R. Duerr, J.C. Goldstein, M.A. Parsons, and H.K. Ramapriyan. "The Importance of Data Set Provenance for Science." *Earth and Space Science News*, Dec. 4. 2015. doi: 10.1029/2015EO040557. https://eos.org/opinions/the-importance-of-data-set-provenance-for-science, accessed Aug. 31, 2018.

[HIO n.d.] HIO, n.d. https://github.com/hpc/libhio, accessed Apr. 24, 2019.

[Hoefler2009] Hoefler, T., A. Lumsdaine, and J. Dongarra. "Towards efficient MapReduce using MPI." *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pp. 240–249. Springer Berlin: Heidelberg, 2009.

[Horrocks2004] Horrocks, I., P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. "SWRL: A semantic web rule language combining OWL and RuleML." W3C Member submission 21, no. 79, 2004.

[Hyperion n.d.] Hyperion, n.d. https://hyperionproject.llnl.gov/index.php, accessed Apr. 24, 2019.

[IBM CAST n.d.] IBM CAST, n.d. https://github.com/IBM/CAST, accessed Apr. 24, 2019.

[Indiana2014] University of Indiana. *Komadu Provenance Collection Framework User Guide*, April 2014.

[Inman2014] Inman, J., G. Grider, and H.B. Chen. "Cost of Tape versus Disk for Archival Storage." Proc. 2014 IEEE 7th International Conference on Cloud Computing, Anchorage, AK, pp. 208–215, 2014.

[Inman2017] Inman, J., W. Vining, G. Ransom, and G. Grider. "MarFS, a Near-POSIX Interface to Cloud Objects." *USENIX Magazine* 42.1 (Spring 2017).

[I/ODoctors2018] I/O Doctors, 2018. http://iodoctors.com/, accessed Apr. 24, 2019.

[Isaila2011] Isaila, F., et al. "Design and evaluation of multiple-level data staging for blue gene systems." *IEEE Transactions on Parallel and Distributed Systems* 22.6 (2011): 946–959.

[Jenkins2012] Jenkins, J., et al. "Analytics-driven lossless data compression for rapid in-situ indexing, storing, and querying." *Database and Expert Systems Applications*, pp. 16–30. Springer Berlin: Heidelberg, 2012.

[Jenkins2017] Jenkins, J., G. Shipman, J. Mohd-Yusof, K. Barros, P. Carns, and R. Ross. A case study in computational caching microservices for HPC. Proc. IEEE International Workshop on Emerging Parallel and Distributed Runtime Systems and Middleware, June 2017.

[Jian2016] Jian, X., V. Sridharan, and R. Kumar. "Parity Helix: Efficient protection for single-dimensional faults in multi-dimensional memory systems." Proc. 2016 IEEE International Symposium on High Performance Computer Architecture, Barcelona, pp. 555–567, 2016.

[Jin2013] Jin, T., F. Zhang, Q. Sun, H. Bui, M. Parashar, H. Yu, S. Klasky, N. Podhorszki, and H. Abbasi. "Using cross-layer adaptations for dynamic data management in large scale coupled scientific workflows." Proc. ACM/IEEE International Conference for High Performance Computing, Networking Storage and Analysis, Denver, Colo., Nov. 2013.

[Jin2015] Jin, Z., et al. "Exploring data staging across deep memory hierarchies for coupled data intensive simulation workflows." Proc. 29th IEEE International Parallel & Distributed Processing Symposium, Hyderabad, India, May 2015.

[Jin2017] Jin, X., X. Li, H. Zhang, R. Soulé, J. Lee, N. Foster, C. Kim, and I. Stoica. "NetCache: Balancing Key-Value Stores with Fast In-Network Caching." Proc. 26th ACM Symposium on Operating Systems Principles, 2017.

[Johnson2014] Johnson, C., et al. "From research to practice: Experiences engineering a production metadata database for a scale out file system." Proc. 12th USENIX Conference on File and Storage Technologies, 2014.

[Jones2017] Jones, T., et al. "UNITY: Unified Memory and File Space." Proc. 7th International Workshop on Runtime and Operating Systems for Supercomputers ROSS 2017, New York, NY, Article 6, 2017.

[Jung2010] Jung, J., et al. "FRASH: Exploiting storage class memory in hybrid file system for hierarchical storage." Trans. Storage 6, 1, Article 3, April, 2010.

[Kadekodi2015] Kadekodi, S., S. Pimpale, and G.A. Gibson. "Caveat-Scriptor: Write Anywhere Shingled Disks." Proc. 7th USENIX Workshop on Hot Topics in Storage and File Systems, 2015.

[Kannan2011a] Kannan, S., et al. "Using active NVRAM for I/O staging." Proc. Petascale data analytics: challenges and opportunities, 2011.

[Kannan2011b] Kannan, S., et al. "Using active NVRAM for cloud I/O." Proc. 6th Open Cirrus Summit, 2011.

[Kannan2013] Kannan, S., et al. "Optimizing Checkpoints Using NVM as Virtual Memory." Proc. 27th International Symposium on Parallel and Distributed Processing, 2013.

[Kettimuthu2012] Kettimuthu, R., G. Vardoyan, G. Agrawal, and P. Sadayappan. "Modeling and optimizing large-scale wide-area data transfers." Proc. 14th IEEE/ACM Symposium on Cluster, Cloud, and Grid Computing, 2014.

[Kim2003] Kim, M., J. Hufferd, M. Chadalapaka, U. Elzur, H. Shah, and P. Thaler. "iSCSI Extensions for RDMA Specification (Version 1.0)," 2003.

[Kim2008] Kim, J., et al. "Technology-driven, highly-scalable dragonfly topology." *ACM SIGARCH Computer Architecture News* 36.3. (2008).

[Kim2015] Kim, Y., and R. Gunasekaran. "Understanding I/O workload characteristics of a peta-scale storage system." *Journal of Supercomputing* 71.3 (2015): 761–780.

[Kimpe2007] Kimpe, D., R. Ross, S. Vandewalle, and S. Poedts. "Transparent log-based data storage in MPI-IO applications." *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pp. 233–241. Springer Berlin: Heidelberg, 2007.

[Klasky2011] Klasky, S., et al. "In situ data processing for extreme-scale computing." Proc. Scientific Discovery through Advanced Computing Program, 2011.

[Knüpfer2011] Knüpfer, A., et al. "Score-P: A joint performance measurement run-time infrastructure for Periscope, Scalasca, TAU, and Vampir." *Tools for High Performance Computing 2011*, pp. 79–91. Springer, Berlin: Heidelberg, 2012.

[Komatitsch2015] Komatitsch, D.; J.-P. Vilotte, J. Tromp, M. Afanasiev, E. Bozdag, J. Charles, M. Chen, D. Goddeke, V. Hjorleifsdottir, J. Labarta, N. Le Goff, P. Le Loher, Q. Liu, A. Maggi, R. Martin, D. McRitchie, P. Messmer, D. Michea, T. Nissen-Meyer, D. Peter, M. Rietmann, S. de Andrade, B. Savage, B. Schuberth, A. Siemenski, L. Strand, C. Tape, Z. Xie, and H. Zhu (2015), SPECFEM3D GLOBE v7.0.0 [software], Computational Infrastructure for Geodynamics, doi: NoDOI: https://geodynamics.org/cig/software/specfem3d_globe/, accessed Apr. 24, 2019.

[Koziol2014] Koziol, Q., ed. *High performance parallel I/O*. CRC Press, 2014.

[Kryza2015] Kryza, B., and J. Kitowski. "File-Less Approach to Large Scale Data Management." Proc. European Conference on Parallel Processing, Aug. 2015. doi: 10.1007/978-3-319-27308-2_3. https://www.researchgate.net/publication/300123924_File-Less_Approach_to_Large_Scale_Data_Management, accessed August 31, 2018.

[Ku2006] Ku, S., et al. "Gyrokinetic particle simulation of neoclassical transport in the pedestal/scrape-off region of a tokamak plasma." *Journal of Physics: Conference Series* 46 (2006): 87.

[Kunkel2013] Kunkel, J.M. "Simulating parallel programs on application and system level." *Comput. Sci. Res. Dev.* 28 (2013): 167. https://doi.org/10.1007/s00450-012-0208-2, accessed Apr. 24, 2019.

[Kunkel2017] Kunkel, J., G.F. Lofstead, and J. Bent. *The Virtual Institute for I/O and the IO-500,* 2017.

[Kunkel2018] Kunkel, J., and G. Markomanolis. "Understanding Metadata Latency with MDWorkbench." Proc. Workshop on Performance and Scalability of Storage Systems, 2018.

[Kurtzer2017] Kurtzer, G. "Containers in HPC: Singularity." Proc. Intel HPC Developer's Conference, 2017.

[LaFon2012] LaFon, J., S. Misra, and J. Bringhurst. "On distributed file tree walk of parallel file systems." Proc. International Conference on High Performance Computing, Networking, Storage and Analysis, 2012.

[Lakshminarasimhan2011] Lakshminarasimhan, S., 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." *Euro-Par 2011 Parallel Processing*, pp. 366-379. Springer Berlin: Heidelberg, 2011.

[Lamport2001] Lamport, L. "Paxos made simple." *ACM Sigact News* 32.4 (2001): 18–25.

[LANL2015] Los Alamos National Laboratory. "Trinity: Advanced Technology System." https://www.lanl.gov/projects/trinity/, accessed Aug. 2, 2018.

[LANL n.d.] Los Alamos National Laboratory. "LANL systems – operational and fault data," n.d. http://institute.lanl.gov/data/, accessed Apr. 24, 2019.

[Leung2007] Leung, A.W., E.L. Miller, and S. Jones. "Scalable security for petascale parallel file systems." Proc. ACM/IEEE conference on Supercomputing, p. 16, 2007.

[Leung2009] Leung, A.W., I.F. Adams, and E.L. Miller. *Megellan: A searchable metadata architecture for large-scale file systems*. UCSC-SSRC-09-07. Technical Report. University of California, Santa Cruz, CA, Nov. 2009.

[Li2013] Li, Y., N.S. Dhotre, Y. Ohara, T.M. Kroeger, E.L. Miller, and D. Long. "Horus: Fine-grained encryption-based security for large-scale storage." Proc. FAST, pp. 147–160, 2013.

[Li2017] Li, B., Z. Ruan, W. Xiao, Y. Lu, Y. Xiong, A. Putnam, E. Chen, and L. Zhang. "KV-Direct: High-Performance In-Memory Key-Value Store with Programmable NIC." Proc. 26th ACM Symposium on Operating Systems Principles, 2017.

[Liao2007] Liao, W.-K., et al. "An implementation and evaluation of client-side file caching for MPI-IO." Proc. Parallel and Distributed Processing Symposium, 2007.

[Lindstrom2006] Lindstrom, P., and M. Isenburg. "Fast and efficient compression of floating-point data." *IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006): 1245–1250.

[Lister2003] Lister, J.B., et al. "The ITER project and its data handling requirements." Proc. 9th ICALEPCS Conference, Gyeongju, Korea, 2003.

[Liu2012a] Liu, N., et al. "On the role of burst buffers in leadership-class storage systems." Proc. IEEE 28th Symposium on Mass Storage Systems and Technologies, 2012.

[Liu2012b] Liu, Z., et al. "PCM-based durable write cache for fast disk I/O." Proc. IEEE 20th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems, 2012.

[Liu2014] Liu, Q., J. Logan, Y. Tian, H. Abbasi, N. Podhorszki, J.Y. Choi, S. Klasky, et al. "Hello ADIOS: the challenges and lessons of developing leadership class I/O frameworks." Concurrency and Computation: Practice and Experience 26, no. 7, pp. 1453–1473, 2014.

[Liu2015] Liu, N., X. Yang, X.H. Sun, J. Jenkins, and R. Ross. "YARNsim: Simulating Hadoop YARN." Proc. 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Shenzhen, pp. 637–646, 2015.

[Liu2017] Liu, Z., P. Balaprakash, R. Kettimuthu, and I. Foster. "Explaining Wide Area Data Transfer Performance." Proc. IEEE International Symposium on High-performance Parallel and Distributed Computing, Washington, DC, June 2017.

[LLNL2017] Lawrence Livermore National Laboratory. "Sierra – High Performance Computing,"2017. https://hpc.llnl.gov/hardware/platforms/sierra, accessed Aug. 1, 2018.

[Lockwood2017a] Lockwood, G.K., W. Yoo, S. Byna, N.J. Wright, S. Snyder, K. Harms, Z. Nault, and P. Carns. "UMAMI: a recipe for generating meaningful metrics through holistic I/O performance analysis." Proc. 2nd Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems, New York, NY, USA, pp. 55–60, 2017.

[Lockwood2017b] Lockwood, G.K., et al. *Storage 2020: A Vision for the Future of HPC Storage*, LBNL-2001072, Berkeley, CA, 2017.

[Lockwood2018] Lockwood, G.K., S. Snyder, G. Brown, K. Harms, P. Carns, and N.J. Wright. "TOKIO on ClusterStor: Connecting Standard Tools to Enable Holistic I/O Performance Analysis." Proc. Cray User Group. Stockholm, May 2018.

[Lofstead2008] Lofstead, J.F., et al. "Flexible IO and integration for scientific codes through the adaptable IO system (ADIOS)." Proc. Challenges of Large Applications in Distributed Environments, 2008.

[Lofstead2016] Lofstead, J., I. Jimenez, C. Maltzahn, Q. Koziol, J. Bent, and E. Barton. "DAOS and friends: a proposal for an exascale storage system." Proc. International Conference for High Performance Computing, Networking, Storage and Analysis, Piscataway, NJ, Article 50, 2016.

[Logan2012] Logan, J., et al. "Understanding I/O Performance Using I/O Skeletal Applications." In: C. Kaklamanis et al. (eds.) *Euro-Par 2012 Parallel Processing*. Lecture Notes in Computer Science, vol. 7484. Springer, Berlin: Heidelberg, 2012.

[LSST2009] LSST Science Collaborations. *LSST Science Book, Version 2.0*, 2009. arXiv:0912.0201.

[Lu2016] Lu, W. "Crossbar RRAM: A New Era of Storage Innovation for a Content Rich World." Proc. Flash Memory Summit, 2016.

[Luettgau2017] Luettgau, J., and J. Kunkel. "Simulation of Hierarchical Storage Systems for TCO and QoS." In J. Kunkel et al. (eds.) ISC High Performance 2017. Lecture Notes in Computer Science, vol. 10524, 2017.

[Luo2017] Luo, X., et al. "ScalaIOExtrap: Elastic I/O Tracing and Extrapolation." Proc. 2017 IEEE International Parallel and Distributed Processing Symposium, Orlando, FL, pp. 585–594, 2017.

[Lustre2010] Lustre-HSM. "Lustre HSM Project," 2010. http://wiki.lustre.org/images/4/4d/Lustre_hsm_seminar_lug10.pdf, accessed Apr. 24, 2019.

[Luu2015] Luu, H., M. Winslett, W. Gropp, K. Harms, P. Carns, R. Ross, Y. Yao, S. Byna, and Prabhat. "A Multi-platform Study of I/O Behavior on Petascale Supercomputers." Proc. 24th International ACM Symposium on High Performance Distributed Computing, June 2015.

[Ma2003] Ma, X., and A.L. Narasimha Reddy. "MVSS: An active storage architecture." *IEEE Transactions on Parallel and Distributed System* 14.10 (2003): 993–1005.

[MacArthur2017] MacArthur, P., Q. Liu, R.D. Russell, F. Mizero, M. Veeraraghavan, and J.M. Dennis. "An Integrated Tutorial on Verbs, Infiniband, and MPI." *IEEE Communications Surveys & Tutorials*, Vol. 19, No. 4, 2017.

[Madireddy2018] Madireddy, S., et al. "Machine Learning Based Parallel I/O Predictive Modeling: A Case Study on Lustre File Systems." In R. Yokota et al. (eds.) *ISC High Performance 2018*. Lecture Notes in Computer Science, vol. 10876, 2018.

[Mandal2007] Mandal, N., E. Deelman, G. Mehta, M.-H. Su, and K. Vahi. "Integrating existing scientific workflow systems: The Kepler/Pegasus example." Proc. 2nd Workshop on Workflows in Support of Large-scale Science, pp. 21–28, 2007.

[Manzanares2016] Manzanares, A., N. Watkins, C. Guyot, D. LeMoal, C. Maltzahn, and Z. Bandic. "ZEA, A Data Management Approach for SMR." Proc. 8th USENIX Workshop on Hot Topics in Storage and File Systems, 2016.

[May2001] May, J.M. "Parallel I/O for high performance computing." Morgan Kaufmann, 2001.

[MDTest n.d.] MDTest, n.d. https://www.nersc.gov/users/computational-systems/cori/nersc-8-procurement/trinity-nersc-8-rfp/nersc-8-trinity-benchmarks/mdtest/, accessed Apr. 24, 2019.

[Mesnier2003] Mesnier, M., G.R. Ganger, and E. Riedel. "Object-based Storage." *IEEE Communications Magazine* 41.8 (2003): 84–90.

[Mikdadi2017] Mikdadi, D. "The Power of Provenance," 2017. https://datascience.nih.gov/PowerofProvenance, July 6; accessed Aug. 31, 2018.

[Milenkovic2018] Milenkovic, O., R. Gabrys, H.M. Kiah, and S.M.H. Tabatabaei Yazdi. "Exabytes in a Test Tube." *IEEE Spectrum* 55.5 (May 2018): 40–45.

[Miller2001] Miller, E., S.A. Brandt, and D. Long. "HerMES: High-performance reliable MRAM-enabled storage." Proc. 8th IEEE Workshop on Hot Topics in Operating Systems, May 2001.

[Miller2010] Miller, R., J. Hill, G. Raghul, G.M. Shipman, and D. Maxwell. "Monitoring tools for large scale systems." Proc. Cray Users Group, 2010.

[Miller2015] Miller, M.C. *Design & Implementation of MACSio*. LLNL-TR-670388. Lawrence Livermore National Laboratory, Livermore, Calif., 2015.

[Moody2010] Moody, A., G. Bronevetsky, K. Mohror, and B.R. De Supinski. "Design, modeling, and evaluation of a scalable multi-level checkpointing system." Proc. International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–11, 2010.

[Moore2011] Moore, M., et al. "OrangeFS: Advancing PVFS." *FAST Poster Session*, 2011.

[MPI2019] "The Message Passing Interface Standard." Versions 2.0, 2.1, 2.2, 3.0, 3.1; 2019. https://www.mpi-forum.org/docs/, accessed Apr. 24, 2019.

[Mubarak2017a] Mubarak, M., et al. "Quantifying I/O and Communication Traffic Interference on Dragonfly Networks equipped with Burst Buffers." Proc. 19th IEEE Cluster Conference, September 2017.

[Mubarak2017b] Mubarak, M., C.D. Carothers, R.B. Ross, and P. Carns. "Enabling parallel simulation in large-scale HPC Network System co-design." Proc. IEEE Transactions on Parallel and Distributed Systems, 2017.

[Muelder2011] Muelder, C., et al. "Visual analysis of I/O system behavior for high-end computing." Proc. 3rd Workshop on Large-Scale System and Application Performance, pp. 19–26, 2011.

[Muniswamy-Reddy2006] Muniswamy-Reddy, K.-K., D.A. Holland, U. Braun, and M. Seltzer. "Provenance-aware storage systems." Proc. USENIX '06 Annual Technical Conference, 2006.

[Nagle2004] Nagle, D., D. Serenyi, and A. Matthews. "The Panasas ActiveScale storage cluster-delivering scalable high bandwidth storage." Proc. ACM/IEEE Conference on Supercomputing, 2004.

[Najm2009] Najm, H.N. "Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics." *Annual Review of Fluid Mechanics* 41 (2009): 35–52.

[NERSC2015a] NERSC. "Cori," 2015. http://www.nersc.gov/users/computational-systems/cori/, accessed Aug. 1, 2018.

[NERSC2015b] NERSC. "MyNERSC gives users easier access to data, jobs, wait times," 2015. http://www.nersc.gov/news-publications/nersc-news/nersc-center-news/2015/mynersc-gives-users-easier-access-to-data-projects-wait-times/, accessed Apr. 24, 2019.

[Ni2012] Ni, X., E. Meneses, and L.V. Kalé. "Hiding checkpoint overhead in HPC applications with a semi-blocking algorithm." Proc. IEEE International Conference on Cluster Computing, 2012.

[NMC2019] New Mexico Consortium. "Parallel Reconfigurable Observational Environment," 2019. http://www.nmc-probe.org/, accessed Apr. 24, 2019.

[Northwestern2014] Northwestern University. "Damsel—A Data Model Storage Library for Exascale Science," 2014. http://cucis.ece.northwestern.edu/projects/DAMSEL, accessed Apr. 24, 2019.

[Nugent 2015] Nugent, P., Y. Cao, and M. Kasliwal. "The Palomar transient factory." February 2015.

[Núñez2010] Núñez, A., J. Fernández, J. Garcia, F. Garcia, and J. Carretero. "New Techniques for Simulating High Performance MPI Applications on Large Storage Networks." *The Journal of Supercomputing* 51 (2010): 40–57.

[NVME2018a] NVM Express. *Non-Volatile Memory Express Base Specification*, 2018. https://nvmexpress.org/wp-content/uploads/NVM-Express-1_3c-2018.05.24-Ratified.pdf, accessed Aug. 1, 2018.

[NVME2018b] NVM Express. *Non-volatile Memory Express over Fabrics Revision 1.0a*, 2018. https://nvmexpress.org/wp-content/uploads/NVMe-over-Fabrics-1_0a-2018.07.23-Ratified.pdf, accessed April 24, 2019.

[Oldfield2007] Oldfield, R.A., L. Ward, A.B. Maccabe, and P. Widener. "Scalable security for MPP storage systems." Proc. International Conference on Security and Management: Special Session on Security in Supercomputing Clusters, Las Vegas, Nev., July 2007.

[ONF2018a] Open Networking Foundation. "OpenFlow Protocol Specifications," 2018. https://www.opennetworking.org/, accessed Aug. 1, 2018.

[ONF2018b] Open Networking Foundation. "Open Networking Operating System," 2018. https://www.opennetworking.org/onos/, accessed Aug. 1, 2018.

[Ongaro2014] Ongaro, D., and J. Ousterhout. "In Search of an Understandable Consensus Algorithm." Proc. USENIX Annual Technical Conference, 2014.

[OPA2018] Intel, Inc. "Intel Omni-Path Fabric—the Next HPC Fabric," 2018. https://www.intel.com/content/www/us/en/high-performance-computing-fabrics/omni-path-fabric-demo.html?_ga=2.112446476.906196838.1534179142-948016287.1534179142, accessed Aug. 1, 2018.

[Optane2018] Intel. Intel® Optane™ Technology, 2018. https://www.intel.com/content/www/us/en/architecture-and-technology/intel-optane-technology.html, accessed Aug. 1, 2018.

[Organick2018] Organick, L., et al. "Random Access in Large-scale DNA Data Storage." *Nature Biotechnology* 36 (2018).

[ORNL2015] Oak Ridge National Laboratory. "Titan," 2015. https://www.olcf.ornl.gov/titan/, accessed Apr. 24, 2019.

[ORNL2018] Oak Ridge National Laboratory. "Summit," 2018. https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/, accessed Aug. 1, 2018.

[Orphus2018] Orphus, C. "Berkeley Lab Researchers Use Machine Learning to Search Science Data," 2018. https://newscenter.lbl.gov/2018/06/19/berkeley-lab-researchers-use-machine-learning-to-search-science-data/, accessed Aug. 31, 2018.

[Ovsyannikov2016] Ovsyannikov, A., M. Romanus, B. Van Straalen, G.H. Weber, and D. Trebotich. "Scientific Workflows at DataWarp-Speed: Accelerated Data-Intensive Science using NERSC's Burst Buffer." Proc. 1st Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems, Nov. 2016.

[Palmer2011] Palmer, B., A. Koontz, K. Schuchardt, R. Heikes, and D. Randall. "Efficient data IO for a parallel global cloud resolving model." *Environmental Modelling & Software* 26.12 (2011): 1725–1735.

[Palmer2015] Palmer, J.T., et al. "Open XDMoD: A Tool for the Comprehensive Management of High-Performance Computing Resources." *Computing in Science & Engineering* 17.4 (2015): 52–62.

[Parallel I/O Tutorial n.d.] "Parallel I/O in practice," n.d. http://sc14.supercomputing.org/program/tutorials.html, accessed Apr. 24, 2019.

[Parker-Wood2010] Parker-Wood, A., C. Strong, E.L. Miller, and D.D.E. Long. "Security aware partitioning for efficient file system search." Proc. IEEE 26th Symposium on Mass Storage Systems and Technologies, pp. 1–14, 2010.

[Parker-Wood2013] Parker-Wood, A., D.D.E. Long, B.A. Madden, I.F. Adams, M. McThrow, and A. Wildani. "Examining extended and scientific metadata for scalable index designs." Proc. 6th International Systems and Storage Conference, New York, NY, Article 4, 2013. doi: https://doi.org/10.1145/2485732.2485754, accessed Apr. 24, 2019.

[Parkinson 2016] Parkinson, D.Y., et al. "Real-time data-intensive computing." *AIP Conf. Proc.*, 1741 (2016): 1–6.

[Paschke2005] Paschke, A., M. Bichler, and J. Dietrich. ContractLog: An Approach to Rule Based Monitoring and Execution of Service Level Agreements, International Conference on Rules and Rule Markup Languages for the Semantic Web (RuleML 2005), Galway, Ireland, 2005.

[Patil2011] Patil, S., and G.A. Gibson. "Scale and concurrency of GIGA+: File system directories with millions of files." Proc. USENIX Conference on File and Storage Technologies, pp. 177–190, 2011.

[Patterson1989] Patterson, D.A., et al. "Introduction to redundant arrays of inexpensive disks (RAID)." Proc. IEEE COMPCON, vol. 89, 1989.

[PDSW n.d.] Parallel Data Storage Workshop, n.d. http://www.pdsw.org, accessed Apr. 24, 2019.

[PFTool2018] "PFTool: Parallel File Tool," 2018. https://github.com/pftool/pftool, accessed Aug. 1, 2018.

[Piernas2007] Piernas, J., J. Nieplocha, and E.J. Felix. "Evaluation of active storage strategies for the Lustre parallel file system." Proc. 2007 ACM/IEEE conference on Supercomputing. ACM, 2007.

[Plimpton2011] Plimpton, S.J., and K.D. Devine. "MapReduce in MPI for large-scale graph algorithms." *Parallel Computing* 37.9 (2011): 610–632.

[PMDK2018] Persistent Memory Development Kit, 2018. https://pmem.io/pmdk/, accessed Aug. 1, 2018.

[Poremba2015] Poremba, M., T. Zhang, and Y. Xie. "NVMain 2.0: A User-Friendly Memory Simulator to Model (Non-)Volatile Memory Systems." *IEEE Computer Architecture Letters* 14.2 (2015): 140–143.

[Priedhorsky2017] Priedhorsky, R., and T. Randles. "Charliecloud: unprivileged containers for user-defined software stacks in HPC." Proc. International Conference for High Performance Computing, Networking, Storage and Analysis, 2017.

[Prometheus2018] Prometheus. Home page, 2018. https://prometheus.io/, accessed Apr. 24, 2019.

[Qin2006] Qin, L., and D. Feng. "Active storage framework for object-based storage device." Proc. 20th International Conference on Advanced Information Networking and Applications, 2006.

[Qin2009] Qin, X., H. Jiang, A. Manzanares, X.-J Ruan, and S. Yin. "A dynamic load balancing for I/O-intensive applications on clusters." *ACM Transactions on Storage*, 5 (2009).

[QMCPack2015] QMCPack. Home page, 2015. http://www.qmcpack.org/, accessed Apr. 24, 2019.

[QuantX2018] QuantX. "3D XPoint® Technology," 2018. https://www.micron.com/products/advanced-solutions/3d-xpoint-technology, accessed Aug. 1, 2018.

[Rajachandrasekar2013] Rajachandrasekar, R., et al. "A 1 PB/s file system to checkpoint three million MPI tasks." Proc. High-Performance Parallel and Distributed Computing, 2013.

[Rajasekar2010] Rajasekar, A., R. Moore, C.-Y. Hou, C.A. Lee, R. Marciano, A. de Torcy, M. Wan, et al. "iRODS primer: integrated rule-oriented data system." Synthesis Lectures on Information Concepts, Retrieval, and Services 2, no. 1, pp. 1–143, 2010.

[Rambus2018] Rambus. "Smart Data Acceleration Platform." https://www.rambus.com/emerging-solutions/smart-data-acceleration/, accessed Aug. 1, 2018.

[Reagana2003] Reagana, M.T., H.N. Najm, R.G. Ghanem, and O.M. Knio. "Uncertainty quantification in reacting-flow simulations through non-intrusive spectral projection." *Combustion and Flame* 132, 3 (2003): 545–555.

[Reed2004] Reed, D.A. (ed.) *Scalable Input/Output: Achieving System Balance*. MIT Press, 2004.

[Ren2014] Ren, K., Q. Zheng, S. Patil, and G. Gibson. "IndexFS: Scaling file system metadata performance with stateless caching and bulk insertion." Proc. International Conference for High Performance Computing, Networking, Storage and Analysis, Nov. 2014.

[Rew1990] Rew, R., and G. Davis. "NetCDF: An interface for scientific data access." *Computer Graphics and Applications* 10.4 (1990): 76–82.

[Riedel1997] Riedel, E., and G. Gibson. *Active disks-remote execution for network-attached storage*. CMU-CS-97-198. Carnegie-Mellon University, School of Computer Science, Pittsburgh, PA, 1997.

[Riedel2001] Riedel, E., C. Faloutsos, G.A. Gibson, and D. Nagle, "Active disks: Disks for Large-Scale Data Processing." Computer 34., 6 (June 2001): 68–74. DOI: https://doi.org/10.1109/2.928624, accessed April 24, 2019.

[Ripeanu2002] Ripeanu, M. "Peer-to-peer architecture case study: Gnutella network." Peer-to-Peer Computing, 2001. Proceedings. First International Conference on. IEEE, 2001.

[Rizzo1997] Rizzo, L. "Effective erasure codes for reliable computer communication protocols." *ACM SIGCOMM Computer Communication Review* 27.2 (1997): 24–36.

[Ross2001] Ross, R., D. Nurmi, A. Cheng, and M. Zingale. "A case study in application I/O on Linux clusters." Proc. ACM/IEEE Conference on Supercomputing, 2001.

[Rumble2014] Rumble, S.M., A. Kejriwal, and J. Ousterhout. "Log-structured Memory for DRAM-based Storage." Proc. 12th USENIX Conference on File and Storage Technologies, 2014.

[Sankaran2005] Sankaran, S., et al. "The LAM/MPI checkpoint/restart framework: System-initiated checkpointing." *International Journal of High Performance Computing Applications* 19.4 (2005): 479–493.

[Samsung2017a] Samsung. "Samsung Key Value SSD Enables High Performance." Press release, 2017.

[Samsung2017b] Samsung. "Ultra-Low Latency with Samsung Z-NAND SSD." Press release, 2017.

[Samsung2018] Samsung. "Samsung Debuts Semiconductor Innovations at Samsung Tech Day that Maximize Data Center Efficiencies and Enable AI, Enterprise and Emerging Technologies." Press release, 2018.

[Savoie2016] Savoie, L., D.K. Lowenthal, B.R. de Supinski, T. Islam, K. Mohror, B. Rountree, and M. Schulz. "I/O Aware Power Shifting." Proc. 2016 IEEE International Parallel and Distributed Processing Symposium, Chicago, IL, pp. 740–749, 2016.

[SC2016] DOE Office of Science. "Statement on Digital Data Management," 2016. http://science.energy.gov/funding-opportunities/digital-data-management/, accessed Apr. 24, 2019.

[Schissel2014] Schissel, D.P., G. Abla, S.M. Flanagan, M. Greenwald, X. Lee, A. Romosan, A. Shoshani, J. Stillerman, and J. Wright. "Automated Metadata, Provenance Cataloging and Navigable Interfaces: Ensuring the Usefulness of Extreme-scale Data." *Fusion Engineering and Design*, Feb. 23, 2014.

[Schopf2002] Schopf, J.M. "A general architecture for scheduling on the grid." *Journal of Parallel and Distributed Computing*, April 2002.

[SciServer2018] SciServer. Home page. http://www.sciserver.org, accessed Aug. 31, 2018.

[Scott2006] Scott, S., et al. "The BlackWidow high-radix Clos network." *ACM SIGARCH Computer Architecture News* 34.2 (2006).

[SCSI2018] SCSI. "SCSI Standards Architecture." http://www.t10.org/scsi-3.htm accessed Aug. 1, 2018.

[Seamons1994] Seamons, K.E., and M. Winslett. "An efficient abstract interface for multidimensional array I/O." Proc. Supercomputing '94, pp. 650–659, 1994.

[Sears2009] Sears, M.P., B.W. Bader, and T. Kolda. "Parallel implementation of tensor decompositions for large data analysis." *SIAM AN09*, July 8, 2009.

[Settlemyer2011] Settlemyer, B.W., J.D. Dobson, S.W. Hodson, J.A. Kuehn, and S.W. Poole. "A technique for moving large data sets over high-performance long distance networks." Proc. of the 2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1 – 6, 2011.

[Settlemyer2012] Settlemyer, B.W., N.S.V. Rao, S.W. Poole, S.W. Hodson, S.E. Hicks, and P.E. Newman. "Experimental analysis of 10Gbps transfers over physical and emulated dedicated connections." Proc. International Conference on Computing, Networking, and Communications, 2012.

[Sevilla2015] Sevilla, M., N. Watkins, C. Maltzahn, I. Nassi, S. Brandt, S. Weil, G. Farnum, and S. Fineberg. "Mantle: a programmable metadata load balancer for the ceph file system." Proc. International Conference High Performance Computing, Networking, Storage and Analysis, 2015.

[Shamis2015] Shamis, P., et al. "UCX: An Open Source Framework for HPC Network APIs and Beyond." Proc. IEEE 23rd Annual Symposium on High-Performance Interconnects, 2015.

[Shan2008] Shan, H., K. Antypas, and J. Shalf. "Characterizing and predicting the I/O performance of HPC applications using a parameterized synthetic benchmark." Proc. ACM/IEEE conference on Supercomputing, 2008.

[Shende2006] Shende, S.S., and A.D. Malony. "The TAU parallel performance system." *The International Journal of High Performance Computing Applications* 20.2 (2006): 287–311.

[Sigelman2010] Sigelman, B.H., L.A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag. *Dapper, a Large-scale Distributed Systems Tracing Infrastructure*. Technical report, Google, Inc., 2010.

[Sjostrand2008] Sjöstrand, T., S. Mrenna, and P. Skands. "A brief introduction to PYTHIA 8.1." *Computer Physics Communications* 178.11 (2008): 852–867.

[Skinner2005] Skinner, D. "Performance Monitoring of Parallel Scientific Applications." LBNL/PUB-5503, 2005. https://www.osti.gov/biblio/881368, accessed Apr. 24, 2019.

[SNIA2017] Storage Network Industry Association. *SNIA NVM Programming Model (NPM) Version 1.2*, 2017. https://www.snia.org/sites/default/files/technical work/final/NVMProgrammingModel v1.2.pdf, accessed Apr. 24, 2019.

[SNIA2018] Storage Network Industry Association. *Swordfish Scalable Storage Management API Specification*, 2018. https://www.snia.org/sites/default/files/technical work/Swordfish/Swordfish v1.0.6 specification.pdf, accessed May 25, 2018.

[Snyder2015] Snyder, S., et al. "Techniques for Modeling Large-scale HPC I/O Workloads." Proc. International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems, 2015.

[Snyder2016] Snyder, S., P. Carns, K. Harms, R. Ross, G.K. Lockwood, and N.J. Wright. "Modular HPC I/O Characterization with Darshan." Proc. 5th Workshop on Extreme-scale Programming Tools, 2016.

[Son2010] Son, S.W., S. Lang, P. Carns, R. Ross, R. Thakur, B. Ozisikyilmaz, P. Kumar, W.-K. Liao, and A. Choudhary. "Enabling active storage on parallel I/O software stacks." Proc. IEEE 26th Symposium on Mass Storage Systems and Technologies, pp. 1-12, 2010.

[Son2017] Son, S.W., et al. "Reducing I/O variability using dynamic I/O path characterization in petascale storage systems." *Journal of Supercomputing* (2017) 73: 2069 (2017). https://doi.org/10.1007/s11227-016-1904-7, accessed Apr. 24, 2019.

[Soumagne2013] Soumagne, J., D. Kimpe, J. Zounmevo, M. Chaarawi, Q. Koziol, A. Afsahi, and R. Ross. "Mercury: Enabling Remote Procedure Call for High-Performance Computing." Proc. IEEE International Conference on Cluster Computing, Sept. 2013.

[Spelman2018] Spelman, L. "Reimagining the Data Center Memory and Storage Hierarchy." *Intel*, 2018. https://newsroom.intel.com/editorials/re-architecting-data-center-memory-storage-hierarchy/, accessed May 2018.

[Subramoni2008] Subramoni, H., G. Marsh, S. Narravula, L. Ping and D.K. Panda. "Design and evaluation of benchmarks for financial applications using advanced message queuing protocol (AMQP) over InfiniBand." Proc. Workshop on High Performance Computational Finance, pp. 1, 8, Nov. 16, 2008.

[Tang2018] Houjun, T., et al. "Toward Scalable and Asynchronous Object-centric Data Management for HPC." Proc. 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2018.

[Tantisiriroj2011] Tantisiriroj, W., S.W. Son, S. Patil, S.J. Lang, G. Gibson, and R.B. Ross. "On the duality of data-intensive file system design: reconciling HDFS and PVFS." Proc. International Conference for High Performance Computing, Networking, Storage and Analysis, p. 67, 2011.

[Thakur1999] Thakur, R., W. Gropp, and E. Lusk. "Data sieving and collective I/O in ROMIO." Proc. Seventh Symposium on the Frontiers of Massively Parallel Computation, 1999.

[Thapaliya2016] Thapaliya, S., P. Bangalore, J. Lofstead, K. Mohror, and A. Moody. "Managing I/O Interference in a Shared Burst Buffer System." Proc. 45th International Conference on Parallel Processing, Philadelphia, PA, pp. 416–425, 2016.

[UChicago2018] UChicago Argonne LLC and Lawrence Livermore National Security, LLC. "VeloC," 2018. http://veloc.readthedocs.io/en/latest/, accessed Apr. 24, 2019.

[UPCIO2004] El-Ghazawi, T. et al. "UPC-IO: A Parallel I/O API for UPC" 2004. http://upc.lbl.gov/publications/UPC-IOv1.0.pdf, accessed Apr. 24, 2019.

[Vairavanathan2012] Vairavanathan, E., S. Al-Kiswany, L. Beltrão Costa, Z. Zhang, D.S. Katz, M. Wilde, and M. Ripeanu. "A workflow-aware storage system: An opportunity study." Proc. 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2012.

[VanEssen2012] Van Essen, B., R. Pearce, S. Ames, and M. Gokhale. "On the Role of NVRAM in Data-intensive Architectures: An Evaluation." Proc. IEEE 26th International Parallel and Distributed Processing Symposium, Shanghai, pp. 703–714, 2012.

[Vazhkudai2017] Vazhkudai, S.S., R. Miller, D. Tiwari, C. Zimmer, F. Wang, S. Oral, R. Gunasekaran, and D. Steinert. "GUIDE: a scalable information directory service to collect, federate, and analyze logs for operational insights into a leadership HPC facility." Proc. International Conference for High Performance Computing, Networking, Storage and Analysis, New York, N.Y., Article 45, 2017.

[Venkataraman2011] Venkataraman, S., N. Tolia, P. Ranganathan, and R.H. Campbell. "Consistent and durable data Structures for non-volatile byte-addressable memory." Proc. USENIX Conference on File and Storage Technologies, pp. 61–75. 2011.

[Vetter2016] Vetter, J., et al. *Exascale Requirements Review for Advanced Scientific Computing Research*, DOE Office of Science, Rockville, Md., Sept. 27–29, 2016.

[Vijayakumar2009] Vijayakumar, K., F. Mueller, X. Ma, and P.C. Roth. "Scalable I/O tracing and analysis." Proc. 4th Annual Workshop on Petascale Data Storage, New York, N.Y., pp. 26–31, 2009.

[Vilayannur2008] Vilayannur, M., Lang, S., Ross, R., Klundt, R., and Ward, L. "Extending the POSIX I/O Interface: A Parallel File System Perspective." Argonne National Laboratory Technical Report ANL/MCS-TM-302, October 2008.

[Vincent2018] Vincent, L., and G. Goret. "Self-Optimized Strategy for IO Accelerator Parametrization." Proc. Workshop on Performance and Scalability of Storage Systems, 2018.

[Vishwanath2011] Vishwanath, V., M. Hereld, V. Morozov, and M.E. Papka. "Topology-aware data movement and staging for I/O acceleration on Blue Gene/P supercomputing systems." Proc. 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, p. 19, 2011.

[Wachs2009] Wachs, M., and G.R. Ganger. "Co-scheduling of Disk Head Time in Cluster-based Storage." Proc. 28th International Symposium on Reliable Distributed Systems, Sept. 27–30, 2009.

[Wallace1992] Wallace, G.K. "The JPEG still picture compression standard." *IEEE Transactions on Consumer Electronics* 38, no. 1 (1992): xviii-xxxiv.

[Wang2002] Wang, A.-I., et al. "Conquest: Better performance through a disk/persistent-RAM hybrid files system." Proc. USENIX Annual Technical Conference, 2002.

[Wang2016] Wang, F., V.G. Vergara Larrea, D. Leverman, and S. Oral. "FCP: A Fast and Scalable Data Copy Tool for High Performance Parallel File Systems." Proc. Cray Users' Group, 2016.

[Watkins2015] Watkins, N., Z. Jia, G. Shipman, C. Maltzahn, A. Aiken, and P. McCormick, "Automatic and transparent I/O optimization with storage integrated application runtime support." Proc. 10th Parallel Data Storage Workshop, November 2015.

[Watson1995] Watson, R.W., and R.A. Coyne. "The Parallel I/O Architecture of the High-Performance Storage System (HPSS)." Proc. 14th IEEE Symposium on Mass Storage Systems, 1995: Storage - At the Forefront of Information Infrastructures, p. 27, Sept. 11–14, 1995.

[Weil2004] Weil, S.A., K.T. Pollack, S.A. Brandt, and E.L. Miller. "Dynamic metadata management for petabyte-scale file systems." Proc. SC'04, Pittsburgh, PA, Nov. 2004.

[Weil2006] Weil, S.A., S.A. Brandt, E.L. Miller, D.D.E. Long, and C. Maltzahn. "Ceph: A scalable, high-performance distributed file system." Proc. Symposium on Operating Systems Design and Implementation, University of California, Santa Cruz, CA, pp. 307–320. 2006.

[Weil2007] Weil, S.A. "Ceph: Reliable, scalable, and high-performance distributed storage." Ph.D. thesis, University of California at Santa Cruz, CA, Dec. 2007.

[Welch2008] Welch, B., M. Unangst, Z. Abbasi, G.A. Gibson, B. Mueller, J. Small, J. Zelenka, and B. Zhou. "Scalable performance of the Panasas parallel file system." Proc. USENIX Conference on File and Storage Technologies, 8 (2008): 1–17.

[Whitlock2011] Whitlock, B., J.M. Favre, and J.S. Meredith. "Parallel In Situ Coupling of a Simulation with a Fully Featured Visualization System." Proc. Eurographics Symposium on Parallel Graphics and Visualization, pp. 101–109, 2011.

[Wieczorek2009] Wieczorek, M., A. Hoheisel, and R. Prodan. "Towards a general model of the multi-criteria workflow scheduling on the grid." *Future Generation Computing Systems* 25.3 (March 2009): 237–256.

[Williams1997] Williams, D.N. "The PCMDI software system: Status and future plans." Program for Climate Model Diagnosis and Intercomparison, University of California, Lawrence Livermore National Laboratory, 1997.

[Windus2015] Windus, T., et al. *Basic Energy Sciences Exascale Requirements Review*, DOE Office of Science, Rockville, Md., Nov. 3–5, 2015.

[Wolstencroft2013] Wolstencroft, K., et al. “The Taverna workflow suite: Designing and executing workflows of Web Services on the desktop, web or in the cloud.” *Nucleic Acids Research* 41(2013): W557–W561.

[Wozniak2014] Wozniak, J.M., M. Wilde, and I.T. Foster. “Language features for scalable distributed-memory dataflow computing.” Proc. Data-flow Execution Models for Extreme-scale Computing, 2014.

[Wu2009] Wu, K., et al. “FastBit: Interactively searching massive data.” *Journal of Physics: Conference Series* 180.1 (2009): 012053.

[Wu2011] Wu, X., and A.L.N. Reddy. “SCMFS: A file system for Storage Class Memory.” Proc. International Conference for High Performance Computing, Networking, Storage and Analysis, 2011.

[Xie2017] Xie, B., Y. Huang, J.S. Chase, J.Y. Choi, S. Klasky, J.F. Lofstead, and S. Oral. “Predicting Output Performance of a Petascale Supercomputer.” Proc. High Performance Distributed Computing (2017).

[Xu2016] Xu, J., and S. Swanson. “NOVA: A Log-structured File System for Hybrid Volatile/Non-volatile Main Memories.” Proc. 14th USENIX Conference on File and Storage Technologies, 2016.

[Xu2016] Xu, T., K. Sato, and S. Matsuoka. “CloudBB: Scalable I/O Accelerator for Shared Cloud Storage.” IEEE 22nd International Conference on Parallel and Distributed Systems, Wuhan, China, pp. 509–518, 2016.

[Yildiz2016] Yildiz, O., M. Dorier, S. Ibrahim, R. Ross, and G. Antoniu. “On the Root Causes of Cross-Application I/O Interference in HPC Storage Systems.” Proc. IEEE International Parallel and Distributed Processing Symposium, Chicago, IL, pp. 750–759, 2016.

[Zhang2010] Zhang, Y., et al. “End-to-end data integrity for file systems: A ZFS Case Study.” Proc. 8th USENIX Conference on File and Storage Technologies, 2010.

[Zhang2016] Zhang, H., D.G. Andersen, A. Pavlo, M. Kaminsky, L. Ma, and R. Shen. “Reducing the Storage Overhead of Main-Memory OLTP Databases with Hybrid Indexes.” Proc. 2016 International Conference on Management of Data (SIGMOD ‘16), 2016.

[Zhang2017] Zhang, F., et al. “In-memory staging and data-centric task placement for coupled scientific simulation workflows.” *Concurrency and Computation: Practice and Experience* 29.12 (2017): e4147.

[Zhao2004] Zhao, B.Y., et al. “Tapestry: A resilient global-scale overlay for service deployment.” *IEEE Journal on Selected Areas in Communications* 22.1 (2004): 41–53.

[Zhao2007] Zhao, Y., M. Hategan, B. Clifford, I. Foster, G. Von Laszewski, V. Nefedova, I. Raicu, T. Stef-Praun, and M. Wilde. “Swift: Fast, reliable, loosely coupled parallel computation.” Proc. IEEE Congress on Services, pp. 199–206, 2007.

[Zheng2014] Zheng, Q., K. Ren, and G. Gibson. “BatchFS: Scaling the file system control plane with client-funded metadata servers.” Proc. 9th Parallel Data Storage Workshop, pp. 1–6, 2014.

## 10 Acknowledgments

The organizers wish to thank a number of people in advance for their assistance with the workshop. The organizers wish to thank Lucy Nowell for sponsoring the meeting and for facilitating and soliciting contributions from key science domains. Additionally, the organizers wish to thank Oak Ridge Institute for Science and Education and Deneise Terry for managing the registration and logistics of the workshop series, and Mary Fitzpatrick, Andrea Manning, Michele Nelson, Vicki Skonicki, and Carolyn Steele for their editing and layout assistance.