# Storage Systems and Input/Output 2018
# Pre-Workshop Document

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


**Meeting Organizers**
Robert Ross (ANL) (lead organizer)
Lee Ward (SNL) (co-lead)
Gary Grider (LANL)
Scott Klasky (ORNL)
Glenn Lockwood (LBL)
Kathryn Mohror (LLNL)
Bradley Settlemyer (LANL)

**Pre-Workshop Document Contributors**
Philip Carns (ANL)
Quincey Koziol (LBL)
Matthew Wolf (ORNL)

# 1 Table of Contents

## 2 Executive Summary

Storage systems are a foundational component of computational, experimental, and observational science today. The success of 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. In September 2018, a diverse group of domain and computer scientists from the Office of Science, the National Nuclear Security Administration, industry, and academia will assemble in Gaithersburg, Maryland, to update and review the SSIO requirements identified during the comprehensive SSIO 2014 workshop for simulation-driven activities associated with DOE's science, energy, and national security missions, to assess the state of the art in key storage system and I/O areas, to consider the implications of data and learning applications on SSIO requirements, and to identify potential research directions in SSIO for extreme scale DOE science.

This activity is planned as a single-track, two day, workshop. The workshop is organized around six topical areas identified by the organizers as areas where gaps exist in capabilities required for DOE science to proceed at pace. Each day begins with an initial set of talks and a panel that provide context and background for the attendees, followed by rigorous moderated discussion of three of the topics:

- Hardware/software architectures
- Metadata, name spaces, and provenance
- Supporting science workflows
- Deepening storage hierarchies
- Understanding storage systems and I/O
- Streaming data

The purpose of this pre-workshop document is to provide important background and context for our attendees. Building from and augmenting the 2015 report, this document captures a summary of mission drivers and an assessment of the state of the art and challenges in each of the six above mentioned areas. Further, this document will serve as the starting point for the final report capturing the outcomes of the 2018 workshop, and as such is a work in progress.

# 3 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 I/O (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, GPFS) and archival storage (e.g., 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, accessed, and their integrity to be maintained. Improvements to SSIO software systems improves the productivity of DOE scientists by facilitating the discovery of and access to their data and reduces the cost to operate storage systems by improving 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 I/O. New nonvolatile memory technologies (e.g., 3D XPoint®[1] and Intel

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

Optane®[2] DIMM), drive technologies (e.g., shingled disks, 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 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.

The purpose of this pre-workshop document is to provide important background and context for our workshop attendees. Building from and augmenting the 2015 report, this document captures a summary of mission drivers and an assessment of the state of the art and challenges in key SSIO research areas. Further, this document will serve as the starting point for the final report capturing the outcomes of the 2018 workshop, and as such is a work in progress.

The report is organized as follows. Section 4 summarizes the DOE mission requirements for SSIO. Section 5 discusses in depth the state of the art and challenges to be addressed to advance SSIO technologies in support of DOE mission. Section 6 discusses some supporting activities that were previously identified as helpful in enabling successful R&D in the SSIO area.

[2] Optane is a registered trademark of Intel Corporation

# 4 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 and 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 SciDAC co-design centers [CoDesign] 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 Office of Science 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 HEP [Habib2015], BES [Windus2015], FES [Chang2016], BER [Arkin2016], 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 tradeoffs 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.

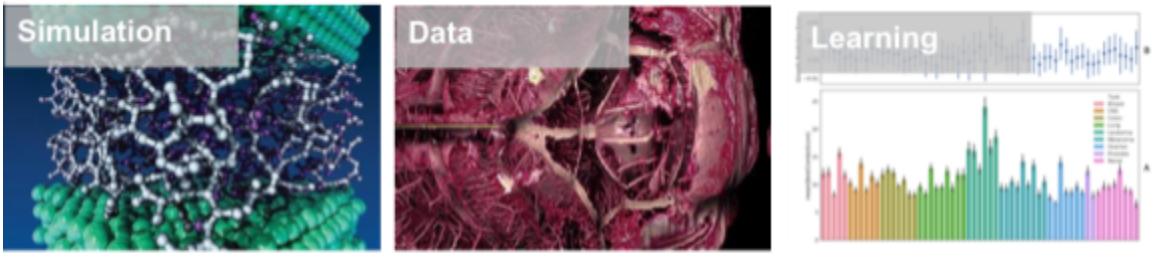



Figure 4.1. The Office of Science computing portfolio has diversified in recent years, beyond the traditional simulation focus, to include significant numbers of activities employing big data analysis and machine and deep learning techniques to accomplish mission goals. Image credit B. Helland (ASCR).

## 4.1 Overview

Historically the Office of Science computing facilities have served science teams employing simulation codes to better understand phenomena of interest to the 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. In fact, *all* 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 V's: 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 ANL, NERSC, and ORNL in 2015), have launched a series of simulations that need to write out 100 PB of data in order to capture all of the turbulence data for runs on the Titan system [Titan2015], the prior OLCF platform. Because of the lack of storage and because of the time needed to write this amount of data, the simulation will be able to write only about 10 PB. This may 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].

**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, since they are generating over 2 TB of data from their simulation every 10 seconds. 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; approach depends both on effective management of high-velocity data and also 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 most of the 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 below). 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 is 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 described above), many choices must be made to understand which data products will have later value. Among other characteristics, the value of the data is also impacted by how much it can be reduced for fast post-processing. 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 to access those data. 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.

## 4.2 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 experimental and observational data analysis (EOD) 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 S.5.3.

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. Some of these tasks produce a large amount of data to be stored persistently. Additionally, 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 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 an uncertainty quantification (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

multi-dimensional 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. Supporting long-running service tasks will require supercomputer centers to change their 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 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 data from one component often at the same time as other components, which can increase the I/O variability.

### 4.2.1 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 an uncertainty quantification (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 disks. 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. Since the intermediate time steps are not written back to global storage, this approach reduces the I/O time. Since 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, since 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.

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

The CMS experiment [CMS n.d.] at the Large Hadron Collider (LHC) [LHC n.d.] 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 600Hz. 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, but simulations and analysis of this 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 multi-phase workflow reconstructs "tracks" (i.e., paths of particles) from detector data, 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 GBytes 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 best use of hardware investments: this is further facilitated by merging.

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

### 4.2.3 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, DEEP n.d.], 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, finding the approach compares favorably to traditional appoaches.

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 [Cori2015] for training. While just one example of the application of learning algorithms toward 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.

## 4.3 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.

### 4.3.1 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. Additionally, as many simulations start 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, while 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.

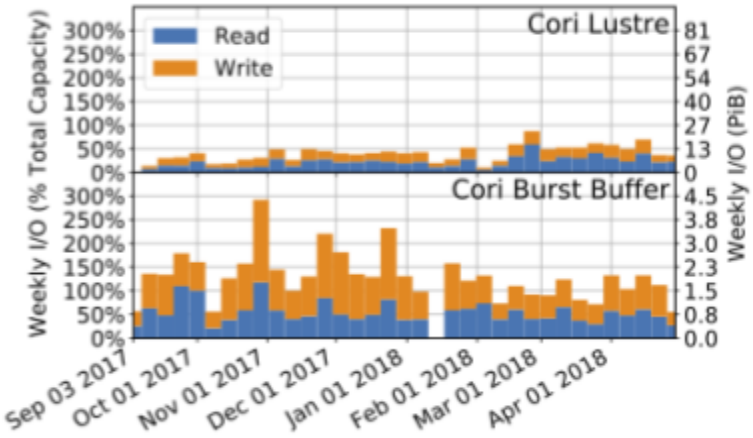



Figure 4.2 While traditionally we have expected HPC storage systems to be dominated by checkpointing writes, modern systems see a more balanced workload. Data from Cori supercomputer at NERSC, both for the Lustre filesystem (top) and the DataWarp burst buffer (bottom) [Lockwood2018].

As parallel computers grow in size, there is considerable interest in moving away from writing checkpoints to a global, parallel file system. 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.

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 is discussed further in S.4.5-6.

### 4.3.2 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, global climate 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 consistent of many, modest-sized files. The topic of streaming data is further discussed in S.5.6.

### 4.3.3 Initial Impact of Nonvolatile Storage

Modern machines have nonvolatile memory (NVRAM) that can be used for storing checkpoint files and frequently-used input data (e.g., through systems like DataWarp [Cray DataWarp n.d.]). 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 the next section. Another common issue is that the analysis may require only a portion of the data instead of the whole data set—for example, only those data records in the region of influence mentioned in the previous use case. 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 S.5.4.

## 4.4 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 petabytes (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, 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].

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 analysis (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 sit*u analysis. Scientists are wary of including other code in their simulation for good reasons. The challenges generally are as follows:

1. Can the analysis routines run when/where there are idle resources? As simulations evolve on systems, we see many unused cycles (due to issues with OpenMP, etc.) that could be used by *in situ* analysis operations. There are funded projects that are working to address these issues, such as using task-based parallelism and dynamically moving tasks to locations where there are free cycles. In much the same way, there is an urgent need for analytics and visualization tasks to move to locations where application scientists allow them, i.e. where they are available for auxiliary services. This same challenge exists for future SSIO services.
2. Can new services be used on all hardware platforms available to an application community? This question applies not only to *in situ* analysis services but also to a variety of potential SSIO service designs. It requires that these services be able to be executed in a variety of ways: on-node – using different cores, on-node – using free cycles on the same cores as the simulation, off-node – on the same exascale resource, or off-machine on a nearby resource. Research is necessary in order to ensure this flexibility of service deployment. This should also allow the inclusion of UQ analytics to be used in this suite of services.
3. Can analysis tasks be shipped to another computer system or preserved for future execution? For analysis that needs to meet hard time constraints, it might be necessary to either ship some analysis tasks to another computer system or save the task for future execution. Enabling this capability has implications for the connections between programming models, workflow, and data abstractions and representations.
4. Can users ensure that the simulation *does not pause* as someone is visualizing or analyzing data at a previous time step? The ability to interact with actively executing simulation codes is becoming a more pressing issue on exascale machines.

Today several *in situ* visualization and analysis services are being used in applications. 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 [Vishwanath2011a] uses a similar methodology in order execute analysis pipelines. Catalyst allows users to embed analysis routines into their simulation, which then call VTK/Paraview [Henderson2004] code, which is similar to LibSim [Whitlock2011]. Each of these frameworks has tradeoffs, and more research is

necessary to understand how to best provide needed data services in support of exascale science.

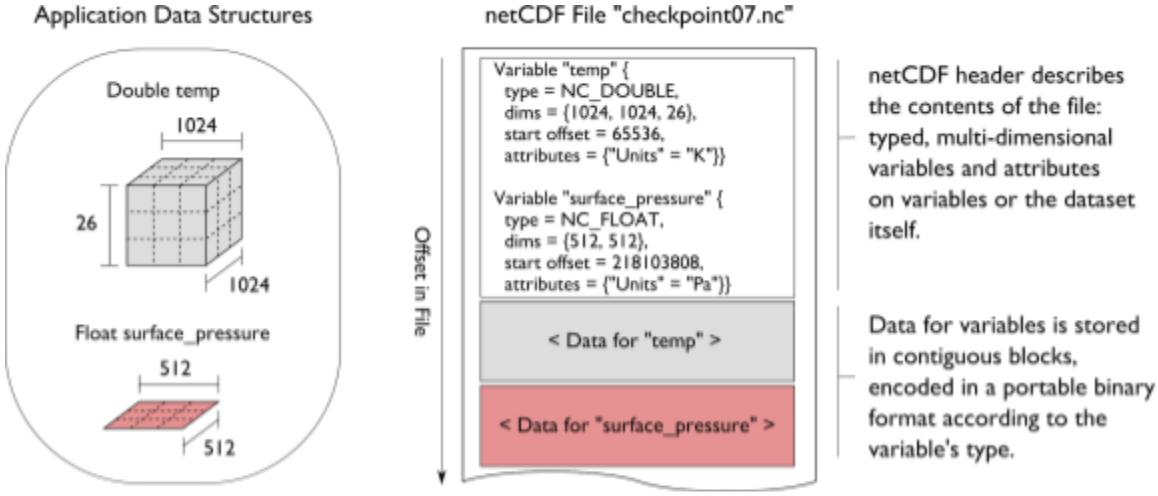



Figure 4.3: Self-describing file formats, such as the netCDF format shown here, capture not only array data but also structural information such as names, units, types, and dimensions.

## 4.5 Data Organization and Archiving

Many application programs running on the current generation of supercomputers still write data in custom formats. However, the majority of the data files being shared by large scientific projects are using popular file formats such as ADIOS BP [Lofstead2008], HDF5 [Folk1999], and netCDF [Rew1990]. 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 to our community that must be addressed for the 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?

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. Research 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. For these libraries to be adopted effectively in the upcoming high-performance computers, the following issues need to be addressed:

- **Performance.** The I/O system must be highly efficient in a wide variety of use cases: uniform meshes, semi-structured meshes, and unstructured meshes. 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 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. New research into file formats that can withstand failures is critical for future adoption. This is often the strategy taken by many Monte Carlo simulations such as QMCPACK and from some PIC simulations.
- **Compression.** Some forms of compression are already supported by the current generation of I/O libraries. In at least one of the applications, it is 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, since 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 MB/node must be compressed and written to the storage system every 10 seconds. Since the data at this scale will overflow the burst buffers on current systems, compression must be very fast in order to greatly reduce the I/O overhead, and it must be significant in cost savings (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 when arriving at the detection or resurrected from disk. Additionally, 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 on disk is often considered online because it can be accessed with common I/O libraries, whereas data in tertiary storage is considered offline because it has to go through an extensive data transfer process before it is usable by a data analysis program. Typically, online data is available in milliseconds, while offline data may require hours or weeks to become available. Such a gap is a tremendous barrier for user access to data in tertiary storage and, so, a challenge to analyzing data stored in this way.

A number of application scientists have expressed a desire to have a near-line storage system with latencies much less than 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 to the SSIO community is to understand how best to organize data across the many tiers of storage.

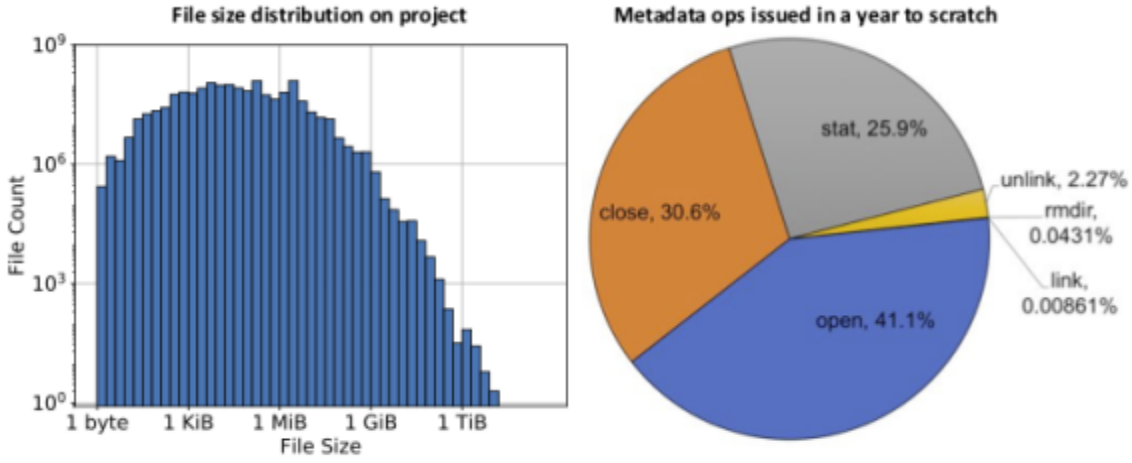



Figure 4.4. Distribution of file sizes and metadata operation counts on NERSC Edison file systems from June 2016 to June 2017. The presence of a massive number of modest-sized files drives an increased focus on metadata operations and performance [Lockwood2017b]

## 4.6 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 automatically capture metadata. 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 [SC n.d.]. 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 data set. 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. For example, scientists might want to capture the different types of algorithms used for analysis to help them understand accuracy vs. power tradeoffs, 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 want to keep information at the granularity of MPI processes.

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 way 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. Additionally, 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.

## 4.7 Summary

Mission scientists see the SSIO community as facing a number of exciting challenges, summarized here in terms of the five Vs presented earlier in this section:

1. Fast data access is essential to large data-intensive applications. High-level self-describing data formats are critical to allow 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, 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, value)
4. *In situ* data services/frameworks are needed that can co-locate tasks (on node, off-node, and on external resources) and can be specialized for specific applications. (Velocity, volume, variety, value)

# 5 Computer Science Challenges

A number of contributors to the overarching challenge of providing high-bandwidth, low-latency, storage access across multiple tiers are discussed. In toto, the sub-sections below are intended to capture the breadth of the problem, with each detailing the particulars and magnitude of the particular, classified, regions.

## 5.1 Hardware/Software Architectures

Architectural changes in Exascale and post-Exascale HPC platforms raise new challenges for effective SSIO hardware and software architectures. The generation, analysis, and management of multiple extreme-scale data sets 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 impact broad swaths of the simulation science community. Multiple years of research efforts [Bent2012, Liu2012a, Barton2014] led to the successful deployment of Flash media into HPC platforms at multiple DOE facilities [Cori2015, Trinity2015, Sierra2017, Summit2017]. 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. Cross-cutting research across these areas may be required for some projects. For example, emerging storage media types 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.

### 5.1.1 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 devices. 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, Abghayev2015], 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, Abghayev2017], flash read/write/erase asymmetries [Gal05], and distributed in-memory stores [Rumble2014, SDA2018] 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 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, Zhang2018] have been developed to exploit performance much greater than 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 in improving storage media interfaces developed 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 [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.

**Challenges**

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

- Challenges exist with incorporating byte-addressable media into distributed HPC storage systems as 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 medias 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 presents 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 re-visiting assumptions throughout the storage stack.

### 5.1.2 Networking Technologies

HPC networks continue to improve in performance dramatically 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. Additionally, 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 100-200Gbps [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], though 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 recently explored.

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

#### 5.1.2.1 Challenges

Future HPC systems will incorporate multiple levels of storage distributed across one or more networks with potentially complex topologies and thousands of storage endpoints. 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 endpoints and the switching hierarchy
- Leveraging direct access to storage devices efficiently and effectively within HPC platforms.

Many of these capabilities will be researched outside the context of SSIO, but 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.

### 5.1.3 Active Storage and 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 density. 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 has enjoyed over a decade of active research [Riedel1997, Amiri2000, Son2010]. The current state of the art includes the extension of active storage concepts to the device level (T10 OSD) [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 looked at 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].

**Challenges**

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 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).
- Exposing acceleration and active storage capabilities with flexible programming interfaces that integrate well with existing HPC I/O middleware.
- 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.

### 5.1.4 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. Temporary and intermediate data sets may leverage different protection schemes than 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 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 been done to explore 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 CAP theorem and peer-to-peer systems principles are also gaining momentum and adoption within the scientific HPC community.

New stacked DRAM technologies provide extremely high levels of memory bandwidth and have the opportunity to provide high performance buffer operations in HPC storage systems; but 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.

**Challenges**

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.
- Leveraging new memory technologies inside of storage systems without memory protection strategies like ChipKill.
- Providing appropriate levels of performance and resiliency for persistent data within the storage hierarchy.

### 5.1.5 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 interfaces for enabling

#### 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 yet 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 [cite Globus, BBCP, XDD] 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 systems 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.

#### Challenges

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:

- Toolsets for implementing policies and managing data are well-suited for the systems and problems for which they were defined, but 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, that are able to explore the system configuration space and measurably improve reliability, performance, and/or TCO for storage systems.

### 5.1.6 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. The issues of balancing authentication, secure access control, secure data transmission, and efficient performance have traditionally been performed by HPC facilities, but 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 is 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 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].

#### Challenges

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 and vendors.
- The dependence on node 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, since 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 solution 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).

## 5.2 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 (e.g., scripts that use UNIX `grep`, `find`, and `awk`) that are not designed for the scale or complexity anticipated for large scientific data sets. In general, managing large data sets on our existing systems requires a level of discipline and organization by the user 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 with extended features that 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 mid-2018) 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, name spaces and provenance challenges described below should support those goals, as well as the needs of the scientists producing the original data.

### 5.2.1 Metadata

Metadata, in this context, refers generally to the information about data as well as the tools and techniques in an SSIO storage system to support the storage and retrieval of such information. 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**

While most HPC file systems support some notion of extended attributes for files, [Lustre2002, 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 data sets more productively, and should be investigated further.

Scalable metadata management for HPC systems has been a known issue for more than a decade; and while many systems have some support for multiple metadata servers [Carns2000, Weil2006], the additional servers often are used for fail-over, not performance [Lustre2002]. Truly distributed metadata servers with strong consistency semantics, such as Ceph's 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.

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 in their infancy. Techniques such as these may enable DOE DMP goals to be achieved in an automated and scalable fashion, with minimal effort by scientists.

**Challenges**

Workshop attendees identified a number of nontraditional use cases for the metadata management system. 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).

If we expand what can be stored as metadata, how do we ensure that all metadata associated with a dataset remains with the data? Particular concerns were about metadata storage at the different storage tiers, storage and recovery of metadata from archive, and the transfer of data sets to different storage systems.

Hashing a namespace balances the load but does not account for locality. Fixed namespace partitioning accounts for locality but creates a load imbalance. How can we combine the two or conceive of better techniques? How do we scale distributed and multi-objective load-balancing algorithms across exascale-sized metadata services that maximize caching and wear- or power-leveling?

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 MPMD (multiple program multiple data) or composite workflows, there is a need to capture the complex dependencies within a single parallel job. Since a job on an exascale machine may have 1-billion-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 billion interrelated parallel tasks. The volume and velocity of the metadata associated with such a fine-grained metadata could present a serious challenge to manage.

### 5.2.2 Namespaces

The namespace is the 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 [cite Kryza2015]. Approaches that index metadata for data sets and present a scalable search mechanism to return results (see earlier science data servers / web-portals citations in section 5.2.2) are also promising.

**Challenges**

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. Eliminating these limitations with object storage systems or higher-level systems could be the fundamental breakthrough needed to scale namespaces to billion-way concurrency.

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

**State of the Art**

Most of today's scientific data sets 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 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.

**Challenges**

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 identified was 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.

## 5.3 Integrating With and Supporting Science Workflows

In the exascale era, we have seen a significant growth of applications which 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 since 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 the extreme-age of computing, computational science is coupled with experiments/observations to validate computational models and predict and control them. This intricate process of "validation" consists of a mixture of inter-related 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 below.

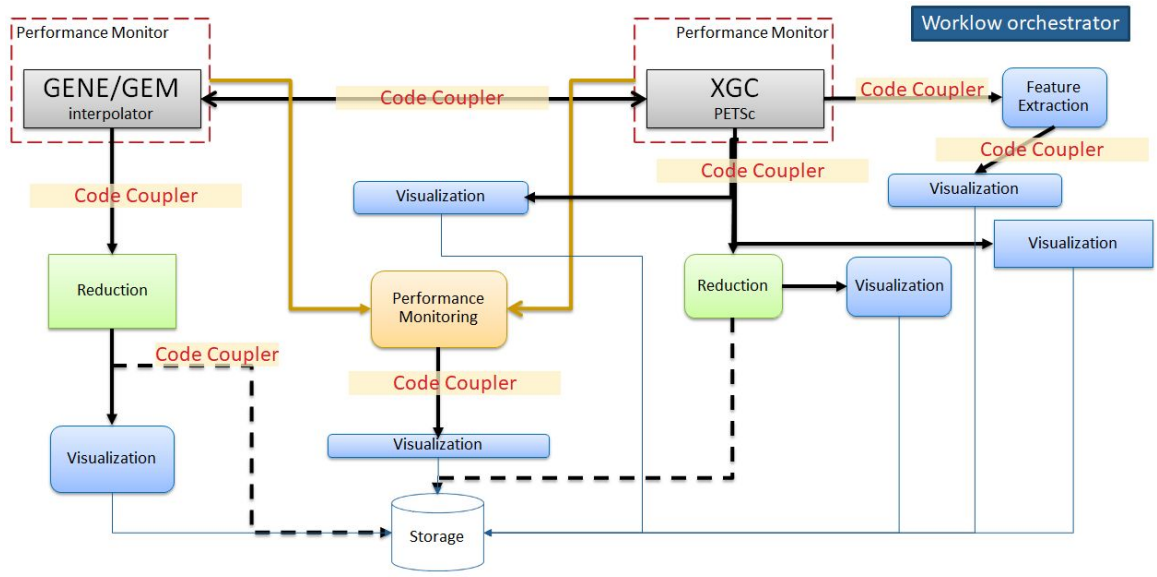
Research and development in SSIO is critical towards enabling scientists to 1) Express their online and offline workflows (**composition**) and allow the workflows to be captured and saved (**provenance**) with their scientific campaign (Programming Model Integration); 2) Integrate and execute sub-routines 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**.

### 5.3.1 DOE Extreme Scale Use cases

In this section, we capture three important DOE application use cases which are already generating a tremendous amount of data (PB's per scientific campaign) and

are in need for the SSIO community to research, develop and create new software to cope with their complex workflows. Many of these findings have been reported in the NSF stream workshop (cite), the future of scientific workflows workshop (cite), as well as within several of our communities

.




**5.3.1.1** The above figure illustrates a complex set of processes to couple two fusion codes on a LCF, along with reducing, analyzing, and visualizing the results.

The first application scenario is the online coupling of simulation codes and visualization codes in whole device modeling. In this scenario, a code which models core transport in tokamak cores is coupled with a code which models wall physics and edge plasma-wall interactions. Coupling of core and edge simulations is necessary for more accurate whole device modeling. As is seen in the figure above, this scenario illustrates the new and growing to integrate, execute and instrument a variety of tasks in a workflow for on-line analysis and coupling. In addition to exchanging data through a code coupler 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 take long time to execute, it is necessary that coupling with analysis and visualization codes be done 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.

The amount of data generated by these simulations on extreme scale machines often exceed 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 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.

This type of workflows are becoming more diverse as applications invest in Machine Learning (ML) frameworks (e.g. TensorFlow) and integrate ML workflows with the application's primary workflows. Another growing requirement is the integration of experimental data with simulation data for validation and prediction of their scientific results.

### 5.3.2 Workflow Composition

The capacity and associated bandwidth of today's file systems are straining to store ephemeral intermediate data in workflows. 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 requires 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 has focused on providing better programming model support for HPC SSIO and imparting more information from the programming model to lower-level storage system interfaces. Other communities outside HPC have conducted R&D (e.g., MapReduce [**Dean2008**]) to better integrate storage within programming models and 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**].

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. 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**], looking at methods for passing information from applications to the runtime (via the programming model) so that tradeoffs 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. To date, however, these efforts have not focused on providing better support for SSIO.

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 [**Wosniak2014**] 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 of 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**].

**Challenges**

Future HPC systems will incorporate multiple levels of memory and storage, including high-bandwidth designs, NVRAM, DRAM, disk, and tape. This 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 operating system 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 which allow for query of the characteristics, state and workload of a storage level and placement of data at a specific level and 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 is 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, integration of such optimizations in the programming model through lower-level interfaces and 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 (see, 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 getting scientific workflow tool users using 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.

### 5.3.3 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, has shown the potential for exploiting data locality in task placement [Duro2014]. Similarly, the ADIOS [**Lofstead2008**, **Lofstead2014**] 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 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.

**Challenges**

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. More research is needed to take advantage of complex storage and memory hierarchies in future systems for dynamically configured workflows. Also strongly needed is linkage to resource management systems for more dynamic allocation of resources within the workflow. 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 and compilers.

Additionally, 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. Successful solutions here will need co-design 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.

### 5.3.4 I/O Middleware and Libraries (Connectivity) - both on-and offline, (not or)

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. 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 component 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 quality of service(QOS) guarantees for data movement costs is essential.

**State of the Art [needs more references, and needs more pieces]**
A variety of middleware libraries are addressing these needs to various degrees. ADIOS [Liu2014], and associated data staging mechanisms SST, SST-2, Dataspaces [**Docan2012**] and Flexpath [Dayal2014], represent a widely adopted and rapidly maturing set of middleware tools. Other libraries aimed this space include Conduit, Damaris [**Dorier2012**], DAOS, and Decaf.

The libraries primarily are 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.

**Challenges**
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. Emerging platforms have different RDMA mechanisms and network fabrics and may host a range of accelerator technologies which may have separate memory and interconnect concerns. 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.

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.

### 5.3.5 Data Abstractions and Representation

Within multi-developer monolithic code bases, it is not unusual to have policy-enforced standards for the layouts of particular data structures. However, as one moves towards 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 3rd 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 [Damsel2014] project investigated methods for storing unstructured arrays.

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 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 PVFS [Carns2000], with PVFS 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 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, multi-dimensional data structures has seen 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.

**Challenges**

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 are needed, 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, based on its location. Expectations of cost of data access must be made available 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 been done one exploring abstractions and runtime mechanisms for application-driven data management across deep memory hierarchies [Jin2015], and associated energy/performance tradeoffs have been explored [Gamell2013]; but much work remains.

Relationships between data are not represented well in current SSIO approaches. In models such as HDF, data can be grouped and organized in a hierarchy within a file, but relationships across files are not readily captured. Overall, a richer method for expressing relationships between data items is needed.

As the cost (in terms of energy or time) of data movement continues to increase, the need for alternatives to standard compression grows. Additional research is needed to allow users to control the size of stored data while understanding the cost in fidelity. In particular, techniques must be devised that can provide a range of bounded error options.

Additional research also is needed 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. Embedding of metadata with data may be necessary to enable certain classes of analysis as well.

As discussed in the data abstraction section, additional information needs to be communicated between applications and layers of the I/O stack. This information should be coupled with mechanisms that allow for adjustment of data representation in response to changing demands, including the possible storage of multiple representations (or partial representations) in anticipation of a variety of upcoming use cases. In situations where data is reorganized, information on how that data was reorganized must be captured, particularly if the fidelity of the data may have been affected by the reorganization.

### 5.3.6 Data Refactoring/compression

**Introduction**

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 l 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. In order 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. Lossless methods have unfortunately been generally unable to achieve the high compression ratios needed to handle the huge quantities of data being produced [**Ainsworth2017**]. Consequently, attention has turned to lossy methods, which brings into question the fidelity of the reduced dataset to the original dataset in addition to the amount of compression they achieve. Fidelity and reduction are in direct competition, so it is worthwhile to consider what exactly is required of a reduced dataset in order for it serve as a scientifically useful surrogate.

**State of the Art**

There are two major classes of data reduction techniques which are being researched and developed in the community: lossless and lossy. New research is needed in both of these areas in order to fully explore how scientific workflows can reduce the volumes of data being moved and processed through next generation routines. Compression is a class of reduction techniques which can reduce the total amount of data that needs to be transferred and stored, but the challenge has quite often been in the speed. Luckily, new algorithms along with faster processors have helped the community move in this direction.

Lossless compression

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 the traditional, non-compressed, direct memory fetch approach via a memcpy() OS call. Blosc is the first compressor that is meant not only to reduce the size of large datasets on-disk or in-memory, but also to accelerate memory-bound computations. It uses the blocking technique so as to reduce activity in the memory bus as much as possible. In short, this technique works by dividing datasets in blocks that are small enough to fit in caches of modern processors and perform compression / decompression there. It also leverages, if available, SIMD instructions (SSE2, AVX2) and multi-threading capabilities of CPUs, in order to accelerate the compression / decompression process to a maximum. Furthermore, 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 data sets and meets the throughput demands of high-performance systems.

Lossy compression

Lossy compression, such as JPEG [12], has long been used in computer graphics and digital images, leveraging the fact that the visual resolution by human eyes is well below machine precision. However, its application in the scientific domain is less well established. Since scientific data are primarily composed of high-dimensional floating-point values, lossy floating-point data compressors have begun to emerge, including ISABELA [13], ZFP [14], SZ [15], and Tucker decomposition[].

## Challenges

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 use lossy compression from a user perspective, e.g., which compressor should be used for a particular dataset, and what level of reduction ratio should be expected. We also need to address questions on: 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 loosening error bound? How does the design of compression influence compression throughput? What is the relationship between compression ratio and throughput? What is the impact of lossy compression on data fidelity and complex scientific data analytics? How to extract data features and accurately predict the compression ratios of various compressors?

Through answering these questions, we need the ability to help HPC end users understand what to expect from lossy compressors. As a completely unbiased third-party evaluation without ad hoc performance tunings, we need to shed light on the limitations of existing compressors, and point out some of the new R&D opportunities for compressor developers and the communities to make further optimizations, thus ensuring the broad adoption of reduction in science production.

### 5.3.7 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; (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 it is 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. These datasets are stored in large files, which may reach hundreds of terabytes in size, and large numbers of 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 analyses 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 a 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 most of the time used for temporary storage. Scientists need to explicitly move their data from the parallel file system to a longer-term storage level before their datasets are deleted. This becomes a significant challenge when a study involves multiple workflows and 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 distribured environments. The Integrated Rule-Oriented Data System (iRODS) [Rajasekar2010] provides a unified virtual collection model across heterogeneous data resources but not other resource types. There are also work on rule based data management frameworks and rule languages [Paschke2005,Horrocks2004,Browne2009]. 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.

## Challenges

Scientists are in need for the SSIO community to research and develop support for managing and tracking data that are produced and consumed throughout the data lifecycle of a scientific study. This support 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 system be aware of the intentions of resource consumers and producers. Data can be viewed as both a resource and a consumer of other resources (storage, network, computation). In this context, analysis workflows describe how the workflow author intends that data be processed within an analysis workflow.

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 storage and retention policies of different storage levels. For example, scientists should be able to specify "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 would require research on rule-based systems and integration of rule-based decision making capabilities in SSIO systems.

## 5.4 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 the increasing complexity this is no longer a viable long-term solution.

To address this problem, we need 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, 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.

### 5.4.1 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 XX).

At the very top of the hierarchy are the storage devices that exist on compute nodes themselves, typically NVRAM such as SSD, but in the near future are expected to

include storage class memory (SCM) devices such as 3D XPoint® memory or Z-NAND flash memory. Node-local storage devices have the fastest access times, with SCM devices being several times faster than SSD, but with smallest capacities and shortest data lifetimes, typically 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 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 PBs capacity and data lifetimes on the order of months. A relatively new addition to the storage hierarchy is campaign storage, 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 extremely slow access times.

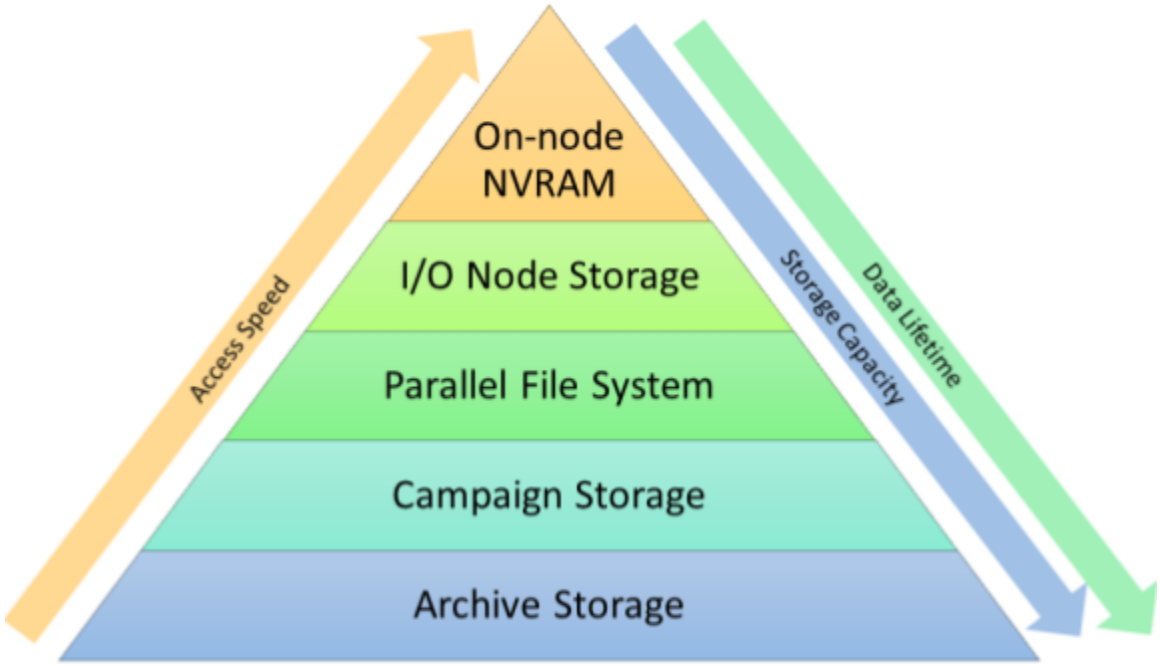



Figure 3. The storage hierarchy on HPC systems is growing more complex. An important consideration is how to best integrate on-node storage class memory devices into the traditional storage infrastructure.

**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 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.

The current state of the art for hierarchical storage management (HSM) in HPC are 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 [VeloC n.d.] are checkpoint/restart libraries that capitalize on the short lifetimes of checkpoint files which are only needed until a more recent replacement checkpoint is written and optimize checkpointing time by transparently utilizing node-local storage to cache checkpoint data and move a select few checkpoints down 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.] and HDF5 with Data Elevator [Dong2016].

**Challenges**

Significant challenges in this area exist, with the deepening of the storage hierarchy and performance characteristics of next-generation storage technologies. Additional research is needed to understand 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. Additionally, 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 of 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.

### 5.4.2 The Role of Nonvolatile Memory in SSIO

Nonvolatile RAM or NVRAM currently exists in HPC systems as simple node-local SSD storage or as burst buffer with system software support. In the near future, we expect HPC systems to additionally include storage class memory (SCM) devices [Freitas2008] such as 3D XPoint® memory or Z-NAND flash memory, and in

next-generation systems devices such as phase change memory or memristor (See Section 5.1.1 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.

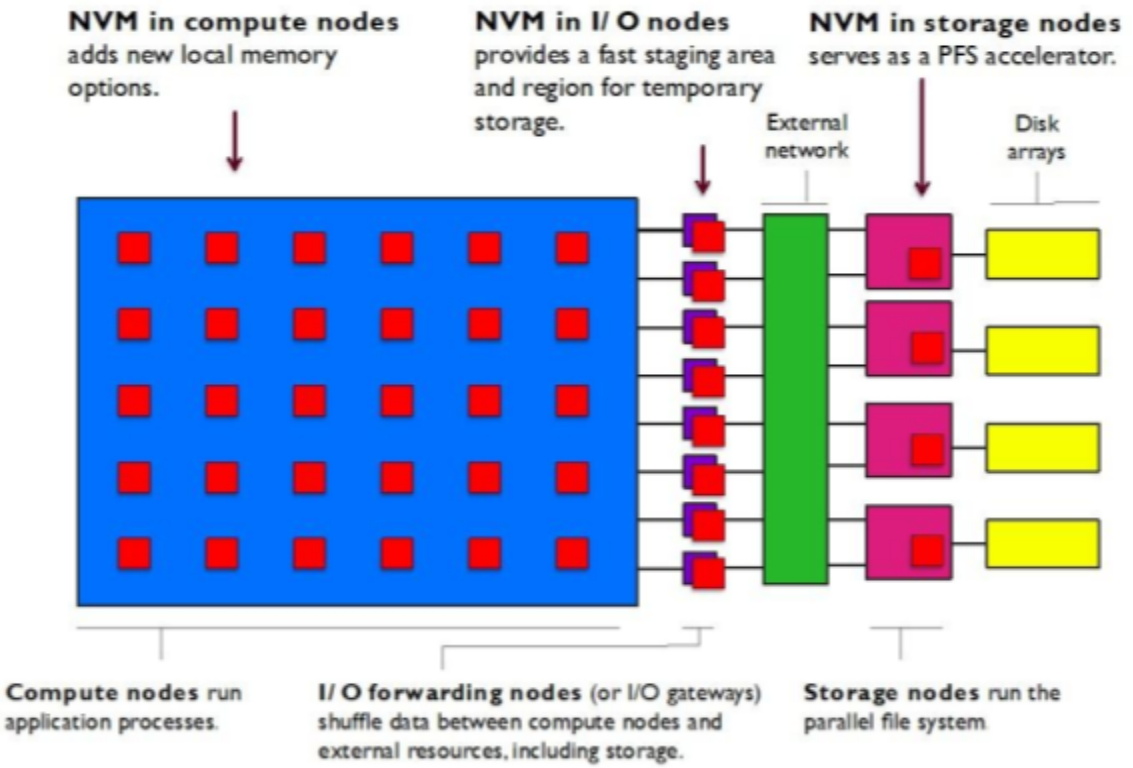



Figure 3: Nonvolatile memory (NVM) is an important component of today's and future SSIO architectures. Placement of NVM in the system influences the utility of the NVM for specific use cases, and additional research is needed to best understand where and how to integrate this technology to provide the 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; 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.

## Challenges

The addition of NVRAM in HPC storage systems increases complexity, arising not only from determining best placement of NVRAM in the system, ranging from compute node to I/O node to the parallel file system (See Figure XX), but also from understanding its role in the storage hierarchy from an application viewpoint. NVRAM devices add to storage system costs and so we need to evaluate the cost/performance tradeoffs for placement of NVRAM in different locations in the storage hierarchy. In addition, for NVRAM placed on compute nodes, we need to understand if the devices are best employed as extensions of memory or fast storage devices in the hierarchy. Considerations for this choice include application I/O workload as well as read and write asymmetries and wear characteristics of the device. A full understanding of the trade-offs will require research into these areas.

### 5.4.3 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 ensure less contention at the storage system and 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 done for grid computing [Schopf2002]. Several works 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 works 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].

## Challenges

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 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. Additionally, the emergence of non-traditional HPC I/O workloads (data-intensive or in-situ) complicates I/O pattern prediction. For reducing inter-job I/O contention, we will need a wealth of information on historical and running application I/O characteristics, storage system load, and the ability to feed this data back to tools and job schedulers.

Workflow management systems present a multidimensional resource provisioning challenge for the SSIO system. Workflow systems could, however, become part of the solution, in that they can provide a priori and runtime information of 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.

### 5.4.4 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 simulation experiments, e.g., to explore the application parameter space, that can last months or longer. Archival storage has long been 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 has recently emerged as a new tier in the storage hierarchy, e.g., on Trinity at LANL. It is intended to provide relatively fast access compared to archival storage to colder application data via a 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 provide data protection in even 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 done 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].

## Challenges

For campaign storage, there is a need for researching scalable mechanisms for handling extremely large data sets with good performance. Aspects of this need include: investigation into the POSIX/near-POSIX requirements for this tier of storage, efficient metadata handling, data set 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 we expect a larger fraction of jobs to employ data analysis *in situ* on the burst buffer, we need to devise scalable explicit and automated HSM approaches for moving data between the tiers. We expect 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 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) effectively verifying the integrity of archive files over time and continuous technology migration, (2) the cost and practicality of retrieving and searching data that are resident on slow TTFB devices, and (3) investigation of new storage media for archive as the cost of tape implementations increases.

## 5.5 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 under-developed. 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.

### 5.5.1 Instrumentation and Monitoring

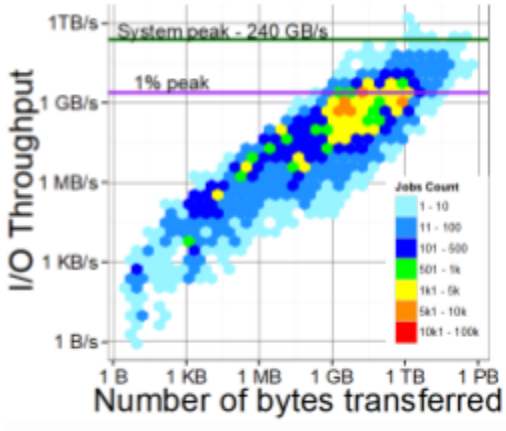
As increasingly sophisticated SSIO systems are deployed, it will become more and more important to extract information about application behavior, the state of constituent devices, and the aggregate condition of the system. These three aspects of system state can be collected via instrumentation or monitoring 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. Research will also be needed to standardize the format of the information being collected and to enable rich analytics of the collected performance data to 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 due to its ability to instrument production applications without perturbing performance. Higher fidelity tools such as TAU [Shende2006], IPM [Uselton2010], 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 5: 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].




## Challenges

The first challenge in instrumentation and monitoring research is dealing with the onslaught of rapidly evolving HPC technology. This includes new application models such as machine learning frameworks [Abadi2016], new hardware devices such as byte-addressable non-volatile 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. Tools such as Darshan, Score-P, and SIOX have made initial steps towards this problem by

modularizing their software architecture [Snyder2016] to permit future expansion, but more work is needed.

Secondly, data integration and alignment is an ongoing challenge, particularly as systems become more heterogeneous and diverse. This is reflected in a lack of standardized formats for storage telemetry that make data sharing and common analysis difficult. It is also reflected in 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. Early work such as [Muelder2011] have demonstrated preliminary capability, but such techniques have not been widely adopted in the field.

Finally, the tension between instrumentation fidelity and runtime overhead have 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 ability to adapt polling frequencies and instrumentation methods at run time, but it is largely the responsibility of administrators and individual users to make ad hoc decisions to address this tradeoff.

### 5.5.2 Interpretation and Application

Instrumentation and monitoring data is only as useful as its application in the service of computer science research or scientific productivity. This can be accomplished in a number of ways, such as 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 under way 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 [NERSC2015] 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,IODoctors2018] have also brought to market I/O analysis tools with an increased focus on the end-user perspective.

**Challenges**

I/O Performance data integration will remain 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 to best enact solutions. Possibilities include guided suggestions, automatic tuning, and automatic reconfiguration of available resources.

### 5.5.3 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 multi-step 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 5.5.1, but it is also important to point out 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 IOR [Shan2008], which focuses on bulk data performance, and MDTest [MDTest], 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]. Additionally, 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 to extend the benchmarks beyond single case executions into regular regression tests that are track performance over time for a system [Lockwood2017,Palmer2015].

**Challenges**

Very little work exists in modeling *future* SSIO workloads. This 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, incorporating 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.

### 5.5.4 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 broadling 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 data sets 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.

**Challenges**

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 5.5.1) into models for ongoing validation and evolution, and how do we combine workload and system modeling (Sections 5.5.2 and 5.5.3) into a coherent framework? The CODES project has made progress on the latter issue, but the former remains an open challenge. Secondly, 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 non-volatile memory (NVM), shingled disks, and heat-assisted magnetic recording (HAMR) are not well-represented in today's models. At the system level, scientific campaigns will increasingly span multiple jobs, multiple system resources, and multiple data sets in order to produce a meaningful scientific result. This calls for storage system models to broaden their scope to include multi-job workloads, 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.

## 5.6 Streaming Data

The storage systems designed for advanced scientific computing have historically been designed to absorb I/O generated from bulk-synchronous checkpoint and restart operations. However, the rate at which experimental and observational data (EOD) 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 only access parts of certain data objects, large collections of data objects, and data objects that may be distributed across disparate physical storage systems. This mode is most often used in large collaborative experiments such as the Compact Muon Solenoid [Bloom 2015] at the Large Hadron Collider where throughput, not latency, of the analysis is the chief optimization point.

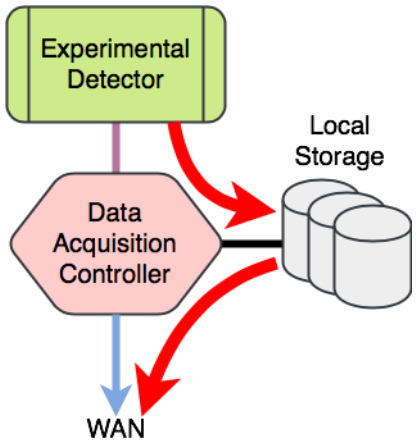
These two modes are by no means mutually exclusive, and a single experimental or observational facility may require that both tightly coupled acquisition and analysis and asynchronous streaming analysis be carried out at a large-scale computing facility. Furthermore, the precise definition of streaming 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 if such a workflow is truly unique from a workflow which involves simple remote file transfers.

### 5.6.1 Tightly coupled acquisition and analysis

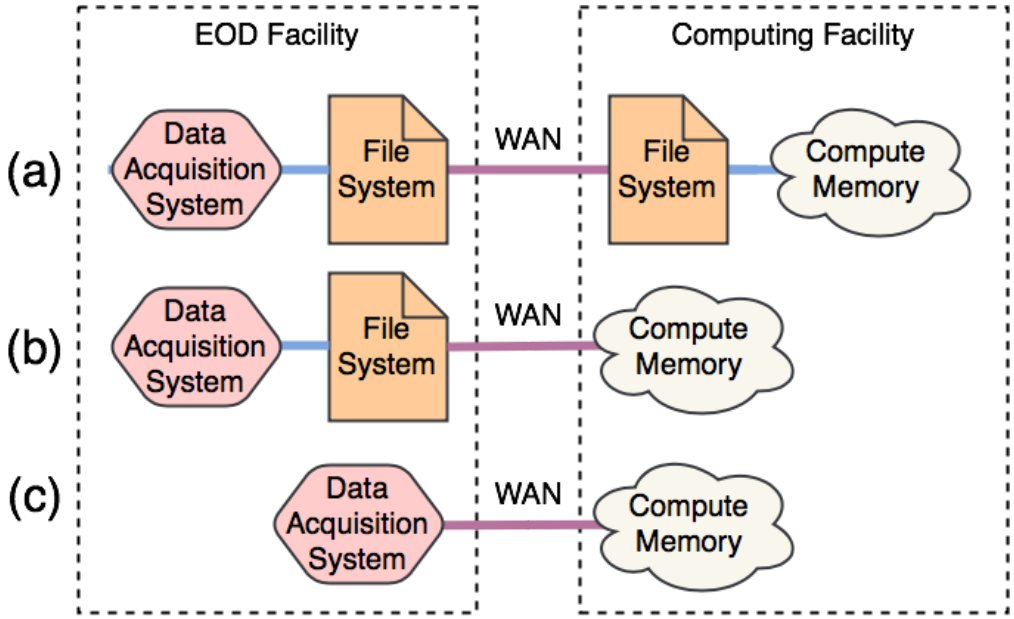
**State of the Art**

The relatively high cost of generating data through an experimental or observational instrument 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 below.




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 below.




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 inter-facility data motion, and POSIX file systems have an extremely well defined API and access semantics. As such though, this mode is not truly streaming so much as it is 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 since file-to-file transfers use well-established tools and semantics.

A major issue with respect to file-to-file transfers are 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 unit from a DAQ need to be read to determine if the data is 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 tradeoff 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 is the notionally optimal configuration of the three modes, 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 WAN or compute facility. 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 mades 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.

**Challenges**

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 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 which impedes moving from mode (a) and towards the more latency-optimized modes (b) and (c) is the lack of well defined interfaces between DAQ systems and the compute 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 is a promising direction that preserves 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.

### 5.6.2 Asynchronous streaming analysis

**State of the Art**

High-value experimental data sets as described in the previous section 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. The mechanisms by which such data sets are distributed to their scientific communities are widely variable though.

Owing in part to the size of these data sets, 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 data set 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 data set.

The APIs and semantics of accessing these remote data sets 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 data sets 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[STREAM2016].

**Challenges**
Because asynchronous streaming analysis often targets only subsets of large distributed data sets, 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, programming models and APIs to perform these tasks are not well defined beyond existing relational and non-relational database interfaces.

This lack of well defined models and semantics can be generalized to other aspects of streaming data analysis in general. 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 towards defining a meaningful data model and access API, it still relies on a network file system to provide a protocol for streaming data.

# 6 Supporting Activities

This section is taken, verbatim, from the 2015 workshop report. Many of the activities described, seemingly, continue to remain though particular examples may be dated. The final report will contain an updated version as modified by input from the attendees.

Although the workshop did not have a dedicated session on supporting activities for SSIO research, throughout the workshop supporting activities were called out as requirements for the research 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 three 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 projection to future systems. The third theme was educational support to enable better understanding of HPC SSIO problems.

## 6.1 Computing, Networking, and Storage Resources

[Introductory text here]

### 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 [Chameleon n.d.] 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/LLNL Hyperion [Hyperion n.d.] and the NSF/DOE/NNSA/LANL PRObE [PRObE n.d.] systems were also mentioned as useful; however, these systems are likely insufficient for future SSIO research without expansion of 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 is the only resource that is specifically designed for researchers to have full access all the way to the hardware and to allow 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 are not well suited for future research. The need is really a combination of new and renewed frequently 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.

### 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 give users bare metal root access. PRObE and Hyperion do provide some aspects of this need area, but they 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.

## 6.2 Availability of Highly Documented Operational Data

[Introductory blurb here]

**State of the Art**

Several facilities are already making well-documented operational data available. Among the releases are data from LANL [LANL Data n.d.] and NERSC [NERSC Data n.d.]. The LANL failure data release represented the largest operational failure data release done 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 is provided by the large-scale Sandia trace data [Sandia Data n.d.] and the Argonne Darshan usage-related data [Carns2013, ANL Data 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. Additionally, storage system namespace statistics are provided by the DOE Petascale Data Storage Institute's FSStats effort [Felix2011, PDSI FSStats Data n.d.]. 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 tracking 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, is 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.

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

## 6.3 Educational Support

Support for education in the SSIO areas includes tutorials at conferences and workshops and materials suitable for classroom teaching, such as reference books.

**State of the Art**

Tutorials such as the Supercomputing Conference recurring tutorial "Parallel I/O In Practice" [Parallel I/O Tutorial n.d.] and recurring informative workshops such as the Supercomputing Conference Parallel Data Storage Workshop [PDSW n.d.] represent state of the art in SSIO educational outreach. Additionally, textbook documentation about SSIO research is best exemplified by "High Performance Parallel I/O" [Koziol2014], "Parallel I/O for High Performance Computing" [May2001], and "Scalable Input/Output: Achieving System Balance" [Reed2004].

**Challenges**

For the most part, textbook creation and, to a great extent, educational outreach activities are not funded for SSIO researchers or SSIO experts at HPC sites.

# 7 Glossary

| | |
|---|---|
| ADIOS | Adaptive I/O System |
| ANL | Argonne National Laboratory |
| ASC | Advanced Simulation and Computing |
| ASCR | Advanced Scientific Computing Research |
| CAP | Consistency, Availability, and Partition tolerance |
| CMU | Carnegie Mellon University |
| CORAL | A collaboration between ANL, LLNL, and ORNL to acquire advanced computing resources |
| DOE | Department of Energy |
| EOD | Experimental and Observational Data |
| FSIO | File Systems and I/O |
| GPFS | General Parallel File System |
| HACC | Hardware/Hybrid Accelerated Cosmology Code |
| HAMR | Heat-Assisted Magnetic Recording |
| HDF | Hierarchical Data Format |
| HEC FSIO | High End Computing File Systems and I/O |
| 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 |
| LBNL | Lawrence Berkeley National Laboratory |
| LLNL | Lawrence Livermore National Laboratory |
| LWFS | Light Weight File System |
| MDS | MetaData Server |
| MIMD | Multiple Instruction Multiple Data |
| MPMD | Multiple Program Multiple Data |
| 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 (see NVMHCIS) |
| NVMHCIS | Non-Volatile Memory Host Controller Interface Specification |
| ORNL | Oak Ridge National Laboratory |
| PDSI | Petascale Data Storage Institute |
| PDSW | Petascale Data Storage Workshop |
| PLFS | Parallel Log-structured File System |
| PNNL | Pacific Northwest National Laboratory |
| POSIX | Portable Operating System Interface |

| | |
|---|---|
| PRObE | Parallel Reconfigurable Observational Environment |
| QoS | Quality of Service |
| RAID | Redundant Array of Independent Disks |
| RAS | Reliability, Availability, and Serviceability |
| RMA | Remote Memory Access |
| SIMD | Single Instruction Multiple Data |
| SNL | Sandia National Laboratories |
| SSD | Solid State Disk |
| SSIO | Storage System and I/O |
| UQ | Uncertainty Quantification |
| XDD | Command line tool for measuring I/O performance |

# 8 References

**[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**

**[Abadi2016] Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin et al. "Tensorflow: a system for large-scale machine learning." In *OSDI*, vol. 16, pp. 265-283. 2016.**

**[Abbasi2010] Abbasi, Hasan, et al. "Datastager: scalable data staging services for petascale applications." *Cluster Computing* 13.3 (2010): 277-290.**

**[Abolfathi2018] Abolfathi, B. et al. 2018. "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." *The Astrophysical Journal Supplement Series* 235 (2).**

**[Acharya1998] Acharya, Anurag, Uysal, Mustafa, and Saltz, Joel. "Active disks: Programming model, algorithms and evaluation." In Proceedings of ASPLOS'98. 1998.**

**[Adams2012] Adams, I. A.; Madden, B. A.; Frank, J. C.; Storer, M. W.; Miller, E. L.; Harano, G. "Usage behavior of a large-scale scientific archive." In Proceedings of 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." In Proceedings of the Particle Accelerator Conference, 2005. PAC 2005, pp. 4129-4131. IEEE, 2005.

**[Agelastos2014] Agelastos, Anthony, et al. "The Lightweight Distributed Metric Service: A scalable infrastructure for continuous monitoring of large scale computing systems and applications." In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '14, pages 154–165, 2014.**

**[Aghayev2015] A. Aghayev and P. Desnoyers. Skylight—A Window on Shingled Disk Operation. In the proceedings of the 13th USENIX Conference on File and Storage Technologies (FAST 15) , pages 135–149, 2015.**

**[Aghayev2017] Abutalib Aghayev, Theodore Ts'o, Garth Gibson and Peter Desnoyers. Evolving EXT4 for Shingled Disks. In the proceedings of the 15th USENIX Conference on File and Storage Technologies (FAST '17), 2017.**

[Ainsworth2017] Mark Ainsworth, Scott Klasky, and Ben Whitney. Compression using lossless decimation: Analysis and application. SIAM Journal on Scientific Computing, 39(4):B732–B757, August 2017.

[Alagiannis2014] I. Alagiannis, S. Idreos, and A. Ailamaki. "H2o: A hands-free adaptive store." In SIGMOD '14, Snowbird, UT, June 22-27, 2014.

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

[Ali2009] Ali, Nawab, Philip Carns, Kamil Iskra, Dries Kimpe, Samuel Lang, Robert Latham, Robert Ross, Lee Ward, and P. Sadayappan. "Scalable I/O forwarding framework for high-performance computing systems." In IEEE International Conference on Cluster Computing and Workshops, 2009. CLUSTER'09, pp. 1-10. IEEE, 2009.

**[Allen2004] Bruce Allen. Monitoring Hard Disks with SMART. In the Linux Journal, January 1, 2004.**

[**Alted2010**] Alted, Francesc. "Why modern CPUs are starving and what can be done about it." Computing in Science & Engineering 12, no. 2 (2010): 68-71.

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

**[Alverson2012] Bob Alverson. Cray High Speed Networking. In the 20th IEEE Symposium on High Performance Interconnects (Hot Interconnects), 2012.**

**[Ames2011] Sasha Ames, Maya B. Gokhale, and Carlos Maltzahn. QMDS: A File System Metadata Management Service Supporting a Graph Data Model-Based Query Language. In Proceedings of the 6th IEEE International Converence on Networking, Architecture and Storage (NAS), 2011, pp. 268, 277, July 28-30, 2011.**

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

[Anderson2000] Anderson, Darrell C., Jeffery S. Chase, and Amin M. Vahdat. "Interposed request routing for scalable network storage." In Proceedings of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4. USENIX Association, 2000.

**[Anderson2018] Patrick Anderson, Richard Black, Ausra Cerkauskaite, Andromachi Chatzieleftheriou, James Clegg, Chris Dainty, Raluca Diaconu, Austin Donnelly, Rokas Drevinskas, Alexander L. Gaunt, Andreas Georgiou, Ariel Gomez Diaz, Peter G. Kazansky, David Lara, Sergey Legtchenko, Sebastian Nowozin, Aaron Ogus, Douglas Phillips, Antony Rowstron, Masaaki Sakakura, Ioan Stefanovici, Benn Thomsen, Lei Wang, Hugh Williams, and Mengyang Yang. Glass: A New Media for a New Era? In the 10th USENIX Workshop on Hot Topics in Storage and File Systems, 2018.**

[ANL Data n.d.] ALCF I/O data repository http://press3.mcs.anl.gov/darshan/data/

**[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.**

[Arpaci-Dusseau1999] Arpaci-Dusseau, Remzi H., et al. "Cluster I/O with River: Making the fast case common." In Proceedings of the sixth workshop on I/O in Parallel and Distributed Systems. ACM, 1999.

[Arpaci-Dusseau2006] Arpaci-Dusseau, Remzi H., Andrea C. Arpaci-Dusseau, Benjamin R. Liblit, Miron Livny, and Michael M. Swift. "Formal failure analysis for storage systems." High End Computing University Research Activity NSF 06-503 (2006)

[Arpaci-Dusseau2014] Remzi Arpaci-Dusseau, Andrea Arpaci-Dusseau, Carlos Maltzahn: Reproducible evaluation of HPC Systems. SSIO White Paper, Dec. 2014.

**[Bethel2015] Bethel, E, et al. Management, Analysis, and Visualization of Experiment and Observational Data (EOD) - The Convergence of Data and Computing Workshop Final Report. Office of Advanced Scientific Computing Research, DOE Office of Science Bethesda, Maryland, September 29–October 1, 2015.**
**https://science.energy.gov/~/media/ascr/pdf/programdocuments/docs/ascr-eod-workshop-2015-report_160524.pdf**

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

**[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 on August 1, 2018.**

[BackBlaze2015] https://www.backblaze.com/hard-drive-test-data.html

[Barseghian2010] Derik Barseghian, Ilkay Altintas, Matthew B. Jones, Daniel Crawl, Nathan Potter , James Gallagher, Peter Cornillon, Mark Schildhauer, Elizabeth T. Borer, Eric W. Seabloom, Parviez R. Hosseini "Workflows and extensions to the Kepler Scientific Workflow System to support environmental sensor Data access and analysis." Ecological Informatics 5 (2010): 42-50.

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

**[Barton2014] Barton, Eric, Bent, John, and Quincey Koziol. "Fast forward storage and IO program documents. " https://wiki.hpdd.intel.com/display/PUB/Fast+Forward+Storage+and+IO+Program+Documents.**

**[Baru1998] Chaitanya Baru, Reagan Moore, Arcot Rajasekar, and Michael Wan. "The SDSC storage resource broker." In Proceedings of the 1998 conference of the Centre for Advanced Studies on Collaborative Research, p. 5. IBM Press, 1998.**

**[Bauer2012] Bauer, Michael, Sean Treichler, Elliott Slaughter, and Alex Aiken. "Legion: expressing locality and independence with logical regions." In Proceedings of the international conference on High Performance Computing, Networking, Storage and Analysis, p. 66. IEEE Computer Society Press, 2012.**

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

**[Bautista-Gomez2011] Bautista-Gomez, Leonardo, Seiji Tsuboi, Dimitri Komatitsch, Franck Cappello, Naoya Maruyama, and Satoshi Matsuoka. "FTI: high performance fault tolerance interface for hybrid systems." In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, p. 32. ACM, 2011.**

[Beck2002] Beck, Micah, Terry Moore, and James S. Plank. "An end-to-end approach to globally scalable network storage." ACM SIGCOMM Computer Communication Review 32. 4 (2002).

**[Behzad2014a] Babak Behzad, Surendra Byna, Stefan Wild, Prabat, and Marc Snir. "Improving Parallel I/O Autotuning with Performance Modeling." In the 23rd International ACM Symposium on High Performance Distributed Computing. June 2014.**

**[Behzad2014b] Babak Behzad, Hoang-Vu Dang, Farah Hariri, Weizhe Zhang, and Marc Snir. "Automatic generation of I/O kernels for HPC applications." Parallel Data Storage Workshop (PDSW) 2014.**

**[Behzad2015] Babak Behzad, Surendra Byna, Stefan Wild, Mr Prabhat and Marc Snir. Dynamic Model-driven Parallel I/O Performance Tuning. IEEE Cluster , Sept.2015.**

[Bennett2012] Bennett, Janine C., Hasan Abbasi, P-T. Bremer, Ray Grout, Attila Gyulassy, Tong Jin, Scott Klasky et al. "Combining in-situ and in-transit processing to enable extreme-scale scientific analysis." In International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2012, pp. 1-9. IEEE, 2012.

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

[Bent2009] Bent, John, Garth Gibson, Gary Grider, Ben McClelland, Paul Nowoczynski, James Nunez, Milo Polte, and Meghan Wingate. "PLFS: A checkpoint filesystem for parallel applications." In Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, p. 21. ACM, 2009.

**[Bent2012] John Bent, Sorin Faibish, James Ahrens, Gary Grider, John Patchett, Percy Tzelnic, and Jon Woodring. "Jitter-free co-processing on a prototype exascale storage stack." In 28th IEEE Symposium on Massive Storage Systems and Technologies, MSST 2012, 2012.**

**[Bent2016] John Bent, Brad Settlemyer, and Gary Grider, "Serving Data to the Lunatic Fringe: The Evolution of HPC Storage," ;login: The USENIX Magazine, Summer 2016, 41(2).**

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

**[Betke2017] Betke E., Kunkel J. (2017) Real-Time I/O-Monitoring of HPC Applications with SIOX, Elasticsearch, Grafana and FUSE. In: Kunkel J., Yokota**

**R., Taufer M., Shalf J. (eds) High Performance Computing. ISC High Performance 2017. 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 arXiv:1711.03573* (2017).**

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

[Blackcomb n.d.] https://ft.ornl.gov/trac/blackcomb

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

**[Bloom2015] K. Bloom, T. Boccali, B. Bockelman, D. Bradley, S. Dasu, J. Dost, F. Fanzago, I. Sfiligoi, A. M. Tadel, M. Tadel, C. Vuosalo, F. Wurthwein, A. Yagil, and M. Zvada, "Any Data, Any Time, Anywhere: Global Data Access for Science," in 2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC), 2015, pp. 85–91.**

**[Boboila2012] Boboila, Simona, Youngjae Kim, Sudharshan S. Vazhkudai, Peter Desnoyers, and Galen M. Shipman. "Active flash: Out-of-core data analytics on flash storage." In IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST), 2012, pp. 1-12. IEEE, 2012.**

**[Bockelman2018] Brian Bockelman. CMS IO Overview. HEP-CCE: Scalable IO for Energy and Intensity Frontier Experiments. Argonne, IL, August 23, 2018.**

**[Bonnie2018] David Bonnie. Massive Scale Metadata Efforts and Solutions. In the Proceedings of the 34th International Conference on Massive Storage Systems and Technology (MSST 2018), 2018.**

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

**[Bornholt2016] James Bornholt, Randolph Lopez, Douglas Carmean, Luis Ceze, Georg Seelig, and Karin Strauss. A DNA-based Archival Storage System. ASPLOS 2016: International Conference on Architectural Support for Programming Languages and Operating Systems, 2016**

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

**[Braam2017] Peter Braam, David Bonnie, "Campaign Storage," MSST 2017.**

**[Brewer2015] Eric A. Brewer. Kubernetes and the path to cloud native. In the Proceedings of the Sixth ACM Symposium on Cloud Computing (SOCC '15), Keynote, 2015.**

**[Brinkmann2014] Brinkmann, A., Cortes, T., Falter, H., Kunkel, J., and Narasimhamurthy, S. "E10 – Exascale IO." E10 Working Group Technical Report. 2014 . https://hps.vi4io.org/_media/research/publications/2014/weeibcfkn14-whitepaper_e10_exascale_io.pdf**

[Broquedis2010] Broquedis, Francois, et. al. "HWloc: A generic framework for managing hardware affinities in HPC applications." In Proceedings of the 18th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP 2010), 2010.

**[Brun1997] Brun, Rene, and Fons Rademakers. "ROOT—an object oriented data analysis framework." Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 389.1 (1997): 81-86.**

[Bugra2008] Bugra Gedik , Henrique Andrade , Kun-Lung Wu , Philip S. Yu , Myungcheol Doo "SPADE: The System S Declarative Stream Processing Engine." In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, June 9-12, 2008, Vancouver, Canada.

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

**[CMS n.d.] The Compact Muon Solenoid experiment at CERN. http://cms.cern/**

[Cachin2006] Cachin, Christian, and Stefano Tessaro. "Optimal resilience for erasure-coded Byzantine distributed storage." International Conference on Dependable Systems and Networks, 2006. DSN 2006. IEEE, 2006.

**[Callahan2006] Steven P. Callahan, Juliana Freire, Emanuele Santos, Carlos E. Scheidegger, Cláudio T. Silva, and Huy T. Vo. Vistrails: Visualization meets data management. In Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, pp. 745-747, 2006.**

**[Canon2016] Shane Canon and Doug Jacobsen. Shifter: Containers for HPC. In the Proceeding of the Cray Users' Group, 2016.**

**[Carey2014] Carey, Varis, Hasan Abbasi, Ivan Rodero, and Hemanth Kolla. "Sensitivity analysis for time dependent problems: optimal checkpoint-recompute HPC workflows." In Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science, pp. 20-30. IEEE Press, 2014.**

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

**[Carns2000] Carns, Philip, Ligon, Walter, Ross, Robert B., and Rajeev Thakur. "PVFS: A parallel file system for Linux clusters." In Proceedings of the 4th annual Linux Showcase and Conference, pp. 391-430. 2000.**

[Carns2009] Carns, Philip, Robert Latham, Robert Ross, Kamil Iskra, Samuel Lang, and Katherine Riley. "24/7 characterization of petascale I/O workloads." In Proceedings of 2009 Workshop on Interfaces and Architectures for Scientific Data Storage. IEEE, 2009.

**[Carns2011] Carns, Philip, Kevin Harms, William Allcock, Charles Bacon, Samuel Lang, Robert Latham, and Robert Ross. "Understanding and improving computational science storage access through continuous characterization." ACM Transactions on Storage 7.3 (2011):8.**

[Carns2013] Carns, Philip, et al. "Production I/O characterization on the Cray XE6." In Proceedings of the Cray User Group meeting. Vol. 2013. 2013.

[Caulfield2009] Caulfield, Adrian M., Laura M. Grupp, and Steven Swanson. "Gordon: using flash memory to build fast, power-efficient clusters for data-intensive applications." ACM SIGPLAN Notices 44.3 (2009): 217-228.

[Chamberlain2007] Chamberlain, Bradford L., David Callahan, and Hans P. Zima. "Parallel programmability and the chapel language." International Journal of High Performance Computing Applications 21.3 (2007): 291-312.

[Chameleon2015] https://www.chameleoncloud.org

**[Chang2008] Chang, Fay, et al. "Bigtable: A distributed storage system for structured data." ACM Transactions on Computer Systems (TOCS) 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, Maryland, January 27-29, 2016.**

[Charles2005] Charles, Philippe, Christian Grothoff, Vijay Saraswat, Christopher Donawa, Allan Kielstra, Kemal Ebcioglu, Christoph Von Praun, and Vivek Sarkar. "X10: An object-oriented approach to non-uniform cluster computing." ACM SIGPLAN Notices 40.10 (2005): 519-538.

[Cheung2015] A. Cheung. "Towards creating application-specific database management systems." In CIDR '15, Asilomar, CA, January 4-7, 2015

**[Chou2011] Jerry Chou, Mark Howison, Brian Austin, Kesheng Wu, Ji Qiang, E. Wes Bethel, Arie Shoshani, Oliver Rübel, Prabhat, and Rob D. Ryne. Parallel index and query for large scale data analysis. In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, number 30, pp, 1-30, November 2011.**

[Chuang1999] Chuang, J., and M. Sirbu. "Stor-serv: Adding quality-of-service to network storage." In Proceedings of Workshop on Internet Service Quality Economics. 1999.

**[Cidon2013] Asaf Cidon, Stephen Rumble, Ryan Stutsman, Sachin Katti, John Ousterhout, and Mendel Rosenblum. Copysets: Reducing the Frequency of Data Loss in Cloud Storage. In the 2013 USENIX Annual Technical Conference, 2013.**

[Clos1953] Clos, Charles (Mar 1953). "A study of non-blocking switching networks". Bell System Technical Journal 32.2 (2011): 406–424. doi:10.1002/j.1538-7305.1953.tb01433.x. ISSN 0005-8580. Retrieved 22 March 2011.

[CloudLab2015] https://www.cloudlab.us

**[Coburn2011] Joel Coburn, Adrian M. Caulfield, Ameen Akel, Laura M. Grupp, Rajesh K. Gupta, Ranjit Jhala, and Steven Swanson. NV-Heaps: making persistent objects fast and safe with next-generation, non-volatile memories. In Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems (ASPLOS XVI), 2011.**

**[CoDesign] Scientific Discovery through Advanced Computing (SciDAC) Co-Design Centers.**
**http://science.energy.gov/ascr/research/scidac/co-design/**

[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] Jeremy Condit, Edmund B. Nightingale, Christopher Frost, Engin Ipek, Benjamin Lee, Doug Burger, and Derrick Coetzee. 2009. Better I/O through byte-addressable, persistent memory. In Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles (SOSP '09). ACM, New York, NY, USA, 133-146.**

**[Coral2018] Coral Collaboration, Coral-2 Benchmarks https://asc.llnl.gov/coral-2-benchmarks/**

**[Cori2015] Cori. http://www.nersc.gov/users/computational-systems/cori/ accessed on August 1, 2018.**

**[Cray2018] Cray Inc. Cray View for ClusterStor. https://www.cray.com/products/storage/clusterstor/view**

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

[Curry2012] M. L. Curry, R. Klundt, and H. L. Ward. Using the Sirocco file system for high-bandwidth checkpoints. Tech. rept. SAND2012-1087, Sandia National Laboratories, Albuquerque, NM, February 2012.

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

**[DDN2018] Data Direct Networks, DDN Insight Data Sheet. https://www.ddn.com/products/storage-monitoring-ddn-insight/**

[Dagum1998] Dagum, Leonardo, and Ramesh Menon. "OpenMP: an industry standard API for shared-memory programming." Computational Science & Engineering, IEEE 5.1 (1998): 46-55.

**[Dai2014] Dong Dai, Robert B. Ross, Philip Carns, Dries Kimpe, Yong Chen. Using Property Graphs for Rich Metadata Management in HPC Systems. In Proceedings of the 9th Parallel Data Storage Workshop, vol. 11, IEEE, 2014.**

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

[Damsel2014] Damsel: A Data Model Storage Library for Exascale Science. http://cucis.ece.northwestern.edu/projects/DAMSEL/

**[Davidson2008] Susan B. Davidson and Juliana Freire. Provenance and scientific workflows: challenges and opportunities. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08, pages 1345-1350. ACM, 2008.**

**[Dawson1983] Dawson, J.M. "Particle simulation of plasmas." Reviews of Modern Physics 55.2(1983): 403. Bibcode:1983RvMP...55..403D. doi:10.1103/RevModPhys.55.403.**

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

**[Declerck2014] Tina M Declerck. Using Robinhood to Purge Data from Lustre File Systems. In the Proceedings of the Cray Users' Group, 2014.**

[Deelman2002] Ewa Deelman, James Blythe, Yolanda Gil and Carl Kesselman "Pegasus: Planning for Execution in Grids." GriPhyN Technical Report 2002-20, 2002.

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

[Deelman2008] Ewa Deelman, Miron Livny, Gaurang Mehta, Andrew Pavlo, Gurmeet Singh, Mei-Hui, Karan Vahi, R. Kent Wenger. "Pegasus and DAGMan from concept to execution: Mapping scientific workflows onto today's cyberinfrastructure." pp. 56–74. IOS, Amsterdam, 2008.

**[DEEP n.d.]**
**http://www.nersc.gov/users/data-analytics/data-analytics-2/deep-learning/deep-networks-for-hep/**

**[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] Aurélien Degremont, Thomas Leibovici. http://cdn.opensfs.org/wp-content/uploads/2013/04/lug13-robinhood.pdf**

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

[DeRoure2008] De Roure, D., Goble, C. and Stevens, R. "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]

**[Devarakonda2010] Devarakonda, R., Palanisamy, G., "Advancements in scientific data searching, sharing and retrieval". URL: https://arxiv.org/pdf/1101.1252.pdf**

**[DES2018] Dark Energy Survey - Data Management, https://des.ncsa.illinois.edu, accessed on August 31, 2018.**

**[Dickson2017] Dickson, James, , Steven A. Wright, Satheesh Maheswaran, J. A. Herdman, Duncan Harris, Mark C. Miller, and Stephen A. Jarvis,"Enabling portable I/O analysis of commercially sensitive HPC applications through workload replication." in Cray User Group 2017 Proceedings (CUG2017) pp. 1-14.**

[Dillow2011] Dillow, David A., et al. "I/O congestion avoidance via routing and object placement." In Proceedings of Cray User Group Conference (CUG 2011). 2011.

**[DiskSim n.d.] disksim. http://www.pdl.cmu.edu/DiskSim/**

**[Docan2012] Docan, Ciprian, Manish Parashar, and Scott Klasky. "DataSpaces: An interaction and coordination framework for coupled simulation workflows." Cluster Computing 15.2 (2012): 163-181.**

**[DOE2018] DOE Policy for Digital Research Data Management. https://www.energy.gov/datamanagement/doe-policy-digital-research-data-management accessed on August 31, 2018.**

[Dong2013] Bin Dong, Suren Byna, and John Wu, "SDS: A Framework for Scientific Data Services." 8th Parallel Data Storage Workshop (PDSW) held in conjunction with SC13, 2013.

**[Dong2016] Bin Dong, Suren Byna, Kesheng Wu, Prabhat, Hans Johansen, Jeffrey N. Johnson, Noel Keen, "Data Elevator: Low-Contention Data Movement in Hierarchical Storage System." HiPC 2016: 152-161.**

[Dorier2012] Dorier, Matthieu, Gabriel Antoniu, Franck Cappello, Marc Snir, and Leigh Orf. "Damaris: How to efficiently leverage multicore parallelism to achieve scalable, jitter-free I/O." In IEEE International Conference on Cluster Computing, pp. 155-163. IEEE, 2012.

**[Dorier2014] Dorier, Matthieu, S. Ibrahim, G. Antoniu and R. Ross, "Omnisc'IO: A Grammar-Based Approach to Spatial and Temporal I/O Patterns Prediction," SC14: International Conference for High Performance Computing, Networking, Storage and Analysis, New Orleans, LA, 2014, pp. 623-634.**

[Dosanjh2014] Dosanjh, Sudip. "Cori (NERSC-8)." Presented at the 2014 Scientific Discovery Through Advanced Computing (SciDAC-3) Principal Investigator Meeting. Washington, DC, August 2014.

**[Dorigo2005] A. Dorigo, P. Elmer, F. Furano, and A. Hanushevsky, "XROOTD - A highly scalable architecture for data access," WSEAS Trans. Comput., vol. 4, no. 4, p. 348, 2005.**

[Draper1999] Draper, Jesse M., David E. Culler, Kathy Yelick, Eugene Brooks, and Karen Warren. "Introduction to UPC and language specification." Center for Computing Sciences, Institute for Defense Analyses, 1999.

[Duro2014] Francisco Rodrigo Duro, Javier Garcia Blas, Florin Isaila, Justin M. Wozniak, Jesús Carretero and Robert Ross. "Exploiting data locality in Swift/T workflows using Hercules." In Proceedings of the Network for Sustainable Ultrascale Computing Workshop, 2014.

[Ekanayake2008] Ekanayake, Jaliya, Shrideep Pallickara, and Geoffrey Fox. "Mapreduce for data intensive scientific analyses." In IEEE Fourth International Conference on eScience, 2008, pp. 277-284. IEEE, 2008.

**[Ellexus2018] https://www.ellexus.com/**

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

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

[Eugster2003] Eugster, Patrick Th, et al. "The many faces of publish/subscribe." ACM Computing Surveys (CSUR) 35.2 (2003): 114-131.

**[Fahey2010] Mark Fahey, Nick Jones, and Bilel Hadri. The Automatic Library Tracking Database. In Proceedings of the Cray User Group, 2010.**

[Felderman1994] Felderman, Robert, et al. "ATOMIC: A high-speed local communication architecture." Journal of High Speed Networks 3.1 (1994): 1-29.

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

**[Felix2006] Felix, Evan J., et al. "Active storage processing in a parallel file system." In Proceedings of the 6th LCI International Conference on Linux Clusters: The HPC Revolution. 2006.**

[Felix2011] Felix, E. "Environmental molecular sciences laboratory: Static survey of file system statistics." [2011-02-23]. http://www.pdsi-scidac.org/fsstats/index, html

[Filguiera2014] Rosa Filguiera, Iraklis Klampanos, Amrey Krause, Mario David, Alexander Moreno, Malcolm Atkinson "dispel4py: A Python framework for data-intensive scientific computing." In Proceedings of the 2014 International Workshop on Data Intensive Scalable Computing Systems, pp. 9-16.

[FireWorks2013] FireWorks workflow software, http://pythonhosted.org/FireWorks. [doi: 10.5281/zenodo.14096]

**[Flynn2011] Flynn, Michael. "Flynn's taxonomy." In Encyclopedia of Parallel Computing, pp. 689-697. Springer, 2011.**

**[Folk1999] Folk, Mike, Albert Cheng, and Kim Yates. "HDF5: A file format and I/O library for high performance computing applications." In Proceedings of Supercomputing, vol. 99, pp. 5-33. 1999.**

**[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, Virginia, March 22-23, 2016.**

[Freche2009] Freche, Jens, Wolfgang Frings, and Godehard Sutmann. "High-throughput parallel-I/O using SIONlib for mesoscopic particle dynamics simulations on massively parallel computers." In PARCO, pp. 371-378. 2009.

**[Freitas2008] R. F. Freitas and W. W. Wilcke, "Storage-class memory: The next storage system technology," in *IBM Journal of Research and Development*, vol. 52, no. 4.5, pp. 439-447, July 2008.**

**[Frings2007] Frings, Wolfgang, and Morris Riedel. "LLview: User-level monitoring in computational grids and e-Science infrastructures." In *German e-Science Conference*. 2007.**

**[Gainaru2011] Ana Gainaru, Franck Cappello, Stefan Trausan-Matu, and Bill Kramer "Event log mining tool for large scale HPC systems." In Euro-Par 2011 Parallel Processing, pp. 52–64. Springer, 2011.**

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

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

[Gamell2013] M. Gamell, I. Rodero, M. Parashar and S. Poole, "Exploring energy and performance behaviors of data-intensive scientific workflows on systems with deep memory hierarchies." In Proceedings of 20th Annual International Conference on High Performance Computing (HiPC 2013), IEEE Computer Society Press, Hyderabad, India, December 2013.

**[GenZ2018] The GenZ Consortium. https://genzconsortium.org/ accessed on August 1, 2018.**

[Geni2006] GENI, NSF. Global environment for network innovations. [2007-12-17][2008-06-05]. http://www.geni.net

**[Goodell2012] Goodell, David, Seong Jo Kim, Robert Latham, Mahmut Kandemir, and Robert Ross. "An evolutionary path to object storage access." In High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion, pp. 36-41. IEEE, 2012.**

**[Goodell2015] Dave Goodell, Paul Grun, Sean Hefty, Howard Pritchard, Bob Russell, Jeff Squyres, Sayantan Sur. A Brief Introduction to Openfabrics' Interfaces libfabrics. In the 23rd IEEE Symposium on High Performance Interconnects (Hot Interconnects), 2015.**

**[Greenberg2015] Greenberg, Hugh, John Bent, and Gary 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," *2016 IEEE International Conference on Cluster Computing (CLUSTER)*, Taipei, 2016, pp. 50-59.**

**[Gu2011] Gu, J., Katramatos, D, Liu, X, Shoshani, A, Sim, A, Yu, D, Bradley , S, McKee, S, StorNet: Co-Scheduling of End-to-End Bandwidth Reservation on**

**Storage and Network Systems for High-Performance Data Transfers. IEEE INFOCOM 2011 High-Speed Networks Workshop (HSN 2011), 2011.**

**[Gupta2010] C. Gupta and M. Govindaraju, "Framework for Efficient Indexing and Searching of Scientific Metadata," *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, Melbourne, VIC, 2010, pp. 553-556. DOI: 10.1109/CCGRID.2010.120. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5493435&isnumber=5493340**

**[Gupta2014] Preeti Gupta, Avani Wildani, Daniel Rosenthal, Ethan L. Miller, Ian Adams, Christina Strong, and Andy Hospodor. An Economic Perspective of Disk vs. Flash Media in Archival Storage. In the Proceedings of the 22th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2014), 2014.**

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

**[HDF2018] The HDF Group. Hierarchical Data Format, version 5, 1997-2015. http://www.hdfgroup.org/HDF5/ accessed on August, 1 2018.**

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

**[HL-LHC n.d.] The High-Luminosity LHC Project. https://home.cern/topics/high-luminosity-lhc**

**[Habib2015] Habib, S., et al. ASCR/HEP Exascale Requirements Review Report. Advanced Scientific Computing Research and High Energy Physics, DOE Office of Science, Bethesda, Maryland, June 10-12, 2015.**

**[Hack2017] Hack, J., et al. Crosscut Report: Exascale Requirements Reviews. Office of Advanced Scientific Computing Research, DOE Office of Science, Tysons Corner , Virginia, March 9-10, 2017.**

[Hammond2011] J. Hammond. "Rationalizing message logging for Lustre." Lustre Users Group, 2011.

**[Hargrove2006] Hargrove, Paul H., and Jason C. Duell. "Berkeley Lab Checkpoint/Restart (BLCR) for Linux clusters." Journal of Physics: Conference Series. 46. 1. IOP Publishing, 2006.**

[HEC-FSIO2011] High End Computing Interagency Working Group (HECIWG) Sponsored File Systems and I/O Workshop HEC FSIO 2011.

**[Henderson2004] Henderson, Amy, Jim Ahrens, and Charles Law. The ParaView guide. Clifton Park, NY: Kitware, 2004.**

**[Herbein2016] Stephen Herbein, Dong H. Ahn, Don Lipari, Thomas R.W. Scogland, Marc Stearman, Mark Grondona, Jim Garlick, Becky Springmeyer, and Michela Taufer. 2016. Scalable I/O-Aware Job Scheduling for Burst Buffer Enabled HPC Clusters. In Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC '16). ACM, New York, NY, USA, 69-80.**

**[Hills2015] Hills, D. J., R. R. Downs, R. Duerr, J. C. Goldstein, M. A. Parsons, and H. K. Ramapriyan (2015), The importance of data set provenance for science, Eos, 96, DOI:10.1029/2015EO040557. Published on 4 December 2015. URL: https://eos.org/opinions/the-importance-of-data-set-provenance-for-science, accessed on August 31, 2018.**

**[HIO n.d.] https://github.com/hpc/libhio**

[Hoefler2009] Hoefler, Torsten, Andrew Lumsdaine, and Jack Dongarra. "Towards efficient MapReduce using MPI." In Recent Advances in Parallel Virtual Machine and Message Passing Interface, pp. 240-249. Springer Berlin Heidelberg, 2009.

[Hyperion n.d.] https://hyperionproject.llnl.gov/index.php

**[IBM CAST n.d.] https://github.com/IBM/CAST**

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

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

**[Inman2017] Jeff Inman, Will Vining, Garrett Ransom, and Gary Grider, "MarFS, a Near-POSIX Interface to Cloud Objects", ;login: The USENIX Magazine, Spring 2017, 42(1).**

**[IODoctors2018] http://iodoctors.com/**

[Ionkov2013] L. Ionkov, M. Lang, and C. Maltzahn. “Drepl: Optimizing access to application data for analysis and visualization.” In MSST ’13, Long Beach, CA, May 6-10, 2013.

[IOR n.d.]
https://www.nersc.gov/users/computational-systems/cori/nersc-8-procurement/trinity-nersc-8-rfp/nersc-8-trinity-benchmarks/ior/

**[Isaila2011] Isaila, Florin, 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, John, Isha Arkatkar, Sriram Lakshminarasimhan, Neil Shah, Eric R. Schendel, Stephane Ethier, Choong-Seock Chang, et al. "Analytics-driven lossless data compression for rapid in-situ indexing, storing, and querying." In *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. In Proceedings of the IEEE International Workshop on Emerging Parallel and Distributed Runtime Systems and Middleware (IPDRM). IEEE, June 2017.**

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

[Jin2015] Jin,Zhang, Q. Sun, H. Bui, M. Romanus, N. Podhorszki, S. Klasky, H. Kolla, J. Chen, R. Hager, C-S Chang and M. Parashar, “Exploring data staging across deep memory hierarchies for coupled data intensive simulation workflows." In Proceedings of the 29th IEEE International Parallel & Distributed Processing Symposium, Hyderabad, India, May 2015.

[Jin2013] T. Jin, 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.” In Proceedings of SC’13, The ACM/IEEE International Conference for High Performance Computing, Networking Storage and Analysis, Denver, CO, USA, November 2013.

**[Jin2017] Xin Jin, Xiaozhou Li, Haoyu Zhang, Robert Soulé, Jeongkeun Lee, Nate Foster, Changhoon Kim, and Ion Stoica. NetCache: Balancing Key-Value Stores with Fast In-Network Caching. In the 26th ACM Symposium on Operating Systems Principles (SOSP 2017), 2017.**

**[Johnson2014] Charles Johnson, Kimberly Keeton, Charles B. Morrey III, Craig A. N. Soules, Alistair Veitch, Stephen Bacon, Oskar Batuner, Marcelo Condotta, Hamilton Coutinho, Patrick J. Doyle, Rafael Eichelberger, Hugo Kiehl, Guilherme Magalhaes, James McEvoy, Padmanabhan Nagarajan, Patrick Osborne, Joaquim Souza, Andy Sparkes, Mike Spitzer, Sebastien Tandel, Lincoln Thomas, and Sebastian Zangaro. "From research to practice: Experiences engineering a production metadata database for a scale out file system." In Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 2014), USENIX, 2014.**

**[Jones2017] Terry Jones, Michael J. Brim, Geoffroy Vallee, Benjamin Mayer, Aaron Welch, Tonglin Li, Michael Lang, Latchesar Ionkov, Douglas Otstott, Ada Gavrilovska, Greg Eisenhauer, Thaleia Doudali, and Pradeep Fernando. 2017. UNITY: Unified Memory and File Space. In Proceedings of the 7th International Workshop on Runtime and Operating Systems for Supercomputers ROSS 2017 (ROSS '17). ACM, New York, NY, USA, Article 6, 8 pages.**

**[Kadekodi2015] S. Kadekodi, S. Pimpale, and G. A. Gibson. Caveat-Scriptor: Write Anywhere Shingled Disks. In 7th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 15) , 2015.**

**[Kannan2011a] Kannan, Sudarsun, et al. "Using active NVRAM for I/O staging." In Proceedings of the 2nd international workshop on Petascale data analytics: challenges and opportunities (PDAC@SC), 2011.**

**[Kannan2011b] Kannan, Sudarsun, et al. "Using active NVRAM for cloud I/O." In Proceedings of the 2011 Sixth Open Cirrus Summit. 2011.**

**[Kannan2013] Kannan, Sudarsun, et al. "Optimizing Checkpoints Using NVM as Virtual Memory." In Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, 2013.**

[Karpathiotakis2015] M. Karpathiotakis, I. Alagiannis, T. Heinis, M. Branco, and A. Ailamaki. "Just-in-time data virtualization: Lightweight data management with VIDa. "In CIDR '15, Asilomar, CA, January 4-7 2015.

**[Kettimuthu12] Kettimuthu, R., Vardoyan, G., Agrawal, G., and Sadayappan, P., "Modeling and optimizing large-scale wide-area data transfers." In 14th IEEE/ACM Symposium on Cluster, Cloud, and Grid Computing (CCGrid2014). IEEE, 2014.**

**[Kim2003] Mike Kim, John Hufferd, Mallikarjun Chadalapaka, Uri Elzur, Hemal Shah, and Patricia Thaler. iSCSI Extensions for RDMA Specification (Version 1.0), 2003.**

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

**[Kim2015] Youngjae Kim, Youngjae and Gunasekaran, Raghul. "Understanding I/O workload characteristics of a peta-scale storage system." The Journal of Supercomputing, Volume 71 Issue 3, pages 761-780, 2015.**

[Kimpe2007] Kimpe, Dries, Rob Ross, Stefan Vandewalle, and Stefaan Poedts. "Transparent log-based data storage in MPI-IO applications." In Recent Advances in Parallel Virtual Machine and Message Passing Interface, pp. 233-241. Springer Berlin Heidelberg, 2007.

[Kimpe2012] D. Kimpe, P. Carns, K. Harms, J. M. Wozniak, S. Lang, and R. Ross. "AESOP: Expressing concurrency in high-performance system software." In Proceedings of the 7th International Conference on Networking, Architecture and Storage (NAS), pp.303–312, Fujian, China, June 2012.

[Klasky2011] Klasky, Scott, Hasan Abbasi, Jeremy Logan, Manish Parashar, Karsten Schwan, Arie Shoshani, Matthew Wolf et al. "In situ data processing for extreme-scale computing." Scientific Discovery through Advanced Computing Program (SciDAC'11) (2011).

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

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

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

**[Ku2006] S. Ku, C. Chang, M. Adams, J. Cummings, F. Hinton, D. Keyes, S. Klasky, W. Lee, Z. Lin, S. Parker, 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.**

[Kung 1981] Kung, Hsiang-Tsung, and John T. Robinson. "On optimistic methods for concurrency control." ACM Transactions on Database Systems (TODS) 6.2 (1981): 213-226.

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

**[Kunkel2017] Kunkel, Julian, Lofstead, Gerald Fredrick, and Bent, John. *The Virtual Institute for I/O and the IO-500.*. United States: N. p., 2017. Web.**

**[Kunkel2018] Kunkel, Julian and George Markomanolis, Understanding Metadata Latency with MDWorkbench. Proceedings of the Workshop On Performance and Scalability of Storage Systems (WOPSSS), 2018.**

**[Kurtzer2017] Gregory Kurtzer. Containers in HPC: Singularity. In the 2017 Intel HPC Developer's Conference, 2017.**

**[LHC n.d.] The Large Hadron Collider Project. http://home.cern/topics/large-hadron-collider**

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

**[LaFon2012] Jharrod LaFon, Satyajayant Misra, and Jon Bringhurst. On distributed file tree walk of parallel file systems. In the Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '12), 2012.**

[Lakshminarasimhan2011] Lakshminarasimhan, Sriram, Neil Shah, Stephane Ethier, Scott Klasky, Rob Latham, Rob Ross, and Nagiza F. Samatova. "Compressing the incompressible with ISABELA: In-situ reduction of spatio-temporal data." In Euro-Par 2011 Parallel Processing, pp. 366-379. Springer Berlin Heidelberg, 2011.

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

[LANL Data n.d.] LANL systems – operational and fault data http://institute.lanl.gov/data/

[LeFevre2014] Jeff LeFevre, Jagan Sankaranarayanan, Hakan Hacıg¨um¨us, Junichi Tatemura, Neoklis Polyzotis, and Michael J. Carey. "Miso: Souping up big data query processing with a multistore system." In SIGMOD '14, Snowbird, UT, June 22-27 2014.

**[Leung2007] Leung, Andrew W., Ethan L. Miller, and Stephanie Jones. "Scalable security for petascale parallel file systems." In Proceedings of the 2007 ACM/IEEE conference on Supercomputing, p. 16. ACM, 2007.**

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

[Li2003] Li, Jianwei, Wei-keng Liao, Alok Choudhary, Robert Ross, Rajeev Thakur, William Gropp, Robert Latham, Andrew Siegel, Brad Gallagher, and Michael Zingale. "Parallel netCDF: A high-performance scientific I/O interface." In Supercomputing, 2003 ACM/IEEE Conference, pp. 39-39. IEEE, 2003.

**[Li2013] Li, Yan, Nakul Sanjay Dhotre, Yasuhiro Ohara, Thomas M. Kroeger, Ethan L. Miller, and Darrell Long. "Horus: Fine-grained encryption-based security for large-scale storage." In FAST, pp. 147-160. 2013.**

**[Li2017] Bojie Li, Zhenyuan Ruan, Wencong Xiao, Yuanwei Lu, Yongqiang Xiong, Andrew Putnam, Enhong Chen, and Lintao Zhang. KV-Direct: High-Performance In-Memory Key-Value Store with Programmable NIC. In the 26th ACM Symposium on Operating Systems Principles (SOSP 2017), 2017.**

**[Liao2007] Liao, Wei-keng, et al. "An implementation and evaluation of client-side file caching for MPI-IO." Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE, 2007.**

[Liewer1989] Liewer, Paulett C., and Viktor K. Decyk. "A general concurrent algorithm for plasma particle-in-cell simulation codes." Journal of Computational Physics 85, no. 2 (1989): 302-322.

[Ligon2006] Ligon, Walter B. “Improving scalability in parallel file systems for high end computing.” High End Computing University Research Activity NSF 06-503 (2006)

[Lindstrom2006] Lindstrom, Peter, and Martin 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., B. P. Duval, J. W. Farthing, T. J. Fredian, M. Greenwald, J. How, X. Llobet, F. Saint-Laurent, W. Spears, and J. A. Stillerman. "The ITER project and its data handling requirements." In 9th ICALEPCS Conference, Gyeongju, Korea. 2003.**

[Liu2004] Liu, Jiuxing, Dhabaleswar K. Panda, and Mohammad Banikazemi. "Evaluating the impact of RDMA on Storage I/O over Infiniband." SAN-03 Workshop (in conjunction with HPCA). 2004.

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

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

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

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

**[Lockwood2017] Lockwood, Glenn K., Wucherl Yoo, Suren Byna, Nicholas J. Wright, Shane Snyder, Kevin Harms, Zachary Nault, and Philip Carns, "UMAMI: a recipe for generating meaningful metrics through holistic I/O performance analysis", In Proceedings of the 2nd Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems 2017 (PDSW-DISCS '17). ACM, New York, NY, USA, 55-60.**

**[Lockwood2017b] Lockwood, G.K. , D. Hazen, Q. Koziol, S. Canon, K. Antypas, J. Balewski, N. Bathaser, W. Bhimji, J. Botts, J. Broughton, T. L. Butler, G. F. Butler, R. Cheema, C. S. Daley, T. Declerck, L. Gerhardt, W. E. Hurlbert, K. A. Kallback-Rose, S. Leak, J. Lee, R. Lee, J. Liu, K. Lozinskiy, D. Paul, Prabhat, C. Snavely, J. Srinivasan, T. Stone Gibbins, and N. J. Wright, "Storage 2020: A Vision for the Future of HPC Storage," LBNL-2001072, Berkeley, CA, 2017.**

**[Lockwood2018] Lockwood, Glenn K., Shane Snyder, George Brown, Kevin Harms, Philip Carns, Nicholas J. Wright. "TOKIO on ClusterStor: Connecting Standard Tools to Enable Holistic I/O Performance Analysis." In *Proceedings of the 2018 Cray User Group*. Stockholm, SE. May 2018.**

**[Lofstead2008] Lofstead, Jay F., et al. "Flexible IO and integration for scientific codes through the adaptable IO system (ADIOS)." In Proceedings of the 6th international workshop on Challenges of Large Applications in Distributed Environments. ACM, 2008.**

[Lofstead2014] G. F. "Lofstead, Q. Liu, J. Logan, Y. Tian, H. Abbasi, N. Podhorszki, J. Y. Choi, S. Klasky, R. Tchoua, R. A. Oldfield, M. Parashar, N. Samatova, K. Schwan, A. Shoshani, M. Wolf, K. Wu, W. Yu, "Hello ADIOS: The challenges and lessons of developing leadership class I/O frameworks." Concurrency and Computation: Practice and Experience 26.7 (2014): 1453-1473.

**[Lofstead2016] Lofstead, Jay, Ivo Jimenez, Carlos Maltzahn, Quincey Koziol, John Bent, and Eric Barton. 2016. DAOS and friends: a proposal for an exascale storage system. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (SC '16). IEEE Press, Piscataway, NJ, USA, Article 50, 12 pages.**

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

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

[Ludwig2007] Ludwig, Thomas, Stephan Krempel, Michael Kuhn, Julian Kunkel, and Christian Lohse. "Analysis of the MPI-IO optimization levels with the PIOViz Jumpshot enhancement." In Recent Advances in Parallel Virtual Machine and Message Passing Interface, pp. 213-222. Springer Berlin Heidelberg, 2007.

**[Luettgau2017] Luettgau J., Kunkel J. (2017) Simulation of Hierarchical Storage Systems for TCO and QoS. In: Kunkel J., Yokota R., Taufer M., Shalf J. (eds) High Performance Computing. ISC High Performance 2017. Lecture Notes in Computer Science, vol 10524.**

**[Luo2017] Luo, Xiaoqing *et al.*, "ScalaIOExtrap: Elastic I/O Tracing and Extrapolation," *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Orlando, FL, 2017, pp. 585-594.**

**[Lustre2010] Lustre-HSM, (http://wiki.lustre.org/images/4/4d/Lustre_hsm_seminar_lug10.pdf)**

[Luu2013] Luu, H, B. Behzad, R. Aydt, and M. Winslett. "A multi-level approach for understanding I/O activity in HPC applications." in IEEE International Conference on Cluster Computing (CLUSTER), Sept. 2013, pp. 1–5.

[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." In the 24th International ACM Symposium on High Performance Distributed Computing. June 2015 (to appear).

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

[Ma2009a] Ma, Kwan-Liu, Peter H. Beckman, and Kamil A. Iskra. "Visual characterization of I/O system behavior for high-end computing." High End Computing University Research Activity NSF 09-530 (2009)

[Ma2009b] Ma, Xiaosong, Frank Mueller, Kai Shen and Marianne Winslett. "Automatic extraction of parallel I/O benchmarks from HEC applications." High End Computing University Research Activity NSF 09-530 (2009)

**[MacArthur2017] Patrick MacArthur, Qian Liu, Robert D. Russell, Fabrice Mizero, Malathi Veeraraghavan, and John M. Dennis. An Integrated Tutorial on Verbs, Infiniband, and MPI. In the IEEE Communications Surveys & Tutorials, Vol. 19, No. 4, 2017.**

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

[Magoutis2003] Magoutis, Kostas, et al. "Making the most out of direct-access network attached storage." FAST. 2003.

**[Mandal2007] Nandita Mandal, Ewa Deelman, Gaurang Mehta, Mei-Hui Su, and Karan Vahi. Integrating existing scientific workflow systems: The Kepler/Pegasus example. In Proceedings of the 2nd Workshop on Workflows in Support of Large-scale Science, WORKS '07, pp. 21-28. ACM Press, 2007.**

**[Manzanares2016] A. Manzanares, N. Watkins, C. Guyot, D. LeMoal, C. Maltzahn, and Z. Bandic. ZEA, A Data Management Approach for SMR. In 8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 16), 2016.**

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

**[MDTest n.d.]**
**https://www.nersc.gov/users/computational-systems/cori/nersc-8-procurement/trinity-nersc-8-rfp/nersc-8-trinity-benchmarks/mdtest/**

**[Mesnier2003] Mike Mesnier, Gregory R Ganger, Erik Riedel. Object-based Storage. In IEEE Communications Magazine, Vol 41, 8, pp 84-90, 2003.**

**[Mesnier2007] Mesnier M.P., M. Wachs, R. R. Sambasivan, J. Lopez, J. Hendricks, G. R. Ganger, and D. O'Hallaron. "Trace: Parallel trace replay with approximate causal events." In Proceedings of the 5th USENIX Conference on**

**File and Storage Technologies, ser. FAST '07. Berkeley, CA, USENIX Association, 2007, pp. 24–24.**

**[Mikdadi2017] Mikdadi, D., "The Power of Provenance", DataScience@NIH, Published on 6 July 2017. URL: https://datascience.nih.gov/PowerofProvenance, accessed on August 31, 2018.**

**[Milenkovic2018] O. Milenkovic, R. Gabrys, H. M. Kiah and S. M. H. Tabatabaei Yazdi, "Exabytes in a Test Tube," in *IEEE Spectrum*, vol. 55, no. 5, pp. 40-45, May 2018.**

**[Miller2010] Ross Miller, Jason Hill, G. Raghul, G.M. Shipman, D. Maxwell. "Monitoring tools for large scale systems." CUG10, 2010.**

**[Miller2001] Miller, Ethan, Brandt, Scott A., Long, Darrell, "HerMES: High-performance reliable MRAM-enabled storage." In Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems, May 2001.**

**[Miller2015] Miller, M. C. *Design & Implementation of MACSio*. No. LLNL-TR-670388. Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2015.**

**[Moody2010] Moody, Adam, Greg Bronevetsky, Kathryn Mohror, and Bronis R. De Supinski. "Design, modeling, and evaluation of a scalable multi-level checkpointing system." In International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2010, pp. 1-11. IEEE, 2010.**

**[Moore2011] M. Moore, D. Bonnie, B. Ligon, M. Marshall, W. Ligon, N. Mills, E. Quarles, S. Sampson, S. Yang, and B. Wilson. OrangeFS: Advancing PVFS. FAST Poster Session, 2011.**

**[Mubarak2017] "Quantifying I/O and Communication Traffic Interference on Dragonfly Networks equipped with Burst Buffers", M. Mubarak, P. Carns, J. Jenkins et al. in 19th IEEE Cluster Conference, September 2017.**

**[Mubarak2017TPDS] "Enabling parallel simulation in large-scale HPC Network System co-design", Misbah Mubarak, Christopher D. Carothers, Robert B. Ross and Philip Carns in IEEE Transactions on Parallel and Distributed Systems (IEEE TPDS), 2017.**

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

**[Muniswamy-Reddy2006] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer. Provenance-aware storage systems. In Proceedings of the Annual Conference on USENIX '06 Annual Technical Conference, 2006.**

[Mysore2014] Mysore, Radhika Niranjan, et. al., "Gestalt: Fast, unified fault localization for networked systems." In 2014 USENIX Annual Technical Conference (USENIX ATC 14), pp. 255–267, 2014.

[NERSC Data n.d.] NERSC systems – operational and fault data http://pdsi.nersc.gov/

**[NERSC2015] MyNERSC gives users easier access to data, jobs, wait times. 2018. http://www.nersc.gov/news-publications/nersc-news/nersc-center-news/2015/mynersc-gives-users-easier-access-to-data-projects-wait-times/**

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

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

**[Nagle2004] D. Nagle, D. Serenyi, and A. Matthews. The Panasas ActiveScale storage cluster-delivering scalable high bandwidth storage. In the proceedings of the 2004 ACM/IEEE Conference on Supercomputing (SC '04).**

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

**[Ni2012] Ni, Xiang, Esteban Meneses, and Laxmikant V. Kalé. "Hiding checkpoint overhead in HPC applications with a semi-blocking algorithm." IEEE International Conference on Cluster Computing (CLUSTER), 2012. IEEE, 2012.**

[Nisar2008] Nisar, Arifa, Wei-keng Liao, and Alok Choudhary. "Scaling parallel I/O performance through I/O delegate and caching system." In International Conference for High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. pp. 1-12. IEEE, 2008.

**[Nugent 2015] P. Nugent, Y. Cao, and M. Kasliwal, "The Palomar transient factory," 2015, vol. 939702, no. February 2015, p. 939702.**

[Nunez2003] http://institute.lanl.gov/data/software/

**[Núñez2010] Núñez, Alberto, Javier Fernández, Jose Garcia, Félix Garcia, and Jesús Carretero. New Techniques for Simulating High Performance MPI Applications on Large Storage Networks. The Journal of Supercomputing, 51:40–57, 2010.**

**[ONF2018a] Open Networking Foundation. OpenFlow Protocol Specifications. https://www.opennetworking.org/technical-communities/areas/specification/open-datapath/ accessed on August 1, 2018.**

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

**[OPA2018] Intel, Inc. Intel Omni-Path Fabirc - the Next HPC Fabric. 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 on August 1, 2018.**

**[Oldfield2007] Ron A. Oldfield, Lee Ward, Arther B. Maccabe, and Patrick Widener. *Scalable security for MPP storage systems.* In International Conference on Security and Management: Special Session on Security in Supercomputing Clusters, Las Vegas, NV, July 2007.**

**[Ongaro2014] Diego Ongaro and John Ousterhout. In Search of an Understandable Consensus Algorithm. In the 2014 USENIX Annual Technical Conference, 2014.**

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

**[Organick2018] Lee Organick, Siena Dumas Ang, Yuan-Jyue Chen, Randolph Lopez, Sergey Yekhanin, Konstantin Makarychev, Miklos Racz, Govinda Kamath, Parikshit Gopalan, Bichlien Nguyen, Christopher Takahashi, Sharon Newman, Hsing-Yeh Parker, Cyrus Rashtchian, Kendall Stewart, Gagan Gupta, Robert Carlson, John Mulligan, Douglas Carmean, Georg Seelig, Luis Ceze, and Karin Strauss. Random Access in Large-scale DNA Data Storage. In the journal Nature Biotechnology, vol. 36, 2018.**

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

**[PFTool2018] PFTool: Parallel File Tool. https://github.com/pftool/pftool accessed on August 1, 2018.**

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

[Palmer2011] Palmer, Bruce, Annette Koontz, Karen Schuchardt, Ross Heikes, and David 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," in *Computing in Science & Engineering*, vol. 17, no. 4, pp. 52-62, July-Aug. 2015.**

[Parallel I/O Tutorial n.d.] Parallel I/O in practice http://sc14.supercomputing.org/program/tutorials

**[Parker-Wood2010] Parker-Wood, Aleatha, Christina Strong, Ethan L. Miller, and Darrell DE Long. "Security aware partitioning for efficient file system search." In IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), 2010, pp. 1-14. IEEE, 2010.**

**[Parker-Wood2013] Aleatha Parker-Wood, Darrell D. E. Long, Brian A. Madden, Ian F. Adams, Michael McThrow, and Avani Wildani. 2013. Examining extended and scientific metadata for scalable index designs. In Proceedings of the 6th International Systems and Storage Conference (SYSTOR '13). ACM, New York, NY, USA, , Article 4 , 6 pages. DOI: https://doi.org/10.1145/2485732.2485754. URL: https://www.ssrc.ucsc.edu/Papers/parkerwood-systor13.pdf**

**[Parkinson 2016] D. Y. Parkinson, K. Beattie, X. Chen, J. Correa, E. Dart, B. J. Daurer, J. R. Deslippe, A. Hexemer, H. Krishnan, A. A. Macdowell, F. R. N. C. Maia, S. Marchesini, H. A. Padmore, S. J. Patton, T. Perciano, J. A. Sethian, D. Shapiro, R. Stromsness, N. Tamura, B. L. Tierney, C. E. Tull, and D. Ushizima, "Real-time data-intensive computing," AIP Conf. Proc., vol. 1741, pp. 1–6, 2016.**

**[Patil2011] Swapnil Patil and Garth A. Gibson. "Scale and concurrency of GIGA+: File system directories with millions of files." In Proceedings of the USENIX Conference on File and Storage Technologies (FAST), pp. 177-190. USENIX, 2011.**

**[Patterson1989] Patterson, David A., et al. "Introduction to redundant arrays of inexpensive disks (RAID)." In Proceedings of the IEEE COMPCON, Vol. 89. 1989.**

[PDSI FSStats Data n.d.] http://www.pdsi-scidac.org/fsstats

[PDSW n.d.] Parallel Data Storage Workshop http://www.pdsw.org

**[Piernas2007] Piernas, Juan, Jarek Nieplocha, and Evan J. Felix. "Evaluation of active storage strategies for the Lustre parallel file system." In Proceedings of the 2007 ACM/IEEE conference on Supercomputing. ACM, 2007.**

[Pillai2014] Pillai, T.S., Chidabram, V., Alagappan, R., Al-Kiswany, S., Arpaci-Dussea, A.c., and Arpaci-Dusseau, R.H, "All file systems are not created equal: On the complexity of crafting crash-consistent applications." In The 11th Usenix Symposium on Operating System Design and Implementation (OSDI '14). Usenix Association, 2014.

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

**[Poremba2015] Poremba, M., T. Zhang and Y. Xie, "NVMain 2.0: A User-Friendly Memory Simulator to Model (Non-)Volatile Memory Systems," in *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 140-143, July-Dec. 1 2015.**

[PRObE n.d.] Parallel Reconfigurable Observational Environment http://www.nmc-probe.org/

**[Priedhorsky2017] Reid Priedhorsky and Tim Randles. Charliecloud: unprivileged containers for user-defined software stacks in HPC. In the Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17), 2017.**

**[Prometheus2018] Prometheus. https://prometheus.io/**

**[Qin2006] Qin, Lingjun, and Dan Feng. "Active storage framework for object-based storage device." 20th International Conference on Advanced Information Networking and Applications, Vol. 2. IEEE, 2006.**

**[Qin2009] X. Qin, 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] http://www.qmcpack.org/**

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

**[Rajachandrasekar2013] Rajachandrasekar, Raghunath, et al. "A 1 PB/s file system to checkpoint three million MPI tasks." In Proceedings of the 22nd international symposium on High-Performance Parallel and Distributed Computing. ACM, 2013.**

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

[Reed2004] Reed, Daniel A., ed. Scalable Input/Output: Achieving system balance. MIT Press, 2004.

**[Ren2014] Kai Ren, Qing Zheng, Swapnil Patil, and Garth Gibson. "IndexFS: Scaling file system metadata performance with stateless caching and bulk insertion." In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC14, Nov. 2014**

**[Rew1990] Rew, Russ, and Glenn Davis. "NetCDF: An interface for scientific data access." Computer Graphics and Applications, IEEE 10.4 (1990): 76-82.**

**[Riedel1997] Riedel, Erik, and Garth Gibson. "Active disks-remote execution for network-attached storage." No. CMU-CS-97-198. Carnegie-Mellon University Pittsburgh, PA, School of Computer Science, 1997.**

**[Rizzo1997] Rizzo, Luigi. "Effective erasure codes for reliable computer communication protocols." ACM SIGCOMM computer communication review 27.2 (1997): 24-36.**

[Ross2001] Ross, Robert, Daniel Nurmi, Albert Cheng, and Michael Zingale. "A case study in application I/O on Linux clusters." In Proceedings of the 2001 ACM/IEEE conference on Supercomputing, pp. 11-11. ACM, 2001.

**[Rumble2014] Stephen M. Rumble, Ankita Kejriwal, and John Ousterhout. Log-structured Memory for DRAM-based Storage. in the Proceedings of the12th USENIX Conference on File and Storage Technologies (FAST '14), 2014.**

[Sandia Data n.d.] Sandia application traces http://www.cs.sandia.gov/Scalable_IO/SNL_Trace_Data/

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

**[SC n.d.] Department of Energy Office of Science. "Statement on Digital Data Management." http://science.energy.gov/funding-opportunities/digital-data-management/**

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

**[SNIA2017] Storage Network Industry Association. SNIA NVM Programming Model 1.2. https://www.snia.org/sites/default/files/technical_work/final/NVMProgrammingModel_v1.2.pdf.**

**[SNIA2018] Storage Network Industry Association. Swordfish Scalable Storage Management API Specification. https://www.snia.org/sites/default/files/technical_work/Swordfish/Swordfish_v1.0.6_specification.pdf, May 25, 2018.**

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

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

**[Savoie2016] Lee Savoie, David K. Lowenthal, Bronis R. de Supinski, Tanzima Islam, Kathryn Mohror, Barry Rountree, and Martin Schulz, "I/O Aware Power Shifting," 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Chicago, IL, 2016, pp. 740-749.**

**[SciServer2018] SciServer - Collaborative data-driven science, http://www.sciserver.org, accessed on August 31, 2018.**

**[Schissel2014] D.P. Schissel, G. Abla, S.M. Flanagan, M. Greenwald, X. Lee, A. Romosan, A. Shoshani, J. Stillerman, J. Wright. Automated metadata, provenance cataloging and navigable interfaces: Ensuring the usefulness of extreme-scale data. Fusion Engineering and Design, Feb. 23, 2014.**

[Schmuck2002] Schmuck, Frank B., and Roger L. Haskin. "GPFS: A shared-disk file system for large computing clusters." In FAST 2 (2002):19.

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

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

**[SDA2018] Rambus Smart Data Acceleration Platform. https://www.rambus.com/emerging-solutions/smart-data-acceleration/ accessed August 1, 2018.**

[Seamons1994] Seamons, Kent E., and Marianne Winslett. "An efficient abstract interface for multidimensional array I/O." In Proceedings of Supercomputing'94, pp. 650-659. IEEE, 1994.

**[Sevilla2015] M. Sevilla, N. Watkins, C. Maltzahn, I. Nassi, S. Brandt, S. Weil, G. Farnum, S. Fineberg. Mantle: a programmable metadata load balancer for the ceph file system. In the Proceedings of SC15: The International Conference High Performance Computing, Networking, Storage and Analysis, 2015.**

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

**[Shamis2015] Pavel Shamis, et al. UCX: An Open Source Framework for HPC Network APIs and Beyond. In the 2015 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." In Proceedings of the 2008 ACM/IEEE conference on Supercomputing. IEEE Press, 2008.**

**[Shende2006] Shende, Sameer S., and Allen D. Malony. "The TAU parallel performance system." *The International Journal of High Performance Computing Applications* 20, no. 2 (2006): 287-311.**

**[Sierra2017] Sierra - High Performance Computing LLNL. https://hpc.llnl.gov/hardware/platforms/sierra accessed on August 1, 2018.**

**[Sigelman2010] Sigelman, Benjamin H., Luiz Andre Barroso, Mike Burrows, Pat Stephenson, Manoj Plakal, Donald Beaver, Saul Jaspan, and Chandan Shanbhag. *Dapper, a large-scale distributed systems tracing infrastructure*. Technical report, Google, Inc, 2010.**

**[Sjostrand2008] Sjöstrand, Torbjörn, Stephen Mrenna, and Peter Skands. "A brief introduction to PYTHIA 8.1." *Computer Physics Communications* 178.11 (2008): 852-867.**

**[Son2010] Son, Seung Woo, Samuel Lang, Philip Carns, Robert Ross, Rajeev Thakur, Berkin Ozisikyilmaz, Prabhat Kumar, Wei-Keng Liao, Alok Choudhary. "Enabling active storage on parallel I/O software stacks." In IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), 2010, pp. 1-12. IEEE, 2010.**

**[Soumagne2013] J. Soumagne, D. Kimpe, J. Zounmevo, M. Chaarawi, Q. Koziol, A. Afsahi, and R. Ross. Mercury: Enabling Remote Procedure Call for High-Performance Computing, IEEE International Conference on Cluster Computing, Sep 2013.**

**[Snyder2015] Snyder, Shane, Philip Carns, Robert Latham, Misbah Mubarak, Robert Ross, Christopher Carothers, Babak Behzad, Huong Vu Thanh Luu, Surendra Byna, and Prabhat. Techniques for Modeling Large-scale HPC I/O Workloads. International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS15).**

**[Snyder2016] Snyder, Shane, Philip Carns, Kevin Harms, Robert Ross, Glenn K. Lockwood, Nicholas J. Wright. Modular HPC I/O Characterization with Darshan. In *Proceedings of 5th Workshop on Extreme-scale Programming Tools (ESPT 2016),* 2016.**

**[Son2017] Son, S.W., Sehrish, S., Liao, W. et al. J Supercomput (2017) 73: 2069. https://doi.org/10.1007/s11227-016-1904-7**

[Spafford2012] Spafford, Kyle and Vetter, Jeffery S. "Aspen: A domain specific language for performance modeling." In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '12, pp. 84:1–84:11, 2012.

**[Spelman2018] Spelman, Lisa. "Reimagining the Data Center Memory and Storage Hierarchy." Editorial. May 2018.**

https://newsroom.intel.com/editorials/re-architecting-data-center-memory-storage-hierarchy/

**[STREAM2016] G. C. Fox, S. Jha, and L. Ramakrishnan, Eds., "STREAM2016: Streaming Requirements, Experience, Applications, and Middleware Workshop," 2015.**

[Subramoni2008] Subramoni, H.; Marsh, G.; Narravula, S.; Ping Lai; Panda, D.K. "Design and evaluation of benchmarks for financial applications using advanced message queuing protocol (AMQP) over InfiniBand." Workshop on High Performance Computational Finance, 2008., pp. 1, 8, 16. Nov. 2008.

**[Summit2017] Summit - Oak Ridge Leadership Computing. https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/ accessed on August 1, 2018.**

[Sun2014] Sun, Zhiwei, et al. "A lightweight data location service for nondeterministic exascale storage systems." ACM Transactions on Storage 10.3 (2014): 12.

[Tantisiriroj2011] Tantisiriroj, Wittawat, Seung Woo Son, Swapnil Patil, Samuel J. Lang, Garth Gibson, and Robert B. Ross. "On the duality of data-intensive file system design: reconciling HDFS and PVFS." In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, p. 67. ACM, 2011.

**[Thakur1999] Thakur, Rajeev, William Gropp, and Ewing Lusk. "Data sieving and collective I/O in ROMIO." Seventh Symposium on the Frontiers of Massively Parallel Computation, 1999. IEEE, 1999.**

**[Thapaliya2016] S. Thapaliya, P. Bangalore, J. Lofstead, K. Mohror and A. Moody, "Managing I/O Interference in a Shared Burst Buffer System," *2016 45th International Conference on Parallel Processing (ICPP)*, Philadelphia, PA, 2016, pp. 416-425.**

[Thereska2013] Thereska, Eno, et al. "Ioflow: A software-defined storage architecture." In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles. ACM, 2013.

[Thottethodi2006] Thottethodi, Mithuna S., Vijay S. Pai, Rahul T. Shah, T. N. Vijaykumar and Jeffrey S. Vitter. "Performance models and systems optimization for disk-bound applications." High End Computing University Research Activity NSF 06-503 (2006)

**[Titan2015] https://www.olcf.ornl.gov/titan/**

**[Trinity2015] Trinity - Los Alamos National Laboratory. https://www.lanl.gov/projects/trinity/ accessed on August 2, 2018.**

[Uselton2009] Uselton, Andrew. "Deploying server-side file system monitoring at NERSC." In Proceedings of the Cray Users Group meeting, 2009.

**[Uselton2010] Uselton, A., M. Howison, N. J. Wright, D. Skinner, N. Keen, J. Shalf, K. L. Karavanic, and L. Oliker, "Parallel I/O performance: From events to ensembles." In 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS). IEEE, 2010, pp. 1–11.**

[Vairavanathan2012] Emalayan Vairavanathan, Samer Al-Kiswany, Lauro Beltrão Costa, Zhao Zhang, Daniel S. Katz, Michael Wilde, Matei Ripeanu (2012): "A workflow-aware storage system: An opportunity study." In Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012).

[Vampir n.d.] https://www.vampir.eu/

**[VanEssen2012] B. Van Essen, R. Pearce, S. Ames and M. Gokhale, "On the Role of NVRAM in Data-intensive Architectures: An Evaluation," *2012 IEEE 26th International Parallel and Distributed Processing Symposium*, Shanghai, 2012, pp. 703-714.**

[Vavilapalli2013] V. K. Vavilapalli et al. "Apache Hadoop YARN: Yet another resource negotiator." In SoCC'13, Santa Clara, CA, October 1-3, 2013.

**[Vazhkudai2017] Sudharshan S. Vazhkudai, Ross Miller, Devesh Tiwari, Christopher Zimmer, Feiyi Wang, Sarp Oral, Raghul Gunasekaran, and Deryl Steinert. 2017. GUIDE: a scalable information directory service to collect, federate, and analyze logs for operational insights into a leadership HPC facility. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (SC '17). ACM, New York, NY, USA, Article 45, 12 pages.**

**[VeloC n.d.] http://veloc.readthedocs.io/en/latest/**

**[Venkataraman2011] Venkataraman, Shivaram, Niraj Tolia, Parthasarathy Ranganathan, and Roy H. Campbell. "Consistent and durable data Structures for non-volatile byte-addressable memory." In FAST, pp. 61-75. 2011.**

**[Vetter2016] Vetter, J. et al. Exascale Requirements Review for Advanced Scientific Computing Research, Advanced Scientific Computing Research, DOE Office of Science, Rockville, Maryland, September 27-29, 2016.**

**[Vincent2018] Vincent, Lionel and Gaël Goret. Self-Optimized Strategy for IO Accelerator Parametrization. In Proceedings of the Workshop On Performance and Scalability of Storage Systems (WOPSSS) 2018.**

**[Vijayakumar2009] Vijayakumar, K, F. Mueller, X. Ma, and P. C. Roth, "Scalable I/O tracing and analysis." In Proceedings of the 4th Annual Workshop on Petascale Data Storage, ser. PDSW '09. New York, NY, ACM, 2009, pp. 26–31**

**[Vishwanath2011a] Vishwanath, Venkatram, Mark Hereld, Vitali Morozov, and Michael E. Papka. "Topology-aware data movement and staging for I/O acceleration on Blue Gene/P supercomputing systems." In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, p. 19. ACM, 2011.**

[Vishwanath2011b] Vishwanath, Venkatram, Mark Hereld, Michael E. Papka. "Toward simulation-time data analysis and i/o acceleration on leadership-class systems." In IEEE Symposium on Large Data Analysis and Visualization (LDAV), 2011, pp. 9-14. IEEE, 2011.

**[Wachs2009] Matthew Wachs and Gregory R. Ganger. Co-scheduling of Disk Head Time in Cluster-based Storage. 28th International Symposium On Reliable Distributed Systems September 27-30, 2009.**

**[Wang2002] Wang, An-I, et al. "Conquest: Better performance through a disk/persistent-RAM hybrid files system." In Proceedings of USENIX Annual Technical Conference, 2002.**

**[Wang2016] Feiyi Wang, Veronica G. Vergara Larrea, Dustin Leverman, Sarp Oral. FCP: A Fast and Scalable Data Copy Tool for High Performance Parallel File Systems. In the Proceedings of the Cray Users' Group, 2016.**

[Watkins2013] N. Watkins, C. Maltzahn, S. Brandt, I. Pye, and A. Manzanares. In-vivo storage system development. In BigDataCloud '13 (in conjunction with EuroPar 2013), Aachen, Germany, August 26, 2013.

**[Watson1995] Watson, R.W.; Coyne, R.A., "The Parallel I/O Architecture of the High-Performance Storage System (HPSS)." In Proceedings of the Fourteenth IEEE Symposium on Mass Storage Systems, 1995: Storage - At the Forefront of Information Infrastructures, pp. 27, 44, Sept. 11-14, 1995.**

**[Weil2004] S. A. Weil, K. T. Pollack, S. A. Brandt, and E. L. Miller. Dynamic metadata management for petabyte-scale file systems. In SC'04, Pittsburgh, PA, Nov. 2004.**

**[Weil2006] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn. Ceph: A scalable, high-performance distributed file system. In Proceedings of the 2006 Symposium on Operating Systems Design and Implementation, pp. 307-320. University of California, Santa Cruz, 2006.**

**[Weil2007] S. A. Weil. "Ceph: Reliable, scalable, and high-performance distributed storage." Ph.D. thesis, University of California at Santa Cruz, December 2007.**

**[Welch2008], Brent, Marc Unangst, Zainul Abbasi, Garth A. Gibson, Brian Mueller, Jason Small, Jim Zelenka, Bin Zhou. "Scalable performance of the Panasas parallel file system." In FAST, vol. 8, pp. 1-17. 2008.**

**[Whitlock2011] B. Whitlock, J.M. Favre, J.S. Meredith, "Parallel In Situ Coupling of a Simulation with a Fully Featured Visualization System." In Eurographics Symposium on Parallel Graphics and Visualization, pp 101-109, 2011.**

[Wieczorek2009] Marek Wieczorek, Andreas Hoheisel, and Radu Prodan. 2009. 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, Dean 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, Advanced Scientific Computing Research and Basic Energy Sciences, DOE Office of Science, Rockville, Maryland, November 3-5, 2015.**

[Wolstencroft2013] Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams, David Withers, Stuart Owen, Stian Soiland-Reyes, Ian Dunlop, Aleksandra Nenadic, Paul Fisher, Jiten Bhagat, Khalid Belhajjame, Finn Bacall, Alex Hardisty, Abraham Nieva de la Hidalga, Maria P. Balcazar Vargas, Shoaib Sufi, and Carole Goble. "The Taverna workflow suite: Designing and executing workflows of Web Services on the desktop, web or in the cloud." Nucleic Acids Research 41(W1): W557-W561, 2013.

[Wozniak2010] Wozniak, Justin M., Bryan Jacobs, Rob Latham, Sam Lang, Seung Woo Son, and Robert Ross. "C-MPI: A DHT implementation for grid and HPC environments." Preprint ANL/MCS-P1746-0410 (2010): 04-2010.

[Wozniak2014] Justin M. Wozniak, Michael Wilde, Ian T. Foster "Language features for scalable distributed-memory dataflow computing." In Proceedings, Data-flow Execution Models for Extreme-scale Computing at PACT 2014

[Wu2009] Wu, Kesheng, Sean Ahern, E. Wes Bethel, Jacqueline Chen, Hank Childs, Estelle Cormier-Michel, Cameron Geddes et al. "FastBit: Interactively searching massive data." Journal of Physics: Conference Series, 180.1, p. 012053. IOP Publishing, 2009.

**[Wu2011] X. Wu and A. L. N. Reddy. SCMFS: A file system for Storage Class Memory. SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, 2011.**

**[Xie2017] Xie, Bing, Yezhou Huang, Jeffrey S. Chase, Jong Youl Choi, Scott Klasky, Jay F. Lofstead and Sarp Oral. "Predicting Output Performance of a Petascale Supercomputer."** ***HPDC*** **(2017).**

**[Xu2016] Jian Xu and Steven Swanson. NOVA: A Log-structured File System for Hybrid Volatile/Non-volatile Main Memories. in the Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST '16), 2016.**

**[TXu2016] Tianqi Xu, Kento Sato and Satoshi Matsuoka, "CloudBB: Scalable I/O Accelerator for Shared Cloud Storage," 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS), Wuhan, 2016, pp. 509-518.**

**[Yildiz2016] O. Yildiz, M. Dorier, S. Ibrahim, R. Ross and G. Antoniu, "On the Root Causes of Cross-Application I/O Interference in HPC Storage Systems," 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Chicago, IL, 2016, pp. 750-759.**

[Zadok2006] Zadok, Erez, Ethan L. Miller and Klaus Mueller. "File system tracing, replaying, profiling, and analysis on HEC systems." High End Computing University Research Activity NSF 06-503 (2006).

**[Zhang2010] Zhang, Yupu, et al. "End-to-end data integrity for file systems: A ZFS Case Study." In the 8th USENIX Conference on File and Storage Technologies, 2010.**

[Zhang2012] F. Zhang, C. Docan, M. Parashar, S. Klasky, N. Podhorszki, and H. Abbasi, "Enabling in-situ execution of coupled scientific workflow on multi-core platform." In Proceedings of the 26th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2012), Shanghai, China, pp. 1352-1363, May 2012.

**[Zhang2016] Huanchen Zhang, David G. Andersen, Andrew Pavlo, Michael Kaminsky, Lin Ma, and Rui Shen. Reducing the Storage Overhead of**

**Main-Memory OLTP Databases with Hybrid Indexes. In the proceedings of the 2016 International Conference on Management of Data (SIGMOD '16), 2016.**

**[Zhang2017] Zhang, Fan, 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, Ben 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, Yong, Mihael Hategan, Ben Clifford, Ian Foster, Gregor Von Laszewski, Veronika Nefedova, Ioan Raicu, Tiberiu Stef-Praun, Michael Wilde. "Swift: Fast, reliable, loosely coupled parallel computation." In IEEE Congress on Services, 2007, pp. 199-206. IEEE, 2007.

[Zhao2014] Zhao, D., Zhang, Z., Zhou, X., Li, T., Wang, K., Kimpe, D., Carns, P., Ross, R., Raicu, I. "FusionFS: Toward supporting data-intensive scientific applications on extreme-scale high-performance computing systems." In Proceedings of the IEEE International Conference on Big Data. 2014.

**[Zheng2014] Qing Zheng, Kai Ren, and Garth Gibson. BatchFS: Scaling the file system control plane with client-funded metadata servers. In Proceedings of the 9th Parallel Data Storage Workshop, PDSW '14, pages 1-6. IEEE Press, 2014.**

# 9 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 ORISE and Deneise Terry for managing the registration and logistics of the workshop series.