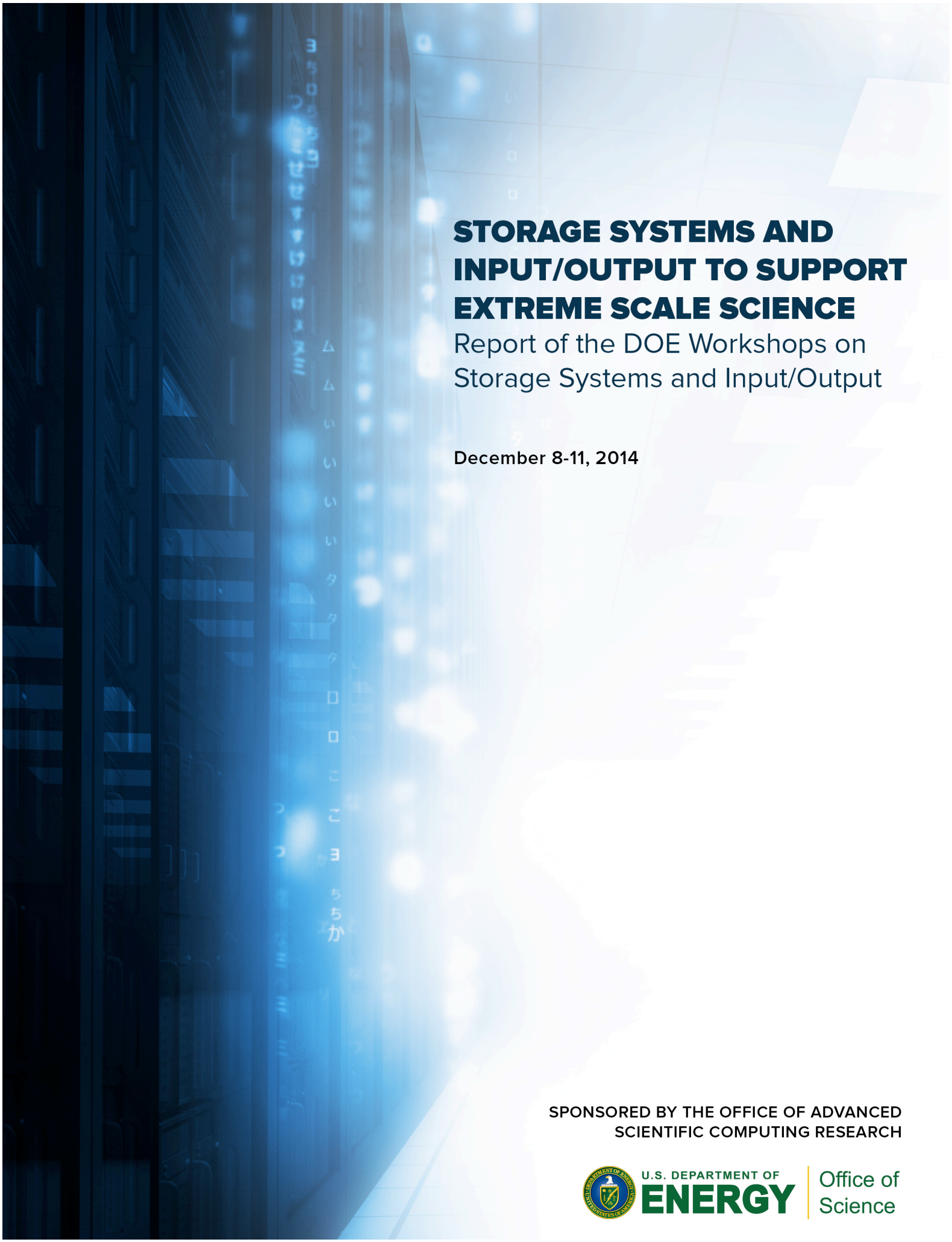

STORAGE SYSTEMS AND INPUT/OUTPUT TO SUPPORT EXTREME SCALE SCIENCE

Report of the DOE Workshops on Storage Systems and Input/Output

December 8-11, 2014

SPONSORED BY THE OFFICE OF ADVANCED SCIENTIFIC COMPUTING RESEARCH

U.S. DEPARTMENT OF ENERGY | Office of Science

Storage Systems and Input/Output to Support Extreme Scale Science

Report of the DOE Workshops on Storage Systems and Input/Output

Rockville, Maryland
December 8-11, 2014

Meeting Organizers

Robert Ross (ANL) (lead organizer)
Gary Grider (LANL)
Evan Felix (PNNL)
Mark Gary (LLNL)
Scott Klasky (ORNL)
Ron Oldfield (SNL)
Galen Shipman (LANL)
John Wu (LBNL)

Science/Mission Representatives

Salman Habib (HEP)
Rob Neely (ASC)
Varis Carey (ExaCT)
David Rogers (ASC)
Dave Richards (ExMatEx)
Michael Glass (ASC)
Dean Williams (Climate)

Crosscutting Computer Science Representatives

Pete Beckman (OS/Runtime)
Kerstin Kleese van Dam (Workflow)
Ian Foster (Collaboration Technologies)
Oliver Rubel (Analysis and Visualization)
Nathan Debardeleben (Resilience)
Maya Gokhale (Analytics and Workflow)
Jay Lofstead (LAN Networking and OS)

Computer Scientist Participants

Hasan Abbasi (ORNL)
Eric Barton (Intel)
Michael Bender (SUNY SB)
John Bent (EMC)
Suren Byna (LBNL)
Phil Carns (ANL)
John Chandy (UConn)
Matt Curry (SNL)
Bronis de Supinski (LLNL)
Garth Gibson (CMU)
Kevin Harms (ANL)
Quincey Koziol (The HDF Group)
Bradley Kuszmaul (MIT)
Wei-keng Liao (Northwestern)
Darrell Long (UCSC)
Carlos Maltzahn (UCSC)
Meghan McClelland (Seagate)
Ethan Miller (UCSC)
Adam Moody (LLNL)
Paul Nowoczynski (DDN)
Manish Parashar (Rutgers)
Narasimha Reddy (TAMU)
Brad Settlemyer (LANL)
Rajeev Thakur (ANL)
Sudharshan Vazhkudai (ORNL)
Lee Ward (SNL)
Brent Welch (Google)
Matt Wolf (GA Tech)
Cornell Wright (LANL)
Wenji Wu (FNAL)
Erez Zadok (SUNY SB)

Table of Contents

1 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 and input/output (I/O) technologies. In December 2014, a diverse group of domain and computer scientists from the Office of Science, the National Nuclear Security Administration, industry, and academia assembled in Rockville, Maryland, to review the storage system and input/output (SSIO) requirements for simulation-driven activities associated with DOE's science, energy, and national security missions and to assess the state of the art in key storage system and I/O areas. The activity was organized into three workshops.

The first workshop consisted of a series of talks from six DOE and NNSA application representatives, many of whom are engaged in exascale application co-design activities. The talks detailed the characteristics of the simulation and analysis tasks that the researchers expect to perform and the I/O characteristics of these tasks; the anticipated use of new storage technologies such as nonvolatile memory in accomplishing their science goals; the ways in which data is organized and searched and how the history of that data maintained; and the expected impact of issues such as increasing error rates and technologies such as compression. Following each talk the group discussed the key points raised and potential areas for future research and development.

The computing landscape is changing rapidly, and so the second workshop focused on how other computer science technologies (i.e., computer architecture, operating systems and runtimes, networking systems, workflow systems, data analysis and visualization algorithms, resilience techniques, and collaboration tools) will influence, and will be influenced by, future SSIO solutions. Seven talks by computer science experts in these fields identified the manner in which these systems are interrelated, and discussion with these experts further focused attention on key issues.

The information from these two workshops fed into the third workshop, during which SSIO community members engaged in open discussion on potential research directions in SSIO to support extreme-scale simulation-based DOE science. During five interactive sessions, the participants openly discussed the state of the art in specific SSIO technology areas and identified challenges and areas where additional research was needed in these areas.

The workshops generated specific findings that are further detailed in this report:

- ***In situ* data analysis is already an important component of many applications.** The question is not whether *in situ* analysis will play a role in future computational and data-intensive science but, rather, how this capability will be manifested.

- **The inclusion of solid state and new disk-based storage layers is dramatically complicating the storage hierarchy.** Standard methods of storage organization (e.g., parallel file systems, archival storage management systems) must significantly change, if not be replaced, to provide effective SSIO for future platforms.
- **To work productively, scientists need an integrated, coherent view of the storage resources at their disposal and a common method of managing and accessing data on these resources.** Meeting this need will require new metadata capabilities and integration with external storage in conjunction with improvements in SSIO architectures.
- **New requirements for public access to digital data required for validation of published results are poised to fundamentally change the role of metadata** in DOE Office of Science and NNSA mission-critical applications. These changes will mandate new approaches for capturing provenance and new methods for exploring extreme scale datasets.
- **The emerging use of alternative programming languages and task-based workflows drives the development of SSIO software.** Such software will need to be more flexible and to better integrate with upper layers in the software stack.
- **Scientists require increasingly complex and specialized data abstractions in order to improve their productivity and the quality of their science**. Significant improvements in SSIO data abstractions and their representations in the storage system are required to support these needs and to simplify upper layers of the stack.
- **Current SSIO designs are hindered by their isolation from system-level resource management, monitoring, and workflow systems.** Cooperation with these critical system services will be mandatory for the success of SSIO in future platforms.
- **Many important aspects of application and system behavior related to SSIO are obscured from view.** Recent successes in capturing application SSIO behavior have highlighted the value of this information for performance debugging, system procurement, and steering of SSIO research; but a better understanding of behavior is critical to SSIO effectiveness.
- **A key need for successful research and development in SSIO is a new and enhanced ecosystem**. This ecosystem must provide community access to rich sources of data on applications and systems, test environments in which new technologies can be evaluated, and investments that bring new talent into the community.

Four priority research directions emerged from this activity:

- In the area of SSIO architectures, additional research is needed to develop solutions to the challenge of managing upcoming deep and heterogeneous storage hierarchies, including storage in the compute system, and to explore alternative paradigms to the current file system model of access and organization.

- In the area of metadata, name spaces, and provenance, research is needed to devise new methods of capturing, organizing, presenting, and exploring rich metadata from DOE science activities, including breaking away from the current file model of data storage prevalent in DOE supercomputing facilities and science.
- In the area of supporting science data, research is needed to develop the next generation of I/O middleware and services in support of the broad collection of HPC and experimental and observational data needs and to integrate with and support new programming abstractions and workflow systems as they are adopted.
- In the area of understanding SSIO, research is needed to improve our ability to characterize the storage activities of DOE scientists and to model and predict the behavior of SSIO activities on future systems.

These findings and priorities are discussed in greater detail in Section 6.

2 Introduction

Computation and simulation advance knowledge in science, energy, and national security. The United States has been a leader in high-performance computing (HPC) for decades, and U.S. researchers greatly benefit from open access to advanced computing facilities, software, and programming tools. As HPC systems become more capable, computational scientists are now turning their attention to new challenges, including the certification of complex engineered systems such as new reactor designs and the analysis of climate mitigation alternatives such as carbon sequestration approaches. Problems of this type demonstrate a need for computing power 1,000 times greater than we have today; and the solution is exascale computing, the next milestone in HPC capability. At the same time, high-performance computing is playing an increasingly critical role in understanding experimental and observational data (EOD) from platforms such as the Large Hadron Collider, which is a key tool in better understanding fundamental questions in physics, and the upcoming Large Synoptic Survey Telescope, which when deployed will provide greater insight into the structure of the Universe. Achieving the power efficiency, reliability, and programmability goals for exascale HPC and for EOD will have dramatic impacts on computing at all scales, from personal computers (PCs) to mid-range computing and beyond; and the broader application of exascale computing can provide tremendous advantages for fundamental science and industrial competitiveness.

The disparity between the rate of increase in the compute capabilities of HPC systems compared with the rate of increase in storage bandwidths creates a widening gap between the capability of vendor storage options and HPC requirements, forcing the adoption of new storage technologies. The explosion in core counts resulting from changes in system architectures is dramatically increasing the number of parties that might concurrently interact with the storage system, mandating new ways of coordinating access. Additionally, the increase in data generation from DOE computational and experimental science applications will result in the need to manage complex data on the scale of tens to hundreds of petabytes or more, requiring new methods of organizing and finding data and of maintaining resiliency. Systems in the FY2017–2018 timeframe will be either hybrid multi-core or many-core systems with dozens to hundreds of homogeneous cores per node and will incorporate multiple levels of memory, including nonvolatile storage, blurring the demarcation point between storage and memory. Storage systems and I/O (SSIO) technologies must be developed and productized to meet science needs on these systems. Systems in the FY2020–2021 timeframe will have an order of magnitude more cores and possibly even deeper storage hierarchies, again mandating rapid development and productization of SSIO technologies.

The purpose of this report is to identify essential areas of research in SSIO technologies necessary to enable the next generation of DOE simulation and data-intensive science. 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.

This report stems from a December 2014 DOE workshop series that brought together application scientists from multiple disciplines; computer scientists from crosscutting areas of research, including data management, analytics and visualization, operating and runtime systems, computer architecture; and storage systems and I/O experts from the Office of Science, the National Nuclear Security Administration, industry, and academia. Representatives from DOE supercomputing facilities also participated.

The first workshop focused on application use cases and critical SSIO requirements for achieving DOE mission goals on future extreme-scale platforms. The second workshop focused on critical SSIO requirements and points for coordination between SSIO and other computer science areas related to extreme-scale DOE systems. The information from these first two workshops informed the third workshop, during which SSIO community members engaged in open discussion on potential research directions in SSIO to support extreme-scale DOE science. The discussion focused on simulation-based science, but the results of the discussion could also be applicable to experimental and observational data (EOD) analysis when this analysis is performed on large scale computing systems. Additional study is warranted to better understand EOD needs.

The report is organized as follows. Section 3 summarizes the DOE mission requirements for SSIO. Section 4 discusses in depth the state of the art, challenges, and required research and development needed to advance SSIO technologies in support of DOE mission goals. Section 5 captures workshop discussion related to other support that is needed for the SSIO R&D community to be most effective.

3 Mission Drivers

To better understand the application requirements, 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 and the SciDAC co-design centers,[1] which are charged to ensure that future architectures are well suited for DOE target applications. These centers contain the combined expertise of vendors, hardware architects, system software developers, domain scientists, computer scientists, and applied mathematicians. Their work is at the forefront of anticipating features and tradeoffs in the exascale hardware, software, and underlying algorithms.

This report also incorporates extensive knowledge from the organizing committee on numerous applications from fusion energy, materials science, climate science, accelerator physics, and other domains.

3.1 Overview

The application drivers 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 them as data-intensive science. In this section we give a quick rundown of their characteristics in terms of the common nomenclature of the five Vs: volume, velocity, variety, veracity, and value.

Volume. Much of the total volume from the HPC applications comes from checkpoint files. Most of this information is written once, and almost never read in. With the inclusion of "burst buffers" we envision that most of the applications will be able 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 current OLCF platform. Because of the lack of storage and because of the time needed to write the amount of data, the simulation will be able to write only about 10 PB. This may lead scientists to miss important artifacts and miss

[1] http://science.energy.gov/ascr/research/scidac/co-design/

opportunity for discovery. Clearly new research is necessary to handle volumes of data this large.

Velocity. Higher velocities (i.e., rates of data generation) will be seen from many leading applications because of the nature of 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 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, and this data needs to be processed, reduced, and stored for later postprocessing. Similar requirements are expected in EOD based science.

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 the large amounts of small data will become increasingly important, including the ability to 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 nonintrusive 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 calibration moves away from the experimental test base.

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 research into new storage tiers that keep more data readily available, retain the provenance of the data, and understand what the different variables may contain will allow the SSIO software system to better manage application data.

3.2 Workload Characteristics

The first challenge facing these large simulation codes is that many of them are monolithic programs designed for architectures of past decades. Some of these codes will likely undergo significant change in order to transition to future architectures. Much of the rewrite will be to add additional functionality to make effective use of hundreds to thousands of cores per node, different types of memory that are evolving in time, and the effective use of data-coupling techniques for uncertainty quantification, code coupling, *in situ* and in-transit analysis and visualization, and different types of data reorganization and data compression techniques for making effective use of postprocessing techniques. Codes used in critical safety- and security-related applications have to go through formal verification and validation procedures that require additional time in the development cycle.

We first observe some common characteristics of anticipated simulation workloads and then dive into a specific application use-case to illustrate some key features of the I/O requirements.

Common observations
In Flynn's taxonomy for classifying computer programming paradigms [Flynn2011], the monolithic codes mentioned above 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 are expressed as separate executables and may each be handling a different aspect of a complex model, such as land, ocean, ice, and atmosphere in a global climate model, or 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 or multiple simulation programs plus a set of analysis routines that compare the outputs from the simulations, or simulations being compared with experimental observations. In many of these cases, there is a stringent demand on communicating a large amount of data from one part of a large parallel job to another. These large parallel simulations will be producing a large amount of data to be stored persistently. Additionally, they may also read a large amount of input data, representing the initial condition required at the start of a simulation, representing the boundary conditions needed at every step of the simulation, for comparison against EOD, or other purposes.

The application scientists mentioned a number of characteristics of these MIMD programs; we briefly highlight three:

- **Homogeneous tasks in MIMD**. The quintessential example of this type 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 of MIMD with homogeneous tasks is an ensemble

run of climate models where each instance of the ensemble uses a different model. Of course, 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 MIMD program may need to be run as persistent services. For example, a number of simulation programs involve complex materials, and a large number of chemical reactions and information about these chemistry processes could best be captured in an equation-of-state service. Existing large-scale simulation programs are run in batch mode, where all executables are terminated when the batch job terminates. These persistent services need to last beyond the end of any single batch job. Supporting these long-running services require supercomputer centers to change their mode of operations. Such a change would also benefit long-running data analysis services.
- **Composite workflows**. A MIMD job composed of heterogeneous tasks is a composite workflow. The approach of connecting different tasks into a larger structure has been used extensively for large-scale distributed data analysis, but it has not been widely used for composing parallel simulations. Considerable work will be needed to develop the workflow composition, scheduling, and execution tools. 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.

An example: Adjoint-based sensitivity analysis

To examine the I/O operations in more detail, we next consider a UQ use case, 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 has 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.

3.3 Input/Output Characteristics

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

Most of the applications require a modest amount of input data at the beginning of the simulation run along with the 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, the 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 the experimental/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. Altogether, the datasets collected from the experiments increase and the data passed to the simulations will also grow.

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.

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. 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 the checkpoint files contain all the information necessary to regenerate the whole state of the simulation, while the productive analysis output needs only to summarize key features of the simulation, the checkpoint files are generally larger than the productive output files. 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]. The application scientists also adjust the frequency of checkpointing to limit the I/O time to a relatively small fraction of the total execution time. Most existing simulation codes perform their checkpointing operations by directly writing data to files, instead of using a checkpointing library.

As parallel computers grow in size, there is considerable interest in moving away from global checkpointing. Hybrid checkpointing schemes, such as the Scalable Checkpoint/Restart (SCR) library [Moody2010], are gaining acceptance among application scientists.

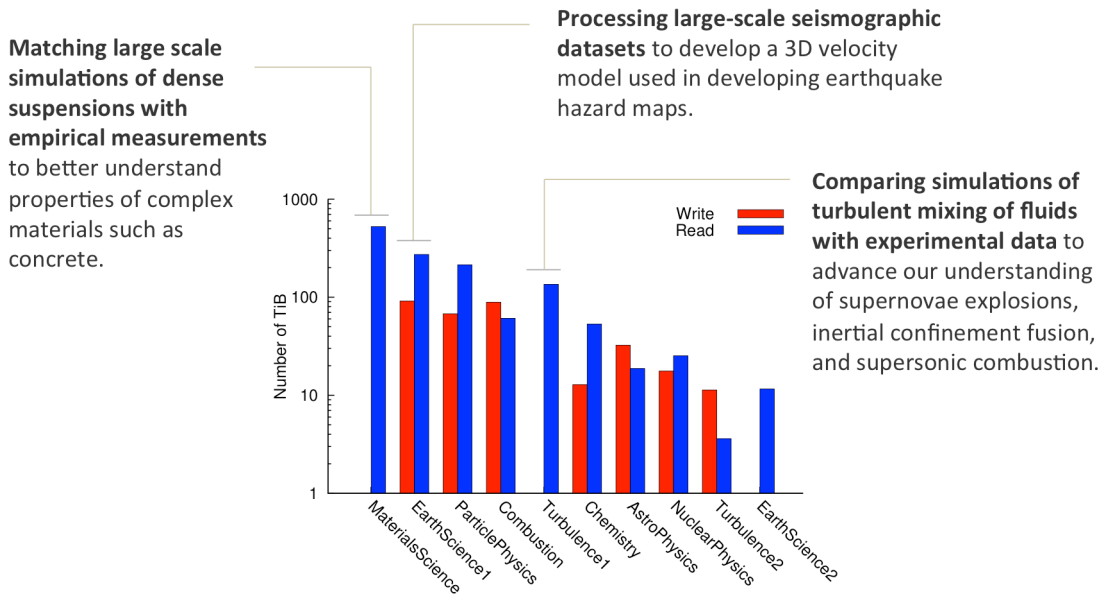

Figure 1: Applications on large-scale DOE platforms can consume great quantities of data. In July 2011, many of the top producers/consumers of data on the Intrepid Blue Gene/P system at Argonne had I/O patterns dominated by reading.

Machines that will be delivered in the 2017-2018 timeframe will have nonvolatile memory (NVRAM) that could be used for storing the checkpoint files. NVRAM could allow the checkpoint data to be written 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. Since NVRAM is not yet available at large-scale, however, the user community is somewhat unsure how to best make use of NVRAM.

When multiple tasks in a parallel job need to 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.

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 number of bytes 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.

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

Many large-scale scientific simulations routinely write out immense amounts of data on today's HPC systems, such as in the case of XGC1 writing 100 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 post-processing, 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 leery 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.). There are funded projects that are working to address these issues, such as by using task-based parallelism and 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 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 will 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 simulations is becoming a more pressing issue on exascale machines.

Today several *in situ* visualization and analysis services are being used in applications. ADIOS [Liu2014] 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.

Finding: In situ data analysis is already an important component of many applications. The question is not whether in situ analysis will play a role in future computational and data-intensive science but, rather, how this capability will be manifested.

3.5 Data Organization and Archiving

Many application programs running on the current generation of supercomputers are still outputting their 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]. As one scientist noted, "Some codes use HDF5, others use a custom data format for performance reasons. Event simulation uses relational databases. Other data organizations are being investigated, catering to the type of scientific information to be represented."

This diversity of data organizations and needs creates challenges to our community, which 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? The application teams commonly articulate that I/O needs to include different application-specific forms of compression as part of the I/O routines themselves. *In*

situ (asynchronous) techniques are often 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 to the file system. Research is necessary to ensure that techniques addressing these issues are in the next-generation I/O libraries.

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 could 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, 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 "swim lanes" for future architectures. 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 a lossy compression is used, it is highly desirable to be able to provide users with ways to quantify the loss introduced by compression. 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. These techniques must be very fast in order to keep up with the high data velocities being presented. For example, in the QMCPack example, 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 and future 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. Compression 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]. The data on disk are often considered on-line because they can be accessed with common I/O libraries, whereas the data in tertiary storage are considered off-line because they have to go through an extensive data transfer process before they are usable by a data analysis program. Typically, on-line data are available in milliseconds, while off-line data may require weeks to become available. Such a gap is a tremendous barrier for users to access the data in tertiary storage and contributes to a failure to analyze data that are 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 off-line storage. Such a system would increase scientific productivity by allowing the scientists to have access to a larger amount of data for a longer period of time even though the access might be somewhat slower. 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

layers of storage. New research in data reorganization must be encouraged in order for applications to take advantage of these features.

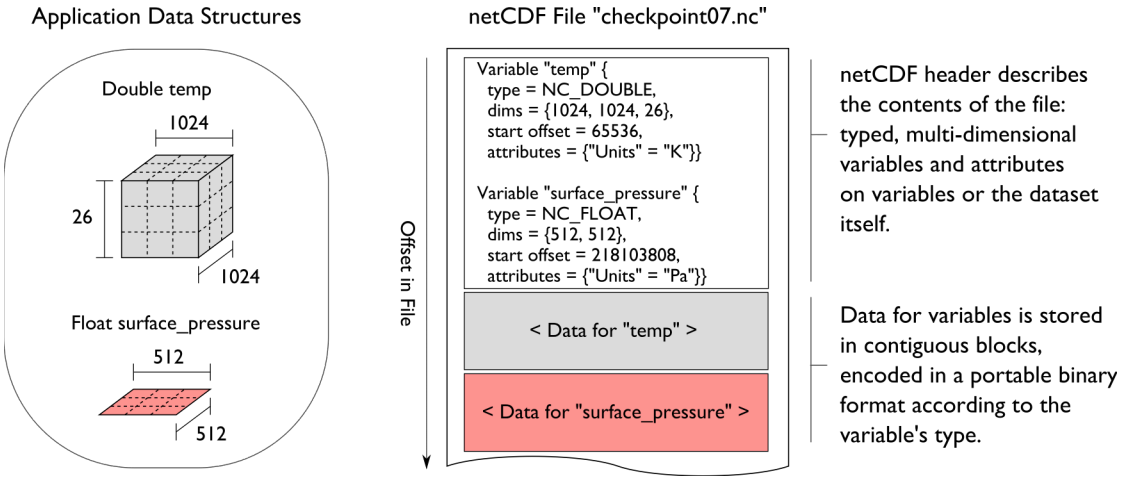

Figure 2: Self-describing file formats, such as the netCDF format shown here, capture not only array data but also structural information such as names, units, types, and dimensions.

3.6 Metadata and Provenance

Metadata is commonly divided into two broad categories: *structural metadata*, which concerns the design and specification of the data structure, and descriptive *metadata*, which comprises all other associated information such as creator, meaning, intended uses, provenance, associations, and context. Historically, such metadata is 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 the scientific data. These contain not only the arrays of the 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 that drive a graph, to be accessible in order to support validation of results [SC n.d.]. This policy has been translated into a number of strong 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 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. In

certain highly regulated computing environments, it may be possible to require 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 the current data management systems, when data files are moved, the associated metadata is often lost.
- The system should accurately capture information about the programming environment and the runtime environment.
- The information captured must be easily searchable or otherwise accessible.
- The system must provide useful feedback about errors and faults. Furthermore, such feedback should be instructive in helping users recover from the errors.

3.7 Summary

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

1. Fast data access is essential to large-scale 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 technology along with other next generation NVRAM. (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 more workflow technologies are integrated into applications, capturing provenance becomes critical for future understanding of what occurred before and after the simulations. (Veracity, value)
4. Research in *in situ* data frameworks is needed that can place tasks (on node, off-node, and on external resources) based on user intentions and resource availability, and this system should accept plug-ins that enable specialization by application scientists. (Velocity, volume, variety, value)

4 Computer Science Challenges

4.1 Hardware/Software Architectures

Architectural changes coming in post-petascale systems raise new challenges for effective SSIO hardware and software. New network technologies and topologies create a new environment in which this software must operate. The inclusion of solid-state storage in systems adds another layer to an ever-deepening storage hierarchy. The introduction of computational capabilities within this hierarchy will open additional challenges. Properties such as voltage scaling and shared network links will create a "noisier" environment in which SSIO systems must reside, and device reliability raises questions about long term data retention.

Findings

The inclusion of solid state and new disk-based storage layers is dramatically complicating the storage hierarchy. Standard methods of storage organization (e.g., parallel file systems, archival storage management systems) must significantly change, if not be replaced, to provide effective SSIO for future platforms.

To work productively, scientists need an integrated, coherent view of the storage resources at their disposal and a common method of managing and accessing data on these resources. Meeting this need will require new metadata capabilities (Section 4.2) and integration with external storage (Section 4.4) in conjunction with improvements in SSIO architectures.

4.1.1 Networks

State of the Art

Most modern HPC networks provide a wealth of 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. In order to scale to tens or even hundreds of thousands of endpoints, these networks employ a variety of topologies such as Clos [Clos1953], mesh [Felderman1994], 3-D torus, butterfly, and variants thereof [Scott2006], [Kim2008].

Some advanced network features have been investigated in the context of SSIO, RMA [Magoutis2003, Liu2004] in particular. QoS [Chuang1999] and request routing [Anderson2000] have been explored to some degree, focusing primarily on commodity networks. A significant amount of funding has been invested in QoS in the past as part of HEC-FSIO activities [HEC-FSIO2011], some of which focused on network based QoS. To date, however, no distributed QoS capability has been demonstrated. Network and storage node topology has been explored in wide-area distributed storage systems [Beck2002], within HPC systems [Dillow2011] and more generally in large-scale datacenters [Thereska2013]. Significant research also

has been conducted in adaptive and dispersive routing that, while not storage specific, is applicable to SSIO.

Challenges

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

- QoS via throttling, performance isolation, and co-scheduling with pre-emption
- 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
- Resource management capabilities for resource sharing and resource isolation

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

R&D Needed

Research is needed to develop QoS mechanisms that integrate QoS capabilities within the network with future SSIO architectures. Mechanisms to dynamically provision network resources for differing SSIO tasks such as bulk data movement, collective communications, and fault notification should be explored. Since fault notification and response protocols such as quorum and gossip are often needed for SSIO, optimal mapping of these protocols to dynamic and reconfigurable network topologies is needed. As SSIO continues to explore the use of distributed transactions, algorithms should be explored that support distributed transactions that leverage advanced network features such as AMOs. Some of this work will be crosscutting, most notably with OS/runtime and resource management topical areas.

Future networks coupled with OS/runtime advances may provide new features such as active message capabilities allowing computation to be scheduled via the network. Furthermore, future networks may enable embedding of computation directly within the network for tasks such as collective reductions. Approaches for utilizing these capabilities within the SSIO environment for active storage and data reorganization (in transit) should be explored.

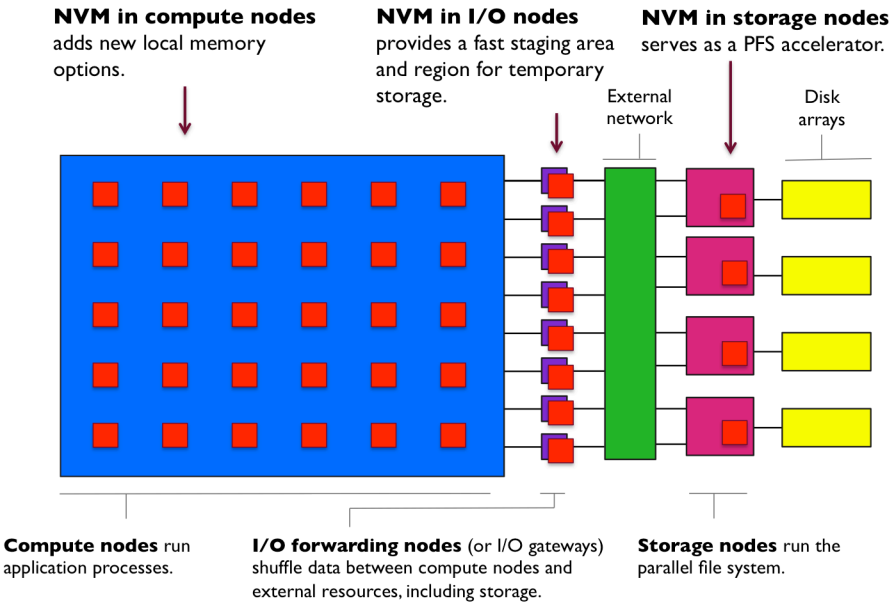

Figure 3: Nonvolatile memory (NVM) will be an important component of future SSIO architectures. Placement of NVM in the system influences the utility of the NVM for specific use cases, and additional research is needed to best understand where and how to integrate this technology to provide the greatest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

4.1.2 Deep Storage Hierarchies and Nonvolatile Memory

State of the Art

Today's state-of-the-art in SSIO includes storage hierarchies that can span multiple system memories [Isaila2011], node-local or I/O node (ION) local SSDs [Caulfield2009], dedicated scratch storage addressable on the HPC platform, global storage addressable across the HPC datacenter, and long-term archival storage addressable across the HPC datacenter. Each layer provides a different set of cost, access rate, capacity, power, and resilience characteristics; and facility teams endeavor to match the overall hierarchy to their projected workloads when they develop specifications for a new system or an upgrade.

While node-local and ION local SSDs are still in their infancy in terms of adoption, most future systems, as evidenced in the proposed FY2017-FY2018 procurements, will incorporate this addition to the storage hierarchy [Bent2012, Liu2012a] because of economic constraints on employing solely disk-based solutions to meet performance requirements on next-generation leadership HPC systems. With the exception of system memories and dedicated scratch storage, these hierarchies are generally managed as explicitly independent resources at large-scale HPC facilities. Client-side caching and extensions to remote compute and ION caching have been widely explored [Thakur1999, Liao2007, Isaila2011, Lofstead2008, Abbasi2010, Qin2009].

As SSDs have become more commonplace, a number of research efforts [Moody2010, Ni2012, Rajachandrasekar2013, Barton2014] have begun exploring the use of this hierarchy of storage within HPC systems. Next-generation NVRAM technologies such as phase change memory and memristor will present yet another layer within the hierarchy with semantics akin to traditional DRAM; preliminary research has begun in this area to assess their impact on file system design [Miller2001, Wang2002] and their use as caches [Liu2012b, Kannan2011a] or for staging and checkpointing [Kannan2011b, Kannan2013], and upcoming systems will incorporate NVRAM into their design (e.g., Cori [Dosanjh2014]). Some preliminary research has been conducted in data location services within deep and potentially geo-distributed storage systems [Sun2014].

Challenges

Significant challenges in this area exist, with the deepening of the storage hierarchy and performance characteristics of next-generation storage technologies. Only a few researchers [Barton2014, Goodell2012, Brinkmann2014] have investigated how best to expose this deepening storage hierarchy. 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. Alternative, possibly transactional, object storage interfaces to replace POSIX need further refinement and community exploration.

One difficulty in creating a new storage interface for future hierarchical storage is understanding what the workload requirements of exascale applications will look like. For example, will bulk synchrony at the application level survive? If so, the entire storage stack may not need radical changes. If applications become increasingly asynchronous as expected, however, they may require new mechanisms in order to manage consistent views of distributed data in the hierarchy.

R&D Needed

Since the storage hierarchy will be utilized for different purposes and competed for by multiple consumers, adaptive resource management mechanisms and policies will be needed. Performance optimization techniques across the hierarchy will need to be explored. While some studies are under way and early products are emerging, additional research in how to expose the hierarchy of storage is required to identify what should be visible to the user, how (and if) the hierarchy should be described to the user (exposition), how users should describe data organization across the hierarchy, and how compositions of the hierarchy are effected. This research has significant overlap with workflow, since workflow systems likely can assist in managing the complexity of data placement and movement within the storage hierarchy. Other challenges, such as enforcing consistency across the hierarchy and dealing with differing units of atomicity, will require new mechanisms, perhaps transactional mechanisms. Integration of computation within the storage hierarchy will require new techniques to expose locality of data within the hierarchy, the

computational resources co-resident with the data, and interfaces to allow computation on this data within the hierarchy.

Moving data efficiently and transparently between data tiers presents a challenge. Knowing where the data should be and getting it there at the right time are both areas that need research. Evolution of existing HSM tools to cater to deep storage hierarchies should be explored; but other, more revolutionary solutions also merit investigation. Methods for incorporating guidance from the user (guided interfaces) are a promising direction that possibly could affect many aspects of management in the hierarchy, including retention policies, reliability requirements, QoS, and data temperature.

In addition to the R&D needed for the general category of deepening storage hierarchies, significant discussion focused on R&D needed specifically for burst buffers. The burst buffer is a new tier of storage within the hierarchy that fills the widening gap of performance between DRAM and traditional magnetic disk drives. The burst buffer will be smaller in capacity than the file system it sits in front of, but it will provide significantly higher bandwidth. The use of burst buffers spans a number of use cases, from defensive and productive I/O to active storage/programmable storage; and care must be taken in the design of burst buffer systems to retain flexibility in how this resource might be used in future systems.

Of particular interest are data models and programming abstractions for burst buffers. Potential interfaces to the burst buffer include transactional interfaces to object storage, traditional file system interfaces, memory abstractions, and durable data structures that enable traditional data structures to be efficiently implemented on persistent memory devices [Venkataraman2011]. R&D is needed in order to understand the tradeoffs of differing interfaces to the burst buffer. System design alternatives need further exploration in coordination with evaluation of deployments, including answering fundamental questions such as where burst buffers will reside within the HPC system (node local, ION dedicated, or separate nodes within the compute area); whether burst buffer resources will be time or space shared among multiple consumers; and how they will be scheduled/allocated including potential time for data draining to a parallel file system or elsewhere on job completion. Better understanding of device characteristics will be required in order to design the most effective system architecture and proper data/programming models for these systems.

4.1.3 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, Arpaci-Dusseau1999, Amiri2000, Son2010]. The current state of the art includes the extension of active storage concepts to the device level (T10 OSD) [Qin2006] 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). Addressing these challenges will open the opportunity to take advantage of new storage media (NVRAM) with increasing computational capabilities (either embedded or coresident) while significantly reducing data movement within future HPC systems.

R&D Needed

Several R&D topics will need to be addressed to make active storage a viable paradigm in future systems. Research is needed in order to understand application use cases relevant to DOE mission, such as data reductions (filtering, subsetting, querying, etc.), data transformations (sorting, mapping, etc.), and ephemeral views, along with their relative tradeoffs in terms of communication and computation. Also needed is better understanding of current device characteristics and capabilities and those needed in the future to support these use cases. Supporting these use cases on next-generation devices also will require advances in data and programming models, allowing the application to describe the computation and its mapping to a high-level data structure. R&D in interfaces with an eye to reusability and portability over time will be required in order to facilitate adoption by a broad set of application communities.

Resource management challenges must be also addressed, since storage resources will incorporate a new dimension of “active” traffic (i.e., computation being performed on storage resources) alongside the traditional “passive” traffic. QoS mechanisms will need to advance to include computational quality of service alongside traditional data movement. Providing a remote execution capability to applications will raise new security challenges (e.g., enforcing security policies in

the context of code running in the storage system). Moreover, workflow will crosscut this area as related issues of data locality, computational placement, and resource scheduling are addressed.

4.1.4 Resilience

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, quorum protocols, and multiple forms of data replication. 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 and Legion, and the application of the CAP theorem and peer-to-peer systems principles are also gaining momentum and adoption within the scientific HPC community.

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. Deepening storage hierarchies, new storage media, increasing storage capacity, and tighter margins in component designs (silent data corruption) will necessitate significant R&D in SSIO resiliency to account for, and take advantage of, device and data properties when providing resiliency of persistent data within the hierarchy and to provide appropriate performance for access to data in the presence of interfering traffic. This latter issue must be addressed in successful hierarchical storage designs.

R&D Needed

A number of research topics will need to be explored in the area of SSIO resiliency. Fault detection, fault communication, and recovery strategies in large-scale distributed storage environments are needed, with particular focus on low-overhead approaches that scale to tens of thousands of endpoints while tolerating jitter. With the advent of new storage technologies, resiliency strategies to individual component failures warrant exploration. Since many application-level resiliency strategies will rely on SSIO, low-overhead scalable techniques for application-level checkpointing strategies should be explored. Arguably, a completely resilient SSIO environment in the face of all failure scenarios will be

impossible; and the costs of increasing levels of resiliency may prove prohibitive for some environments. Thus, resiliency techniques that trade space, bandwidth, and other overheads for differing levels of resiliency should be explored. Application-level interactions with this environment may also be important, enabling the application to assist in making these tradeoffs. Similarly, mechanisms for containment and communication of failures to the application could allow application-level resilience strategies to be employed (recomputation of partial data is an example), but additional SSIO research is needed in order to understand how such mechanisms can work cooperatively with storage software to meet resilience requirements.

4.1.5 Understandability

State of the Art

Understanding SSIO systems in general is covered in Section 4.5. Here we focus on the issue of understanding the behavior of SSIO architectures and how one might architect these systems for better understandability.

The state of the art in this area involves various storage components providing independent logging of what the developers considered to be relevant or critical events. These log entries may be integrated into a systemwide RAS database, but more likely they are accessible with proprietary tools. Continuous monitoring tools are also available for some storage architecture components, such as disk controllers [Kim2014] and file system servers [Uselton2009]. More recently, efforts have been made to include limited shared storage system statistics as a component of system-level monitoring tools [Agelastos2014]. In addition to logs, PLFS [Bent 2009] creates maps that are helpful in identifying actual user IO patterns.

In the past, efforts were made to integrate the logging of user behavior across multiple layers of the SSIO system [Ludwig2007], but this system was not productized, and the approach was not adopted in other systems. More recently, a technique for correlating fine-grained events across the storage architecture was developed for visualization purposes [Muelder2011]; however, the technique is too invasive for routine production use. Another approach [Hammond2011] looked at rationalizing log messages to make them more analyzable in the context of Lustre file systems.

In the area of tools for designing understandable SSIO systems, a project by Remzi and Andrea Arpaci-Dusseau at Wisconsin began to address correctness and formal failure models for SSIO systems [Arpaci-Dusseau2006]. This work has most recently been extended to examine the consistency of file systems in the face of system crashes, including the development of two tools for verifying file system resilience and correctness following a crash [Pillai2014]. Separately, a state machine language called Aesop has been of some assistance in enabling correct behavior in storage systems [Kimpe2012]. Warranting investigation is the treatment of SSIO systems development as a first-class application needing its own tools and languages.

A number of methods have been developed to express HPC computer system models and topologies for purposes such as optimization [Broquedis2010], performance modeling [Spafford2012], or root cause analysis [Mysore2014], but analogous techniques have not been adopted for use in SSIO storage architectures.

Challenges

Several major challenges exist in our ability to understand current SSIO architectures. First, information coming from multiple layers in the SSIO architecture stack is not coordinated or integrated, meaning that a holistic view of how the architectural components work together is not available. Second, no common abstract model is shared across SSIO system implementations, precluding the development of tools to interrogate behavior that are portable across multiple environments. Third, storage systems do not expose information about how individual applications utilize the storage architecture (e.g., what components were used and to what degree). This lack of information makes identifying bottlenecks or gauging the impact of storage system design tradeoffs difficult.

R&D Needed

Research is needed to develop rich abstract storage models that capture the relevant properties of future SSIO architectures. These models must incorporate notions of locality, multiple tiers, as well as traditional properties such as MTBF, capacity, and bandwidth. Methods of presenting views of current system state using these models would be helpful as a first step in better understanding SSIO architectures, but more quantitative tools for analysis are also needed. The development of abstract SSIO architecture models also has the potential to affect crosscutting topics such as storage system simulation, root cause analysis, and autonomic tuning.

Approaches for integrating information from multiple subsystems into coherent "views" (i.e., human-understandable representations) must be developed, including methods for capturing the accuracy of this information (e.g., is this an estimated value for current time or an accurate value from two minutes ago?). Multiple views are needed to address the needs of different consumers of this information, including administrators, developers debugging the system, and scientists who want to use this information to adjust their applications' behavior. Human readability and understandability are critical to the success of these approaches.

Additionally, research is needed to realize the potential of integrated views of user activity. This work may require and benefit from collaboration with researchers in other areas of performance understanding on future systems and could be of value in the development of autonomic SSIO solutions.

4.1.6 Autonomics

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.

Despite a great deal of research into autonomics in the context of storage generally, truly "autonomic" storage systems have not emerged in HPC. Instead, limited adaptation is present in most parallel file systems in order to recover from failed components, and some middleware responds to slow servers by shifting data placement to use faster servers. Accurately representing HPC I/O workloads can assist with enabling autonomics; however, this work is also in its infancy.

Challenges

The increasing complexity of SSIO systems means that practically no deployments are operating in an optimal configuration, because the space of possible configurations is huge, interactions between components lead to unexpected behaviors, and tools do not currently exist to efficiently explore the configuration space and find good configurations for the system as a whole. Autonomic approaches are greatly needed in SSIO to address reliability, performance, and TCO issues.

R&D Needed

Autonomic approaches are needed to increase the reliability of future SSIO systems. A key component of successful approaches to increasing reliability will be accurate methods for detecting (and possibly predicting) faults so that actions can be taken in a timely manner. Clearly a connection exists between successful autonomic systems and accurate fault detection and prediction; in fact, monitoring itself could be adaptive, adjusting what is monitored and at what fidelity in response to significant events.

Autonomic approaches to improving performance are also needed. A wide variety of issues remains open in this area, including methods for optimizing data placement in response to changes in component status or workload; approaches for implementing autonomic principles across multitier storage systems; and techniques for implementing autonomic approaches in a decentralized, scalable manner. Each of these areas has a potential connection to active storage. In particular, active storage components could be the vehicle for implementation of autonomic principles, but this approach has not been thoroughly investigated.

Additional research is needed to better improve the understandability and predictability of autonomic systems. This includes methods for allowing human understanding of how and when decisions were made and ensuring that

autonomous systems do not interact in unexpected, disadvantageous ways. This information may need to be presented in different formats depending on the consumer; for example, an administrator might want to see information in terms of devices, whereas a scientist might want to understand the data sets impacted by an event.

New methods for defining policies are needed to control the behavior of autonomic SSIO systems. Methods for understanding these systems must incorporate mapping back to policies in order to enable administrators to tune policies that lead to unexpected behavior.

4.1.7 Security

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 system architectures create a number of challenges in applying the current security strategy. First, 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 manage multiple tiers, possibly integrating storage from multiple vendors. Second, 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. Third, security must be supported at a range of granularities that may leverage knowledge of file layout (e.g., HDF5 or netCDF). Fourth, any new security solution must be decentralized and allow fast paths for common operations; security needs to be as performance-transparent as possible. Fifth, security solutions should integrate with resource

management to deter denial of service (e.g., consumption of available storage space or bandwidth).

R&D Needed
Security R&D for SSIO is needed that integrates solutions to all five issues cited above while minimizing impacts to performance.

Additionally, solutions that allow data to be securely stored and subsequently rendered inaccessible are needed to facilitate the use of nonvolatile, in-system storage for sensitive computations. The tradeoffs between software-based and hardware-based solutions must be understood as well.

New solutions are needed to ensure that information is not leaked between running applications when data is transmitted over HPC networks, while retaining high performance for communication.

The relationship between resource management and security also merits further study, particularly as relates to the possibility of denial of service attacks targeting SSIO resources.

4.1.8 New Paradigms

State of the Art
Today's state of the art in SSIO includes traditional parallel file systems (e.g., Lustre, GPFS [Schmuck2002], PanFS [Welch2008], PVFS [Carns2000]) and a small collection of middleware tools being used for in-system storage management for fault tolerance (e.g., FTI [Bautista-Gomez2011], SCR [Moody2010]) and data sharing/code coupling (e.g., DataSpaces [Docan2012], C-MPI [Wozniak2010]).

Researchers are also investigating solutions for specific access patterns (e.g., [Curry2012], [Zhao2014], [Sun2014]) and new methods incorporating adaptive storage layout and access operator synthesis to increase performance while limiting data movement and ingest overhead (e.g., [Ionkov2013]). In addition, researchers are borrowing concepts from scalable source code control systems to allow reprogramming of storage systems while maintaining high availability [Watkins2013].

Challenges
Outside the HPC space, various new paradigms for data storage are being explored, including automatic storage engine and query operator design using program analysis and software synthesis (e.g., [Cheung2015], [Karpathiotakis2015]), distributed resource management and scheduling frameworks such as Apache YARN [Vavilapalli 2013] to manage a wide variety of computations within distributed storage systems, key-value stores, object stores of various types, and scientific databases. Similarly, alternative methods of managing data of interest are being successfully employed outside the HPC space (e.g., [LeFevre2014],

[Alagiannis2014]). The merits of these methods in the context of HPC and EOD analysis must be understood, and extensions to these methods developed in order to accelerate scientific discovery.

R&D Needed

Key-value stores, storage methods that manage multiple representations to accelerate specific access patterns, and methods for managing data in the system to facilitate sharing of data between jobs are all promising techniques for accelerating data analysis tasks relevant to DOE science.

In terms of improving the performance of our systems in the context of concurrent access, alternative approaches to storage semantics such as transactions, optimistic methods[2] [Kung 1981], and speculative execution clearly need additional study. Closer connections to programming models and better support for science data models (e.g., multidimensional arrays) promise further usability of these systems. A closer connection to programming models enables more efficient I/O operators within storage systems because of the availability of high-level semantics. With efficient I/O operators also comes the need for distributed resource management and scheduling frameworks that successfully abstract over the resources of heterogeneous system architectures. A closer connection to programming models also implies that storage systems may have to evolve much faster than before while remaining highly available.

Managing data is an increasingly critical component of successful SSIO, in both HPC and EOD contexts. Efficient indexing techniques as alternatives to traditional name spaces provide new methods of data discovery and can enable "piling" rather than "filing"—which in turn opens up the possibility of new ways to manage bursty write traffic. Similarly, finding the right tradeoff between the cost of accessing raw data and the overhead of ingesting data into structured formats optimized for certain access patterns is an increasingly important challenge, especially for applications combining simulation and (possibly real-time) observational data.

In addition, new methods are needed to enable users to express future I/O needs, so that I/O operations can be more effectively scheduled over long periods of time. This is partially a programming model or interface issue, but these methods must be coupled with mechanisms for data movement across the memory/storage hierarchy, including incremental data movement, so that data can be effectively positioned for computation. Because of the increasing complexity of system architectures and heterogeneity of devices, users will need to increasingly rely on a high level of automation in order to identify future I/O needs. Information available during compile time and within runtimes could be a critical component of successful solutions.

[2] a lock-free approach to managing concurrent access to resources

4.2 Metadata, Name Spaces, and Provenance

As the complexity and scale of systems, applications, and data continue to grow, there is an increasing need to develop robust capabilities that enable both systems and user 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) 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.

Finding
New requirements for public access to digital data required for validation of published results [SC n.d.] are poised to fundamentally change the role of metadata in DOE Office of Science and NNSA mission-critical applications. These changes will mandate new approaches for capturing provenance and new methods for exploring extreme scale datasets.

4.2.1 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 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-Gonzales2014, Leung2009], but most of these systems rely on

effective indexing, which has its own scalability and data-consistency challenges [Chou2011]. Recent work has investigated the use of integrated databases for metadata storage [Johnson2014], but this technique has not been applied in an HPC storage system.

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.

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.

R&D Needed

Scalable approaches for metadata management on HPC platforms are still inadequate; and the need to manage additional metadata for provenance and to provide capabilities for searching creates new and exciting R&D challenges. How would metadata management systems scale up to deal with the massive amount of metadata from parallel job with one billion parallel threads of execution? Are there lessons we can learn and apply from cloud and web technologies?

Storage and metadata systems with implicit capabilities for efficiently querying and analyzing metadata and application data are still needed. Particular efforts are needed to develop scalable algorithms, data-storage architectures for metadata, and performance analysis for these capabilities.

While file system permissions, access control lists, and encryption-based security have been used for data, applying the same security features to the corresponding metadata must be explored. Easier interfaces to alter the security and privacy features of metadata may be needed in order to open or restrict sharing of the metadata. For any new storage system feature, additional R&D to understand security implications of these features is necessary. For example, a particular concern for searchable files systems arises, since search results may be used to infer the existence of files that cannot be read.

4.2.2 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].

Challenges

The existing work generally is hierarchical. A number of researchers, however, have argued that such hierarchical namespaces impose inherent limitations on concurrency. Eliminating these limitations could be the fundamental breakthrough needed to scale the namespaces to billion-way concurrency.

R&D Needed

One challenge is providing a coherent view of similar data from different systems (i.e., federating namespaces).

Much work still remains to develop algorithms and approaches that allow low-latency traversal and search of very large and dynamic namespaces. Indexing is the primary tool; but keeping indexes consistent is a challenge, especially for systems with many small writes. Bulk ingest also presents a challenge for indexing.

Science teams need methods that preserve naming schemes and semantics across systems and at different levels of the storage hierarchy. Of interest is the related problem of data integration, for which the database community has developed approaches for some years.

Techniques also are needed to support the coexistence of multiple namespaces for a single collection of data. These techniques allow, for example, one namespace view while data is generated and another for analysis of multiple experiments.

4.2.3 Provenance

Provenance is broadly defined as metadata that describes the lineage of data. In simple terms, provenance contains details on how a particular file was generated; these details can be used to reproduce scientific results. For large-scale computational problems, the information can include the origin of data (sometimes experimental data); algorithms, libraries, and associated parameters and versions 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 [SC n.d.] 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.

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. 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.

R&D Needed

Research is needed to understand how much provenance information is sufficient for the validation of results and whether there are different types of provenance information that must be captured for different use cases. The provenance data required to rerun experiments may not be the same as the provenance data required for debugging or security; however, they are both necessary for different levels of reproducibility [Arpaci-Dusseau2014]. How do we automatically capture provenance from third-party libraries and system software? Do we define a standard API for emitting/capturing provenance data? Do we provide options to select/filter this data? Research to evaluate and understand the tradeoffs and possible solutions is necessary.

The prospect of collecting a wide range of provenance data could lead to significant overheads, not just on the storage system, but also on the network and associated applications. These overheads include costs of acquiring provenance, transferring in the memory/storage hierarchy, storing provenance, and searching for useful information. The community must understand these overheads and perform R&D to explore software and hardware systems to mitigate these costs at HPC and other Office of Science user facilities.

Keeping provenance data accessible is an important task, through either standard formats or standard interfaces. This capability would enable third-party tools to evaluate and analyze provenance data, allow for the integration of provenance data from multiple sources, and provide a mechanism for encapsulating all data and associated metadata from a set of experiments into one logical unit. R&D is necessary to explore viable approaches to enable standard access, attribution, and establish relationships between data and metadata. Closely related, we need an infrastructure that supports the *curation* of metadata and data associated with simulations, so they can be validated across changing software and hardware environments.

4.3 Supporting Science Data

DOE science teams operate with large and complex datasets representing a wide variety of phenomena. In order to be productive, their tools must make storing, managing, and analyzing this data convenient. Research and development in SSIO is needed to bridge the gap between general-purpose storage services and specific science needs, including supporting these data abstractions in programming models and workflows – areas where data abstraction support in HPC has traditionally been lacking.

Findings
The emerging use of alternative programming languages and task-based workflows drives the development of SSIO software. Such software will need to be more flexible and to better integrate with upper layers in the software stack.

Scientists require increasingly complex and specialized data abstractions in order to improve their productivity and the quality of their science. Significant improvements in SSIO data abstractions and their representations in the storage system are required to support these needs and to simplify upper layers of the stack.

4.3.1 Programming Model Integration

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. In particular, 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.

Challenges
Future HPC systems will incorporate multiple levels of memory and storage, including high-bandwidth designs, NVRAM, DRAM, disk, and tape. The current approach to programming does not expose any information regarding this hierarchy to the programmer; rather, aspects of this hierarchy are transparently managed by the operating system and hardware, while libraries and other services explicitly manage other components. Providing programming model support 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. Via 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.

R&D Needed

R&D needs focus primarily on the three challenges mentioned above: support for using complex storage hierarchies seamlessly, mapping of computation to persistent data (active storage), and resilience through persistence.

In the area of complex memory and storage hierarchies, research is needed to understand how the deep storage hierarchy can be presented to users and managed as a whole in the scientific computing context. This research includes methods for mapping complex data structures across the storage hierarchy, and exposing locality of data within the storage hierarchy, as well as methods for persisting data structures stored across the hierarchy, possibly in a transactional manner.

In the area of mapping of computation to persistent data, additional research is needed in order to understand how to efficiently execute computation (simulation or analysis) on distributed, persistent scientific data structures; and in fact this work must be performed in concert with work to understand mapping of these structures within the storage system. Research in methods to enable computation to occur deep in the storage hierarchy (e.g., active storage) is also needed; and the functions to be executed might need to be adjusted interactively, while an application is running.

In the area of fault tolerance, applications need to be able to express their desired persistence properties of data. Additional research is needed in order to understand how application developers can effectively communicate persistence requirements through the programming model and how the storage system can better express (and guarantee) fault properties (and potentially fault containment) to the programming model so that educated decisions can be made by the application.

More generally, given the complexity and constraints (power, failure rates) in emerging extreme-scales systems, programming models and abstractions must enable the application to guide an autonomic runtime so that it can effectively evaluate tradeoffs (performance, power, resilience) and make decisions (mapping, data placement, data prefetching, scheduling, etc.) in a cross-layer and application-aware manner.

4.3.2 Workflows

State of the Art

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.

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. While some research has been done on this topic, more work is needed. 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.

R&D Needed

Research is needed in order to better understand the best methods of interfacing between workflow management systems and their tasks. For example, it is unclear how best to present “locality” in the context of a deep memory hierarchy so that decisions can be made regarding whether to move computation to data or data to

computation or whether to further partition computation or data in order to facilitate a mix of approaches. Additionally, scalable methods for publish/subscribe paradigms [3] [Eugster2003] may be needed to assist in efficient, distributed scheduling of workflow tasks.

Since much data generated by workflow tasks will be transient, methods are needed to provide data protection over short timeframes when data are stored in the system. An obvious connection exists here with other activities, such as SCR [Moody2010] and FTI [Bautista-Gomez2011], but these activities focus on checkpoint/restart cases; more general solutions are needed.

Multiple intermediate representations of data and how they interact with a multistage workflow must be explored. The data have to be augmented with provenance information that is provided to the workflow management system. This research strongly crosscut with the resource management system with which the workflow management system must interact in order to make optimal decisions about resource allocation.

Research is needed in order to understand how provenance information from workflows can best be captured and managed by the SSIO system. This could include new interfaces for passing this data into the storage system, methods of piggybacking provenance information on data payloads, or methods for cross-referencing between generated data and workflow and resource management constructs (e.g., tasks, jobs, workflows, and system resources).

Moreover, the overlap between big data programming models and scientific data programming models needs to be explored. MapReduce is an important big data model, and initial activities have explored its utility for scientific data [Ekanayake2008], but deeper study of programing models for big data workflows is required.

4.3.3 I/O Middleware and Libraries

State of the Art

I/O middleware has established itself as an important component in the I/O stack for HPC; and libraries such as ROMIO [Thakur1999], HDF5 [Folk1999], Parallel netCDF [Li2003], ADIOS [Lofstead2008, Lofstead2014], SIONlib [Freche2009], and Damaris [Dorier2012] are routinely used in scientific codes to ease the burden of data organization and management. These libraries largely support dense, multidimensional arrays, with more specialized data models typically being supported on top of these building blocks.

[3] a model of communication where messages are published and delivered to receivers who have subscribed to messages matching certain criteria.

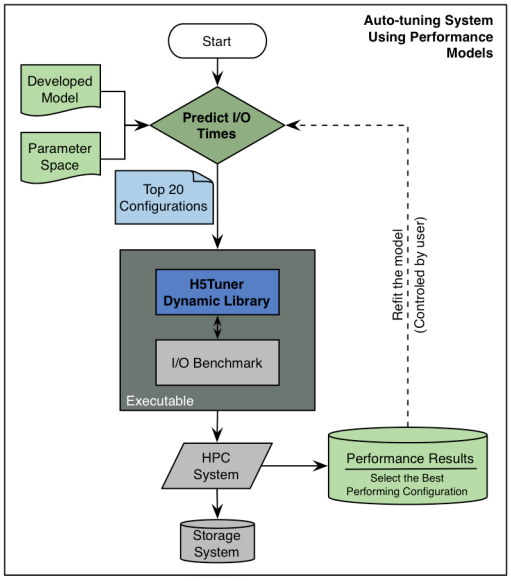

Figure 4: I/O middleware is hampered by its ability to self tune. Work such as [Behzad2014a] is investigating approaches for autotuning of I/O middleware that can provide dramatically improved performance with limited search time.

The libraries primarily are linked into application codes and perform data transformation and optimization on the nodes on which the computation is running. However, researchers also used set-aside processes or cores to perform I/O tasks [Nisar2008, Vishwanath2011a] and to integrate processing into the data path [Bennett2012, Vishwanath2011b]. This research begins to blur the line between "traditional" I/O middleware and I/O system software such as I/O forwarding packages [Ali2009] and data management services [Dong2013]. Additional research (e.g., [Docan2012]) focuses on middleware approaches for data sharing between codes in HPC systems.

Challenges

A significant issue that must be resolved is resource allocation to I/O middleware. Currently, either I/O middleware gains control over cores or nodes when functions are called, or these resources are handed to the middleware in an ad hoc manner by the application. I/O middleware needs to be extended to interact with resource management for allocation (and reallocation) of resources at runtime. This work includes enabling I/O middleware to execute beyond the scope of a single job.

The ability of I/O middleware to make best use of system resources is limited by the information that it presents to users, other layers of middleware, and low-level OS services. These communication paths and the ability to compose services from multiple layers must be improved so that multiple layers of storage software can be more easily composed into effective solutions for specific science teams. A more refined abstract model for storage may be a necessary component.

The range of models supported by middleware also must be improved. Dense, multidimensional arrays may not be the ideal building block for certain other models (e.g., graph representations, adaptive data structures); but it is unclear how best to organize middleware to support these models. Further, access methods should account for the fact that users have differing levels of expertise in accessing data.

R&D Needed

Research is needed in order to understand how advanced I/O middleware architectures should be implemented. Issues include how multiple layers of

middleware can be composed and coordinate efficiently with one another and with users, how security can be managed for multiuser I/O middleware, and how faults and fault domains are managed and reported. Coordination with work in how programming models support use of deep storage/memory hierarchies is mandatory.

Additional research is needed into data models and interfaces, beyond the pervasive dense multidimensional array model. More work is needed to understand how asynchrony can be better supported, how adoption of new interfaces can be facilitated, and how users and other system services can direct data traffic explicitly when necessary.

4.3.4 Data Abstractions and Representation

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.

R&D Needed

New research is needed in order to identify what are the best underlying storage data models for supporting science data models and correspondingly to determine how to best map these science models down onto complex storage hierarchies as well as emerging architectural solutions such as burst buffers. This research includes not only the organization of data but also the semantics for access to that data—where to implement certain capabilities in the SSIO stack and how to efficiently specialize models for specific use cases.

Alongside the science data abstraction, a separate storage abstraction capturing the salient properties of the data is needed. This includes a range of attributes discussed in other sections of this document: resilience properties and locality information, as well as information on how the data has been used and is expected to be used. Furthermore, these abstractions should expose architecture aspects and constraints as well as available tradeoffs in a semantically meaningful way; in other words, while it is meaningful to talk about power at the hardware or OS level, it may be more meaningful to consider resolution or frequency of analytics, which have power implications, at the application level

Research is needed into methods for capturing relationships between data items. For example, one approach could focus on alternative name spaces (i.e., new methods of organizing storage in the large). Applying graph-based or other methods of associating data elements could also lead to promising solutions.

As mentioned under data abstractions, research is needed into how to map complex science data models onto hierarchical storage architectures. In the context of data representations, additional research is needed to understand what roles different organizational techniques should play in the system as a whole and how these can be applied in concert to achieve science goals. This includes both different ways to organize data (and metadata) on storage and different methods for finding that data.

Similarly, assuming that new information is made available up and down the SSIO stack, a number of concerns need to be addressed. Issues to be considered include the intent of users, the status of layers, history and expectations of performance, and the relationships between data elements, research is needed to develop the mechanisms that make best use of this information. This objective is tightly tied to the success of autonomic approaches as well.

In the context of compression and fidelity, additional research is needed to understand what types of lossy compression are acceptable to scientists, what level of loss is tolerable, how and where to support these methods in the SSIO stack, and how information on fidelity and errors is communicated.

4.4 Integration with External Services

SSIO systems can no longer exist as autonomous islands of data that users move data into and out of manually or through custom scripting. Storage systems need to work seamlessly and be shared across multiple HPC resources within a datacenter, an activity that introduces additional complexities for providing consistent data access and modification. New layers in the memory hierarchy further complicate the picture. Interfaces are not currently well defined that would allow SSIO systems to work with archives, resource schedulers, monitoring systems, and workflow systems. Clearly needed are interfaces and software that will allow a user or application to tell the system what its storage needs are and to ask the system about its storage hierarchy and capabilities. These needs can be addressed by researching ideas such as standard APIs, fault monitoring and attribution, coscheduling, dynamic provisioning, and discovery mechanisms.

Finding
Current SSIO designs are hindered by their isolation from system-level resource management, monitoring, and workflow systems. Cooperation with these critical system services will be mandatory for the success of SSIO in future NNSA and ASCR HPC platforms.

4.4.1 Scheduling and Resource Management

Integration with scheduling and resource managers is becoming increasingly important to effectively use and manage large storage systems that will include archive, disk, burst buffers, and other NVRAM devices that are either on or off node. Better coordination between the storage system and the scheduler can ensure less contention at the storage system and result in improved job runtimes. To achieve this goal, we will need a wealth of information on application I/O characteristics, storage system load, and the ability to feed this data back to 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 on current and future SSIO requirements of the workflow components to schedulers and resource managers or can adapt their execution to work within the available resources. In the future, 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.

State of the Art
Batch scheduling has been used for supercomputers for some time, and initial work has been done on checkpoint/restart integration using burst buffer devices (e.g., Cray, HIO@LANL). Work involving moving data and scheduling jobs has been done for grid computing, and the multischeduler issue has been addressed [Schopf2002]. The BadFS work has shown ways to integrate storage with the scheduler [Bent2004].

Challenges
A major challenge concerns how schedulers should interface with storage systems in such a way as to provide various storage resources to jobs that are executing on multiple shared systems. Users want elasticity and on-demand provisioning controlled by a running job.

R&D Needed
Research is needed in numerous areas. Scheduler modifications include storage hierarchy integration for data movement, long-running jobs needing resources at different times, monitoring integration and reaction, multisystem use of shared resources, QOS scheduling, active storage scheduling, and integrating an understanding of new resource models.

4.4.2 System Monitoring

As the massive collection of devices associated with SSIO systems is deployed, knowledge of the state of every device, its relation to the system as a whole, and its real-time maintenance needs will be critical to the efficient use of storage systems. Scalable collection of performance, fault, power, and usage data will allow such system monitors to provide online analytics that provide key data to system managers with the lowest possible imposed overhead. 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 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 perform rich analytics of the collected performance data in order to provide the predictions of expected load necessary for efficient scheduling.

State of the Art
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 data already exist; and many of these tools are combined by using custom scripts [Miller2010].

Challenges

An open monitoring system is needed that can collect analyze and report failures as well as highlight the cause of performance issues that are actively occurring across multiple compute, storage, and archive systems. The system should be capable of supporting O(100,000) elements. Creating such a monitoring system is difficult, however, because of the current lack of any standard or interface for gathering all performance, log, and failure data.

R&D Needed

Research is required in a number of areas. For example, a common and scalable approach to data collection and organization is needed, which relates to schemas for storing information and could be used for additional purposes beyond SSIO data collection. More standards are needed for collecting performance and failure data, as well as the ability to conduct online analyses of this information over the whole system so that normal slowdowns and performance issues can be separated from faults and other events. Additional research is needed into automating root cause analysis. The ability to map monitoring information back to explanations that inform the user of the storage system about the validity, availability, and durability of data is also needed. Monitoring storage system power use is important for system management and scheduling. In order to support this research, more data about the current running systems is needed to validate working models; to this end, production centers should be encouraged to provide operational data in a privacy-protected way.

4.4.3 Workflow and Orchestration

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, including ones to extended memory hierarchies, will be required that allow for the discovery of system characteristics and the integration of data from disparate sensors and multiple storage and compute systems.

State of the Art

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

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.

R&D Needed

Topics that need research include a system publish/subscribe model that ties in with programming and workflow models, resource managers that cross multiple systems and sites, services that are available always or on demand for feeding workflows, producer/consumer models for sensors and other data movement, and discovery mechanisms for storage properties.

4.4.4 Archives

Scientific data storage has long used large archives based on tape storage. These archives need to be better integrated into the storage system of large-scale computing systems. In the 2018 timeframe, there may be a real need for the archival system to integrate not just with the machine's scratch storage but also with the center-wide long-term analysis storage. In the exascale timeframe, as much of the data analysis begins to be performed *in situ* on the burst buffer, there may also be a need for the archival system to interface directly with the burst buffer. As this situation develops, the architecture and design of archives 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 flows, and rich metadata services. This may necessitate the use of alternative archive technologies beyond tape, such as power-managed disk or optical storage.

State of the Art

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.

Challenges
Two challenges of particular concern are (1) effectively verifying the integrity of archive files over time and continuous technology migration and (2) the cost and practicality of retrieving and searching data that are resident on slow TTFB devices.

R&D Needed
The question of how to interface with archival systems raises a long list of topics for research. These include burst buffer direct to archive transfer; scalable data integrity checking; resilient and scalable archive searching and indexing, including semantic search; ease of archive expansion and federation; programming models for archive access; active archives; new archive systems that may not be tape or as monolithic; cross-system namespaces including other archives, file systems, and compute systems; security and privacy protection; and long-term curation interfaces. Research in these areas should dovetail with efforts both to integrate archives more closely with namespaces and resources that span the entirety of the storage hierarchy and to allow for intelligent searching and indexing of archive metadata and data.

4.5 Understanding Storage Systems and I/O

Research tools are needed for end-to-end measurement and understanding of SSIO systems. In parallel application building and tuning, numerous correctness and performance tools are available to applications. In the area of SSIO systems, however, few generally applicable tools are available. Tools and benchmarks for use by application programmers, library developers, and SSIO system managers would be an enormous aid. Also needed is research into evolutionary ideas such as layered performance measurement, benchmarking, tracing, and visualization of related performance data. More radical ideas also need to be explored, including end-to-end modeling and simulation of SSIO stacks.

In addition, tools for designing SSIO systems for understandability are desirable. Given the need for extreme concurrency while having to handle coherence, consistency, and correctness over many tiers of storage devices with many orders of magnitude differences in latency and performance, SSIO systems have been difficult to build with provable reliability and performance. Just as applications on extreme-scale systems have become difficult to reason about, SSIO systems are equally difficult or even perhaps harder to reason about, given the multiuser environments in which they operate. Researchers need to treat SSIO systems development as a first-class application warranting its own tools and languages. This topic is covered in greater detail in Section 4.1.5.

Finding
Many important aspects of application and system behavior related to SSIO are obscured from view. Recent successes in capturing application SSIO behavior have highlighted the value of this information for performance debugging, system

procurement, and steering of SSIO research; but a better understanding of behavior is critical to SSIO effectiveness.

4.5.1 Workload Characterization

Workload characterization, emulation, and understanding are some of the more important concerns that call for micro-applications that capture current application SSIO patterns/flows. Automated tools for workload characterization like I/O kernel extraction, automated workload characterization, and of course benchmarks were also discussed. There are no real standard tools for workload characterization or benchmarking. There are examples of current state of the art tools but there was consensus that these tools are not used broadly enough and dissemination of the data they produce is not ubiquitous enough to enable broad enough research. There are even fewer I/O kernels of important applications being provided and kept up to date for broad use by the research community. So while tools do exist, the quality of upkeep of the tools, broadness of use, and availability and quality of data they produce is just not sufficient for a broad community engagement.

In addition to characterizing and understanding today's workloads, the attendees felt that some methods for understanding future workloads/workflows including local and wide area data management needs are needed. There are very few if any workflow characterizations for future applications.

State of the Art

In the area of workload characterization, emulation, and understanding, multiple projects have focused on tracing, replaying, and analyzing workloads, including workload-aware storage [Zadok2006], visualization of I/O behavior [Ma2009a], and automatic extraction of parallel I/O benchmarks [Ma2009b, Behzad2014b]. Other HPC I/O tracing tools have contributed features such as trace compression [Vijayakumar2009], multilevel instrumentation [Luu2013], and detection of internode dependencies [Mesnier2007]. A large set of tools exist for characterizing and understanding I/O workload, addressing areas such as automation, compression, replaying, and visualization; but the tools are piecemeal, each designed to measure or

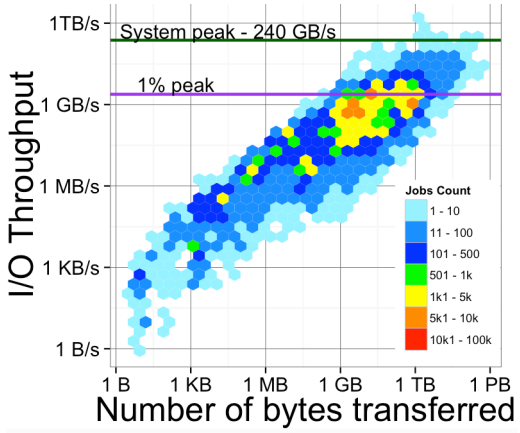

Figure 5: Darshan data can be used to understand the behavior of applications running on production systems. This graph shows the relationship between number of bytes transferred and effective I/O throughput for applications on the Mira Blue Gene/Q platform [Luu2015].

understand one part of the workload or system. Sorely lacking is a comprehensive tool set that can characterize all the way from application to storage devices and back at flexible fidelity/overhead. Further, there is no standardization of tools or the data they produce; and therefore there is not a large set of data that characterize the entire problem space disseminated broadly enough to engage a large research community effort.

Benchmarking is also an important part of SSIO research. Many HPC SSIO-related benchmarks exist, such as IOR [Shan2009], FSTest [Nunez2003], and MDTest [MDTest]. Additionally, the recent DOE CORAL procurement included an HPC application-centric SSIO benchmark called HACC [HACC n.d.]. Similar to the characterization tools above, however, the benchmark area suffers from lack of standardization and piecemeal tools. Most of the benchmarks are for measuring low-level SSIO activities, with only a few that are intended for full stack understanding.

Profiling techniques for workload characterization also exist, including Darshan from Argonne National Laboratory [Carns2011, Carns2009], IPM from LBNL [Uselton2010], and Vampir from TU Dresden [Vampir n.d.]. These profiling tools could be used to provide much more data to the research community if adopted and used more broadly and if HPC sites felt that providing this insight into SSIO workloads was worthwhile.

Darshan [Carns2011] is the most used and relevant SSIO workload profiling tool in use by a large set of HPC sites today. While other tools do exist, Darshan has a particularly broad impact because it can be transparently enabled for production use. The tool captures histogram-level information for I/O activities performed by a job, for example, the number of writes binned into size bins. The simplicity of the tool and its light touch on production systems makes it highly leverageable.

From a benchmarking point of view, IOR [IOR n.d.] is probably the most used HPC SSIO low-level benchmark for understanding data movement from compute memory to and from storage devices. MDTest [MDTest n.d.] is probably the most used mostly for metadata measurements, such as file name/attribute creation, deletion, and query rates.

In the area of understanding future workloads, almost no work exists. The most salient ideas in this area revolve around having future facing mini-I/O applications that are coordinated with the co-design center mini-applications. This work includes integration with future potential programming and execution models. At a minimum, a suite of mini-I/O applications is needed that is tied to co-design center mini-applications written in or utilizing next-generation runtimes.

Challenges

For the most part tools such as IOR and MDTest can mimic current application I/O patterns, but at times more application-like benchmarks would be useful. Another

challenge is in effectively leveraging characterization data to improve applications and systems (e.g., expert systems and other automated methods). Additionally, complex workloads have not been well characterized, especially complex workflow-based workloads. Tools for capturing complex workflow effects on SSIO systems are largely nonexistent.

Another critical need is an effort to enable much more broadly disseminated, validated results of characterization. The lack of incentive for large HPC sites to run collection tools and routinely provide this information is a real inhibitor. Nothing would help the research community more than orders of magnitude more high quality operational and characterization data.

R&D Needed

R&D needed in this area include more kernel extraction and scalable mini-applications of more complex workflows, characterizations of multi-application ensembles or platform-wide workloads, tools for exploring new architectures and mapping new application concepts onto new architectures, more R&D on workload-driven understandable SSIO systems designs, and better analysis and automation techniques to translate characterization data into actionable information. The small amount of available operational and characterization data limits the number of researchers willing to invest time into looking at deep understanding of the SSIO problem space. The tools mentioned above are worth producing only if they are run routinely and produce high-quality data that is released regularly. Funding more tool development without finding a way to incentivize HPC sites to produce ongoing data will not have a huge payoff. Producing a tool for a single point study might make for a great conference paper but lacks the leverage of engaging the community at large.

4.5.2 Modeling and Simulation

Clearly needed are much better modeling and simulation tools for SSIO systems, the environment they live in, and the workloads they face. Modeling and simulation could be applied to trying new workloads, new concepts, and new technologies, all virtually. Just as with the tools mentioned above, however, without follow-up to ensure these tools are maintained and do not keep up with current thinking on future applications, architectures, and systems, the need will still exist.

State of the Art

The state of the art for SSIO systems-related modeling and simulation is the CODES project [Lang2010]. Utilizing the massive DOE computing complex to model SSIO systems and their environments is a big payoff activity, and CODES is an excellent start in this direction. CODES can be thought of as a combination of modeling and simulation of SSIO systems and interactions between SSIO system parts and other parts of the environment. Use of the massive DOE computing resources for furthering the understanding of SSIO systems is something that should be

leveraged; but sustained effort on a small number of consolidated model and simulation frameworks that could eventually be well validated has not occurred.

Smaller efforts have included the development of new mathematical techniques, such as differential regression [Settlemyer2012] and machine learning models [Kettimuthu2012], for developing analytical methods for modeling the pairing of storage devices with wide-area networks, as is common for long-distance file transfers with tools such as XDD and GridFTP.

Additionally, work on the HECIOS simulator done by Clemson [HECIOS n.d.] and the SIOX work done by the High Performance Computing Center in Stuttgart (HLRS) [SIOX n.d.] are likely the closest things to the state of the art, but neither is generally regarded as the entire answer to this need. These efforts approach simulation of SSIO systems from the ground up. Other work has been done at Purdue on modeling disk-bound applications [Thottethodi2006] and at Clemson on SSIO simulation frameworks [Ligon2006]. Additional efforts include device-oriented modeling and simulation tools from the DiskSim project managed at CMU [DiskSim n.d.] and the Blackcomb project at ORNL [Blackcomb n.d.]. DiskSim has been used as the basis for understanding disk behaviors for over a decade and has been an underlying tool to hundreds of academic efforts.

Challenges

The principal challenge in this area is the need to include modeling of the application in conjunction with the entire computing/networking/storage environment. Most modeling and simulation of SSIO systems are not comprehensive enough to improve our understanding of the linkage between the application, networking, and storage. Furthermore, no current modeling and simulation activities adequately deal with the complex workflows in shared usage environments, so the scenarios considered by these tools are “best case” and often not realistic about what users see or will see.

Other important issues with simulation tools include intellectual property issues 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. Funding for these types of activities does not promote standardization or de facto acceptance and broad use by the community.

R&D Needed

Numerous R&D activities are needed. Automated extraction of multi-resolution models from codes, runtime, and traces would enable rapid tailoring of models to applications. Modeling reliability of SSIO systems, including source, duration, and distributions of subtle failures, would allow SSIO designers to explore the resilience design space with realistic failure modes. End-to-end flexible modeling and simulation tools could facilitate exploring new design spaces of new systems without having to produce entirely new models. Composed models that cover applications, networks, and SSIO systems and their interactions would enable much

more realistic understanding in more typical environments users will see. Exploration of potential domain-specific languages for modeling SSIO systems would enable rapid exploration of design and use space. Models that enable study of SSIO systems over long time periods (e.g., years) would provide information to SSIO designers to assist sites in management issues. Exploring models with mixed modeling techniques and mixed resolutions would enable flexible efficient exploration of interesting features. Moreover, an issue that needs to be addressed is the lack of a generally agreed-upon small set of reference SSIO simulators and models that are supported over many years so that the quality of the data and results could become highly trusted.

5 Supporting Activities

Although the workshop did not have a dedicated session on supporting activities for SSIO research, throughout the workshop supporting activities were called out as 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.

Finding
A key need for successful research and development in SSIO is a new and enhanced ecosystem. This ecosystem must provide community access to rich sources of data on applications and systems, test environments in which new technologies can be evaluated, and investments that bring new talent into the community.

5.1 Computing, Networking, and Storage Resources

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.

Support Needed

Funding is needed to ensure that testbed environments are made available to the community and that the systems within the environments are periodically refreshed. Support for instrumentation and fault injection tools is required in order for researchers to make better use of these testbed environments. Further, the testbeds and their testing environments need to be maintained well: if the testing environments are not well cared for and do not have support for the broad community use, the leverage is lost.

5.2 Availability of Highly Documented Operational Data

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.

Support Needed
Funding and coordination of R&D and HPC site activities are needed in order to make large-scale operational data-sharing a reality. Additionally, all funded R&D should have a data management and dissemination plan for data used in the research. A consistent mechanism for giving credit in research publications to the HPC sites and tool producers for providing the data would create extra incentive for HPC sites and researchers to work together more.

5.3 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.

Support Needed
Support could be provided by request for specific outreach activities in research solicitations, similar to NSF solicitations.

6 Summary of Findings and Priority Research

6.1 Findings

Finding 1: *In situ* data analysis is already an important component of many applications. The question is not whether in situ analysis will play a role in future computational and data-intensive science but, rather, how this capability will be manifested (p. 16).

Finding 2: The inclusion of solid state and new disk-based storage layers is dramatically complicating the storage hierarchy. Standard methods of storage organization (e.g., parallel file systems, archival storage management systems) must significantly change, if not be replaced, to provide effective SSIO for future platforms (p. 22).

Finding 3: To work productively, scientists need an integrated, coherent view of the storage resources at their disposal and a common method of managing and accessing data on these resources. Meeting this need will require new metadata capabilities and integration with external storage in conjunction with improvements in SSIO architectures (p. 22).

Finding 4: New requirements for public access to digital data required for validation of published results are poised to fundamentally change the role of metadata in DOE Office of Science and NNSA mission-critical applications. These changes will mandate new approaches for capturing provenance and new methods for exploring extreme scale datasets (p. 35).

Finding 5: The emerging use of alternative programming languages and task-based workflows drives the development of SSIO software. Such software will need to be more flexible and to better integrate with upper layers in the software stack (p. 41).

Finding 6: Scientists require increasingly complex and specialized data abstractions in order to improve their productivity and the quality of their science. Significant improvements in SSIO data abstractions and their representations in the storage system are required to support these needs and to simplify upper layers of the stack (p. 41).

Finding 7: Current SSIO designs are hindered by their isolation from system-level resource management, monitoring, and workflow systems. Cooperation with these critical system services will be mandatory for the success of SSIO in future platforms (p. 49).

Finding 8: Many important aspects of application and system behavior related to SSIO are obscured from view. Recent successes in capturing application SSIO

behavior have highlighted the value of this information for performance debugging, system procurement, and steering of SSIO research; but a better understanding of behavior is critical to SSIO effectiveness. (p. 53)

Finding 9: A key need for successful research and development in SSIO is a new and enhanced ecosystem. This ecosystem must provide community access to rich sources of data on applications and systems, test environments in which new technologies can be evaluated, and investments that bring new talent into the community (p. 59).

6.2 Priority Research Directions

In the area of **SSIO architectures**, additional research is needed to develop solutions to the challenge of managing upcoming deep and heterogeneous storage hierarchies, including storage in the compute system, and to explore alternative paradigms to the current file system model of access and organization. This work is needed to address integration of in-system storage and campaign storage with traditional parallel file systems and archive (or their successors). Additionally, new paradigms for data access and organization are needed to eliminate the POSIX file system bottlenecks that increasingly hinder the usability of current SSIO stacks.

In the area of **metadata, name spaces, and provenance**, research is needed to devise new methods of capturing, organizing, presenting, and exploring rich metadata from DOE science activities, including breaking away from the current file model of data storage prevalent in DOE facilities and science. Scalable methods for metadata management are critical to supporting a more rich set of science activities on future systems and to providing provenance capture and search capabilities needed for verification of results. Alternatives to the traditional POSIX name space are needed to eliminate performance bottlenecks and to create a more flexible environment for storage and analysis of science data.

In the area of **supporting science data**, research is needed to develop the next generation of I/O middleware and services in support of the broad collection of HPC and experimental and observational data needs and to integrate with and support new programming abstractions and workflow systems as they are adopted. New methods of coordinating the activities of I/O middleware and services are needed in order to maximize the utility of SSIO deployments, and new and more streamlined methods of interfacing between programming models, workflow systems, and SSIO are needed to enable seamless use of the deep memory/storage hierarchy.

In the area of **understanding SSIO**, research is needed to improve our ability to characterize the storage activities of DOE scientists and to model and predict the behavior of SSIO activities on future systems. This information is critical to the successful design of future platforms, it enables the optimization of SSIO

technologies and applications using them, and it contributes to the growth of a vibrant SSIO research community.

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
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
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

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

[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.

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

[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.

[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.

[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.

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

[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.

[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.

[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.

[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.

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

[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

[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.

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

[Brinkmann2014] Brinkmann, A., Cortes, T., Falter, H., Kunkel, J., and Narasimhamurthy, S. "E10 – Exascale IO." E10 Working Group Technical Report. 2014 . http://www.eiow.org/home/E10-Architecture.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.

[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.

[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.

[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.

[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.

[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

[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.

[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.

[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.

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

[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.

[Degremont2013] Aurélien Degremont, Thomas Leibovici. http://cdn.opensfs.org/wp-content/uploads/2013/04/lug13-robinhood.pdf

[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]

[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.

[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.

[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.

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

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[HACC] https://asc.llnl.gov/CORAL-benchmarks/#hacc

[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.

[HDF5] The HDF Group. Hierarchical Data Format, version 5, 1997-2015. http://www.hdfgroup.org/HDF5/

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

[HECIOS n.d.] http://www.parl.clemson.edu/hecios/

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

[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

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

[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.

[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.

[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.

[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.

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

[Kim2014] Youngjae Kim, Youngjae and Gunasekaran, Raghul. "Understanding I/O workload characteristics of a peta-scale storage system." The Journal of Supercomputing, pages 1–20, 2014.

[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).

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

[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.

[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.

[Lang2010] Lang, Sam and Chris Carothers. "CODES: Enabling co-design of multi-layer exascale storage architectures." Advanced Architectures and Critical Technologies for Exascale Computing ASCR FOA-10-0000255, 2010.

[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.

[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.

[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.

[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.

[Lustre2002] Lustre: A scalable, high-performance file system. Cluster File Systems Inc. white paper, version 1.0, November 2002. http://www.lustre.org/docs/whitepaper.pdf.

[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)

[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.

[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/

[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.

[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.

[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.

[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.

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

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

[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.

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

[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.

[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.

[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.

[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.

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

[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/

[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.

[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/

[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.

[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.

[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.

[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.

[SIOX n.d.] http://www.hlrs.de/research/current-projects/siox/

[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.

[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.

[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.

[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.

[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/

[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/

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

[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.

[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.

[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.

[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.

[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.

[Wosniak2014] 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.

[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." FAST, 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.

[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 Lucy Nowell for sponsoring the meeting and Lucy Nowell and Thuc Hoang 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. Last, but not least, the organizers also wish to recognize Gail Pieper for her invaluable assistance in editing this document.