ASCR Workshop on In Situ Data Management

Enabling Scientific Discovery from Diverse Data Sources

Final Report

Workshop Organizers and Report Co-authors:

<table>
<thead>
<tr>
<th>NAME</th>
<th>ROLE</th>
<th>INSTITUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom Peterka</td>
<td>chair</td>
<td>Argonne National Laboratory</td>
</tr>
<tr>
<td>Deborah Bard</td>
<td>organizer</td>
<td>National Energy Research Scientific Computing Center</td>
</tr>
<tr>
<td>Janine Bennett</td>
<td>organizer</td>
<td>Sandia National Laboratories</td>
</tr>
<tr>
<td>E. Wes Bethel</td>
<td>organizer</td>
<td>Lawrence Berkeley National Laboratory</td>
</tr>
<tr>
<td>Ron Oldfield</td>
<td>organizer</td>
<td>Sandia National Laboratories</td>
</tr>
<tr>
<td>Line Pouchard</td>
<td>organizer</td>
<td>Brookhaven National Laboratory</td>
</tr>
<tr>
<td>Christine Sweeney</td>
<td>organizer</td>
<td>Los Alamos National Laboratory</td>
</tr>
<tr>
<td>Matthew Wolf</td>
<td>organizer</td>
<td>Oak Ridge National Laboratory</td>
</tr>
</tbody>
</table>

Sponsored by

The U.S. Department of Energy Office of Science Advanced Scientific Computing Research

DOE/Office of Science Program Office Technical Contact: Dr. Laura Biven, ASCR

Email Laura.Biven@science.doe.gov

Executive Summary

In January 2019, the U.S. Department of Energy, Office of Science program in Advanced Scientific Computing Research, convened a workshop to identify priority research directions for in situ data management (ISDM). The workshop defined ISDM as the practices, capabilities, and procedures to control the organization of data and enable the coordination and communication among heterogeneous tasks, executing simultaneously in a high-performance computing system, cooperating toward a common objective. The workshop revealed two primary, interdependent motivations for processing and managing data in situ. The first motivation is that the in situ methodology enables scientific discovery from a broad range of data sources over a wide scale of computing platforms: leadership-class systems, clusters, clouds, workstations, and embedded devices at the edge. The successful development of ISDM capabilities will benefit real-time decision-making, design optimization, and data-driven scientific discovery. The second motivation is the need to decrease data volumes. ISDM can make critical contributions to managing large data volumes from computations and experiments to minimize data movement, save storage space, and boost resource efficiency, often while simultaneously increasing scientific precision.

A fundamental finding of this workshop is that the methodologies used to manage data among a variety of tasks in situ can be used to facilitate scientific discovery from many different data sources—simulation, experiment, and sensors, for example—and that being able to do so at numerous computing scales will benefit real-time decision-making, design optimization, and data-driven scientific discovery across the Office of Science mission space. Applications wanting to use the in situ capabilities include those where data analysis feeds back to the simulation, decisions are made autonomously, big data or machine learning is among the tasks to be coordinated, and computations need to be completed in real time.

The workshop identified six priority research directions that highlight the components and capabilities needed for ISDM to be successful for the wide variety of applications discussed: making ISDM capabilities more pervasive, controllable, composable, and transparent, with a focus on greater coordination with the software stack and a diversity of fundamentally new data algorithms.

**Pervasive ISDM:** Apply ISDM methodologies and in situ workflows at a variety of platforms and scales.

*How can ISDM methodologies help meet the needs for real-time, high-velocity data applications at the edge and other non-high-performance computing platforms? How can ISDM enable science at experimental and observational facilities?*

A changing landscape of use cases is driving new applications of ISDM. The ability to execute the same ISDM tasks and workflows across a spectrum of computational platforms, spanning high-performance supercomputers to experimental detectors and even embedded devices, will reduce human effort and improve portability by applying consistent computing methods.

**Co-designed ISDM:** Coordinate the development of ISDM with the underlying system software so that it is part of the software stack.

*What abstractions, assumptions, and dependencies on system services are needed by ISDM? What information must be exchanged between the ISDM tools and the rest of the computing software stack to maximize performance and efficiency?*

Understanding the interlayer dependencies so that ISDM becomes part of the software stack can facilitate connections between software layers, communicate semantic meaning, and realize efficient performance in high-performance computing and other software stacks.

**In Situ Algorithms:** Redesign data analysis algorithms for the in situ paradigm.

*How should in situ algorithms be designed to make the most of the available resources? What new classes of data transformations can profit from in situ data access in the presence of constraints imposed by other tasks?*

The in situ environment for data processing and analysis differs substantially from the post hoc environment, requiring fundamentally new algorithms and approaches. Progress will benefit from multidisciplinary approaches that holistically consider the opportunities, constraints, and user needs of in situ analysis.
**Controllable ISDM:** Understand the design space of autonomous decision-making and control of in situ workflows.

*What metrics best describe the ISDM design space? How can that space be defined, codified, and evaluated to support design decision-making and control?*

Understanding the space of ISDM parameters is crucial to making intelligent design decisions, both by humans and autonomously. The capability to optimize a constrained ISDM design space will enable predictable performance and scientific validity. Design metrics will promote knowledge sharing across communities.

**Composable ISDM:** Develop interoperable ISDM components and capabilities for an agile and sustainable programming paradigm.

*Can the composition of ISDM software components maximize programmer productivity and usability? What design decisions of ISDM software components promote their interoperability in order to ensure the long-term utility of ISDM software for the science community?*

The flexible composition of interoperable ISDM software components will enable developers and end users to choose from an array of widely available tools, thereby increasing productivity, portability, and usability, and will ultimately result in agile and reusable software.

**Transparent ISDM:** Increase confidence in reproducible science, deliver repeatable performance, and discover new data features through the provenance of ISDM.

*How can provenance and metadata support data discoverability, reuse, and reproducibility of results? How can these artifacts be captured automatically and analyzed in situ, at the scale of the U.S. Department of Energy science?*

In situ provenance and metadata are crucial to understanding scientific results, assessing correctness, and connecting underlying models and algorithms with workflow execution. The ability to capture and query provenance and metadata at scale and in situ will enable many diverse science needs.
# Table of Contents

**Executive Summary** ......................................................... iii

1 **Introduction** ............................................................... 1

   1.1 Purpose of the Workshop .............................................. 1
   1.2 Potential for Greater Scientific Impact from In Situ Data Management ................................................. 1
   1.3 Science Drivers ......................................................... 2
   1.4 Summary of Present State ............................................. 3
   1.5 Workshop Topics and Priority Research Directions ................................................................. 4

2 **Priority Research Directions** ........................................... 7

   2.1 Pervasive ISDM .......................................................... 7
       2.1.1 Key Challenges and Opportunities ...................... 7
       2.1.2 State of the Art .................................................. 8
       2.1.3 New Research Directions .................................... 9
       2.1.4 Potential Scientific Impact .................................. 10
   2.2 Co-designed ISDM ...................................................... 11
       2.2.1 Key Challenges and Opportunities ...................... 11
       2.2.2 State of the Art .................................................. 12
       2.2.3 New Research Directions .................................... 13
       2.2.4 Potential Scientific Impact .................................. 14
   2.3 In Situ Algorithms ....................................................... 15
       2.3.1 Key Challenges and Opportunities ...................... 15
       2.3.2 State of the Art .................................................. 16
       2.3.3 New Research Directions .................................... 17
       2.3.4 Potential Scientific Impact .................................. 18
   2.4 Controllable ISDM ....................................................... 18
       2.4.1 Key Challenges and Opportunities ...................... 18
       2.4.2 State of the Art .................................................. 19
       2.4.3 New Research Directions .................................... 20
       2.4.4 Potential Scientific Impact .................................. 20
   2.5 Composable ISDM ....................................................... 21
       2.5.1 Key Challenges and Opportunities ...................... 21
       2.5.2 State of the Art .................................................. 22
       2.5.3 New Research Directions .................................... 23
       2.5.4 Potential Scientific Impact .................................. 24
   2.6 Transparent ISDM ....................................................... 24
       2.6.1 Key Challenges and Opportunities ...................... 25
       2.6.2 State of the Art .................................................. 26
       2.6.3 New Research Directions .................................... 27
       2.6.4 Potential Scientific Impact .................................. 28
## 3 Workshop Topics

### 3.1 Science Applications
- Brief Summary of Invited Talks ......................................................... 30
- Description of Open Panel Discussion .................................................. 32

### 3.2 Data Models: Connection and Communication
- Enabling Data Models for Multimodal Science, Data Science, and Learning Interoperability ..... 33
- Enabling Reusability by Responding to Intent Expressions in ISDM .................... 35

### 3.3 Computational Platforms and Environments
- HPC System Software Support for ISDM ............................................... 37
- Co-design and Use of Heterogeneous Hardware for ISDM ............................ 39

### 3.4 Analysis Algorithms
- Reduced Representations Enabling Complex Workflows ........................... 41
- Intelligent Run-Time Decision-Making .................................................. 42
- New System Platforms and Data Outputs .............................................. 43

### 3.5 Provenance and Reproducibility
- Capturing Scalable and Portable Provenance for ISDM .............................. 45
- Processing Performance Provenance In Situ ......................................... 46
- Leveraging Provenance for Reproducible Science .................................... 48

### 3.6 Programming and Execution Models
- PEMs for Elastic and Dynamic Resources In Situ .................................... 50
- PEMs for Scheduling, Mapping, and Optimizing In Situ Workflows ............... 52
- PEMs for Composable In Situ Workflows ............................................ 53
- PEMs for Streamed Data ................................................................. 54

### 3.7 Software Architecture for Usability and Sustainability
- Diverse Use Cases Driving Evolution in Software Design ......................... 56
- ISDM Software Usability and Sustainability ......................................... 58
- Software Tool Interoperability ......................................................... 60
- User Confidence and ISDM Software ................................................. 62
- Science Facilities Partnerships ......................................................... 63

## 4 Conclusions

References ......................................................................................... 67

Appendix 1: List of Abbreviations ................................................... 85

Appendix 2: Workshop Agenda ....................................................... 87

Appendix 3: Workshop Participants .................................................. 89
1 Introduction

The U.S Department of Energy (DOE) Office of Advanced Scientific Computing Research (ASCR) convened a workshop on in-situ data management (ISDM) on January 28–29, 2019, at the Bethesda North Marriott in Rockville, MD. This report provides background information on ISDM and information about the purpose of workshop and summarizes the outcomes and findings of the workshop.

1.1 Purpose of the Workshop

Scientific computing will increasingly incorporate a number of different tasks that need to be managed along with the main simulation tasks. For example, last year's SC18 [Supercomputing Conference, 2018] agenda featured in-situ infrastructures, in-situ analytics, big data analytics, workflows, data-intensive science, machine learning, deep learning, and graph analytics—applications unheard of in a high-performance computing (HPC) conference just a few years ago. Perhaps most surprising, more than half of the 2018 Gordon Bell finalists featured some form of artificial intelligence, deep learning, graph analysis, or experimental data analysis in conjunction with or instead of a single computational model that solves a system of differential equations.

For the purpose of this workshop, **we define ISDM as the practices, capabilities, and procedures to control the organization of data and enable the coordination and communication among heterogeneous tasks, executing simultaneously in an HPC system, cooperating toward a common objective.** This workshop considered in-situ data management, in addition to its traditional roles of accelerating simulation I/O and visualizing simulation results, to more broadly support future scientific computing needs (Figure 1). The workshop identified priority research directions (PRDs) for ISDM to support current and future HPC scientific workloads, which include the convergence of simulation, data analysis, and artificial intelligence, requiring machine learning, data manipulation, creation of data products, assimilation of experimental and observational data, analysis across ensemble members, and, eventually, the incorporation of tasks on non-von Neumann architecture.

![Figure 1: Changing role of ISDM over time and the motivations, opportunities, and potential challenges for a renewed research effort in this area.](image)

1.2 Potential for Greater Scientific Impact from In Situ Data Management

Simulations on HPC systems can generate data up to five orders of magnitude greater than the maximum data volume that can be exported to the storage system. Current approaches to managing this bottleneck focus on executing data analysis and visualization tasks in situ to produce data products that can be orders of magnitude smaller than the full state data. The visualization and input/output (I/O) communities have developed a range of in-situ data processing and analysis technologies as a way of achieving data analysis capabilities despite I/O bottlenecks in HPC systems.

Managing data in situ can lead to better use of storage resources and better science: it eliminates some of the negative impacts of the I/O bottleneck; saves storage space; allows the data analysis and processing tasks to access the full data from the simulation as opposed to just the output data; reduces data movement;
and reduces time to solution. In fact, in a growing number of cases, in situ data techniques are the only way to process and analyze data. The in situ paradigm, however, also complicates some operations. For example, human interaction, exploratory investigation, and temporal analysis may be easier to conduct post hoc. Hence, there is a rich design space for carrying out computation in situ: determining which data products are needed for post hoc analysis and the graph of in situ tasks needed to create these; scheduling and executing in situ tasks; and managing the data and communication flow among these tasks.

1.3 Science Drivers

A motivation for this workshop is that ISDM capabilities could be expanded and leveraged for a broader range of current and future HPC applications. In addition to helping meet the challenges of extreme-scale simulation data, ISDM technologies can facilitate applications that merge simulation and data analysis, simulation and machine learning, or the processing and analysis of experimental data. This workshop identified a diversity of future workloads, shown in Figure 2 and listed below, for which ISDM has the potential to enable new capabilities.

☐ Smart simulations featuring online feedback, computational steering, multiphysics, and/or surrogate modeling

☐ Ensemble analysis of stochastic or rare events, uncertainty studies, or model calibration

☐ High-fidelity, highly scalable data analysis and visualization for debugging, diagnostics, and high temporal and spatial resolution

☐ Workflows featuring the convergence of big data and HPC software and tools: for example, graph analytics, database storage, and streaming

☐ Use of machine learning and deep learning alongside simulations or experiments for data-driven analysis methods

☐ Real-time experimental and observational data analysis and assimilation of streaming, potentially noisy, and time-critical data

---

Figure 2: In situ data management will broadly support future scientific computing needs, in addition to the traditional roles of accelerating simulation I/O and visualizing simulation results.

Following are two examples, illustrated in Figures 3 and 4, of workflows embodying some of the listed elements. The workflows are represented as directed graphs: nodes are tasks, and edges represent data flow between two tasks. Figure 3 is a future vision of smart simulations in a High Energy Physics (HEP)-ASCR partnership [Sousa et al., 2018]. The goal of this activity is to automatically simulate millions of concurrent Monte Carlo proton-proton collisions, search the response surface for minimum statistical difference with experimentally observed collisions, and advance toward new regions of the parameter space to investigate. Figure 4 is an example of ensemble analysis. It shows a Basic Energy Sciences (BES)-ASCR collaboration to simulate nucleation as a material cools and crystallizes [Yildiz et al., 2019]. I/O bottlenecks are avoided by detecting crystal structures in situ and only storing features of interest. Instead of one large simulation, many smaller instances are launched dynamically until a rare event is detected—a pattern that has widespread applicability to other domains such as protein folding, self-assembled structures, and genetic algorithms.
Figures 3 and 4 also show another aspect of modern ISDM uses: complex and dynamic workflow graph topologies. In the past, in situ usage was often limited to a single analysis or visualization coupled to one simulation. The task graph consisted of two nodes and one edge. Perhaps computational steering was included, in which case there were two edges. In contrast, emerging today are more complex directed graphs of a larger number of tasks arranged in arbitrary configurations. Both the graphs in Figures 3 and 4 have multiple tasks and contain cycles with feedback. The graph in Figure 3 is static, but the graph in Figure 4 is dynamic, launching as many instances of simulation and feature detection subgraphs as are needed to detect a rare, stochastic phenomenon. Dynamic task graphs for machine learning and artificial intelligence problems can contain thousands of tasks because of the large amount of training needed to learn complex nonlinear scientific behavior.

1.4 Summary of Present State

The end of Moore’s law and Dennard scaling has led to increased concurrency and heterogeneity in computing units and reliance on both general and dedicated-purpose accelerators. Disparity in data movement latency, bandwidth, and energy consumption compared with the rate of floating-point operations has led to deeper memory and storage hierarchies. To put the imbalance between computing and data management in perspective, the rate of data that can be computed on the Summit [Oak Ridge Leadership Computing Facility, 2019] supercomputer (assuming 1 byte generated per clock cycle) is five orders of magnitude greater than the bandwidth of its parallel file system. The I/O bottleneck is one driver of in situ analysis.

Current approaches to manage this bottleneck focus on data analysis, visualization, and the production of data products in situ. The resulting data products are often orders of magnitude smaller than the full state data, thereby eliminating some of the negative impacts of the I/O bottleneck and saving storage space. In situ analyses can also lead to better science. While the infrequency of data outputs limits the fidelity of post hoc analysis, in situ analysis can have much higher fidelity because analysis tasks have access to simulation data directly and are not throttled by the I/O. The in situ paradigm, however, also complicates some operations. For example, human interaction, exploratory investigation, and temporal analysis are easier to conduct post hoc. In situ methods also add complexity to the workflow because of the larger number of interconnected, concurrent tasks that need to be managed.

A survey of past and present in situ methods and tools [Bauer et al., 2016] demonstrates how reusable in situ software evolved separately from the storage and visualization communities. Storage solutions originally were used for staging a simulation’s state for checkpointing, restarting, or saving outputs for later post hoc analysis. Even though such tools have expanded their applications beyond I/O staging, their I/O style of interface and data model remain. Meanwhile, the scientific visualization community developed in situ equivalents of their post hoc tools. Coming at the in situ problem from a visualization direction, these tools feature the VTK data model and scripts for connecting and executing pipelines of VTK filters.
Today, new tools are being developed for more generic data producer/consumer tasks with the potential to manage a general graph of tasks communicating custom data types. Lacking, however, is a common vision for core capabilities to be delivered to users, as well as sufficient attention to making these tools interoperable. To more broadly support scientific computing needs, this workshop provided a forum to address generic in situ data management capabilities, for example for machine learning, automated spawning of ensemble runs, automated triggering and production of data products, and tasks run on non-von Neumann architectures. Workshop participants also had opportunities to discuss provenance and uncertainty as data are managed across tasks, as well as ways to facilitate workflows across multiple data and computing resources through interfaces between distributed and in situ workflow systems.

1.5 Workshop Topics and Priority Research Directions

The workshop addressed the seven topic areas shown in Figure 5. Discussion sessions covered each area, and workshop participants were asked to identify key challenges and opportunities, list the potential benefits to the DOE-ASCR mission of addressing those challenges, and synthesize candidate research topics for each topic.

![Figure 5: Workshop topic areas.](image-url)
The workshop distilled the outcomes from those discussions into six priority research directions, illustrated in Figure 6. The PRDs highlight the components and capabilities needed for ISDM to be successful for the wide variety of applications discussed: making ISDM capabilities more pervasive, controllable, composable, and transparent, with a focus on greater coordination with the software stack and a diversity of fundamentally new data algorithms.

![Figure 6: Priority research directions at a glance.](image-url)

The rest of this report is organized into two main parts: PRDs (§ 2) and workshop topics (§ 3). The material in § 2 explains the highest-level findings of the workshop, while the material in § 3 provides details of the breakout sessions that generated the information from which the PRDs were derived.

Each breakout session resulted in several research areas, listed in the left-hand column of Table 1. These research areas were synthesized into the six PRDs in the top row of Table 1. An “X” in the body of the table indicates a contribution to a PRD by an outcome of a breakout session. The density of the table shows that all the PRDs are broadly supported by the conclusions of the breakout sessions. Table 1 can also be used to search further details for a PRD in the corresponding breakout session.
Table 1: Mapping of workshop breakout session research areas to resulting PRDs

<table>
<thead>
<tr>
<th>Breakout Session Research Areas</th>
<th>PRDs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pervasive ISDM</td>
</tr>
<tr>
<td>DATA MODELS</td>
<td></td>
</tr>
<tr>
<td>Multimodal data science and ML</td>
<td>X</td>
</tr>
<tr>
<td>Intent expression and reuse</td>
<td>X</td>
</tr>
<tr>
<td>COMPUTATIONAL PLATFORMS</td>
<td></td>
</tr>
<tr>
<td>System software support</td>
<td></td>
</tr>
<tr>
<td>Heterogeneous hardware</td>
<td></td>
</tr>
<tr>
<td>ANALYSIS ALGORITHMS</td>
<td></td>
</tr>
<tr>
<td>Reduced representations</td>
<td>X</td>
</tr>
<tr>
<td>Run-time control</td>
<td></td>
</tr>
<tr>
<td>New platforms and outputs</td>
<td>X</td>
</tr>
<tr>
<td>PROVENANCE AND REPRODUCIBILITY</td>
<td></td>
</tr>
<tr>
<td>Scalable, portable provenance capture</td>
<td>X</td>
</tr>
<tr>
<td>In situ provenance processing</td>
<td></td>
</tr>
<tr>
<td>Provenance for ML and reproducibility</td>
<td></td>
</tr>
<tr>
<td>PROGRAMMING AND EXECUTION MODELS</td>
<td></td>
</tr>
<tr>
<td>Elastic, dynamic resources</td>
<td></td>
</tr>
<tr>
<td>Scheduling and optimizing execution</td>
<td></td>
</tr>
<tr>
<td>Composable workflows</td>
<td>X</td>
</tr>
<tr>
<td>Streaming</td>
<td></td>
</tr>
<tr>
<td>SOFTWARE ARCHITECTURE</td>
<td></td>
</tr>
<tr>
<td>Use cases driving design</td>
<td>X</td>
</tr>
<tr>
<td>Usability and sustainability</td>
<td></td>
</tr>
<tr>
<td>Software tool interoperability</td>
<td></td>
</tr>
<tr>
<td>User confidence</td>
<td></td>
</tr>
<tr>
<td>Science facilities partnerships</td>
<td></td>
</tr>
</tbody>
</table>
2 Priority Research Directions

The priority research directions described in this chapter highlight the components and capabilities needed for ISDM to be successful for the wide variety of applications discussed: making ISDM capabilities more pervasive, controllable, composable, and transparent, with a focus on greater coordination with the software stack and a diversity of fundamentally new data algorithms.

2.1 Pervasive ISDM

Apply ISDM methodologies and in situ workflows on a variety of platforms and scales.

Key questions How can ISDM methodologies help meet the needs for real-time, high-velocity data applications at the edge and other non-HPC platforms? How can ISDM enable science at experimental and observational facilities? How do ISDM methodologies for traditional computational modeling compare with ISDM methodologies for experimental and observational facilities, including edge devices such as sensors and detectors? Can recognizing commonalities among such disparate use cases increase the adoption of ISDM among scientists across the DOE mission and bridge scientific communities?

Research opportunities A changing landscape of use cases is driving new applications of ISDM, which increasingly require ISDM near instruments or sensors for data analysis in near-real time. To what extent can similar ISDM approaches be applied both to HPC computing and to edge computing in experimental or observational facilities? The next generation of ISDM research will often require combining and coordinating workflows across multiple such computational platforms in order to answer fundamental science questions. Clearly needed, then, is the ability to execute the same ISDM tasks and workflows across a spectrum of computational platforms, spanning from high-performance supercomputers to experimental detectors and even embedded devices. The workshop called this desired capability "pervasive ISDM." Pervasive ISDM requires streaming data over heterogeneous computing and networking scales and satisfying real-time demands of experimental instruments. Also essential are new algorithms and reduced data representations that are designed for low-power embedded processors with limited memory or bandwidth. Experimental data can be noisy, containing errors or missing data points; and assimilating real and simulated data requires managing disparate levels of data quality, in addition to managing disparities in scale, resolution, and data organization among the various data sources of the overall science workflow.

Potential benefits Considering the combined system of instruments and computing, scientists want to be able to place data analysis where it is required, based on timeliness and other constraints. Being able to deploy ISDM on a variety of platforms would enable this flexibility. Data reduction and analysis are key to dealing with the high volume and velocity data, and performing some of these operations as early as possible will increase the efficiency of later processing stages. Pervasive ISDM would reduce human effort by reusing software tools, algorithms, and frameworks and would improve understanding of performance and science by applying consistent computing methods. The resulting unified approach would have broad applicability across DOE. The ability to deploy in situ approaches traditionally used for computational modeling pervasively, across platforms and scales, would advance experimental, observational, and computational science.

2.1.1 Key Challenges and Opportunities

An increasing number of applications across the Office of Science mission space, including those driven by experimental and observational data, need to execute diverse computations on many different platforms: commodity clusters, HPC centers, and specialized hardware that may be part of an experimental data pipeline. Deployment of software for experimental and observational science (EOS) across these diverse computing facilities includes resource marshalling, incorporating heterogeneous data resources, and spanning distributed physical locations. If ISDM software tools can be pervasively deployed across these different architectures, they will be useful to a broader scientific audience.

The flow of data in a combined system that includes instruments and possibly several different types of compute architectures and facilities poses new challenges to data management. Information can flow from instrument to HPC facility (possibly via edge computing) and back again on several different time scales. Feedback from real-time data analysis can be used to steer experiments and/or observations; short-turnaround (~24 hour) data analysis can inform the next day’s observation plans; over longer time frames, scientists
need to perform more detailed analysis—reprocessing and sharing of data. Many of these use cases rely on specialized hardware that may be at the edge, adjacent to, or incorporated within an HPC facility. For example, field-programmable gate arrays (FPGAs) may be used for fast filtering of a data stream, or neuromorphic devices may be used for analysis. Supporting in situ processing and the movement of data, metadata, and computation with such hardware is a significant challenge.

Disparities may exist between data and programming models used at the edge and on HPC systems. Data from an instrument may be at the atomic scale, whereas data from simulations may be meso- or macroscale. In situ analyses may be multimodal, requiring data from different kinds of experiments and simulations. Workflows that include both HPC and edge data analysis will require reduced representations of data to alleviate storage pressure and to enable streaming from detectors with high data rates. Algorithmic research and infrastructure can enable decision-making within a streaming in situ regime for experimental and observational data, for example, to decide how to adjust an x-ray beam or where to position a sample next. In situ algorithms for edge data need to be performance-portable and composable, and new algorithms having these capabilities will need to be developed. Programming and execution models (PEMs) used in each location must be composable as well. Facility-specific PEMs may not be interoperable; for example, edge-based PEMs are often optimized for streamed data, a mode that may not be supported by HPC PEMs.

Provenance data for in situ analysis is difficult to capture for a variety of experimental and observational sources, such as sensors, detectors, and beamlines. Additionally, it is challenging to build in situ provenance processing into a real-time workflow and to allocate compute and storage resources for this processing. In situ machine learning (ML) models used at the edge will also need provenance to justify real-time experimental or observational decisions guided by ML inference.

### 2.1.2 State of the Art

Data models for multistage computing on diverse resources exist in some form at several experimental facilities. For instance, the NSLS-II data acquisition and management infrastructure includes a generic event model that merges into a single channel data streams from the multiple data and metadata sources available at each detector, while preserving provenance and timestamps [NSLS-II Data Acquisition and Management Group, 2019].

While some existing software frameworks can support experimental and observational data models, streaming data analysis requires a specialized data model. For example, in the Kafka model [Narkhede et al., 2017], commonly used in cloud and commercial applications, data must be an ordered, replayable, and fault-tolerant sequence of immutable data records. Recent work is moving toward developing streaming of data from an instrument to edge or HPC computing, for example in particle physics [Magini et al., 2018].

On traditional HPC platforms, batch schedulers coordinate the compute jobs. One example is Flux [Flux, 2019], an advanced scheduler that could be applied to ISDM. Such schedulers can be interfaced to a workflow management system (WMS) for coordinating data movement and compute across multiple systems. Examples of these WMSs include FireWorks [Jain et al., 2015], Parsi [Babuji et al., 2019], Pegasus [Deelman et al., 2015], and the Open Science Grid (OSG) [Pordes et al., 2007]. For example, Pegasus was used by the LIGO experiment (Figure 7) to detect gravitational waves by managing almost 4,000 inter and intracluster workflows with more than 9 million tasks. This allowed scientists to complete their offline data analyses in days instead of weeks, enabling fast confirmation of alerts [Brown et al., 2007].

Numerous in situ analysis frameworks exist for traditional HPC platforms (Alpine [Larsen et al., 2017], ParaView [Ayachit, 2015], VisIt [Childs et al., 2012a], Adios [Liu et al., 2014], SENSEI [Ayachit et al., 2016b], Decaf [Dreher and Peterka, 2017], VTK-M [Moreland et al., 2016]), but they currently do not extend across platforms.

ISDM for collection and processing of provenance data at the edge is often application specific. The Bluesky library for experiment control and collection of scientific data and metadata [Koerner et al., 2019, Allan et al., 2019] is being used at more than one light source user facility. Tools that are available for handling growing volumes and complexity of provenance data in the HPC domain are XALT [Agrawal et al., 2014] and the metadata graph model [Dai et al., 2014].
PEMs for pervasive ISDM will need to bridge domains (e.g., edge to HPC). PEMs that support streaming of data, which is often required in non-HPC domains, include Psana [Damiani et al., 2016], ADARA [Shipman et al., 2014], ICEE [Choi et al., 2013], and the NSLS-II event model [NSLS-II Data Acquisition and Management Group, 2019].

Software infrastructures addressing high-volume, high-throughput data streams from light source instruments include Xi-cam, an extensible platform for data management, analysis, and visualization [Pandolfi et al., 2018], and Nanosurveyor [Daurer et al., 2017], which supports workflow execution either locally or remotely. The HEP community has identified the need for software architecture approaches for large-scale experiments, which have lifetimes that can span multiple decades [Hildreth et al., 2018]. Some software platforms for EOS exist in production at HPC facilities. The National Energy Research Scientific Computing Center (NERSC), as part of its effort to better support EOS, has provided web-based APIs for accessing services [National Energy Research Scientific Computing Center, 2019b] as well as adding real-time queues to service time-critical workloads. When combined with networking infrastructure optimized for high-volume scientific data movement [Dart et al., 2014], the resulting set of resources is helpful for meeting the needs of on-demand, high-throughput processing from non-HPC sources.

2.1.3 New Research Directions

In order for the efficiencies and capabilities of ISDM to pervade across computational scales and platforms, a number of new research directions need to be pursued. Work is needed to discover how the requirements of EOS differ from those of traditional HPC.

Data model formalisms developed for streaming data will need to incorporate both edge and HPC systems and the relevant metadata from multiple systems. Data from instruments may include temporal constraints, uncertainty, and intent for downstream processing that could be far from where the data were collected. The data flow in EOS is bidirectional; the results of real-time analysis can feed back and steer an instrument or experiment or guide future observations. Therefore, streaming APIs are needed to help connect streamed data throughout the stages of the workflow, from instrument to computing cluster to HPC, and potentially back again.

Computational platform research will be needed to explore, schedule, and utilize in situ emerging and heterogeneous hardware at the edge and other non-HPC environments. New challenges arise in system design, performance, and administration regarding I/O sources and sinks that are remote (external I/O). Pervasive ISDM will require advances in external I/O because, for example, the data flow in real-time streams rather than in bulk-synchronous checkpoints, as in traditional HPC. How to present external I/O to ISDM applications and how ISDM concepts and technologies relate to external data channels such as instruments, human users, and other computer systems are open questions.

The increasing deployment of computing near to experimental detectors, whether in traditional or specialized processors, complicates in situ workflows and necessitates the development of new frameworks, algorithms, and data management tools to handle the mix of computing resources and the different ways data are processed. New research is needed to develop methods to explore, schedule, and utilize in situ processing in non-HPC settings; such research could include co-design among ISDM, system software, programming models, and edge hardware vendors.
New algorithms also need to be developed for in situ analysis on edge devices, particularly for emerging hardware. Algorithms developed for HPC systems or small-scale x86 computing resources may not be appropriate or optimized for hardware such as FPGAs or neuromorphic devices at the edge. Single-pass algorithms will be needed to generate reduced representations in a streaming regime, and research in trigger-based and other decision-making capabilities will be important for control of the experiment based on in situ analyses.

Provenance at the edge will require a cultural shift at scientific user facilities to implement new operational policies regarding access and data. Research is needed in using artificial intelligence (AI) at the edge for provenance processing to detect real-time anomalies as data are generated. Unique identification is needed to track data over a lifespan of movement across a variety of locations.

Programming and execution models that are composable and support streamed data for in situ processing will need to allow access to both edge and HPC data, be scheduled across multiple platforms, and have standardized interfaces for both edge and HPC applications, as opposed to one-off solutions. PEMs that support streaming will need to be developed in order to minimize communication time for real-time in situ processing at the edge with reliability and quality guarantees.

Assumptions and dependencies: Realizing pervasive ISDM requires corresponding advances in algorithms (§ 3.4) to be portable, data models (§ 3.2) to incorporate multiscale and multimodal representations, provenance (§ 3.5) for edge data, and PEMs (§ 3.6) that support streaming and composition of heterogeneous data sources.

2.1.4 Potential Scientific Impact

The increase in data rates from modern experimental and observational facilities means that more data reduction and analysis, from multiple sources, will take place at the edge and at a variety of other computational scales and platforms. Exploiting and deploying the ISDM techniques from HPC to those new paradigms for experiments can improve computational efficiency and productivity.

Furthermore, there is an opportunity for ISDM to make an impact on experimental science, in the way that scientists design their instruments and experiments. If extreme data rates can be handled in situ, some of the restrictions faced by instrument designers today can be lifted, and powerful new instruments can be deployed with no loss of resolution (e.g., for the National Center for Electron Microscopy [Johnson et al., 2016]). Incorporation of real-time feedback in experimental operation will enable scientists to control and steer instruments and maximize scientific productivity. By utilizing specialized and optimized computational platforms for edge computing, the efficiency of data processing at an instrument could be significantly increased, allowing experiments to handle higher data rates.

Data models incorporating multiple stages in experimental workflows will increase the interoperability of in situ analysis codes across edge, cloud, and HPC resources. Such models will lower the barriers for sharing and reuse of in situ analysis tools across HPC and experimental and observational facilities and could aid, for example, in their adoption by the entire light source community, rather than individual beamlines reinventing the same capabilities. Similarly, algorithms that make effective use of emerging hardware will increase scientific productivity. Improved data reduction and compression at the edge will inform judicious use of resources and maximize downstream analysis. Automating decisions about resource usage and data handling, steps that previously required a human in the loop, would permit scientists to devote their time and attention to more important research.

Tracking the provenance of experimental data during their life cycle can help make experimental and observational scientific discoveries explainable.

PEMs that can handle EOS workflows could result in flexible and composable movement of data from the instrument, through edge computing, to HPC, and back to the instrument. Sometimes known as the “digital twin” [Parkinson et al., 2016], the coupling of simulations with real-time streamed data can benefit both modes of inquiry: the results of the simulation can guide how the experiment is conducted, and the real-time data analysis can guide how the simulation is run. A software infrastructure that can enable and coordinate pervasive ISDM on multiple platforms will broaden the impact of DOE’s investments in experimental facilities and increase the impact of ISDM research.
2.2 Co-designed ISDM

Coordinate the development of ISDM with the underlying system software so that it is part of the software stack.

Key questions  What abstractions, assumptions, and dependencies on system services are needed by ISDM? What information must be exchanged between the ISDM tools and the rest of the computing software stack to maximize performance and efficiency? How can we ensure seamless communication between the ISDM software layer and other parts of the system software stack?

Research opportunities  Understanding the interlayer dependencies so that ISDM becomes part of the software stack can facilitate connections between software layers, communicate semantic meaning, and realize efficient performance in HPC and other software stacks. Defining the dependencies between ISDM and the rest of the software stack can enable autonomous data management, efficient algorithmic performance, and verifiable science. Dependencies do not only extend down the software stack toward system software layers (the service providers to ISDM); just as important are upward connections toward application and workflow layers (the users of ISDM) in higher levels of the software stack. There are also opportunities for connections across multiple software stacks, for example, HPC and big data.

Potential benefits  The co-design of new services, new platforms, and application workflows has the potential to revolutionize ISDM. Thinking of ISDM as an integral part of the computational platform has numerous advantages over developing ISDM software independently from the computing hardware, system software, and applications or workflows. Competition for computational and data resources among in situ tasks can be avoided. Significant power savings are realizable by using next-generation hardware such as neuromorphic chips or nonvolatile memory. Better communication across system software layers can save developer time, provide consistent interfaces to users, and improve the use of computing and data resources. Integration of other software stacks and frameworks (e.g., from industry or big data) can exploit economies of scale, because those tools often have many more developers contributing to them compared with HPC software. More complex workflows can be supported by co-design of new hardware and software systems informed by ISDM use cases.

2.2.1 Key Challenges and Opportunities

The co-design of HPC system software to better support the needs of ISDM would be a departure from the traditional way HPC machines are designed, which is to maximize the performance of linear algebra benchmarks and computational kernels derived from simulation codes. In the past, data analysis methods were developed for the machine after it was delivered. In the future, identifying unique characteristics of ISDM use cases and unique needs not represented by other use cases could help HPC system designers incorporate ISDM workloads in the design of systems.

Another challenge for ISDM is to effectively employ hardware developed by market-driven providers. As non-von Neumann and other emerging architectures become more prevalent in industry, an opportunity arises for the ISDM community to understand the fundamental differences between conventional HPC systems and emerging platforms. Compute nodes are rapidly evolving beyond CPUs and graphics processing units (GPUs) to include neuromorphic chips, tensor processors, FPGAs, and eventually quantum computers. Network heterogeneity, especially when multiple data sources or sinks are connected to the ISDM framework in different ways (e.g., InfiniBand and Ethernet), poses synchronization and scheduling problems among ISDM tasks.

Dealing with new I/O channels—levels in memory/storage hierarchy, multiple data sources and sinks, data streams from experiments and sensors—is a challenge directly related to the hardware and software systems on computational platforms. Utilizing the abstractions provided in system software and co-designing new abstractions—in order to access nonvolatile random-access memory (NVRAM), high-bandwidth memory, local disks or solid-state devices, and buffers for streaming external data—are necessary for efficient ISDM on a variety of computer architectures.

Coordination between ISDM and the system software stack poses numerous challenges and opportunities that affect ISDM usability and scalability, and integrating ISDM with the rest of the system is needed to support
more complex workflows. Today, there lacks a comprehensive understanding of the information that must be exchanged between ISDM and the system software stack. For example, what is needed for the ISDM framework to instruct the operating system/run-time (OS/R) to launch tasks dynamically on demand and provide communication between tasks in an ISDM workflow? Conversely, what information does the ISDM framework need to get from the software stack regarding performance metrics about system usage and system health monitoring, in order to make informed control decisions?

One example of a complex workflow is the data assimilation of simulations and multimodal hard x-ray imaging in synchrotron light sources. Such workflows [Deelman et al., 2017] combine molecular dynamics simulations, simulated diffraction patterns, measured diffraction patterns at the beamline, 3D volumetric reconstructions from the diffraction patterns, and identification of regions of strain (shown in Figure 8). Information in such scenarios flows to and from the ISDM framework both up and down the software stack. Up the stack, users of ISDM—applications and workflow systems—communicate with the ISDM system. Down the stack, the ISDM framework communicates with service providers of the computing platform: the OS/R, I/O systems, schedulers, resource managers, and so forth. Information can include science data, science metadata, and system control data. System control data includes instructions from upper layers to the ISDM system, from the ISDM system to lower layers, and provenance in the opposite direction: from lower levels to the ISDM system and from the ISDM system to upper levels.

An increasing number of data analysis tasks require heterogeneous software stacks. Many open source and enterprise AI, ML, streaming, and database tools exist but require different software stacks from the traditional HPC one. Hence, there is an opportunity to apply existing high-level libraries and algorithms for machine learning and data streaming and to adapt the data awareness and fault tolerance mechanisms of other software systems such as big data frameworks to scientific computing. Unfortunately, today big data and HPC frameworks remain largely incompatible: programming models and software development tools are inconsistent [Ahmad et al., 2018], and trying to mix models can create memory overheads and poor scalability [Chaimov et al., 2016]. The disparity between storage architectures in big data and HPC systems can degrade performance [Yildiz and Ibrahim, 2018]; and merging models presents limitations, such as high memory consumption and low communication efficiency between workflow tasks [Calín-Lores et al., 2015].

2.2.2 State of the Art

Workflow management systems for distributed computing were originally developed for grid environments. Here, as in a prior DOE workshop on scientific workflows [Deelman et al., 2016], we call these distributed workflow management systems to distinguish them from in situ workflows. Distributed WMSs are characterized by being able to coordinate geographically separated tasks, communicating between tasks using files and serializing dependent tasks. In other words, if two tasks have a producer-consumer dependency, the producer runs to completion and writes its output to storage before the consumer can begin executing. Representative examples of distributed WMSs are described in several surveys [Yu and Buyya, 2005, Deelman et al., 2009].

In situ workflows, in contrast, launch all tasks concurrently in one HPC facility (often in the same supercomputer), and tasks either proceed or wait depending on whether they have received all of their current inputs. Communication occurs over shared memory or through the interconnect of the machine (e.g., using MPI) depending whether tasks are colocated on the same compute node or not. These technologies evolved separately from the storage and visualization communities, as the survey of Bauer et al. [Bauer et al., 2016] demonstrates. Today, new tools are being developed for generic data producer/consumer tasks with
the potential to manage a general graph of tasks communicating custom data types; but as Dreher et al. [Dreher et al., 2017] show, HPC platforms do not support all the capabilities needed for in situ workflows.

Today, ISDM applications are mapped to existing hardware and system software constraints rather than co-designing systems with the needs of ISDM in mind, resulting in inefficient workarounds to accomplish needed ISDM capabilities. For example, HPC machines have static job scheduling and static resource allocation. Computer scientists work around these limitations by allocating the total number of resources to be needed statically and leaving them idle until a task requires them. This approach wastes power and ties up scarce resources that others could be using. HPC systems also lack flexible security domains. In situ workflows assume security but in fact provide no guarantees that data will not be shared or corrupted by others. Supercomputers do provide security through a separation and lack of connectivity between tasks, but this separation actually prevents tasks that are cooperating from being able to communicate. Low-performance workarounds to intentional task isolation (e.g., using TCP/IP or UDP sockets or storage systems as communication channels) for in situ workflows are inadequate.

Performance portability layers developed under DOE funding (e.g., Raja [Beckingsale et al., 2019], Kokkos [Edwards et al., 2014], and VTK-m [Moreland et al., 2016]) are leading the way for HPC systems to provide somewhat standard programming models for diverse hardware; however, gaps still remain with respect to NVRAM and emerging platforms. Individual NVRAM abstraction layers (e.g., libhio [Hjelm and Wright, 2017], Data Elevator [Dong et al., 2016], BurstMem [Wang et al., 2014], DataWarp [Henseler et al., 2016], Mochi [Dorier et al., 2018, Jenkins et al., 2017, Carns et al., 2016]) exist, but community-wide standards have yet to evolve.

OS/R research funded by the Exascale Computing Project (ECP) and ASCR (Argo [Perarnau et al., 2013], Hobbes [Brightwell et al., 2013]) investigates system support for unconventional HPC programming models, support for multiple concurrent runtimes, and advanced virtualization capabilities that could be leveraged to support desired ISDM capabilities.

Another active area of research is the use of non-von Neumann and emerging hardware for AI, ML [Schuman et al., 2017, James et al., 2017], and scientific computing [Severa et al., 2016]. These and other surveys [Agarwal et al., 2016] point to the potential for significant energy savings, although the use of neuromorphic hardware for general-purpose scientific computing is still limited.

Early work has been done in integrating software stacks, primarily the Apache big data stack with the HPC stack. MapReduce models can use the Message Passing Interface (MPI) in various ways [Caino-Lores et al., 2018, Malitsky et al., 2017, Bicer et al., 2017, Gao et al., 2017, Wang et al., 2015, Gittens et al., 2018]. Such tools either introduce a new MapReduce language for HPC that resembles native MapReduce but is implemented in MPI, or they inject an adaptor layer between the two programming models that switches contexts between them as needed. Neither solution is ideal. The former does not allow existing tools built on top of MapReduce to be reused, while the latter introduces data copies and overheads associated with the context switches between models. More work is needed to find a low-overhead integration that does not require changing the constituent programming models and allows existing tools to use the integrated software system.

### 2.2.3 New Research Directions

New research is required to develop system software abstractions and APIs that link ISDM with applications, workflow management systems, and HPC system software. For example, advances in system resource management are needed to allow dynamic resource allocation and dynamic task management. Scheduling support must be developed to provide real-time queues, reservations for experiments, and remote connections to other facilities. Security features need to be implemented to allow an HPC system to accept remote connections (incoming and outgoing) from and to other facilities, and varying levels of security (e.g., unencrypted, encrypted) need to be available for individual users as well as members of science teams. High-speed network connectivity among multiple tasks in an ISDM workflow needs to be implemented and supported.
Research should also be directed at support for multiple, concurrent software stacks. Rather than user-level add-ons and adaptors between different programming environments, full-system installation of multiple stacks in the same system is needed, as well as system-level integration or communication among the stacks. For example, the same underlying I/O system should support parallel file systems, key-value stores, and databases. The same resource manager should be able to launch tasks consisting of (for example) Spark executors and MPI ranks. Individual developers should not have to develop custom workarounds for such capabilities that are needed by many users.

Specifications need to be developed detailing what information is exchanged between the system software and ISDM infrastructure or tasks running within it. The specifications should include not only data format but also metadata conveying intent. For example, the ISDM software ought to be able to tell the OS/R about the desired quality of service or constraints imposed by the user on resources, so that the OS/R can intelligently manage resources and data (e.g., deciding when to move data and when to relocate a task). Conversely, the OS/R ought to relate profiling information to the ISDM software regarding system health and resource usage levels, so that the ISDM framework can adjust intent or quality levels.

Co-design among ISDM, system software, programming models, and computing vendors is needed to ensure being able to adapt and evolve with the various communities—and perhaps influence design to jointly meet needs of both computing architecture and ISDM research. Coordinated explorations should take place with vendors of emerging—post-Moore, non–von Neumann—hardware so that such hardware can benefit ISDM. Some of these hardware systems are early enough in their development that there is still an opportunity to influence their design. Systems and hardware should be mapped to the needs of ISDM in addition to or instead of the other way around. While the conventional approach is to develop software to match capabilities of hardware, given an expectation of highly heterogeneous platforms, there is opportunity for innovative ideas to align hardware to the needs of ISDM, leading to improved utilization of computing platforms.

ISDM abstractions are needed for new hardware. Storage/memory devices in the storage/memory hierarchy are central to ISDM. In the same way that the HPC community has been evolving performance-portability abstractions for CPUs and GPUs, there remains research potential for storage and memory abstractions in support of ISDM, particularly NVRAM. Co-design is needed between the ISDM community and researchers from ASCR programs such as storage systems and I/O [Ross et al., 2019] and extreme heterogeneity [Vetter et al., 2019] as well as memory and storage vendors. Abstractions for heterogeneous computing units are also needed for ISDM. Beyond performance-portability abstractions for CPU and GPU threads, programming models will also be needed for FPGAs, tensor processing units, neuromorphic chips, and eventually quantum cores.

**Assumptions and dependencies:** Co-designed ISDM assumes a high degree of coordination among research activities described in this report such as programming and execution models (§ 3.6), algorithmic development (§ 3.4), and support for quality of service in delivering data (§ 3.2). It also requires the study of the ISDM interface to the rest of the system and the separation of responsibilities of data management. Co-designed ISDM both is influenced by and can influence rapidly changing technology, security challenges, and facility policy. Co-designed ISDM is also related to ASCR research in storage systems and I/O and in extreme heterogeneity.

### 2.2.4 Potential Scientific Impact

Significant power savings are possible by using low-power neuromorphic hardware for in situ science and engineering applications. Similar power savings are realizable for edge devices and other low-power computing units.

Exploiting commodity hardware for DOE applications can lower the cost of building a new system by realizing economies of scale. Enabling in situ applications on diverse hardware has the potential to increase usability and availability of diverse features such as tensor processing in hardware for scientists.

Realizing efficient performance in HPC and other software stacks will enable greater access to system services for in situ applications and will increase the number of applications that can run in situ. Making system services available in ISDM tasks rather than tasks rewriting those services can lead to efficient algorithmic performance and reduce duplication of developer time. Efficient reuse of system services can save power and time and ultimately accelerate the pace of scientific discovery and promote verifiable science.
2.3 In Situ Algorithms

Redesign data analysis algorithms for the in situ paradigm.

Key questions How should in situ algorithms be designed to make most of the available resources? What new classes of data transformations can profit from in situ data access in the presence of constraints imposed by other tasks? What algorithms are needed for multiscale, multimodal, and multiphysics in situ coupling of tasks and data?

Research opportunities The capability to access every datum of a computation or an experiment poses unique opportunities and challenges for algorithm design, both for traditional visualization and analysis and for ML and AI. The in situ environment for data processing and analysis differs substantially from the post hoc environment, requiring fundamentally new algorithms and approaches. Analysis and processing tasks execute on streaming data in a dynamic, resource-constrained environment; and algorithms that are scalable and intelligent are needed to exploit the high spatial resolution and temporal fidelity of in situ data. However, limitations imposed by the coexistence of multiple in situ tasks, sequential data access, and the removal of human interaction from in situ workflows also complicate algorithm design. Portable algorithms that deliver peak performance are needed for both in situ and post hoc execution over multiple computational platforms in order to realize efficient utilization of each computational platform, maximize programmer productivity, and facilitate software maintenance.

Potential benefits Intermediate data products in memory can elicit new types of analyses on those data and enhance the power of scientific analysis. Progress will benefit from multidisciplinary approaches that holistically consider the opportunities, constraints, and user needs of in situ analysis. It is well known that in situ algorithms reduce time to solution compared with post hoc, mainly by avoiding I/O roundtrips to/from disk storage. In situ analysis also provides unprecedented access to every value computed by a simulation or generated by an experiment, provided that algorithms are scalable and intelligent and can utilize this capability to the fullest extent possible. The ability to assimilate multimodal and multiscale data in situ would provide a comprehensive view of scientific phenomena.

2.3.1 Key Challenges and Opportunities

Data analysis algorithms that operate in situ have unique characteristics and requirements compared with those designed for post hoc execution. Because in situ algorithms can potentially be scaled to the full spatiotemporal resolution of data being produced by the application, at the rate being produced, efficient in situ operation requires advances in scalability, platform portability, and novel algorithm design.

Sequential access to every data point presents both a challenge and an opportunity for in situ algorithm design. In the traditional post hoc regime, a relatively low-frequency sample of time steps (e.g., checkpoints or analysis output files) is stored for the entire duration of a simulation. Therefore, analysis algorithms (e.g., for feature detection and tracking) are designed for high spatial resolution, low temporal resolution, and random access to time steps. The in situ regime is different. Analysis algorithms have access to full spatial and temporal resolution—every datum at every time step—but with sequential access in the time dimension. Random access to time steps in the past or the future is not available. Hence, algorithms need to plan and store data (e.g., in memory or NVRAM) if multiple passes are needed or several time steps are required to, for example, track a feature over time (Figure 9).

Resource constraints encountered in the in situ setting present time and space challenges for analysis algorithms and open opportunities to rethink whether full precision is always necessary. The overall budget for a workflow execution (e.g., total number of core-hours allocated to the run) dictates the required wall-clock time and memory footprint available for each constituent task. These constraints on the analysis tasks may motivate the redesign of analysis algorithms to minimize data movement and energy or conserve other resources.

Additional complexity is caused by the removal of human interaction from in situ algorithms. Users have become accustomed to interactively exploring datasets post hoc. This sort of manual “human in the loop” interactivity is difficult to architect in situ. Computational steering based on currently available results is one approach [Sanderson et al., 2018]. Even with this capability, however, a degree of separation exists between
the human and the machine, requiring increased autonomy in analysis algorithms. The increased diversity and complexity of in situ workflows and science use cases require informed decision-making capability regarding the control of analysis, simulation, and/or experimental tasks. This algorithm control capability is expected to feature more autonomy and less human intervention than in the past.

An evolving diversity in system platforms is driving a corresponding increase in the diversity of science use cases. Generation of new outputs on new system platforms presents additional challenges and opportunities, and algorithms need to address challenges across both the system platform and science use-case axes. Algorithms must be performance portable, and new algorithms will need to be developed to support emerging in situ science use cases, for example, assimilating computational models with observations and experiments in real time. Climate models initialized by weather observations are one example [Kondo et al., 2017]; integrating molecular dynamics and finite-element simulations with high-energy x-ray experiments are another [Cherukara et al., 2016]. Opportunities also exist to use intermediate data representations and derived data products for analysis that otherwise would not be available post hoc. An example is a Voronoi tessellation [Peterka et al., 2014] used for resampling the density of a point cloud of dark matter particles in cosmology simulations [Peterka et al., 2016] onto a regular grid. The tessellation can be used in situ to compute histograms of cell volume or to extract features based on cell statistics [Peterka et al., 2012]. In contrast, only the final output, which is the density estimation, would be available post hoc because the Voronoi tessellation is too large to save.

Figure 9: In situ feature tracking of supercurrent vortex core lines in the simulation of a superconductor (image courtesy of H. Guo).

2.3.2 State of the Art

Reduced representations have been used for data analysis and surrogate modeling to mitigate large data size, high dimensionality, or long computation times. For example, low-rank approximation is one way to reduce dimensionality and data size while preserving salient features [Austin et al., 2016]. Other approaches feature statistical [Hazaria et al., 2018, Thompson et al., 2011, Biswas et al., 2018, Dutta et al., 2017], topological [Morozov and Weber, 2013, Morozov and Weber, 2014, Gyulassy et al., 2012, Gyulassy et al., 2019, Landge et al. 2014], wavelet [Li et al., 2017, Salloum et al., 2018], compression [Di and Cappello, 2016, Lindstrom, 2014, Brislawn et al., 2012], and feature detection [Guo et al., 2017] methods. Surrogate models and multifidelity models can be geometric, for example, fitted by piecewise-continuous approximations [Peterka et al., 2018, Nashed et al., 2019], or statistical, for example, employing Gaussian processes [Lawrence et al., 2017, Lohmann et al., 2017].

The ability to steer a computational model and conduct exploratory analysis while a simulation is running has long been an objective of the in situ visualization community. Early work includes coupling a multiphase reservoir simulator with visual data analysis and exploration tools for user-guided problem setup and computational steering [Bethel et al., 1994]. Other early work dates back to 1995 with the SciRun [Parker and Johnson, 1995] system. In 2018, the Uintah code was coupled with VisIt through a dashboard [Sanderson et al., 2018] offering steering capability.

Several in situ workflow infrastructures exist [Dorier et al., 2016, Dreher and Peterka, 2017, Lofstead et al., 2008, Ayachit et al., 2016b, Larsen et al., 2017, Whitlock et al., 2011, Ayachit et al., 2015], but diversity over computing platforms and use cases is still limited. A dichotomy remains between in situ and distributed workflows [Deelman et al., 2015, Friedman-Hill et al., 2015, Yu and Buyya, 2005, Deelman et al., 2009] despite efforts to integrate these communities [Deelman et al., 2016]. Even though heterogeneous hierarchical combinations
of multiple workflow models are starting to appear in the research literature [Yildiz et al., 2019], workflows in practice are still limited to single use cases such as visualization of VTK data.

Fairly recent advances have increased support for adaptive, data-driven in situ workflows that minimize the need for a human in the loop. Some of these works identify time steps of interest for deeper analysis [Larsen et al., 2018, Bennett et al., 2016, Salloum et al., 2015, Woodring et al., 2011, Dutta et al., 2018]. Other works aim to identify input parameter values based on simulation state [Weber et al., 2007] enabling autonomous deployment of algorithms that historically have required a human in the loop.

2.3.3 New Research Directions

In situ data analysis workflows impose resource constraints on the individual analysis algorithms. These limitations motivate algorithms that use approximate and reduced representations, low-rank methods, functional approximations, and the combination of low- and high-fidelity surrogate models. Single-pass, multipass, and sliding window methods of accessing streaming data in situ need to be developed and combined, based on the algorithm. Moreover, the accuracy and quality requirements and guarantees of approximate approaches need to be quantified and validated in the context of the workflow. Even though uncertainty is compounded as more tasks are connected in a workflow, each task introducing additional approximation error, quantifying uncertainty and validating correctness in a workflow task graph have not been adequately addressed in the past.

Research is required to modify existing post hoc algorithms and develop new in situ algorithms to satisfy the needs of modern use cases on emerging system architectures that can feature massive scale, many cores, deep memory hierarchies, or embedded lightweight edge devices. Required are performance-portable algorithms that can be productively deployed by scientists for analyzing real-time, noisy, streaming data from physical experiments or sensors, in conjunction with traditional simulation models.

In addition to traditional simulations, data may stem from experiments, observations, multiphysics simulations, and/or ensemble workflows. Outputs will be increasingly high-dimensional, multifidelity, and multimodal and may be available only in a streaming fashion. Examples of analyzing multiple data modalities and runs include integration of experimental and simulation data, ensemble data analysis, intelligent parameter setting in the absence of a human-in-the-loop, the reconciling of the differences between multifidelity simulations, and the use of high-fidelity simulations to inform low-fidelity simulations.

New classes of data analysis algorithms in ML [Baker et al., 2019] and AI are needed to augment traditional visualization, topological, and statistical analyses. Examples include graph analytics (e.g., clustering, community detection) and semi- or unsupervised methods featuring reinforcement learning or transfer learning. Although these methods exist in other contexts, applying them to scientific data, as opposed to images, on emerging platforms (both HPC and edge devices) remains elusive. Redesigning AI algorithms for in situ execution is crucial because existing algorithms that rely on massive amounts of training data and require long training times cannot be run, in their current state, in situ.

Algorithms need to be designed around common and custom data models that provide both uniformity and extensibility. Moving from ad hoc data models to shared, adaptable data models enables not only new capabilities for simulations but also better connection to a wide array of data science analysis and learning capabilities. Augmenting data models with metadata indicating expressed intent or desired quality of service can inform algorithmic choices concerning level of accuracy or amount of approximation in the algorithm.

Algorithmic development will incorporate multiple programming and execution models. Bulk-synchronous processing, while straightforward and effective in many instances, can perform poorly or be difficult to employ when computational or communication load is imbalanced or when the problem is highly irregular or does not allow iterations with strict synchronization points. New research directions include developing algorithms using flexible and dynamic programming and execution models and the composition of multiple such models in the same in situ workflow.

Assumptions and dependencies: In situ analysis algorithms will be dependent on and require advances in software composability (§ 3.7), resource management (§ 3.3), data models (§ 3.2), programming and execution models (§ 3.6), and provenance (§ 3.5).
2.3.4 Potential Scientific Impact

The impact of researching in situ algorithms is multifold. In situ algorithms, compared with post hoc, reduce data and time to solution. Both human and machine resources are conserved as a result. In situ algorithms enable real-time debugging and computational steering. Rapid feedback and dynamic control of the workflow allow informed decision-making in terms of both the computer science and the application science. The workflow can be optimized, and scientific insights can be realized because of real-time selection of critical features and attribution of events.

The ability to see every single data point in full spatial and temporal frequency provides unequaled data fidelity. Access to derived and intermediate data products can further our understanding. Multiscale, multiphysics, and multimodal data are crucial to DOE science that spans a range of scales from quantum to continuum to phenomenological, and the ability to assimilate multiple data modalities can enable an integrated understanding of phenomena across scales that was not possible when data were studied separately. New classes of AI algorithms can provide surprise findings because they are driven by the data, producing results that we could not predict.

2.4 Controllable ISDM

Understand the design space of autonomous decision-making and control of in situ workflows.

Key questions What metrics best describe the ISDM design space? How can that space be defined, codified, and evaluated to support design decision-making and control? How can we dynamically adjust the organization, placement, and utilization of data to improve performance and satisfy user requirements?

Research opportunities Decisions concerning scheduling and placement of computations, choosing data structures and algorithms, planning for reliability, and satisfying user requirements create a complex design space for ISDM. Understanding that space of ISDM parameters is crucial to making intelligent design decisions, both by humans and autonomously. The capability to optimize a constrained ISDM design space will enable predictable performance and scientific validity. Codifying the design space and developing metrics and benchmarks to evaluate it will also promote sharing of metrics and parameters across communities. The capability to optimize such a constrained ISDM design space will enable predictable and repeatable performance and scientific validity.

Potential benefits Some analysis decisions in the in situ regime depend on the results of earlier computations, requiring autonomous control of the scientific workflow and of data and resource management. Automating workflows can save human and machine resources because scientists can focus on domain-specific challenges, and computing platforms can be used more efficiently than with human control based on trial and error. Capturing the right information automatically about the workflow can lead to improved understanding of the workflow performance and of the science results. The development of performance models and other standardized metrics to evaluate the control operations will promote the sharing and reuse of information and the training of computer and domain scientists.

2.4.1 Key Challenges and Opportunities

ISDM faces an increasing diversity and complexity of in situ use cases and platforms. Many of these cases require the ability to make decisions based on prior computation. In the past, control decisions could be made a priori by humans because workflows were simple (e.g., one simulation and one visualization) and had static resource requirements that were known in advance. This is no longer true, as tasks are becoming more difficult to control manually (e.g., deep learning hyperparameter optimization) and because the overall workflows are becoming larger and more complex, featuring multiple data sources and sinks with dynamically changing requirements. Automated, rather than manual, control of in situ workflows is needed, based on rigorous, explainable, reliable, and trustworthy decision-making.

A key challenge is determining whether to derive control logic from the bottom up (by generalizing specific examples) or to design it from the top down (by implementing analytical models), or something in between. A bottom-up data-driven controller would be built from training data used to learn the behavior of the system. Deep neural networks (DNNs), for example, can capture nonlinear behavior that is difficult or impossible to
describe analytically. Much training data and training time, however, are needed. Another potential limitation is assessing whether DNNs can be trusted to automate decision-making, especially when encountering unanticipated inputs that are different from the training data.

Top-down control can take the form of equations that model the system behavior (e.g., the time to complete a task as a function of problem size and number of resources). The analytical models can be augmented with hints or requirements describing user preferences or needs. Desired quality of service or time limitations to complete a task are examples. Agreements on quality of service can be used as constraints when optimizing the parameters of the analytical model. In the case of bottom-up control, there is less opportunity for constraints to be included in, for example, a deep neural network (DNN).

Control decisions have implications on the quality of service and corresponding quality of results. Either criterion—service or results—can be stipulated as a constraint to the controller, either as a hard constraint (the metric must be met) or as a soft constraint (a hint for best effort). In either case, one must understand the tradeoffs between quality of service and quality of results in order for humans and the control system to make those decisions judiciously. For example, a visualization can often be delivered faster at lower resolution or slower at higher resolution. Balancing the tradeoffs depends on various factors, and some visualization systems will generate a low-quality preview quickly, followed by a high-quality image later. This simple example illustrates the need for specifying quality constraints, system responses, and sensible default behaviors in the absence of more information.

### 2.4.2 State of the Art

The control of an in situ system consisting of two tasks, one simulation coupled to one analysis or visualization, is often called computational steering [Sanderson et al., 2018]. Current and future workflows can incorporate many tasks coupled in a directed graph that can include multiple cycles; hence, control of ISDM systems is considerably more challenging and requires a greater degree of automation.

Several categories of algorithms can be used to determine whether something “interesting” deserves further attention, such as spawning an additional task. One body of research focuses on “quick sketches” used to trigger some action [Larsen et al., 2018, Bennett et al., 2016, Salloum et al., 2015, Woodring et al., 2011, Dutta et al., 2018]. Topological methods [Morozov and Weber, 2013, Morozov and Weber, 2014] and mesh integrals [Guo et al., 2017] can be applied to detect and track critical points, regions of interest, or other features. Information-theoretic methods [Biswas et al., 2013, Wang and Shen, 2011] can quantify the overall information content of a temporal or spatial interval, and changes in information entropy can indicate potential areas of further investigation. Machine learning methods [Wozniak et al., 2018, Kurth et al., 2018, Joubert et al., 2018] can elucidate features that cannot be described by other methods.

Defining a design space is a prerequisite to optimizing its control. A common taxonomy for in situ parameters was begun in 2016 [Childs et al., 2016], with certain definitions such as time partitioning and space partitioning replacing older terms such as in situ versus in transit. Today, we consider both tasks that share the same memory space and tasks in separate spaces part of ISDM. The objective of this taxonomy was to align terminology of the in situ user interface. It does not, however, address implementation details of the underlying system, such as what communication mechanism or scheduling algorithm can be used to generate the desired result.

Conveying user-defined constraints can tailor the autonomous control of an ISDM system to the intentions of a user. User intent and constraints in an in situ system can be expressed in at least three ways. The first is through a file I/O interface similar to the way that application scientists write checkpoints and data files for post hoc analysis. File formats used for this purpose are the native BP format used in ADIOS [Liu et al., 2014] as well as NetCDF [Davis et al., 2017] and HDF5 [Folk et al., 1999] formats. The second way is through generic data containers that are created through an API inside the user’s tasks [Dreher and Peterka, 2016] or in data contracts [Mommessin et al., 2017, Donier et al., 2017] outside of those tasks. The third is through service-level agreements (SLAs) that express and match the user’s intents and constraints on how data are to be used with metrics and mechanisms for evaluating the usage. The Common Object Request Broker Architecture [OMG, 2000] is one example of intent-matching subscriptions in an event management system. Self-describing and extensible interfaces, for example, Scientific XML [Widener et al., 2002] and External Data Representation [Srinivasan, 1995], have been used for similar purposes.
2.4.3 New Research Directions

Research is needed to develop rigorous, explainable, reliable, and trustworthy decision-making in ISDM software. The complexity, nonlinearity, and dynamism of in situ workflows mandate augmenting or replacing human control with autonomous control. Autonomous control would enable real-time feedback and fine tuning of the ISDM components, potentially at a much higher frequency and accuracy than possible by a human.

Optimization, whether static or dynamic, of ISDM parameters requires codifying the metrics that best describe the design space. Researchers need to agree on the minimal set of orthogonal basis vectors in a common taxonomy and language for describing ISDM. The in situ terminology project [Childs et al., 2016] was a step in that direction; but beyond agreeing on what we call something, a common framework is needed for generating and comparing test results. A set of community-wide benchmarks and test suites (i.e., miniapps for canonical workflow problems) is needed. Only then can the results of various optimization and control strategies be meaningfully compared.

Methods are needed for incorporating user intent (constraints or hints) in the ISDM design space. There are four aspects of incorporating constraints: specification, recording, execution, and provenance. Constraints may be specified by using SLAs or quality-of-service (QoS) contracts via the programming model (§ 3.6). A record of the constraints needs to be stored in the system somewhere, for example, as part of the data model (§ 3.2). The ISDM framework needs to execute the constraints in its communication and execution models. The outcome of the execution (to what extent the constraints could be honored) needs to be recorded in the provenance (§ 3.5) of the execution and linked to the impact on scientific results.

Both challenges and opportunities exist in understanding and incorporating the tradeoffs between quality of service and quality of results into the control of ISDM. Measuring or theoretically deriving the effect of algorithmic and operational parameters on performance (e.g., time to solution) and data quality (e.g., amount of error introduced) is a necessary step. Mapping data quality to scientific quality (e.g., confidence interval) is also required, but this is an open problem. Research is needed to understand this relationship. Eventually, we need to generalize those mappings over various science domains and codify them into a small set of control parameters of the ISDM system, independent of application domain. Sensible default values for those control parameters also need to be determined.

Assumptions and dependencies: Controlling ISDM corresponds to complex in situ workflows not currently supported by programming and execution models (§ 3.6). New data models (§ 3.2) are needed to capture provenance, intent, and other user constraints. New analysis algorithms (§ 3.4), including ML techniques, are needed to control ISDM and ensure reproducibility (§ 3.5).

2.4.4 Potential Scientific Impact

Investigating research directions for controllable ISDM can yield numerous benefits. Automating workflows that previously required human intervention can save human and machine resources. Judicious use of system resources will result from successful dynamic control of in situ workflows. Human effort will be saved by automating mundane control and optimization tasks, which could require considerable trial and error, freeing scientists to focus on more interesting problems. Beyond improving existing workflows, autonomous control can also enable some in situ workflows that were impossible before.

Some of the ISDM parameters control the frequency and granularity of provenance capture, and optimization of these parameters can produce selective intelligent provenance. Capturing the right information can enable real-time debugging, monitoring, tuning, and attribution of events.

Other parameters control the functionality of data analysis algorithms, enabling automated rapid or real-time selection of critical scientific features. Critical analyses can be performed with predictable and repeatable performance and with scientific validity.

Performance models, benchmarks, and test suites also promote sharing of information across communities and reuse of workflows. Being able to reproduce performance and scientific results is required for rigorous science, as well as for training the next generation of scientists.
2.5 Composable ISDM

Develop interoperable ISDM components and capabilities for an agile and sustainable programming paradigm.

Key questions Can the composition of ISDM software components maximize programmer productivity and usability? What design decisions of ISDM software components promote their interoperability in order to ensure the long-term utility of ISDM software for the science community? How can we eliminate the burden on users wanting to transition current analysis and processing methods from post hoc to in situ?

Research opportunities Long-lived sustainable ISDM frameworks that are adopted and used by the science community require being able to compose in situ workflows from modular interoperable building blocks. The flexible composition of interoperable ISDM software components will enable developers and end users to choose from an array of widely available tools, thereby increasing productivity, portability, and usability, and will ultimately result in agile and reusable software.

Potential benefits Increasing the number and breadth of analysis tools to bring to bear on science problems would be a direct result of interoperable and composable ISDM software. Encouraging users to rely on reusable ISDM software frameworks developed and maintained for general use would provide measurable cost savings across the DOE Office of Science because individual application teams would not have to reinvent software infrastructure that is needed across the board. Such a model requires, however, that ISDM infrastructure be easy to use, robust, and sustainable. Given appropriate research in the design of interoperable software components coupled with support through computing facilities and industry partnerships, ISDM software would avail application scientists of all the advantages that the in situ computing model offers.

2.5.1 Key Challenges and Opportunities

One driver of ISDM software design is the wide variety of use cases, ranging from computational models to experiments in user facilities to sensor data in the field (Bethel et al., 2016). Another is the diversity of computing platforms that need to be supported, from leadership-class supercomputers, to departmental computing clusters, to individual workstations, to devices embedded in detectors (Vetter et al., 2019). In addition to increasing diversity in use cases and underlying hardware, a third motivation comes from different software stacks and software environments used in HPC, industry, science, and engineering, and big data. For example, computer and computational scientists tend to develop software in C++. Experimentalists often use Python. MATLAB and R are popular among applied mathematicians, statisticians, and data analysts. Spark is popular in the big data community. The AI community builds deep learning networks using tools such as TensorFlow or Keras. More than just different programming languages or tools, each of these frameworks depends on different underlying libraries and software stacks. Scientists tell us (§ 3.1) that even though in situ data management can be the bridge between different communities, the challenges to bridging these communities are significant and affect many science domains.

Composable ISDM software would presumably consist of components or modules in one or more ISDM software frameworks, as opposed to custom in situ solutions in each application. The granularity and form of those modules (libraries, standalone tools, etc.), the conventions imposed by the framework (e.g., uniform data model), and the degree of abstraction between the user and the ISDM functionality are open research questions. In other words, the ability to deliver capabilities in an established framework that covers a wide range of use cases is not trivial. Common data, programming, and execution models with clearly defined interfaces and operations are desired; but history tells us that one size does not fit all (e.g., Common Component Architecture [Kumfert et al., 2006]), and such standards are difficult to extend beyond a single application community (e.g., ROOT in the HEP community [Antcheva et al., 2009]).

Rather than a strict top-down organization, a more practical approach may be a functional decomposition of the core ISDM capabilities, with various offerings for each capability, coupled in a loosely compatible way that allows diverse modules to communicate in a common language, in addition to their own native APIs. Designing such an ISDM software ecosystem remains an open problem requiring high-level architectural design. These are computer science research questions beyond software engineering.
ISDM software must be easy to use if it is expected to be adopted by scientists. Usability refers to how well a software tool performs a specific function and to what degree the tool insulates the user from complexity. In a composable ISDM ecosystem, another measure of usability is the ease with which software products can be combined. However, the diversity of use cases, hardware, and software stacks can reduce usability. Research is needed to overcome these challenges.

ASCR must also continue to strive to improve the dependability of ISDM software. One of the main impediments for science teams adopting others’ software is lack of confidence that the software is well designed, implemented, tested, and maintained. Software engineering practices of quality assurance and testing can produce higher-quality ISDM software. While component-level testing is reasonably well understood, testing of multistage ISDM workflows, especially those involving experimental/observational facilities, is significantly more complex. Testing performance at scale is another challenging aspect, requiring performance analysis, characterization, and modeling to understand the difference between expected and measured performance. Leadership-computing resources are required for full-scale performance tests. Also needed is multistage testing, from coarse grained to fine grained, including data movement between tasks, storage traffic, memory footprint, and operation counts in heterogeneous processing elements.

The reuse and inclusion of software tools from both ASCR-funded as well as third-party sources are necessary in order to realize the future of ISDM. Reusing tools from elsewhere (e.g., software from industry) is needed in order to integrate more machine learning and data science into DOE applications. Deep learning and AI are relatively new directions in DOE science, and capitalizing on software from other communities in these areas would be advantageous. Challenging problems include how to bring that software into the HPC ecosystem, demonstrate scalability, and integrate in situ into DOE applications.

Another challenge is to sustain and grow ISDM software over a long lifetime, for example, spanning several new machine versions over 15–20 years. Sustainability refers to several different concepts [Venters et al., 2014], including preserving the function of a system over a defined time span, maintaining the ability to modify a software system or component after delivery, and providing community support for adoption over time.

2.5.2 State of the Art

ISDM software design is motivated by science use cases, and the increasing diversity of ISDM applications described in § 2.1—observations, experiments, ensembles, edge computing, multifacility federated workflows, to name a few—require ISDM software that is available, stable, reusable, and maintainable.

Today, however, there is a lack of widespread tool and data model interoperability. There is consistency at the level of specific domain science communities, such as the ROOT programming and data model in HEP [Antcheva et al., 2009]. However, ROOT is designed for usability and portability but not for HPC in situ scalability. That is why present efforts in the DOE’s SciDAC program are addressing rewriting HEP codes using scalable software usable by multiple domains [Fermilab, 2019].

Other forms of interoperability are possible through files, data models, and programming models. “File-level” interoperability is made possible through common file formats, for example, NetCDF in the climate community [Davis et al., 2017]. “Data-level” interoperability is achieved through common memory models, for example, NumPy arrays. Data-level interoperability facilitates “programming-level” interoperability with a consistent user-facing API through front-end “bridge” adaptors such as SENSEI [Ayachit et al., 2016b], where a common data model, VTK, underlies the bridging.

Recent efforts in ASCR and the ECP have begun to address software development and deployment standards. The IDEAS [McInnes, 2019, Balay et al., 2016] project was an early attempt to make math libraries interoperable by enforcing consistent guidelines for their development and delivery. These efforts have been extended through the ECP xSDK [xSDK Project, 2019, Bartlett et al., 2017] standards to encompass all ECP software tools. Delivery of ECP software is being done through the Spack [Gamblin et al., 2015] package manager as well as through containers such as Shifter [Kincade, 2015] and Singularity [Argonne Leadership Computing Facility, 2019].
DOE supercomputing facilities—NERSC, OLCF, and ALCF—are beginning to support features needed for ISDM, although much work still remains. For example, NERSC’s upcoming Perlmutter system will be the first machine specifically designed for real-time data analysis [National Energy Research Scientific Computing Center, 2019a]. Other features, however, such as inter-job communication and dynamic process management, are still not available on DOE’s flagship resources.

### 2.5.3 New Research Directions

An ISDM ecosystem built from composable software modules would benefit from community guidelines for broadly accepted specifications for ISDM software. In order for these conventions to represent a broad cross-section of DOE science use cases, such accepted common conventions should not be derived from the needs of a single science program. Rather, a program such as ASCR, which is central to the other science programs—BES, BER, HEP, FES, NE—can develop such conventions, with input and ownership from the other programs. ASCR research can enumerate representative use cases for ISDM to drive software requirements and needs. The goal of such an activity is to define a minimal set of community guidelines for interfaces and data models to promote interoperability. Such guidelines must strike the right balance between commonality, generality, and extensibility. The resulting specifications would enable the design of modular software architectures that use best engineering practices, combined with a clear focus on science mission needs.

Continued research is needed to design middleware (also called frameworks or infrastructure) to shield users from implementation differences and to provide platform portability. This middleware should be designed based on the lightweight independent functional decomposition of software pieces, at multiple granularities, from very coarse (entire applications) to very fine (single-purpose libraries). The decomposition should allow the ISDM community to exploit third-party software tools from other software ecosystems. The ability to include external software would free ISDM developers from reinventing existing functionality and allow them to focus their attention on more complex problems.

ISDM software must employ software engineering practices of quality assurance and testing. Component-level testing, testing at scale, performance analysis and characterization, and multistage testing of entire workflows over distributed area are all needed. While unit testing of individual tasks is well understood, full-scale testing of entire complex workflows requires research. Lacking are ISDM workflow test benchmarks or suites and sufficient performance models (analytical and/or empirical) to predict expected behavior. The lack of test workflows (the equivalent of miniapps for single applications) is also an impediment for vendors to be able to co-design systems for ISDM (§ 2.2). Testing on different platforms requires installation on those platforms. Also needed are deployment strategies for packaging and delivering software over combinations of OS/R versions, compilers, and software dependencies. These deployment strategies will also be needed by users and facilities who need to use the installed software in their daily work.

In addition to outreach and training for users, ASCR-funded research should target working with science facilities (computational and experimental) and industry partners to develop, deploy, and support ISDM software over an extended lifetime of up to 20 years. Deeper engagement with DOE computing facilities is necessary in order to deploy software and define use policies for ISDM (such as on-demand job queues or opening network connections with remote facilities). Facilities can also help with education, outreach, and training for users. Long-term sustainability also requires partnering with vendors and industry to eventually subcontract ongoing software testing and maintenance, small feature additions, and new hardware support over a longer period of time beyond the initial ASCR research activities. Typically, this transition from active research and development to maintenance occurs five or six years from the start of the research and can continue for the next 10–15 years. Such longevity is required in order for ISDM software to be widely used in DOE.

**Assumptions and dependencies:** Composable ISDM is tied to research in programming and execution models (§ 3.6), data models (§ 3.2), operational policies at HPC facilities, distributed workflow management systems, data-streaming support, provenance (the reproducibility of software reuse, traceability, transparency, § 3.5), and software architecture (§ 3.7).
2.5.4 Potential Scientific Impact

Research in composable ISDM would have wide-ranging benefit in three broad areas: user productivity, scientific confidence, and return on investment.

First, meeting the challenges outlined above would increase productivity. Increasing the interoperability of software tools would increase ISDM software usability and would increase availability of a broad array of tools for users to select. Combined, these advantages would reduce cost and increase productivity for science code teams and ISDM researchers and developers alike.

Moreover, high-quality ISDM software could encourage broad use and adoption of ISDM technologies because of the increased confidence in the software infrastructure. Ultimately, the availability of high-quality thoroughly-tested software would increase certainty in the scientific conclusions drawn from in situ workflows.

Improving ISDM software sustainability would also maximize ASCR’s return on the initial investment in research and development of that software. Amortizing the initial cost over a longer lifetime and getting a longer return on that investment by adding a moderate cost for maintenance can be less expensive than developing a new product every five years.

2.6 Transparent ISDM

*Increase confidence in reproducible science, deliver repeatable performance, and discover new data features through the provenance of ISDM.*

**Key questions**

How can provenance and metadata support data interpretability, discovery, reuse, and reproducibility of results? How can these artifacts be captured automatically and analyzed in situ, at the scale of DOE science?

**Research opportunities**

A recent National Academies of Science and Engineering report [National Academies of Sciences, Engineering, 2019] defines reproducibility as obtaining consistent computational results using the same input data, computational steps, methods, code, and conditions of analysis. Replicability, a broader concept, implies obtaining consistent results across studies performed by different teams answering the same scientific question, each using its own data.¹ Provenance and metadata are needed for both reproducibility and replicability. In situ provenance is crucial to understanding scientific results, assessing correctness, and connecting underlying models and algorithms with workflow execution. The capability to capture and analyze provenance data within the time and space requirements of the domain science can both improve ISDM performance and ensure scientific validity. Also needed is an understanding (and quantifying) of the data uncertainties in the underlying computational models and algorithms, in particular understanding how uncertainties are compounded and propagated by multiple in situ tasks. The ability to capture and query provenance and metadata at scale and in situ will support replicability and reproducibility, post hoc analysis, data discovery, performance diagnostics, and many other science needs.

**Potential benefits**

Efficient capture of provenance information coupled with in situ analysis of the captured metadata would increase confidence in scientific conclusions and accelerate performance of in situ workflows. Workflows that are validated through the provenance of data would be more trustworthy, reusable, and explainable than those whose data lineage is unknown. Confidence in workflow results can increase collaboration within and across science teams. Understanding captured performance metrics also can lead to predictable and repeatable execution of workflows, ultimately reducing data size and shortening time to solution.

¹ The ACM swaps the definitions of replicability and reproducibility [Heroux et al., 2018]. In this report, we use reproducibility and replicability consistent with the NAS definitions.
2.6.1 Key Challenges and Opportunities

Motivations for collecting and analyzing provenance in situ include validating scientific results and investigating performance variability across runs. Levels of replicability and reproducibility are often defined by community thresholds that provide different goals in different domain sciences. Numerical reproducibility is used in some domains such as climate, for example, to test new code features in large distributed teams. Other scenarios and sciences may employ less strict interpretations of replicability and reproducibility.

Provenance capture at the scale of DOE science is a challenging problem. Large volumes of diverse provenance data can potentially be produced by heterogeneous in situ tasks and the ISDM framework [Vetter et al., 2019]. The volume, complexity, resource constraints, and impact on other tasks imply that the brute-force approach of capturing all data at the same level of granularity will not work in most cases. This would result in enormous data size, which would be impossible to manage, store, process, query, or analyze. Smarter approaches are needed to customize capture to specific architectures or desired uses of captured data, coupled with in situ triaging to decide what to keep. Methods for filtering captured data, deciding importance, querying and analyzing provenance, and relating data artifacts with the application code and system environment provide new opportunities for research.

Introspection of provenance can facilitate debugging and code optimization [Pouchard et al., 2017]. Provenance introspection and analysis have traditionally been performed offline and post hoc. However, in situ tasks and in situ workflows need to make informed decisions and adapt in real time to detected anomalies and other features present in provenance profiles. Therefore, in situ processing of provenance is needed in order to be useful for ISDM, in other words, while an in situ workflow is still executing. In situ processing of provenance can also reduce the volume of data collected, addressing the problem of scalable provenance capture. Processing provenance data in situ faces many of the same challenges as other types of in situ analysis: additional complexity involved in execution, code optimization, and debugging due to increased heterogeneity and performance variability. Collecting and analyzing performance provenance in situ will enable smarter compute placement and have the potential ultimately to provide substantial energy savings.

Provenance data can have a potentially long lifespan, longer than just the time that a workflow is currently executing. These data need to be searched, queried, and audited over that lifespan. Post hoc analysis of provenance data usually requires connecting data with the software that produced them and determining the scientific implications of the provenance. This requires capture and in situ analysis of additional metadata: not only raw performance numbers, but references to user tasks, functions, lines of code, and system information, so that the graph of an execution (i.e., the informative links between these components) can be reconstructed afterwards.

Provenance can aid in discovering and trusting scientific results. Except for a few science domains and particular experiments, however, scientific data products cannot be easily searched or shared by the scientists who were not involved in the experiment or simulation. Performing composite searches across multiple datasets or sharing data between scientists is also difficult. This is particularly true of provenance that is largely application oriented and typically meant to meet the immediate needs of a particular experiment or study.

In ISDM systems where data undergo multiple lossy and irreversible transformations, replicability and reproducibility will need to be studied more carefully. For example, quantifying uncertainty that is compounded by the composition of multiple in situ tasks has not been adequately addressed. Moreover, deeper integration of ML in scientific computing poses new challenges and opportunities for replicability and reproducibility. Unpredictability and sensitivity are increased when ML is used in ISDM, and traceability becomes more difficult in such situations. Provenance collection and analysis can help with transparency and explainability of ML results.
2.6.2 State of the Art

Today, no tools capture provenance of in situ workflows or of ISDM frameworks. Related efforts in other domains are discussed below.

Numerous standards and tools for provenance exist in communities other than the HPC and DOE communities, but they are not always applicable to high-performance, heterogeneous, or edge computing. For instance, the W3C-Prov [World Wide Web Consortium Working Group, 2013] collection of standards specifies provenance “to enable the interoperable interchange of provenance information in heterogeneous environments such as the Web.” Standards for smart sensors such as IEEE 1451 and the Open Geo Consortium Sensor Web Enablement [Pouchard et al., 2009] have been developed that encode the provenance of a sensor signal.

Today, there is a patchwork of profiling and capture tools from various sources, each independently profiling specific telemetry data, such as HPC Toolkit [Tallent et al., 2008] and SONAR [Lammel et al., 2016]. For example, TAU [Shende and Malony, 2006] and ScoreP [Knüpfer et al., 2012] extract performance profiles from applications, but these tools are not optimized for large-scale workflows, nor do they collect comprehensive provenance information required for detailed introspection and analysis. Other tools, such as XALT [Agrawal et al., 2014], are developed for system monitoring at HPC facilities, and they do not extract provenance. The Chimbuko [Pouchard et al., 2018] system supports workflow-level performance analysis, but provenance is not analyzed in situ.

Tools for replicability and reproducibility that collect provenance, such as ReproZip [Chirigati et al., 2013] and initiatives that facilitate reproduction of studies based on provenance, such as the Open Science Framework [Center for Open Science, 2011], are useful but may not scale and do not support ISDM.

Instrument-specific solutions from experimentalists do exist, but they are application specific and not general purpose. For instance, Bluesky, the data acquisition and management software developed at NSLS-II, keeps track of experimental conditions at the detectors for each beamline, but it does not track provenance for the analysis of derived data [Koerner et al., 2019, Allan et al., 2019].

Some related work in the storage of metadata comes from the I/O community. Provenance in the storage systems context is a description of the computing environment or of the I/O subsystem in particular. Tools such as Darshan [Snyder et al., 2016] and TOKIO [Lockwood et al., 2018] measure the performance of the I/O system and applications’ interaction with it. Currently, however, no storage system solutions link provenance-related metadata with the associated scientific datasets. HPC facilities use tools such as an automatic library-tracking database [Fahey et al., 2010] to track which libraries are linked with which applications. Graphs have been explored [Ames et al., 2013, Dai et al., 2014] for provenance storage, but these efforts are small-scale and not deployed in production systems. Participants of a recent DOE workshop on storage systems and I/O [Ross et al., 2019] suggested research in hierarchical, domain-specific namespaces to support “scientists’ view of the data.” The workshop also identified the need for lightweight metadata capture and storage, possibly in a different format from scientific data (e.g., in a database).

Provenance systems exist that can alert developers of anomalous system behavior, but such systems require post hoc analysis, for instance in the security domain [Pasquier et al., 2018], and for Spark dataflows [Interlandi et al., 2015].

State-of-the-art provenance capture is I/O intensive, however, and places undue burden on applications. Research has shown [Singh et al., 2016] that supervised ML algorithms trained post hoc can alleviate the in situ I/O burden of provenance collection by performing intelligent triage.

In situ visualization and analysis tools such as ParaView and VisIt [Ayachit, 2015, Childs et al., 2012a] are not designed for provenance data, and it is not clear whether they can manage the volume and complexity of these data. One visualization tool that displays the provenance of visualization analysis tasks is VisTrails [Silva et al., 2007].
2.6.3 New Research Directions

Scalable and portable provenance capture is needed for ISDM applications. The potentially large volume of captured provenance data will require selecting provenance according to goals or desired levels of granularity and adjusting the level of capture so that provenance does not dominate the data management budget. Scalable compression of provenance data can augment judicious selection, and developing compression methods tailored for provenance is another potential research avenue. A principled approach guided by ML methods that predicts anomalies can both reduce the collected provenance data and increase the granularity at the predicted point of focus. Not being able to keep all provenance data will require rethinking analysis and data retention methods, requiring changes at the technical level, in facility policy, and in user expectations. Also required are methods for selecting between synchronous collection of metrics and execution placement, and asynchronous collection and communication of small data items that are latency bound rather than bandwidth bound.

In situ provenance processing is necessary in order to provide decision support for ISDM frameworks and the tasks running in them: for example, to tune the execution parameters or handle anomalies in real time. New algorithms are needed to analyze provenance in situ and enable real-time control. Artificial intelligence and complex event processing can be applied to decision support for real-time anomalies in a decentralized fashion [Radovic et al., 2018]; research is still needed, however, to determine efficient forms of provenance that can be used by both methods. This research could include different data reduction and enrichment techniques for the provenance dataflows. Determining which in situ algorithms to apply also requires development of cost models for provenance collection, processing, and retention.

Provenance will need to be integrated from multiple sources. Multimodal science use cases will generate multisource provenance, and these provenance data will need to be assimilated in a fashion similar to that for other data from multiple sources. Provenance introspection at varying levels of granularity will be needed because ISDM will drive wider adoption and innovation in provenance technologies across a broad spectrum of use cases. Unique identifiers that may be persisted, in streaming and in transit, are one approach to tracking multilevel provenance. The successful development and deployment of such tools will depend on the existence of standardized formats and libraries to capture provenance across numerous domain sciences. Uncertainty quantification (UQ) is closely related to provenance. Different types of uncertainty related to data, algorithms, implementation, and statistical representation require different types of provenance. New research will be needed to understand what, if any, biases are introduced by in situ algorithms, both for processing scientific data and for provenance data. In particular, extensive use of ML will pose challenges and opportunities for provenance. DNNs introduce unpredictability and sensitivity in application behavior, making reproduction and interpretation of results even more challenging. Traditional approaches to scientific validation and reproduction will have to be re-evaluated in light of the dynamic decisions made by in situ tasks that perform function approximation, analysis, or experiment design. Calculations that integrate these techniques will need specialized provenance techniques that capture relevant information about the analysis method and that can answer provenance-like questions after the computation is complete.

Provenance systems should be co-designed with computational platforms and integrated with system-level provenance systems, as well as with standards and components developed in other communities. Computational platforms have their own provenance of subsystems such as resource managers, storage systems, memory, or interconnects. ISDM provenance systems will need to interact with these platform subsystems. Integrating application-level data with system data can potentially yield new functionality and utility. These research activities pose software and data standardization challenges; however, overcoming them is necessary in order to enhance provenance collection in complex workloads.

Progress in ISDM provenance research would also benefit from recognizing work related to digital data preservation, data access, and provenance performed in other communities, such as libraries and digital archives. These organizations have developed standards, supporting software, and best practices for provenance and preservation that, while not directly applicable to HPC, could be adapted and complemented. In addition, emerging technologies such as blockchains or containers, which are essential tools for assuring the integrity of data in web and enterprise applications, may prove valuable for ISDM.
Assumptions and dependencies: Ensuring the transparency of ISDM requires corresponding advances in data models for provenance data (§ 3.2), algorithms for provenance introspection in situ (§ 3.4), and research in leveraging system-level provenance from computational platforms (§ 3.3). Transparent ISDM can also benefit from standards developed by other communities, such as digital preservation.

2.6.4 Potential Scientific Impact

Prioritizing research in transparent ISDM would improve at least three distinct aspects of ISDM usage and technology: increased confidence in results, improved understanding and scientific validity, and accelerated performance of in situ workflows.

Increased confidence in the scientific validity of the workflow leads to software reuse, because a validated workflow can be trusted and used again. Software reuse, in turn, affords improved and accelerated code development. Trusted software also leads to collaboration among and increased efficiency across facilities and user communities.

Another reason to collect information about an in situ workflow is to understand the scientific implications of the workflow output. Transparency not only increases confidence in scientific results; it also allows scientists to explain discoveries with increased scientific validity.

Provenance also facilitates predictable and repeatable in situ algorithm and workflow performance. Not only would the performance be able to be reproduced, but its comparison with models would point to bottlenecks or underperforming parts of the workflow. The end result of a properly tuned workflow execution is reduced data volume and shorter solution time.
3 Workshop Topics

The workshop agenda appears in Appendix 1. Day 1 of the workshop featured a plenary session and breakout sessions on individual topics. Day 2 began with short informative talks about other related workshops, with the remainder of the day dedicated to distilling ideas from Day 1 into the PRDs in § 2.

The plenary session began with talks from application domain scientists with the aim of helping workshop participants think about ISDM in new ways. For this plenary session, the organizers invited the following six speakers to present short talks.

- Jacqueline Chen, Distinguished member of the technical staff at the Combustion Research Facility, Sandia National Laboratories
- Ann Almgren, Senior scientist and group lead of the Center for Computational Sciences and Engineering in the Applied Math Department, Lawrence Berkeley National Laboratory
- Steve Legensky, Founder and general manager of Intelligent Light
- Daniel Jacobson, Chief scientist for computational systems biology, Oak Ridge National Laboratory
- Amber Boehnlein, Chief information officer, Jefferson Lab
- Bruce Hendrickson, Associate director for computation, Lawrence Livermore National Laboratory

All the speakers were then invited to participate in a panel for 30 minutes of open discussion. The moderator began the discussion by summarizing the talk topics and posing some leading questions. Participants and panelists discussed those and other questions for the remaining time. The dialogue served as motivation and background for the breakout sessions that followed.

For these breakout sessions, six groups were organized into three sessions of two parallel breakouts each. Each breakout group focused on a topic area from the list below:

- Science applications
- Computational platforms
- Data and communication models
- Programming and execution models
- Provenance and reproducibility
- Analysis algorithms
- Software architecture

Details of the science plenary session and the six breakout sessions are given in the following sections.
### 3.1 Science Applications

The scientific applications topic area explored commonalities among applications and examined categories of use cases that drive many of the other topic areas of the workshop, including the state of the practice that will identify gaps in ISDM. The goals of ISDM include enabling useful and insightful in situ analysis at the desired level of fidelity in order to provide feedback to the user or to automatically steer the simulation or analysis. Users are broadly defined as simulation scientists, experimentalists, computer scientists, and application developers—for example, physicists researching plasma fusion in Tokamak reactors (Figure 10)—and they often want ISDM to provide efficient utilization of in situ data for analysis with little impact on running simulations or disruption of streaming input data. Assuming such constraints can be satisfied, ISDM can provide potential benefits to applications. It can decrease the copying and conversion of in situ algorithm input and output data, provide appropriate data structures that capture necessary information, improve computational performance of in situ algorithms within the application, and potentially provide provenance and resilience.

**What ISDM capabilities are needed to best meet science application requirements?** [Gerber et al., 2018] The session speakers and attendees were asked to consider this question from several different vantage points, both in the invited talks (§ 3.1.1) and in the open panel discussion that followed (§ 3.1.2). One aspect was to identify commonalities of current and future applications that are using in situ analysis in similar ways. For example, we asked whether one can find a useful categorization of scientific applications using in situ analysis, whether common data models and/or programming models exist, and what in situ analysis algorithms and in situ frameworks typically are used for each application category.

Another aspect was to consider current in situ frameworks in terms of their usability. For example, we considered missing elements in existing frameworks that prevent wider adoption, how frameworks will need to evolve to meet the needs of science applications and architectures in the future, and what skills will be needed by science teams to develop these frameworks. We also studied what is needed in workforce development to address the needs of future applications and to address education gaps in order for applications to use in situ analysis techniques. We also considered the tradeoffs between domain-specific and generic in situ frameworks and how performance and scalability of in situ frameworks affect their adoption.

A third aspect was the use of ISDM in science application workflows. In this case, the “application” is not necessarily a single task but a combination of multiple coordinated tasks, perhaps spanning multiple facilities. In such instances, we wish to understand how ISDM can support a science application workflow that includes both in situ and distributed-area or interfacility components and how workflow management systems can integrate with ISDM tools for in situ analysis.

#### 3.1.1 Brief Summary of Invited Talks

The invited talks included five short-format talks and two long-format talks.

Jackie Chen from Sandia National Laboratories described using in situ data analysis, machine learning, and visualization for turbulent combustion simulations at extreme scales. She described how massive scientific simulations are a source of challenges and opportunities in ISDM. The current typical workflow of compute first and analyze later does not scale even to today’s HPC machines. A widening gap between compute power and I/O is driving the need to integrate analysis in situ. The solutions in this space need to minimize performance impact on the simulation and support many varied analysis algorithms with different data dependencies, scalability, and communication patterns.
Ann Almgren from Lawrence Berkeley National Laboratory presented AMReX, which provides an adaptive mesh refinement framework for application codes in combustion, cosmology, astrophysics, accelerator physics, microfluidics, multiphase flow, phase field, and many other areas. These applications contain a multilevel mesh hierarchy, particles, and cut cells for complex geometry and feature mesh flexibility and algorithmic heterogeneity. The AMReX ISDM wish list includes efficient I/O for new architectures, run-time automated optimization, smart compression, and coupling of codes (e.g., simulations and analysis).

Steve Legensky from Intelligent Light discussed his company’s work in the ISDM space, particularly in data visualization and analytics. Their products reduce data size via the targeted extraction of useful quantities from simulation and via data compression. He pointed out that the results from simulation and from measured data need equal focus, which is not always the case. Steering for smart simulations is an emerging trend, using in situ machine learning to make decisions in the simulation framework.

Dan Jacobson from Oak Ridge National Laboratory introduced the characteristics of biological data, which are multidimensional and have many layers, including temporal and longitudinal information. Big data and HPC can be used together to reveal underlying biological signatures. ISDM plays a role for machine learning workflows, and in particular Dan emphasized that the field needs to understand some of the questions around training frameworks: how to orchestrate iterative training and testing; the impact of adding a network layer in situ; how many iterations are needed for convergence; and how to architect ensemble methods. Science needs explainable artificial intelligence to reveal the interactions between the variables that lead to classification results.

Amber Boehnlein from Jefferson Lab gave an in-depth talk describing how understanding the structure of the proton and quantum chromodynamics at Jefferson Lab will require in situ analysis and data management. Computer science is often discussed in compute-centric language, which is inherently biased toward the operators. Data must be “managed” in order to enable the computation. Amber emphasized that there should be parity in the roles of data and computation. Existing computational models for experiments in nuclear physics are highly serialized and task-based. The information flow between the various stages in the experimental computing model is illustrated in Figure 11. Rethinking the computing model relies on ISDM for real-time calibration, analysis, and comparison with theory.

Bruce Hendrickson from Lawrence Livermore National Laboratory gave an in-depth talk about the role of ISDM resources that can be used more efficiently if we can detect when a problem has occurred or when sufficient data have been collected. However, this detection requires a major change in workflow to accommodate simulations before and during an experiment and real-time interpretation of data. Running a real-time simulation continually updated by observations can help protect critical infrastructure (such as the electrical grid) and keep the system running optimally. ISDM needs include real-time system software, computational steering, real-time analysis of simulation and observational data, efficient algorithms to enable fast-enough simulations, and real-time updates to simulations based on observations. Useful references in this area are the OSTP White Paper on Data-Intensive Science in the US DOE [Ahrens et al., 2011] and the Data Crosscutting Requirements Review [U.S. DOE, 2013].
3.1.2 Description of Open Panel Discussion

After the talks, all six speakers were invited to participate in a panel discussion moderated by members of the workshop organizing committee. The panel discussion covered two main questions. The first question was **What are the pain points and frustrations in ISDM today?** The main responses to this question were as follows:

- ISDM patterns do not fit into scheduling systems of HPC systems, so it is difficult to run these frameworks today.
- There is no programming model for ISDM.
- Science needs to be expressed all the way down the stack. Nonintuitive interfaces will not be adopted quickly.
- Scientists and researchers are struggling to keep up with the changes in computing architecture—it feels like goalposts are constantly moving.
- Perhaps one framework that does everything will be impossible, and instead smaller components would be easier to implement.
- Current implementations are designed from the visualization point of view and often do not take into account the culture of the science code developers: for example, language choices (C++ vs FORTRAN) and library dependencies.

The second question was **Where are the education and skill gaps for using ISDM frameworks?** Panelists’ responses included the following:

- Computing is a problem for experimental scientists, and few experimentalists have any computer science expertise.
- There are more software elements and frameworks than any individual (or institution) can track.
- Co-design teams have been helpful where scientists work with computer science experts. The ASCR funded centers to create teams could be combined, focused on motifs in ISDM.
- Universities are not producing students with full skills needed in HPC, so the field has needed to train incoming talent. These training programs are popular and could always be expanded.

Some common themes were identified in this session. ISDM is an emerging need for many workloads, including simulation, data analysis, machine learning, and experimental facilities. More parity is needed between the roles of computation and data, and ISDM can be the bridge between computational and data science. Software that connects multiple communities and is composed of heterogeneous building blocks is needed. This requires composing programming models, data models, and software stacks.

Another common theme was usability. Obviously, software frameworks need to be easy and intuitive to use, and ISDM is no exception. However, the right level of granularity or modularity of an in situ framework varies with the user; for example, scientists have different levels of interaction from those of application developers. Software is needed where data semantics, user constraints, quality of service, accuracy, and levels of control over the in situ workflow can be expressed and validated. These features need to be present both in the ISDM infrastructure and in the user tasks—data analysis algorithms, for example.

ISDM can be daunting to application scientists, and solutions are needed to help overcome the barriers to adoption of ISDM frameworks. However, panelists unanimously agreed that ISDM is essential in order to combat data movement bottlenecks and to improve the quality of science. ISDM can help simulations be detailed and informative. It can enable multiple simulations to be executed at the same time, such as ensembles for design optimization, uncertainty quantification, and margin analysis. ISDM can allow scientists to produce results as fast as or faster than real-world events, enabling new combinations of simulation and observation with real-time computational and experimental steering.
3.2 Data Models: Connection and Communication

For this workshop data models were viewed as abstractions and implementations describing how a set of values in memory should be interpreted as a relevant scientific object, as well as the middleware and/or communication tools needed to access data in situ. As such, tools to register, manipulate, communicate, publish, and query data models at multiple levels are key to moving beyond having each in situ component being a bespoke, one-off implementation. Some communities have adopted a single unified data standard that allows them to implicitly address issues regarding data model matching. Similarly, some programming models (§3.6) enforce particular data approaches such as zero-copy pointer access that enable simpler run-time assumptions but may sacrifice usage flexibility.

Both of these approaches can limit programmer productivity and software reusability, however. More explicit and robust tools, methods, and frameworks are required in order to improve ISDM data descriptions and communications without providing undue burden on the programmer or end user. Hence, data models for ISDM cover several overlapping issues: structural definition (e.g., is it an integer or a 64-bit floating point value), semantic definition (e.g., does this linked list represent a graph), and access definition (e.g., are data serialized as a message, or do pointers access scattered memory locations).

Are there data model commonalities or motifs for description and access that we can identify that will promote programmer productivity and software reusability? We considered at least three different components of data models to answer this question. In the area of data interchange, participants were asked to consider what framework or tool services could automate the conversion between differing producer and consumer data models and how one could best address the mismatch between producer and consumer data model definitions.

In the area of performance and usability of data models, we suggested investigating the interplay between data model and data communication in light of evolving heterogeneity in systems (§3.3) and performance portability (§3.6). We also asked what developments are needed so that data format, zero-copy, and structure definitions are more universally available.

In the area of metadata, we seeded the discussion by suggesting participants consider the interplay between computational, provenance, performance portability, and archival data models (§3.5); whether there are different time scales where they are relevant; whether we need metadata schema for analysis, visualization, deep learning, and other in situ components (§3.4); or whether there are schemaless approaches that offer advantages.

Participants engaged in a lively discussion on a number of subtopics: the inclusion of scientific reasoning in data models, the need for multiscale and multiphysics models, physical and logical models, meshless models, the tension between domain-specific and generic models, data layout as it relates to portability and performance, usability, support for variable quantization, and the ability to query models. Eventually, the conversation coalesced around two main themes: (1) enabling data models for multimodal science, data science, and learning interoperability and (2) enabling reusability by responding to expressions of intents in ISDM, including timeliness and streaming requirements.

3.2.1 Enabling Data Models for Multimodal Science, Data Science, and Learning Interoperability

Key Challenges and Opportunities

Multiscale, multiphysics, and multimodal application suites play an important role for DOE science, ranging from quantum to continuum to phenomenological scales. As data are exchanged between these different layers, a variety of implicit and explicit services and operations both enforce policies and provide mechanisms for the exchanges. In particular, such complex applications have distinct requirements for in situ data management when compared with scenarios driven by single, monolithic codes. In situ data transfers are between science components that may function over different length scales, time scales, and data representations. Moving from ad hoc, bespoke interfaces between such codes to shared, adaptable data models enables new capabilities not only for simulation construction but also for better connection to a wide array of data science analysis and learning capabilities.
Machine learning approaches (deep learning, random forest, etc.) share many of the same concerns about timeliness of delivery and cleanliness and regularization of data as do coupled physics codes. There is great promise in using ML as part of the coupling infrastructure in such multiscale, multimodal environments, in addition to using ML to classify and model the output data as part of an analysis pipeline. By using in situ data model services and standards to help prepare and precondition data streams, an application can better optimize for later, exploratory deep learning, neural networks, or other AI technologies.

As a concrete example, consider an ensemble of atomistic-scale simulations to determine an appropriate adjustment of parameters in a mesoscale material model, while maintaining information about the error tolerances of the result. This sort of multiscale code coupling could be relevant to overall trust in the results of the simulation because it enables validation and uncertainty quantification of the model parameters, as well as enabling connections to surrogate models generated through machine learning techniques. One could implement such coupled systems fully by hand, with implicit data model assumptions between the components. Without the support of strong in situ data models to enable reusability, however, continuing to develop and to update such implementations would be difficult.

State of the Art

Many multiscale applications have been implemented, for instance, in materials design studies where continuum and atomistic models of materials are coupled [McDowell and Olson, 2008, Cuitiño et al., 2001, Elliott, 2011] and in climate modeling, where ocean, atmosphere, ice, and other components each contribute to the overall model [Golaz et al., 2019]. Each of these has particular solutions for moving the data structures out of one level of resolution, performing an interpolation and projection into the other model’s framework, and then injecting the new data structure into the next code in the circuit. These examples and their frameworks for custom interoperation motivate more general capabilities for ISDM.

Some related issues arise in multiresolution solvers. Techniques such as adaptive mesh refinement (AMR) have a core data model that is extensible, where one can continue to add new refinement data overlays to get finer detail where needed. Different techniques for doing AMR, however, motivate a deeper investigation into the distinction between data structure matching and semantic matching, supported by ISDM.

New Research Directions

One must consider error propagation and variability in the data model specification for ISDM. Simulation data commonly are treated as exact, while experimental data are treated as needing cleaning and statistical interpretation. However, there is a growing need to apply such statistical methods to all data sources, especially as in situ technologies connect disparate model implementations. As such, the in situ data management mechanisms need to be able to recognize and support different classes of operators. Some of these operators may be applied to windows of data to do temporal averaging, or they may have particular timeliness requirements for delivery (as when trying to insert experimental or observational data into ongoing simulations).

During parallel or distributed training for machine learning models, data movement patterns can significantly differ from traditional MPI collective communications. ISDM must ensure that data movement between training, scoring, and generation of models is efficient and best meets the requirements for convergence of the process.

The following capabilities and directions could be enabled by enhanced data models and representations:

- Capture of important aspects of data such as temporal and spatial properties
- Additional information such as provenance, uncertainty, and intent
- Lowered barrier for developing such data models across multiple communities and multiple scales
- Application of community software (applications, analysis tools) in new contexts, and coupling of such applications in new ways
- Adaptation of interfaces to efficiently enable machine learning to connect different scales and models
Potential Scientific Impact

Interoperability of at-scale codes across time and space is a grand challenge in many science domains. Enabling such interoperability would dramatically increase our ability to incorporate fundamental understanding into application-scale simulations, thus increasing their predictive power. Exploiting the increased statistical and data understanding that comes from incorporating machine learning inference and predictive capabilities for in situ coupling further increases the potential impact.

3.2.2 Enabling Reusability by Responding to Intent Expressions in ISDM

Key Challenges and Opportunities

Data models must capture the intention of the scientist in extracting information and uncertainties from the data, beyond specifying simple quantities. A key challenge in data models and interfaces is navigating between overly general frameworks that are too abstract for wide adoption and bespoke single-code implementations. Research to develop a third path that respects the constraints and requirements of the producers and consumers of the in situ data streams would lower the barrier to use and adoption by diverse scientific communities.

Instrumental in finding this middle approach are two key challenges: (1) finding concise ways to express and match the usage intents between producers and consumers of the data and (2) developing metrics and mechanisms for evaluating them that aid in quantifying performance and robustness constraints. If an end user is to adopt a new set of techniques for ISDM, such adoption must be based on understanding how the data must be packaged and what the user can expect in return for using the ISDM tools, rather than simply writing another one-off code implementation.

Concretely, one might think of intent expression as the creation of a sort of “master schema” where all producers and consumers can express a particular interest in timeliness, degree of uncertainty in the data, and so on. Indeed, for limited communities, a sufficient approach might be to adopt a specific schema definition. However, enabling broader reusability within the scientific community requires thinking more in terms of technologies such as the Semantic Web [Kashyap et al., 2008]. Both static and user-defined “schema” definitions should be supported as they are declared, evaluated, and matched.

State of the Art

Data models used by related systems such as I/O systems and data analysis systems provide potentially familiar metaphors for application users. Some I/O systems, such as ADIOS [Liu et al., 2014] and HDF5 [Folk et al., 1999], have made efforts to link to in situ workflows either directly [Dayal et al., 2014] or through file-like I/O on a “burst buffer” system architecture [Dong et al., 2016]. This familiarity could reduce the time needed for the users to develop their in situ processing tasks. Leveraging existing data models and/or APIs could provide more stable and longer-lasting ISDM systems.

Component interface definitions was an important part of the Common Component Architecture effort, where techniques such as Babel [Epperly et al., 2012] were important for achieving interoperability. The Common Object Request Broker Architecture (OMG, 2000) tried to integrate such intent-matching subscriptions in the event management system. Self-describing and extensible interfaces have been explored in a number of ways, for example, Scientific XML [Widener et al., 2002] and the External Data Representation (XDR) [Srinivasan, 1995].

Generic data models that allow arbitrary fields to be appended in a “container” are also possible [Dreher and Peterka, 2016]. The interface to such a model can appear in the form of programmer API calls embedded directly into a user’s task code, as in Decaf [Dreher and Peterka, 2017], or in the form of a “contract” specified outside of the tasks [Mommessin et al., 2017]. Contracts are an important concept in many tools in enterprise computing, as well as in situ visualization tools such as VisIt [Childs et al., 2012a]. The contract mechanism can provide additional benefits such as compile-time and run-time error checking, as well as optimizing data movement. The contract mechanism can be applied to other areas besides in situ processing, such as managing I/O in storage [Dorier et al., 2017].
New Research Directions

An emerging feature in ISDM involves use cases that leverage the in situ capability to provide control and other time-dependant feedback to the application, user, and/or execution environment. Streaming and adaptive control models of execution, as described in other sections (§ 3.4 and § 3.6), also introduce core issues for the data model and middleware implementations. While timeliness can be expressed as service level agreements in cloud computing frameworks, the formalisms developed for those use cases do not necessarily easily translate over to HPC and ISDM.

Another key technological issue for meeting user intents involves addressing changing representations as data move into and out of software components supporting new architectural capabilities, such as mixed-precision floating point numbers on tensor cores. Similarly, using deep memory hierarchies and new interconnect technologies that leverage advanced capabilities (e.g., NVRAM, RDMA) will be key architectural optimizations for middleware. Binding the structural representations to intent, while simultaneously addressing the heterogeneity of future hardware platforms, will be a key component of any data middleware infrastructure.

Some key science use-cases, such as the propagation of uncertainty, have further requirements for dynamic metadata extension and cross-application schema matching. The degree of uncertainty in an in-situ-computed statistical summary from an ensemble of runs lies not with the individual instances but rather with the online statistic. Although one could create bespoke models for a particular set of science runs, doing so would require porting every new change or feature from the individual runs to the statistical summary. Enabling reusability of in-situ-enabled workflow components therefore should focus on new directions that embrace the dynamic extensions of metadata and advanced intent-based schema matching.

Specific other research directions include the following:

☐ Autonomous workflow creation in order to match the processing expectations of a data consumer to the data available from a producer

☐ Models of in situ queries over multiple data streams

☐ Development of appropriate quality-of-service or quality-of-information metrics that can be used for high-performance in situ management.

☐ Data reduction that respects downstream requirements and scientists’ intentions (e.g., features of interest in the cross-product of two fields)

☐ Support for higher-level connection semantics with middleware and workflow execution capabilities (e.g., to make decisions about placement and to support interoperability)

☐ Preparatory analytics to reorganize data structures and indices based on knowledge of future query intents (e.g., space-filling curve or column-ordering of arrays)

☐ Exploration and co-design of in situ capabilities for different computational and data management motifs

Potential Scientific Impact

The core impact is to increase access to broader analysis capabilities, with a low barrier to the reuse of tools and frameworks. Further work in this direction also will reduce or eliminate errors due to uncertainties that are currently not fully expressed in today’s data models.

3.3 Computational Platforms and Environments

While previous in situ efforts reacted, in a defensive manner, to changes in computational platforms and environments (the I/O bottleneck, for example), a more strategic approach is needed to ensure that platforms and environments meet the analysis and visualization needs of computational scientists and that in situ software is flexible enough to exploit emerging technologies. In this session, we elicited ideas that exploit recent and anticipated changes to high-performance computational platforms and environments as opportunities for the in situ analysis research community. For example, nonvolatile memory is expanding...
per-node storage capacity, affording potential opportunities for creative data management and new analysis techniques; and new computational platforms are being deployed to support machine learning. In addition, we expect pervasive processing capabilities through complex heterogeneous node architectures, such as systems with GPUs, FPGAs, processing in memory, processing in network, and neuromorphic hardware. Such advances create exciting opportunities for in situ analysis research.

**How do we develop ISDM technologies to adequately exploit emerging computational platform and environment capabilities?** In the area of memory and storage architectures, we considered how ISDM technologies could exploit or influence storage systems and input/output (SSIO) innovations in storage and data management (for example, multilevel memory, NVRAM, and in-system data services) and what interfaces with storage system software are advantageous.

In the area of heterogeneous node architectures and pervasive computing, we investigated what opportunities for ISDM arise from the computing characteristics of GPUs, FPGAs, neuromorphic hardware, and processing in memory, and what level of portability can be ensured for ISDM capabilities.

Regarding operating system and run-time requirements, session attendees were asked to think about what the operating system/run-time (OS/R) requirements are for in situ analysis and how ISDM technologies should share resources (e.g., memory, storage, and accelerators) among tasks and with other parts of the software stack (e.g., application, file system, run-time system). We also examined what advances in OS/R technology will be needed in order to fully realize a convergence of HPC and big data analysis, such as machine learning and experimental data streaming, and how ISDM capabilities should interface with or influence these new technologies.

Looking at the role of ISDM in co-design, we asked the participants to consider whether ISDM puts unique stresses on hardware or contributes unique needs not already represented by other use cases and, if so, how ISDM should be considered by HPC system designers.

The conversations in this session clustered into four subtopics: hardware co-design of heterogeneous memory and compute hardware; the use of commodity platforms for ISDM; monitoring, modeling, and resilience in ISDM; and system software support for ISDM. Eventually, two research areas were found to capture all the topics: (1) HPC system software support for ISDM and (2) co-design and use of heterogeneous hardware for ISDM.

### 3.3.1 HPC System Software Support for ISDM

**Key Challenges and Opportunities**

The community needs to investigate potential changes to HPC system software to better support the needs of ISDM. In particular, ISDM needs abstractions for resource management (scheduling, allocation, etc.) that link ISDM with applications, workflow management systems, and HPC system software. To enable a broad range of in situ capabilities, HPC systems also need support for multiple, concurrent OS/R software stacks.

**State of the Art**

Current scientific workflow management systems such as Kepler [Altintas et al., 2004], Sandia Analysis Workbench [Friedman-Hill et al., 2015], Pegasus [Deelman et al., 2015], and others [Deelman et al., 2009, Liu et al., 2015] are intended to provide domain scientists an abstraction for designing and automating execution of complex computational experiments. In HPC systems, these workflow management systems are built on top of existing system services that support the explicit requirements of individual applications, but not necessarily workflows. HPC system software and usage policies, conceived decades ago, were designed based on assumptions about security, scheduling, allocation, and communication that are not necessarily valid for application workflows today nor for future exascale ISDM systems [Dreher et al., 2017].

As an example, consider communication on HPC platforms. While every major HPC vendor supports MPI [Message Passing Interface Forum, 2012] for parallel communication within an application, standards for application-to-application communication, required for workflows, are not commonly supported across HPC vendors. Network connectivity among jobs is disabled on some existing systems and is an
afterthought on others, requiring low-performance protocols such as TCP/IP or research-grade transport layers (e.g., Mercury [Soumagne et al., 2013], Nessie [Oldfield et al., 2006], CCI [Atchley et al., 2011]) that use wrappers around proprietary interconnects to provide the desired “in-transit” capabilities [Bennett et al., 2012, Moreland et al., 2011, Oldfield et al., 2014]. While a number of research projects actively use these transport layers for ISDM [Liu et al., 2014, Ulmer et al., 2018, Carns, 2016], to ensure portability, production-grade workflow systems still use parallel file systems as a communication mechanism between workflow components, creating undue stress on a data management system designed for a completely different purpose.

Similarly, schedulers such as the open source Simple Linux Utility for Resource Management (SLURM) [Yoo et al., 2003] and IBM’s Load-Sharing Facility [IBM, 2017], used by a large fraction of the Top500 systems [Strohmaier et al., 2015], provide space-shared, fixed-sized allocations of compute nodes for “jobs,” not necessarily workflows. ISDM could benefit greatly from dynamic allocation of resources (processing, NVRAM, network). For example, ISDM should be able to “spawn” an analysis task; the run-time should be able to “attach” to an application to extract, for example, diagnostic information; and data services should be able to come and go, expanding and contracting to meet elastic demands of ISDM resource requirements. Such capabilities raise fundamental technology challenges, as well as policy and accounting complexities to overcome.

While operating systems research projects such as Hobbes [Brightwell et al., 2013] and Argo [Perarnau et al., 2013] explored explicit support for ISDM capabilities through “application composition” and containers to enable coupling of codes, and even support for multiple concurrent run-times, more work is required (perhaps through OS/R research) to sufficiently explore and harden these ideas to a point where they are production-ready for exascale and beyond.

New Research Directions

The most important first step toward identifying research directions for system software support for ISDM is to establish a common set of requirements and terminology for ISDM capabilities with respect to data and programming models, platform hardware, algorithm development, and others. In situ terminology is beginning to be standardized by the scientific visualization community [Childs et al., 2016], but any system software research will have to be a co-designed effort with multiple communities. Particular issues of interest are the following.

- **Types of system data from ISDM**: Understanding of the type and volume of information exchanged between the ISDM layer and the rest of the computing software stack is needed in order to maximize performance and efficiency. This can include explicit system support for extraction of system-related provenance data (§ 3.5).

- **System support for programming models and resource management**: Design of system software to support ISDM programming models is important to ensure effective utilization of resources. What features of system software are needed to enable coupling tasks with ISDM? For example, does ISDM require support for multiple, concurrent OS/Rs? What features of the system software enable ISDM programming models and scripting languages to execute efficiently on HPC platforms? What interfaces should exist to allow the ISDM layer to express and communicate “intent” to the OS/R, to enable the system to be smart about how to properly manage resources and data (e.g., when and where the system should move data vs. moving tasks)? How important are dynamic resource allocation and quality of service (QoS) for processing, storage, memory, and network (e.g., for streaming real-time data from edge devices); and what challenges arise for system software development and deployment as a result? Are new concepts needed to reason about progress and control in ISDM systems? What changes to system software will enable developers to manage complexity in ISDM applications and obtain useful “what happened” information regarding correctness and performance? See § 3.6 for a more detailed discussion of ISDM programming models.

- **External I/O**: Data management and movement are critical pieces of ISDM technology that could be significantly different from traditional HPC-style I/O. For example, ISDM data flow may resemble real-time streaming more closely than bulk-synchronous checkpoints. How do ISDM concepts and technologies relate to external data channels such as instruments, human users, and other computer systems? How does the system present external data sources and sinks to ISDM applications? What are the system challenges, performance bottlenecks, and system administration complexities to external I/O in various use cases?
Potential Scientific Impact

System software designed to support ISDM technology will likely have impact in other communities besides scientific computing. For example, the data flow model in ISDM also matches the workflow requirements for analysis of data streaming from experimental facilities and other types of sensors. System support for dynamic resource allocation and management is also needed for real-time data analysis and load management for QoS. System software to support these capabilities on HPC platforms could open up powerful, previously unavailable, use cases for HPC systems in cybersecurity, geospatial analytics, verification, and validation using a mix of experimental and simulation data.

3.3.2 Co-design and Use of Heterogeneous Hardware for ISDM

Key Challenges and Opportunities

A significant challenge for the ISDM community (and HPC in general) is to ensure that software can effectively use hardware—for example, memory, processing, Internet of Things, edge, and embedded devices—that the marketplace is developing. Meeting that challenge creates opportunities to explore innovative solutions with emerging computing marketplaces such as quantum and neuromorphic computing and custom VLSI. As with all new capabilities, we do not expect a general solution to emerge immediately. Instead, such explorations with the vendors will lead to new hardware and software solutions for ISDM through an iterative process of research and development (R&D).

Another important aspect of co-design is ensuring that ISDM software meets the direct needs of scientific applications, workflows, and other user- and application-facing layers of the software stack. Although these are important factors, they are discussed in other sections (§ 3.1, § 3.4.2, § 3.5.1, § 3.6.2, § 3.7.1), while this particular section focuses on system software and hardware.

State of the Art

Performance portability layers such as Raja [Hornung and Keasler, 2014] and Kokkos [Edwards and Trott, 2013] are leading the way for HPC systems to provide standard programming models for diverse hardware; however, gaps still remain with respect to nonvolatile memory (NVRAM) and some of the more experimental computational platforms. While several research projects and vendors have developed abstractions for HPC-embedded NVRAM (e.g., libhio, Data Elevator [Dong et al., 2016], BurstMem [Wang et al., 2014], DataWarp [Henseler et al., 2016], Mochi [Dorier et al., 2018, Jenkins et al., 2017, Carns et al., 2016]), each has limitations, and community standards have yet to evolve. The DOE national laboratories have historically played important roles in establishing community standards for various aspects of HPC computing such as networking (MPI), storage (HPSS, HDF5), and power (PowerAPI). The potential exists to do the same for NVRAM.

While neuromorphic hardware has an obvious role in artificial intelligence and machine learning [Schuman et al., 2017], the HPC community is also evaluating the use of neuromorphic hardware for scientific computing [Severa et al., 2016] because of its significant energy advantages [Agarwal et al., 2016, James et al., 2017]. While the utility of neuromorphic hardware for scientific computing is still limited, it is an active area of research. ISDM technologies could play a central role in designing and developing interfaces that enable the use of neuromorphic accelerators for in situ analysis in scientific computing workflows.

New Research Directions

As with most co-design activities, significant research is needed in order to develop sustainable hardware and software architectures that address our most challenging science-mission problems in a computationally efficient way. Among the many ideas discussed at the ISDM workshop, three particular research directions emerged: performance portability for the complex evolution of storage and memory; effective leveraging of emerging technologies in nonconventional architectures for ISDM; and use of an inverse approach to resource management by investigating ways to map systems and hardware to application requirements, rather than the other way around.
Applying storage/memory abstractions: In the same way that the HPC community has been evolving performance-portable abstractions for processing [Hornung and Keasler, 2014, Edwards and Trott, 2013], there remains research potential for storage and memory abstractions in support of ISDM, particularly NVRAM. Co-design is needed between the ISDM community and researchers focused on, for example, SSIO [Ross et al., 2019] and extreme heterogeneity (EH) [Vetter et al., 2019], as well as memory and storage vendors.

Exploiting emerging hardware for ISDM: As nonconventional architectures become more prevalent in industry, both the ISDM and the broader HPC communities have an opportunity and an obligation to understand the fundamental difference between conventional HPC systems and emerging platforms (e.g., neuromorphic and quantum). Co-design among ISDM, system software, programming models, and computing vendors is needed in order to adapt and evolve along with other platforms. In some of these communities, we may still be able to influence hardware design to meet needs of HPC and ISDM.

Mapping system/hardware to the application: While the conventional approach is to evolve software to match capabilities of hardware, in emerging highly heterogeneous platforms there is an opportunity for innovative ideas to align or map hardware to the needs of the application, leading to improved utilization of ISDM platforms. This research requires close collaboration and co-design between system software (e.g., schedulers) and programming models experts.

Potential Scientific Impact

Co-designing systems and platforms with the needs of ISDM can lead to cost savings and new functionality that are not possible after the fact, once a system is designed and deployed. For example, portability layers for NVRAM can be designed from the ground up to serve the needs of both the ISDM and SSIO programs, rather than being designed for one capability and repurposed for the other. Adapting a system or subsystem to do something for which it was not designed usually causes inefficiency, which could be avoided through the co-design process.

Co-design can lead to cost savings when, for example, commodity hardware can be employed effectively for both DOE computational and experimental applications. While the design of such hardware is often out of the scope of ISDM, emerging technologies such as neuromorphic or embedded devices still offer the opportunity for the ISDM or HPC communities to influence design. For example, significant power savings can potentially be realized if neuromorphic hardware can be designed for science and engineering problems. Ultimately, the co-design and use of heterogeneous hardware for ISDM can enable in situ and streaming computation on a diverse range of hardware, from supercomputers to sensors in the field and in experimental facilities. Such co-design is crucial in order to realize the pervasive potential of ISDM outlined in § 2.1.

3.4 Analysis Algorithms

Data analysis algorithms that operate in situ have unique characteristics and requirements compared with those designed for post hoc execution. Algorithms designed for in situ execution can potentially be scaled to the full spatiotemporal resolution of data being produced by the application, at the rate being produced, and on platforms of extreme concurrency and significant heterogeneity. In order to be effective in this regard, advances in scalable and platform-portable methods are a high priority.

How do we architect analysis algorithms, including ML, to be scalable and platform portable and to support the increased diversity and complexity of in situ science use cases on future generations of computational platforms? In answering this question we suggested discussions in the areas of performance portability, data-driven algorithms, emerging use cases, resource-constrained approximate methods, and the relationship between ISDM frameworks and algorithm design.

Regarding performance portability, we asked how performance and scalability can be achieved across applications (computational and experimental), workflows (in situ and distributed), and heterogeneous architectures (current and future).
For data-driven algorithms and emerging in situ use cases, participants were asked to contemplate how existing algorithms in big data tools and frameworks can be used and which algorithms must be redesigned. They also considered how scalable, parallel, and explainable ML algorithms that obey physical models and/or constraints should be developed and, in the absence of human interactivity, how parameters can intelligently be set in computationally-steered workflows. We further considered what algorithmic challenges arise in complex in situ workflows (e.g., real-time model calibration; integration of experimental, observational, and simulation data; and high-fidelity uncertainty quantification) and what new science opportunities exist for analysis methods when working with full spatiotemporal resolution data.

Resource constraints motivate the use of approximate methods of various types. For example, one may study how low-complexity approximate solution techniques (e.g., sampling approaches, surrogates, and/or reduced-order models) can be used and whether analysis algorithms can be redesigned to minimize data movement and energy or conserve other resources (e.g., communication-avoiding algorithms or stochastic communication).

The relationship between the ISDM framework and algorithmic design has multiple facets. For example, what algorithmic primitives should an ISDM framework provide? What can or should be borrowed from big data frameworks (e.g., Spark’s reduce by key), and what is specific to HPC or DOE science? The interplay between ISDM frameworks, algorithms, and data models also may offer opportunities and challenges for co-design of these components. Ultimately, one would want ISDM to be an enabler and not a barrier to developing performance-portable, sustainable, and interoperable algorithms, so that ISDM can enable sharing and simplify development of new methods.

Topics in this session included analysis of streamed data from experimental facilities; multipass methods of analyzing simulation data; machine learning and data-driven analysis methods; surrogate modeling; searching of simulation parameter spaces; steering, automation, and monitoring of in situ analyses; data and model reduction; feature extraction; multifidelity simulations; and performance portability of algorithms across diverse architectures. Eventually, participants synthesized the diverse topics under three high-level research areas: algorithms for reduced representations, run-time decision-making and algorithmic control, and algorithms for new system platforms and new types of outputs.

### 3.4.1 Reduced Representations Enabling Complex Workflows

#### Key Challenges and Opportunities

Emerging HPC system platforms provide for increased simulation fidelity and diversity in workflows. However, I/O rates will preclude storage of the corresponding increased data sizes to disk. Consequently, post hoc exploratory analysis will be of limited use and value, and in situ workflows of varying complexity that support an increased diversity of science use cases will be commonplace. Algorithms for creating reduced representations of data will play a crucial role in enabling these complex in situ workflows.

#### State of the Art

Reduced representations of data are not new; in fact, a wealth of previous research exists in this space, including statistical [Hazarika et al., 2018, Thompson et al., 2011] and topological feature extraction [Morozov and Weber, 2013, Morozov and Weber, 2014, Gyulassy et al., 2012, Gyulassy et al., 2019, Landge et al., 2014], wavelets [Li et al., 2017, Salloum et al., 2018], compression [Di and Cappello, 2016, Lindstrom, 2014, Brislawn et al., 2012], statistical summarization [Dutta et al., 2017, Biswas et al., 2018], geometric modeling [Peterka et al., 2018, Nashed et al., 2019], and surrogate modeling techniques [Lawrence et al., 2017, Lohrmann et al., 2017]. However, many of these techniques were developed in a post hoc setting, in which a “rich enough” set of the simulation data persisted to service all science use case requirements.

Today, there is increased diversity in the types of workflows in which reduced representations will be deployed. Reduced representations may be used to enable analysis post hoc, in situ, or to steer the simulation. Analysis could also enable integration of experimental and simulation data, ensemble data analysis, intelligent parameter setting in the absence of a human in the loop, the reconciling of the differences between
multifidelity simulations, the use of high-fidelity simulations to inform low-fidelity simulations, and the building of a “knowledge base” that spans beyond a single run. Each of these use cases has its own requirements, in terms of reduced representations, that previously may not have been significant: for example, guarantees that can be made about accuracy, fidelity, and the ability to explore the data in downstream analysis.

**New Research Directions**

Numerous research questions must be asked and answered to develop a rigorous and comprehensive collection of reduced representation capabilities that support the increased diversity and complexity of in situ workflows. For example, what are the different requirements for reduced representations for the myriad of science use cases? Furthermore, what guarantees on accuracy, fidelity, the ability to further explore the data, and/or resource usage are placed on the algorithm? Can we develop algorithms that support these requirements across a range of science use cases? Are new reduced representations required assuming that the full simulation data will not persist? How do we implement these reduced representations in a streaming regime, for example, using single- and/or multipass algorithms? Significant ramifications exist if these challenges are not addressed. For example, exploratory analysis post hoc may not be possible if the right data representations are not selected. Furthermore, important science may be lost if quality guarantees on the reduced representations are not used to inform decision-making in situ (e.g., regarding where to perform deeper analysis and/or storage of increased fidelity).

Quality guarantees and measures are closely related to provenance (§ 3.5). Reduced representations should conform to data models that support composability of tasks within complicated workflows. Advances in PEMs (§ 3.6) should be leveraged to enable performant, portable deployment of reduced representation algorithms on future system architectures (§ 3.3). Good software practices (§ 3.7) underlie adoption, usability, and reliability of reduced representation tools.

**Potential Scientific Impact**

Successful development of this capability would result in improved data reduction and compression methods, enabling scientists to make informed, judicious usage of resources while maximizing the insights gleaned from their data. Furthermore, in situ reduced representations with quality measures and guarantees will play a pivotal role in supporting autonomous data-driven workflow capabilities.

**3.4.2 Intelligent Run-Time Decision-Making**

**Key Challenges and Opportunities**

The term computational steering has historically been used to describe the notion of analysis informing simulation computations [Parker and Johnson, 1995, Bethel et al., 1994]. Today, we see increased diversity and complexity of emerging in situ workflows and science use cases beyond traditional simulation. Many of these workflows require some form of informed decision-making capability in situ in order to control analysis, simulation, or experimental tasks. While reduced representations will play a significant role in enabling complex workflows (§ 3.4.1), additional algorithmic research and software infrastructure are required in order to enable decision-making within a streaming in situ regime for the diversity of system platforms and use cases.

**State of the Art**

Over the past ten years, we have seen several bodies of research aimed at decision-making in situ. Examples include “quick-sketch” trigger-based methods to spawn additional actions based on some analysis [Larsen et al., 2018, Bennett et al., 2016, Salloum et al., 2015, Woodring et al., 2011, Dutta et al., 2018], information-theoretic methods [Biswas et al., 2013, Wang and Shen, 2011], and topological techniques to explore isovalues in situ [Weber et al., 2007]. Recently, there has been an increase in ML workflows [Wozniak et al., 2018, Kurth et al., 2018, Joubert et al., 2018]. While these bodies of work are promising, the use of ML for decision-making in situ is nascent; and much algorithmic research remains in developing scalable, rigorous, explainable, reliable, and trustworthy decision-making capabilities in situ.
New Research Directions

This research focuses on the development of algorithmic capabilities to enable in situ decision-making in support of complex workflows. Research in trigger-based and other traditional decision-making capabilities will continue to be important, and fundamental algorithmic questions regarding the role of ML in automating workflows will need to be addressed. These include research in explainable ML to assess whether ML models can be trusted to automate decision-making [Lapuschkin et al., 2019]. Research also may include applied algorithmic challenges in ML, such as in situ generation of training data and methods for updating ML models in situ (e.g., incremental training to evolve the fundamental model). Success will produce new rigorous and reliable decision-making capabilities that enable a diverse set of workflow use cases. Many risks are involved in automating the process of decision-making in situ. For example, when decision-making algorithms are poorly understood, they may work well within the regimes in which they were tested but may not perform well when faced with unanticipated inputs. This research relies on work in PEMs (§3.6) and data models (§3.2), particularly from the broader big data community. It also relates to the research in provenance capture (§3.5.1).

Potential Scientific Impact

Successful development of robust decision-making capabilities in situ would enable many strong scientific impacts: automation and controllability of workflows that previously required a human in the loop; judicious usage of system resources, real-time debugging, monitoring, and tuning; selective intelligent provenance capture; automated methods for rapid real-time selection of critical features; and attribution of events.

3.4.3 New System Platforms and Data Outputs

Key Challenges and Opportunities

The diversity and evolution in system platforms are driving a corresponding increase in the diversity of science use cases. Consequently, algorithms need to respond to challenges across both axes. Algorithms must be performant, portable, and composable (with user-friendly APIs for developers and user interfaces for application scientists), and new algorithms will need to be developed to support emerging in situ science use cases.

State of the Art

Several active research efforts address facilitating in situ workflows, including Alpine [Larsen et al., 2017], ParaView [Ayachit, 2015], VisIt [Whitlock, 2018], ADIOS [Liu et al., 2014], SENSEI [Ayachit et al., 2016b], and Decaf [Dreher and Peterka, 2017]. Furthermore, there are efforts to facilitate effective use of system resources [Moreland et al., 2016]. These tools actively expose advances in computational platforms, PEMs, and data models to in situ algorithms developers. However, these tools do not yet fully support emerging architectures and science use cases; in fact, significant research is required.

New Research Directions

Research is required to modify existing post hoc algorithms to make effective use of emerging system architectures (massive scale, many cores, and deep memory hierarchies). The changing nature of data being processed on HPC resources increases the complexity of this topic. In addition to traditional simulations, data may stem from experiments, observations, multiphysics simulations, and/or ensemble workflows. Outputs will be increasingly high dimensional and multifidelity, will have uncertainties that must be accounted for, and may be available only in a streaming fashion. Specific research directions include streaming techniques (both single-pass and multipass), dynamic scheduling, performance modeling and measurement, and enabling applications to use their own data representations while also developing community-adopted standards. Success would result in performant and composable algorithms that can be productively deployed by scientists. If the challenges outlined above are not addressed, then significant barriers will preclude the adoption of analysis tools by the scientific community. This research will rely heavily on research in computational platforms (§3.3), PEMs (§3.6), and data models (§3.2); and additional lessons may be learned from the big data/cloud community.
Potential Scientific Impact

Conducting this research would increase the productivity of scientists by providing them performant, composable tools that they are willing to adopt. Furthermore, supporting the increased complexity and diversity of data outputs will lead to better, more advanced scientific insights.

3.5 Provenance and Reproducibility

ISDM creates a degree of separation between scientists and data, making in situ data provenance essential in order to trust the results of in situ workflows (i.e., workflows where data are transformed at run time for analysis and in most cases only derived data products are available post hoc). For example, with the increasing adoption of machine learning in ISDM, capturing hyperparameters, noise levels, decision points, and training data can facilitate replicating results and increase confidence in predictions. A record of data products and their transformations serves multiple purposes: trust, code debugging and optimization, data quality and audit, fault tolerance and resilience, and replication and reproducibility. The potential volume and verbosity of provenance information, the cost of capture, and the complexity of supporting metadata make a principled, targeted, and systematic approach to in situ data provenance a must for the success of future scientific discoveries.

For in situ applications—that is, applications that build on top of in situ workflows—what is the minimum set of provenance information that needs to be selected and extracted so that captured information will be useful for various purposes later? Workshop participants elicited various uses of in situ provenance—for performance analysis, validation, replication, and reproducibility—and considered requirements for targeted in situ data selection and extraction.

Participants were also invited to investigate questions such as how selected and extracted provenance information will be collected in situ and used post hoc and whether provenance can enable or improve search, trust, quality assurance, performance analysis, and replication and reproducibility. They examined the application and architecture-dependent goals for which in situ provenance needs to be collected; and they considered the extent to which ISDM software and frameworks should support the selection, capture, reduction, interpretation, and usage of provenance information.

The potential scale of logged information requires careful planning for the impact of provenance collection on scientific application execution, given that resources are already shared among tasks that may be heterogeneous. Design criteria for minimally invasive provenance extraction and collection and maximum usability in ISDM were considered, along with the tradeoffs between enabling replication and reproducibility and potential impact on application performance and resource usage.

Provenance can be used to validate or reproduce scientific results. One may ask what replication and reproducibility means for in situ workflows. One way to answer this question is to define the minimal provenance required for a data product, created in situ, to be replicable, reproducible, and reusable and to examine how that differs from products created post hoc. This is especially significant if machine learning is used for in situ analysis, where models are data driven and lack theoretical foundations and known error bounds.

The breakout session featured a variety of challenges dealing with various aspects of provenance replication, and reproducibility. The volume, complexity, varying levels of detail of provenance information, and run-time impact of collection were mentioned several times. The uses of these collected data also were central themes, with comments concerning cross-community needs for provenance information, searchability, and ease of use of provenance information. For example, auditability and bug tracking, gauging performance and performance variability, and uncertainty quantification of results were listed as potential uses of provenance information. Defining the lifespan of a provenance artifact and the identifiability and relationship of a provenance artifact to its origin (whether a single line of code or an entire software environment) were also mentioned as important capabilities. These and other topics were organized into three main research areas: scalable and portable provenance collection, processing of the collected information, and intelligent provenance for reproducibility.
3.5.1 Capturing Scalable and Portable Provenance for ISDM

Key Challenges and Opportunities

Comprehensive scalable tools and frameworks for capturing provenance data for in situ applications are needed in order to selectively capture provenance customizable to HPC architectures.

Different types of provenance data can be collected from a variety of sources, such as system-level data, software-level data, and data from the user’s environment. Large amounts of system data currently live in HPC facilities but are not captured and collected by ISDM software. When provenance data at the system level is collected for in situ applications, the data volume and diversity substantially increase. A simple brute-force way of collecting these data can result in enormous data sizes and can require its own file system, making the collection infeasible. Also needed are methods for filtering data; determining what data to keep and discard; and understanding the relationship between data artifacts, code, and the user environment.

State of the Art

No comprehensive, portable framework exists today for intelligently capturing system provenance. While experimentalists have rich provenance capabilities, these capabilities are application and instrument-specific. For instance, Bluesky, developed at NSLS-II for metadata and data management at light sources, is being adopted by the Advanced Light Source and Linac Coherent Light Source. But collection focuses on experimental details such as intensity, motor and sample positions, and detector configuration [Koerner et al., 2019, Allan et al., 2019]. Some tools (XALT and SONAR) pertain to HPC centers as a whole and collect job and user data [Agrawal et al., 2014, Dai et al., 2014]. Most tools and frameworks available for aiding in the collection, organization, and storage of metadata originated in the storage and I/O domains, with research into I/O formats and other metadata collection systems. Several I/O libraries have been developed for improved application performance in storing metadata [Lofstead et al., 2009, Wang et al., 2017]. Patchwork solutions exist from different vendors for telemetry data specifically for their own systems, but none of those solutions are comprehensive enough to collect a broad and diverse set of system-level data, let alone other types of provenance. In addition, systems such as TAU [Shende and Malony, 2006] that collect performance metrics do not relate these metrics to provenance and do not scale to thousands of processors on real science applications.

New Research Directions

Specific research directions include the development of tools and frameworks to enable the selection of provenance according to goals and desired levels of granularity, intelligent collection, and the design of a cost model for provenance. These frameworks should port to different architectures, be flexible enough to tailor different solutions for different HPC subsystems (i.e., compute, network, storage), and be applicable to scientific applications from different domains. The modularity of provenance will enable more generic and portable provenance of in situ applications to be used across facilities.

Collecting provenance from all the data output by a simulation means that many of the "small" variables have a disproportionate, negative impact on the I/O time. For example, in the GTC plasma fusion code, when GTC writes small variables, the percentage of time spent in provenance/metadata can be large compared with the total I/O time. How can exascale simulations maintain the wealth of provenance data without creating a large I/O overhead? Research studies are needed in order to understand how users can specify cost models and how provenance can be captured at scale, without violating those cost models. In this context, self-describing file formats must be able to scale to a large number of writers, readers, time steps, variables, and workflow tasks. Furthermore, research studies are necessary to define and use new techniques for data compression capable of curbing rising I/O costs.

Selecting key aspects of the provenance trace to be kept for in situ applications—whether for single tasks or for workflows of tasks—will represent a culture shift for users who are currently accustomed to keeping and storing all data. Another potential cultural shift may be needed at scientific user facilities to implement new operational policies with regard to access and use of provenance data.
Potential Scientific Impact

One potential scientific impact is increasing meaningful provenance data that supports explainable scientific discoveries with increased validity. Specifically, the targeted selection of key provenance indicators, the reduction of provenance data, and the retention of sufficient information to explain scientific results and variations in performance will be crucial to explain new scientific discoveries. Ultimately, understanding how to integrate and derive meaning from provenance data from multiple sources and how to make data collection tools scalable and portable from one system to another will be key to provenance data having maximal scientific impact.

3.5.2 Processing Performance Provenance In Situ

Key Challenges and Opportunities

Understanding application execution behavior on heterogeneous architectures is more challenging in situ than post hoc. Code optimization is also more complex because performance data exhibit greater variability in situ. Provenance data related to performance metrics—performance provenance—can help. Data engineering tasks to find, reduce, and interpret performance provenance are conducted today post hoc and are resource-intensive. With in situ applications bringing new levels of complexity in execution, processing provenance to obtain actionable information will also be more complex; with the increasing scales of systems, from petascale to exascale, additional processing will be needed. Keeping track of which nodes, the presence of hardware accelerators, and which library versions or configuration parameters were used for a run will help scientists pinpoint issues with debugging, resilience, and scalability.

State of the Art

ISDM applications need to make informed adaptations in response to detected run-time anomalies and system environmental factors. To that end, provenance systems are leveraged to inform applications of the pertinent anomalies and/or system environment changes. However, the state-of-the-art provenance listeners and provenance representations extract a heavy I/O and processing burden on the application [Patil et al., 2012]. Previous research has shown that supervised ML algorithms can be used to partly replace traditional provenance acquisition at run time, significantly reducing the I/O burden by training algorithms with post mortem provenance [Singh et al., 2016].

Visualization and analysis tools for single applications and for workflows are available to the community. For example, Vampir [Knüpfer et al., 2008] and Jumpshot [Zaki et al., 1999] visualize performance data in detail as users zoom into a time window. However, these tools scarcely address the growing volume and complexity of provenance data sources in general, and of in situ execution in particular. In addition to performance metrics, the structure of the call stack, with type and execution of a function, is provided in [Xie et al., 2018] and illustrated in Figure 12, allowing some detailed provenance introspection.
ENABLING SCIENTIFIC DISCOVERY FROM DIVERSE DATA SOURCES

New Research Directions

Frameworks and tools that facilitate the use of performance provenance are an important research direction. Specific algorithms to analyze provenance in situ will allow provenance introspection. Artificial intelligence and complex event processing (CEP) can offer a means of handling decision support for real-time anomalies in a decentralized fashion [Radovic et al., 2018], but research is still needed to determine efficient forms of provenance that can be used by AI and CEP. This includes different data reduction and enrichment techniques for the provenance data flows.

Data discovery and reuse depend fundamentally on resource identifiers that are globally or universally unique. Identifiers that may be persisted, in streaming and in transit, can help support the analysis of provenance data at many levels of granularity. Persistent unique identifiers provide a means of retaining resource identification and information about the resource regardless of the lifespan of the object. Identifiers take many forms and are used extensively as part of the provenance record connecting resources to usage, changes, and generation activities. While location information (e.g., directory or IP address) may be associated with provenance information about a resource, it should not be confused with the identifier itself. Determining identification and retention policies of the resource identifier based on particular needs will also require research.

Scalability is a key problem with many provenance-capturing and anomaly detection methods. Since the data per compute node is generally small, communication at frequent intervals remains a fundamental challenge. The ability to understand the overhead and the potential advantages of recording the provenance using a variety of approaches is a needed development. Asynchronous techniques to analyze and move the provenance data are one potential research direction to address the scalability challenges.

The successful development and deployment of the needed tools also depend on the existence of standardized formats and libraries to capture provenance across domain sciences.

Potential Scientific Impact

Potential scientific impacts include more efficient code development and reduced time to solution. Advances in data reduction and improved performance of in situ workflows will increase the use of in situ analysis techniques.
3.5.3 Leveraging Provenance for Reproducible Science

Key Challenges and Opportunities

ISDM implies increased automation of application workloads, creating challenges and opportunities for provenance. The challenge is that more provenance must be captured in order to be able to understand the choices that the automated controller made; the opportunity is that since users are not making these decisions interactively, provenance can and must be automated as well. The associated automated data selection and control decisions made by in situ analyses, however, may introduce biases in results that are difficult to detect; for example, when analysis routines autonomically focus on certain simulation products while discarding others.

In order to address these scientific concerns, ISDM will drive wider adoption and innovation in provenance technologies and standards; it will also accelerate the scientific cycle through better management of simulation behavior, outputs, and artifacts (e.g., performance data). Deeper integration of machine learning into all levels of scientific applications, both in computing and in data analytics, will pose additional challenges and opportunities for provenance. ML modules (especially deep neural networks) introduce unpredictability and sensitivity in application behavior, making reproduction and interpretation of results even more challenging. Traditional approaches to scientific validation and reproduction will have to be re-evaluated in light of the dynamic decisions made by ML modules performing function approximation, in situ analysis, or experimental design. Calculations that integrate these techniques will need specialized provenance techniques that capture relevant information about the ML application; those techniques should enable answering provenance-like questions after the computation is complete.

Incorporating provenance in workflow management systems will require connecting the underlying model uncertainty with uncertainty in data. The anomalous behavior of in situ analysis algorithms can be difficult to trace to its origins. Different types of uncertainty related to data, algorithms, implementation, and statistical representation require different types of provenance. Some aspects of uncertainty, including the order of operations and concurrency issues, are not represented in traditional error bounds. In order to address provenance and UQ, a quantitative theory of the effects of uncertainty in metadata, in addition to the data, is needed, including the following:

- Quantitative theory of the effects of various types of uncertainty on end goals
- Impact of missing data or other sources of numerical uncertainty
- Propagation of uncertainty in metadata
- Quantification of the “cost” of the uncertainty
- Identification of what information would be best to acquire next, in order to reduce the remaining cost of uncertainty
- Development of uncertainty classes of metadata models
- Delineation of uncertainty in the data and metadata: whether there is overlap or whether that distinction is valid
- Use of a graphical model approach to solving the problem
- Evaluation of uncertainty in the initial conditions of a stochastic algorithm (including Monte Carlo), for example, when training with small amounts of data

State of the Art

Scientific and performance reproducibility studies in computer science are rare and tend to be customized to specific applications [Pouchard et al., 2019]. Some initial work tracks the provenance of ML (what models were used and how those models were obtained), for example [Schelter et al., 2017], but these are not in wide adoption and are rarely incorporated into the main ML frameworks.
Performance reproducibility faces challenges related to unique, rapidly changing architectures. Multithreading can make program execution nondeterministic, with performance variations due to the order in which memory is accessed on multicore systems [Gramoli, 2016]. Scientific replicability can require extensive modification of existing tools such as workflow management systems to run batches of simulations and capture the parallel execution environment [Hunold and Träff, 2013].

Without improving dataset and provenance discovery, reproducibility is hardly feasible. With the exception of a few domains and experiments, scientific data products, particularly those produced by experimental and observational facilities, cannot be easily searched or shared by scientists who were not involved in the experiment [Windus et al., 2017]. Users today must sift through data by hand; contact the user who produced the data; and work hard to find, filter, sort, and search for data. Performing composite searches across multiple datasets is difficult, particularly in the case of provenance that is largely application oriented and designed to meet the immediate needs of a particular experiment or study.

New Research Directions

Scalable and portable systems for recording and analyzing provenance would aid in the discovery of data and in the reproducibility of science. Comparing the provenance of scientific datasets produced by diverse computational systems and experimental facilities will ensure trust in the results. Provenance systems will have to be co-designed with emerging resource managers, operating systems, I/O libraries, workflow systems, and accelerators that underlie ISDM workloads (§ 2.2 and § 3.3). These systems each have their own ideas of provenance. Novel provenance techniques and systems that cooperate with these supporting subsystems and integrate application-level data with system data stand to gain a great deal of functionality and utility. Tools, algorithms, and frameworks need to be designed to fuse provenance (e.g., correlations and dependencies) from multiple sources. These integrations pose software and data standardization challenges that must be overcome; however, this research is necessary in order to fully benefit from enhanced provenance collection in complex workloads.

Potential Scientific Impact

Trusting the results of ISDM methodologies augmented with selective provenance capture will allow a reduction of data volumes, while simultaneously permitting an increase in the resolution of scientific data. The reason is that increased confidence in scientific results, with a concurrent increased confidence in software reuse, in an environment where uncertainty is high, will support the ability to select relevant data and discard others.

The management and operations of facilities are the largest components of the Office of Science budget. Enabling data produced at one facility to be reused, validated, or compared with other datasets can facilitate greater scientific output for each experiment conducted at another facility. Furthermore, replication and reproducibility supported by provenance will facilitate collaboration and more efficient operation of user facilities.

3.6 Programming and Execution Models

Programming and execution models are critical to the discussion of ISDM because the manner in which in situ data are accessed and managed across HPC resources largely depends on the programming model. The PEM also controls how in situ computations are run (parallelism and ordering of computations), and where they are run (what node and what processing unit). Often PEMs include data models or have constraints on how data are managed (see § 3.2). In this context, we consider both the outer-level PEMs used to develop and use the ISDM framework and the inner-level PEMs used to define individual tasks within that framework. These PEMs need not be identical, and a composition of multiple heterogeneous PEMs is often required when programming and executing in situ workflows.

Examples of PEMs include bulk-synchronous and asynchronous models, task-based models, big data (databases, message-queueing systems, MapReduce, etc.), and mixed (e.g., MPI + X) or converged models (big data + HPC).
How can PEMs support ISDM for effective and efficient in situ data analysis? When attempting to answer this question, we considered topics such as the suitability of PEMs for ISDM, support for dynamic computations and data generation, and the usability of PEMs.

On the suitability of PEMs for in situ data management, participants were asked to list aspects of PEMs that best support certain types of in situ data analysis or in situ data types and to consider whether a long-term, performance-portable PEM solution could be found that would not require a code rewrite every five years. Furthermore, we considered how in situ data analysis can be supported within systems for ISDM workflows that may use multiple PEMs and/or converged big data and HPC PEMs.

The expanding diversity of uses of ISDM will require support for dynamic computations or data generated at different rates and on different resources. Participants were encouraged to think about how PEMs can support dynamic in situ data analysis for irregular or unpredictable input data generation, how PEMs can support performance-portable in situ data analysis; and how PEMs provide visibility into the performance tradeoffs of in situ data analysis. To support experimental and observational use case cases, PEMs will need to keep up with input data generation, especially for applications with real-time or pseudo-real-time requirements.

We also considered questions of usability of PEMs for in situ analysis. For example, can domain-specific interfaces better enable in situ analysis or be more usable for different types of users and levels of user expertise than generic interfaces can? Participants also considered how PEMs can support provenance capture during in situ analysis, with the goal of understanding the lineage of the results, and how PEMs can support resilience when in situ computations may fail.

The breakout started with a discussion acknowledging many and vastly different PEMs and how they can interoperate while maintaining performance portability. Then the question arose, What makes PEMs for ISDM special? Many times the discussion went back to granularity of work: coarse- or fine-grained units, small or large building blocks that are tightly or loosely coupled. New interfaces to PEMs were suggested; and how interfaces could enable more flexible, dynamic, and complex workflows was also discussed. The need for in situ analysis of streamed data was mentioned in the context of experimental and observational facilities. The issue of QoS was raised in the context of complex workflows. The discussion also included using ML to help determine resource usage for PEMs. Eventually, these discussions converged to four areas of PEM research: dynamic PEMs, optimization and QoS policies for resource usage in PEMs, composable PEMs, and PEMs that support streamed data.

### 3.6.1 PEMs for Elastic and Dynamic Resources In Situ

#### Key Challenges and Opportunities

In situ workflows need to support the ability of multiple codes (e.g., analysis, scientific computation, and provenance capture and processing) to be coupled together in a dynamic fashion. Workflows can change during their execution, for example, because of the behavior of the physics or the dynamic aspect of experimental data rates. In order to support in situ analysis on such dynamic workloads, PEMs must be able to expand, contract, and adapt to the number of resources that are required. Today, we must specify the maximum number of required resources and adjust the balance between constituent tasks, or we must checkpoint the state and then restart the workflow with a different amount of resources. Assuming schedulers and batch systems will support dynamic resource allocation in the future, PEMs will need to work with these schedulers to signal the need for resources and to acquire resources as they become available. PEMs could even be used to predict resource needs, in order to alert the scheduler to request additional resources before they are required.

Consider a set of nonlinear partial differential equations producing data, such as the combustion S3D code (Figure 13). When data analysis is coupled to these data, gradients (in space and time) are steeper near to flame fronts; and when performing topological analysis, the amount of data that needs to be processed in the flame-front region is different from elsewhere. As the simulation progresses and the intense gradient regions change spatiotemporally, the workload has to be rebalanced. Adaptive mesh refinement is used to calculate relative truncation errors in order to understand when and where the simulation mesh needs to be refined. Such refinements affect the topological analysis as well, changing the resource requirements for both the simulation and the analysis.
ENABLING SCIENTIFIC DISCOVERY FROM DIVERSE DATA SOURCES

(a) Topological analysis. The yellow-colored structures in the merge tree correspond to the hot ignition kernels, whereas the red-colored structures represent the lifted high-temperature flame base.

(b) DNS of a turbulent lifted hydrogen/air jet flame stabilized on hot ignition kernels located upstream of the lift-off position with topological nodes overlaid by the cyan-colored network.

Figure 13: In situ topological feature detection in turbulent combustion simulations is used to segment and track localized intermittent ignition and extinction features for steering downstream analysis, I/O, and mesh refinement. The in situ workflow is data dependent, and programming models that support elastic data-dependent workloads are required (images courtesy of J.H. Chen).

State of the Art

Workflow management systems do coarse-grained resource management. Pegasus [Deelman et al., 2015], for example, has been combined with cloud infrastructures such as ExoGENI [Baldine et al., 2012] and ORCA [Ruth et al., 2012] to build elastic distributed-area workflows.

YAWL [Van Der Aalst and Ter Hofstede, 2005] supports dynamic workflows through worklets. Task-based systems can dynamically spawn tasks at finer granularity, for example, in Legion [Treichler et al., 2013] or in OmpSs [Bueno et al., 2011]. Some work has been done with Adaptive MPI [Huang et al., 2004] and Charm++ [Kale and Krishnan, 1993] for malleable workflows, but traditional MPI cannot currently add and subtract resources on the fly. Lacking is a dynamic-granularity PEM that can respond to in situ needs. Interfaces to enable sharing of system resources among programming models have been developed [Castain et al., 2018]. Machine learning has been used to predict usage of resources for applications [Matsunaga and Fortes, 2010] but has not been applied to in situ analysis. Work in dynamic resource usage has also been done in the cloud arena [Apache Software Foundation, 2019c].

New Research Directions

New research directions for this topic are PEMs that support flexible and dynamic resource allocation for in situ processing. Specifically, in the case of task-based PEMs, a research direction would be flexible and dynamic task creation in response to in situ needs, along with flexible granularity of tasks. In the area of anticipating computational needs of in situ analysis, a research direction is using machine learning to predict the resources of a dynamic workflow before they change, in order to minimize the wait time for additional resources.

Workflow dynamics can involve changes to resources or to workflow topology. The former case is less disruptive because the workflow graph nodes and edges remain the same; only their attributes change. Workflow topology changes (i.e., the addition or deletion of graph nodes and edges) involve additional complexity because more steps are required, during which the workflow graph may be in an inconsistent or invalid state.

A reasonable response to a topological change is for the workflow to save its entire state, reconstruct the new workflow, load the saved state, and resume operation. Checkpoint/restart of the state of a workflow will need to be developed. Global checkpoint/restart of workflows will be a useful capability for other reasons such as fault tolerance as well. One could also investigate to what extent workflow changes can be handled without
stopping and starting the entire workflow execution. Ideally, resource changes (e.g., redistribute the number of MPI ranks between simulation and analysis) and some topology changes could occur without interrupting the workflow. A simulation, for example, should continue to run while an analysis task connects to it, without disrupting the simulation. Theoretically, changes to a subgraph of the workflow may be possible while other parts of the workflow graph not directly connected to the affected graph edges continue operating.

**Potential Scientific Impact**

Having a PEM handle malleable, evolving, and adaptive workloads has many benefits: system utilization, job resiliency, proactive system fault-tolerance, job preemption, and backfill potential. One positive impact of this research will be the ability to maximize HPC resource utilization. Another will be to increase performance of in situ analysis. Still another will be the ability to add to a growing list of in situ analytics and to dynamically adjust the resources according to their dynamic and data-driven behavior.

Dynamic PEMs depend on operating systems and resource discovery as well as dynamic resource schedulers and batch managers (§ 3.3). These schedulers and batch managers must allow PEMs to interface with them regarding anticipated resource usage.

### 3.6.2 PEMs for Scheduling, Mapping, and Optimizing In Situ Workflows

**Key Challenges and Opportunities**

Users want to have control over how their applications run. With the addition of in situ analysis, users want to continue to ensure that their applications will have the desired performance and quality of results, while balancing needs and benefiting from in situ analysis and provenance capture and processing. Users would like to avoid ad hoc in situ resource management decisions that are not coordinated with application resource usage. Doing so is challenging, however, because computing needs may not be known until run time. Expected performance and resource usage can be difficult to predict in advance of doing the computation. If coordinated scheduling, mapping of resources, and optimization could be done, they would enable more in situ analyses, less impact on applications, and possible new efficiencies to be realized: improving overall time to solution of a complex workflow and its associated components, minimizing impact on the application’s progress, and minimizing power and data movement.

**State of the Art**

The literature on constrained optimization of scheduling [Sultanik et al., 2007] and DAG scheduling [Selvi and Manimegalai, 2017] is extensive, including single and multiple schedules. During the height of the “grid” era, a number of research efforts focused on scheduling workflows with priorities, budgets, deadlines, and quality of service, including performance, reliability, capacity, and cost [Campbell et al., 1994]. QoS has also been explored for heterogeneous systems [Zhang et al., 2017] using different policies. Since the advent of cloud computing, research efforts have often included scheduling to accommodate billing for services [Rodriguez and Buyya, 2017]. In the area of machine learning to help understand computation performance and guide scheduling, a number of efforts address intelligent workflow scheduling in the cloud arena [Cui et al., 2018].

**New Research Directions**

Several research directions involve PEM computation scheduling with constraints, intelligent scheduling, mapping hardware to an application, expression of constraints in a workflow, and development of useful cost models. Research can answer questions such as the following: What is a useful cost model for in situ analysis coexisting with applications such that it addresses QoS constraints? What kind of PEM interfaces can be provided to such a cost model? How can resource allocation and scheduling be tailored to an in situ cost model, and are new algorithms needed? What is the best way to express an analysis, application, or workflow to provide a resource allocation budget? How can intelligent scheduling (e.g., using ML) help with developing schedules for in situ analysis workflows? Can such scheduling be inclusive, involving network, storage, compute, and edge hardware?
Scheduling, mapping, and optimizing PEMs for in situ workflows relate to usability and performance; a good interface to this functionality will allow easily specified constraints so that the PEM can map the machine to the application and meet those constraints. Scheduling also relates to composability of PEMs where multiple schedules may need to be considered (§ 3.7).

Potential Scientific Impact

Scientific impacts in this area support and maintain application quality while benefiting in situ analyses. Optimized workflows could expand the use of in situ analyses to domains where they may not have been previously feasible. PEMs that support QoS guarantees and advanced scheduling capabilities can improve the portability of the software that uses them because workflows can be rescheduled on new resources if necessary. Optimal scheduling can have scientific benefit by allowing applications to run on multiple platforms. Success in this area can enable in situ computations on non-HPC platforms (e.g., edge devices) where automated workflow scheduling was previously infeasible because of unique performance and timing requirements.

3.6.3 PEMs for Composable In Situ Workflows

Key Challenges and Opportunities

Scientific applications can benefit from in situ composition of more than one analysis software component or framework. Often, in situ software components or frameworks use PEMs that are different from simulations and/or from each other, making it much more challenging to connect them. PEMs can include bulk-synchronous models, task-based models, synchronous or asynchronous models, scripting languages such as R or Python that have restrictions on their execution (e.g., no true multithreading), web or browser-based functionality, on-node or distributed databases that may be dependent on servers, container-based analyses, and services or microservices that have their own scheduler.

Connecting PEMs is challenging for numerous reasons. One is that PEMs have different ways of acquiring resources and may be unaware of each other’s use of resources—cores, processors, nodes, memory, storage, accelerators—that can occur at many levels of the operating system. Also, data may need to be shared across PEMs, but PEMs have different data models, requiring copying, which takes time and may lead to redundancy and potential inconsistency. Furthermore, some programming models rely on different underlying file systems (such as MapReduce and HDFS), which can create further barriers to composition. PEMs may have differing isolation policies; some programming models may be purely functional in nature, not allowing side-effects within functions, while others may allow global variables. Debugging across programming models is also challenging because call stacks can exist in different address spaces.

State of the Art

With some in situ frameworks, one PEM is chosen (usually the same as the application PEM), and the in situ functionality is available in a tightly coupled manner (e.g., as a library call within the program execution space or called via the same PEM used by the main program). Programming models also exist for loosely coupled in situ workflows (the in situ functionality is available via another PEM outside of the application execution space), usually in a restricted way and usually between just two PEMs [Docan et al., 2012, Dorier et al., 2012, Dreher and Peterka, 2017, Vishwanath et al., 2011, Capul et al., 2018]. In situ workflows can also be composed with distributed task-based ones in a two-level hierarchy [Yildiz et al., 2019]. Another example is the interface between two programming models, such as allowing Python to call R functions and vice versa. An interesting application example is the use of the Legion task-based PEM in conjunction with MPI in the S3D combustion application; a handoff (a data copy) is done between MPI and Legion, and analyses are done by Legion tasks [Pebay et al., 2016].

Some researchers are studying converged programming models. For example, IBM has DataBroker [D’Amora, 2018], an in-memory distributed key-value store that enables data sharing between applications in a workflow. The big data extreme-computing community has also looked toward “convergence” [Asch et al., 2018]. Projects such as Apache Arrow [Apache Software Foundation, 2019a] are investigating
composability of big data PEMs and other tools via a common in-memory data layer. Other projects such as Spark-DIY [Caino-Lores et al., 2018] converge Apache big data PEMs such as Spark [Zaharia et al., 2010] with DOE HPC ones such as DIY [Morozov and Peterka, 2016].

**New Research Directions**

New research can connect programming models, run-time systems, and data models by asking some of the following questions. What is needed to access different data models by PEMs? How can memory be managed for in situ computations, optimized across all layers and across programming models and run-time systems? How can PEM policies co-exist and work with each other? How will computations be scheduled across programming models? How can code that runs across multiple PEMs be debugged? What are the best sizes for composable building blocks that would be supported by a PEM for in situ motifs or patterns?

In the future, composing multiple PEMs will feature deeper hierarchies with compositions of more than just two PEMs. PEMs need to be composed with finer-grained integration than the coarse-grained handoffs between tools in present-day examples. Also needed are standardized interfaces for composing PEMs, beyond the present-day one-off solutions.

Success in composability of programming models depends on data model research (§ 3.2). PEM research should also look toward providing abstractions that enable software system interoperability (§ 3.3)—software architecture will depend on those abstractions, and computing platforms can also provide appropriate abstractions to PEMs for system resources.

**Potential Scientific Impact**

One of the biggest potential impacts of composable PEMs is the ability to enable composable workflows that include applications, analyses, and other software services in ways that allow custom in situ analyses or generalized capabilities that were not possible before. Efficient coupling of PEMs could facilitate performance improvements for coupled applications and in situ analyses that previously had to communicate via data copying. Usability of workflows would increase as users extend in situ analyses to experimental and observational regimes as well as to multiphysics applications.

3.6.4 **PEMs for Streamed Data**

**Key Challenges and Opportunities**

Streaming data to in situ data analytics and visualization is a mission-critical need for many application areas, such as measurements from experiments, observations from instruments, data from simulations, or provenance information from various sources.

Streaming challenges are related to different levels or types of coupling between stream producers and consumers, whether at the edge or in HPC facilities. Data from multiple applications can be streamed, and each application may run on a different set of resources. Several challenges arise in minimizing communication time, mitigating errors, being resilient to disruption, ensuring data quality, and programming the workflow.

Stream coupling between remotely distributed instruments and HPC systems poses additional challenges. Data from the experiment need to be ingested into a running simulation; however, some preliminary analysis, data reduction, and data cleaning may need to be done at the edge before data leave the instrument. A human may be needed in the loop to make decisions about what data are streamed to HPC, and additional computational support may be needed for experiment-time decision-making involving streamed data.

**State of the Art**

Streaming management, translation, and reduction services are provided through message queues. For example, the Accelerating Data Acquisition, Reduction and Analysis system (ADARA) [Shipman et al., 2014] at the Spallation Neutron Source streams data from acquisition to a publish-subscribe system where data reduction is done. Another tool that uses a message-queueing PEM for streaming data analysis at
ENABLING SCIENTIFIC DISCOVERY FROM DIVERSE DATA SOURCES

experimental facilities is Xi-cam [Pandolfi et al., 2018]. An edge-based streaming PEM is the Psana system [Damiani et al., 2016] at SLAC. ICEE [Choi et al., 2013] is an in-transit data analysis framework over a wide area network (WAN) operating on Tokamak advanced research (KSTAR) data. It uses indexing and querying to search streamed data efficiently. Strongly coupled physics and data streaming has been done for large-scale kinetic turbulence codes across an interface [Dominski et al., 2018]. High-energy physics data analytics are done on streamed experiment data using the Turbo Stream framework [Aaij et al., 2016], and in situ ray tracing can be done on simulation data [Usher et al., 2016] using the Uintah framework. ML is used on data streams in the cloud using various frameworks [Bifet et al., 2018, Apache Software Foundation, 2019b].

New Research Directions

Numerous research questions can be investigated. How can communication time in streaming PEMs be minimized? How can we ensure that no errors exist in the data? How can the most important information be prioritized such that data are streamed at a certain quality level, with the ability to end the stream at any point, while ensuring a certain level of accuracy? Ultimately, how are such requirements programmed in a workflow?

Research is needed in the areas of prioritizing the information from the data, estimating the error from some parts of the data, and ensuring that the consumer data processing is not the bottleneck. For wide-area data streams, co-scheduling is a research issue, as well as resiliency, redundancy, and limiting of data movement over the WAN.

Research is needed in programming model support for coupling of streamed data consumers and producers and in understanding their resilience to stream interruptions or lack of connection. Also needed are asynchronous programming models that support the coupling of streamed data and the ability to express a “timeliness” factor to match sources and sinks. Research into the relationships of a collaborative streaming workflow is needed to better understand and support collaborative streaming via programming models.

This research depends on understanding a wide variety of workflows, including in situ analysis on streamed data, in order to glean the differences and commonalities between these workflows and see where PEM research can be leveraged. How much data are streamed? What are the latencies? What data types and formats are used? Which consumers can wait? What is the graph of stream producers and consumers; is there sufficient connectivity, and how could stream coupling be improved with additional support from PEMs? Another dependency of this research is analysis algorithms (§ 3.4) for streamed data.

Potential Scientific Impact

Support for streaming in PEMs has three benefits. The ability to stream data from a simulation to another simulation, experiment, or analysis means that the physics can be coupled together with potentially other workflows. The cost of using experimental facilities can be reduced because experiments can potentially use a wealth of simulation and ML data to steer clear of certain instabilities or erroneous conditions. Ultimately, PEMs supporting streaming data can ensure better experiments.

3.7 Software Architecture for Usability and Sustainability

The ISDM “software environment” is diverse and multifaceted; this topic area examines the design of high-level software architecture that promotes the usability, utility, and longevity of ISDM methods and infrastructure. In the ISDM context, the software environment may be thought of as enabling the execution of a dynamic graph of multiple data producers and consumers. Producers may be simulations, experiments, or combinations of both. Consumers ingest and process data and may in turn become data producers for other downstream consumers. Nodes in the graph are processing tasks (e.g., analysis algorithms § 3.4), and edges dictate data dependencies. These processing tasks may be diverse and could be scheduled for execution on different types of hardware (§ 3.6).

What design considerations will promote longevity of ISDM software adoption and use by the science community? Three subtopics were proposed in order to help answer this question: the ISDM software ecosystem, usability and adoption, and sustainability and growth.
Software and ecosystem architecture. The participants were asked to describe what an “ideal” ISDM software ecosystem would look like from the view of a researcher or developer, end user, and the ASCR research portfolio. In designing such an ecosystem, what choices promote reuse and inclusion of software tools from many sources in a composable fashion, such that an ISDM software ecosystem maximizes use of technology from diverse sources?

Usability and adoption. Attendees considered the pros and cons of using ISDM software infrastructure and tools as opposed to deploying bespoke technologies and methods directly into an application. They identified different types of software that users and developers will interact with, extend, develop, and deploy. From the user’s (application, library, or analysis developers) point of view, participants identified the impediments to using ISDM software and how these issues should be addressed. Furthermore, they considered what reusability means in this context and identified barriers to reusability.

Sustainability and growth. Participants identified impediments to the sustainability of ISDM software methods and tools, along with the research needed to address these issues. The discussion included topics related to the advantages and challenges of using diverse and emerging capabilities, such as machine learning tools, and alternative design and execution patterns, as part of the ISDM approach. We also asked attendees to examine what ISDM software research, development, and deployment needs from other areas, such as programming models, OS/R, operational policy at HPC centers, distributed workflows, and data streaming.

Discussion points during the breakout covered numerous topics. For example, functional decomposition of software into lightweight, independent pieces would allow composability of software modules built on common infrastructures. This would promote reuse at various levels, from full stacks to individual components. The need for community standards defining broadly accepted interface specifications was also raised. Given interface standards, software abstractions in middleware are an effective shield against changing implementations, and they are a mechanism for portability across various platforms, from HPC to edge devices. Exploiting third-party tools from sources other than DOE has several advantages, but those tools also have their own dependencies, often entire software stacks and run-times. Testing must include performance, not only correctness, as a necessary property of correctly tested and working software. The need for facility engagement was also recognized as a key component of software sustainability.

The following five research areas captured most of the discussion points. An increasing diversity of science use cases will drive the evolution of ISDM software design and architecture and its use at science user facilities (§ 3.7.1). Carefully thought-out software architecture and engineering practices, combined with a clear focus on science mission need, can increase ISDM software usability and sustainability (§ 3.7.2). Increasing the interoperability of software tools will increase productivity for science code teams and ISDM developers alike (§ 3.7.3). Software engineering practices of quality assurance and testing produce better ISDM software and will encourage broader use and adoption of ISDM technologies (§ 3.7.4). In supporting increasing diversity in science use cases, ISDM software will be developed, used, shared, and deployed in ways that go well beyond single-user and single-machine configurations (§ 3.7.5).

3.7.1 Diverse Use Cases Driving Evolution in Software Design

Key Challenges and Opportunities

The increasing diversity of science use cases drives the evolution of ISDM software architecture and design. One source of diversity is experimental and observational science projects, where data from an instrument may undergo processing (analysis, transformation, etc.) close to the instrument prior to being transmitted to a large-scale HPC facility for additional uses, such as digital twin studies and preparation of data products [Bethel et al., 2016]. This type of distributed processing includes a scientific version of what is often referred to as “edge” computing, where computing is performed “close to” the data when the data are first generated or acquired [Dastjerdi and Buyya, 2016].

Another source of diversity is a rapid growth in the heterogeneity of the underlying computational platforms and data movement and communication fabric [Vetter et al., 2019]. Concurrently, there is an expanding heterogeneity of the software ecosystem throughout the entire software stack, from third-party applications and libraries down to different run-time systems.
All this diversity and heterogeneity pose challenges for ISDM software architecture design. The opportunity is to identify and pursue key design concepts that will help achieve platform portability that is scalable both to large problem sizes and to high concurrency; that enables use of diverse software components, including third-party libraries such as machine learning libraries from industry and academia (e.g., TensorFlow [Abadi et al., 2016], scikit-learn [Pedregosa et al., 2011], etc.)

While this discussion focuses on how science use-case diversity presents challenges for software architecture, there is the matter of deployment and operation both at ASCR HPC user facilities and at other Office of Science user facilities; these issues are discussed in § 3.7.5. Other related issues, such as the interplay between ISDM software architecture and provenance and reproducibility, are discussed elsewhere in this report (§ 3.5).

State of the Art

In computational environments, the SENSEI generic in situ interface [Ayachit et al., 2016b] supports a use model where a code is instrumented once and then can connect to and make use of a diverse set of software tools for analysis and visualization. The design and software implementation have been shown to scale to over 1-million-way concurrency with production simulation and visualization codes [Ayachit et al., 2016a]. SENSEI also can accommodate diverse toolchains used for in situ analysis, including user-written Python, and third-party Python libraries, also run at scale [Loring et al., 2018]. Like SENSEI, the ASCENT in situ environment [Larsen et al., 2017] enables a simulation code to connect to different types of analysis or visualization tools on HPC platforms. One idea central to both these systems is the notion of a transport data model, which is a unit of exchange between producer and consumer. This idea is given consideration in this software architecture section, as well as elsewhere in this report (§ 3.2).

For addressing high-volume, high-throughput data streams from light source instruments, recent work has produced the Xi-cam software, an extensible platform for data management, analysis, and visualization [Pandolfi et al., 2018]. In Xi-cam, a user designs data-processing steps as a graph-based workflow. Underneath, these workflows are managed by Nanosurveyor [Daurer et al., 2017], which supports workflow execution either locally or remotely. Remote execution utilizes high-performance computing or delocalized resources, allowing for the effective reduction of high-throughput data. Xi-cam’s plugin-based architecture targets cross-facility and cross-technique collaborative development, in support of multimodal analysis. Such architectures are useful in approaching the broader challenges of scientific data lifecycle management [Bethel, 2017].

Other large-scale experiments, such as ATLAS and CMS, have made a significant investment in software infrastructure for managing data and computations. More broadly, the HEP community recognizes the need for software architecture approaches that have foresight for needs from large-scale experiments, which have lifetimes that can span multiple decades [Hildreth et al., 2018].

New Research Directions

A good starting point for research is to identify a set of representative science use cases and their requirements for ISDM. This set of use cases is analogous to the “Thirteen Computational Dwarfs” from 2006 [Asanovic et al., 2006]. Those dwarfs provided a scaffold for thinking about different types of R&D challenges in the HPC space, ranging from programming models (§ 3.6) and run-time systems (§ 3.3) to key algorithmic kernels (§ 2.3). A similar set of “dwarfs” for ISDM use cases would in turn provide the science-grounded scaffolding for ISDM software design, development, and deployment. Beyond dwarfs, there is no substitute for use cases at full production scale, where problems may appear that do not arise at smaller or more limited scale. Such problems may occur with hardware, run-time, programming environment limits, and so forth.

Other research directions include using these dwarfs as the basis for design of ISDM software components and their common data model approaches for run-time management of software components and resources (data storage, staging, and movement; compute; resource co-scheduling, etc.). Such an approach is often called “co-design.”

---

2 See http://als.lbl.gov the Advanced Light Source at Lawrence Berkeley National Laboratory, and http://www.aps.anl.gov, the Advanced Photon Source at Argonne National Laboratory
Crosscutting these computational use cases are the design and implementation of methods that can run on diverse platforms: laptops, HPC platforms, and edge devices (e.g., an FPGA close to the instrument). These require the design of software infrastructure and algorithms that are platform portable, are scalable, and have longevity through some amount of insulation from changes in the technology of the underlying computational platforms.

The diversity of use cases necessitates engagement with various collaborators and stakeholders: science and HPC facilities, experimental and simulation sciences, and the gamut of computer science research: OS/R, programming languages, data science, and performance measurement and analysis.

**Potential Scientific Impact**

For science (simulation and experimental), access to a richer collection of software tools for performing data-centric operations could lower cost, since individual teams do not have to create them “from scratch” on their own. A corresponding increase in the efficiency of science will result by leveraging off-the-shelf tools, rather than having to write all new software. Such tools will also broaden the reach of ASCR investments, with a positive impact on more science projects and beyond-HPC-only use scenarios. For engineering, uncertainty quantification in multiphysics environments will benefit from simplified interoperability at scale. Here, ISDM is vital for code coupling as well as analysis. Off-the-shelf availability again will be important to speed adoption. For industry, “simulate to certify” is a goal for aerospace and automotive manufacturers. This is forcing a move to more unsteady calculations and ensembles or high-fidelity runs for uncertainty quantification. Here, too, ISDM is an essential enabling technology.

### 3.7.2 ISDM Software Usability and Sustainability

**Key Challenges and Opportunities**

The term *usability* refers to the extent to which a (software) tool or product may be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use [Nielsen Norman Group, 2019]. This broad concept is often tailored for a specific use scenario, such as in the case of user experience applications, where usability means “how effectively, efficiently, and satisfactorily a user can interact with a user interface.” [Usability.gov, 2019] Within the context of ISDM software, usability refers to similar concepts: the degree to which a software tool performs a specific function with correctness and efficiency and the relative ease of use of ISDM software tools. Here, relative ease of use is to be considered broadly and includes concepts such as insulating users from complexity of software systems and computational platforms.

The term *sustainability* refers to several concepts [Venters et al., 2014]. First, *preservation* refers to the idea of preserving the function of a system over a defined time span. Second, *maintainability* refers to the ability to modify a software system or component after delivery to correct faults, to improve performance or other attributes, or to adapt it to a changed environment. This concept also includes the notion of community support and adoption over time.

Given the increasing diversity of the science use cases that need ISDM software capabilities, combined with the increasing diversity of software tools and components being brought to bear on a spectrum of scientific data understanding challenges (§ 3.7.1), a significant usability challenge is realizing ease of use from user, developer, and maintainer perspectives. For example, ISDM software efforts would like to reuse tools from elsewhere, such as third-party software libraries from industry for performing ML. A significant amount of code has been designed for different applications and ecosystems from third parties that science projects would like to use in order to avoid incurring costs of reimplementation and ongoing maintenance. Open questions include how to bring those tools into our scientific HPC ecosystem, how to demonstrate scalability, and how to integrate into scientific codes for in situ use.
State of the Art

Software is both a digital artifact and "machinery" that takes inputs, performs operations on those inputs, and produces outputs. A diverse set of stakeholders—representing academia, industry, funding agencies, and scholarly publishers—came together to design and jointly endorse a concise and measurable set of principles referred to as the FAIR Data Principles [Wilkinson et al., 2016]: Findability, Accessibility, Interoperability, and Reusability. The intent is that these principles apply not only to "data" in the conventional sense but also to the algorithms, tools, and workflows that led to those data. The idea is that all components of the research process must be available, to ensure transparency, reproducibility, and reusability.

Several noteworthy examples of "community software" tools and environments provide positive dimensions of both usability and sustainability. One well-known example is the Visualization Toolkit (VTK) [Schroeder et al., 2006], which is a collection of data structures and algorithms for visualization and graphics. This foundational project has served as the delivery vehicle for research methods in both visualization and graphics from diverse sources, including academia, government labs, and industry. As a library, VTK's data structures, data models, and algorithms provide the underpinning for multiple HPC-capable visualization applications, such as VisIt [Childs et al., 2012a] and ParaView [Ayachit, 2015]. To highlight the importance of these key community projects, the DOE SciDAC2 Visualization and Analytics Center for Enabling Technology (VACET)3 leveraged VisIt [Childs et al., 2012a] as the basis for deploying and delivering new visual data analysis capabilities to the worldwide scientific community. VACET's accomplishments improved the usability of production-quality, petascale-capable visualization software, as demonstrated by deployment of new scalable methods run on problems of unprecedented size and scale: 2 trillion grid cells on 32,000 cores of JaguarPF [Childs et al., 2012b].

At the community level, individuals make contributions to a larger collection, with examples including image analysis and computer vision libraries (OpenCV [Bradski, 2000], scikit-image [van der Walt et al., 2014]), operating systems (Linux), machine learning frameworks and libraries (TensorFlow [Abadi et al., 2016], PyTorch [Paszke et al., 2017], Keras [Chollet et al., 2015]), compilers (gcc), scientific visualization libraries (VTK [Schroeder et al., 2006]), and applications (VisIt [Childs et al., 2012a] and ParaView [Ayachit, 2015]).

The ROOT software system is a mature software package in widespread use by the NP/HEP community [Brun and Rademakers, 1997]. It provides capabilities for data storage and access, analysis, and visualization. To improve usability, the ROOT team has a deployment with containers that helps insulate users from the complexity of changing dependencies in complex HPC software ecosystems. ROOT origins date back to earlier work from the 1980s, the Physics Analysis Workstation (PAW) [Brun et al., 1989]. Over time, a duplication in functionality has emerged. For example, ROOT provides charting/graphing capabilities that are duplicates of capabilities in R. Such duplication can result in inadvertent isolation of one user community from developments elsewhere in other communities.

Elsewhere, the Department of Defense’s CREATE program is an example of a large, HPC-enabled multiphysics analysis project that has mastered both the framework and data model issues in a production suite. All the components are linked via an ISDM backbone, including use of DOE-funded VisIt/Libsim [Whitlock, 2018], which has been integrated to create data extracts for subsequent analysis [Post et al., 2015].

New Research Directions

Several different research directions address different dimensions of usability. We need to identify factors that contribute to making ISDM software more usable for a diverse population that includes tool developers, science users, and researchers; what may be ideal for one group may be not that useful for another. Underpinning the concept of software usability is the degree to which software can be easily located, used (or reused), integrated, and deployed. Central to this characteristic is the need to identify the appropriate types of interfaces and well-defined data models (§3.2) for ISDM software, both at the system level and at the tool/component/method level. Another direction is considering alternatives for usability, reusability, and “componentization” at different levels of granularity, from a coarse-grained level, such as an application, to a very fine-grained level, such as a highly specialized library.

---

3 VACET website: http://www.vacet.org
Following the FAIR principles for scientific data management [Wilkinson et al., 2016], both the ISDM R&D community and scientific user communities would benefit in finding approaches for community repositories and collections of ISDM software tools. As a result, software would be easy to find, have appropriate metadata (e.g., documentation), and be more reusable (e.g., have interfaces and data models that are consistent with community guidelines and practices).

To promote usability, science users and stakeholders ideally will be engaged in the full, iterative design and evaluation process. That is, users who are domain scientists should make a commitment to participate in both formative and summative tests. The development of most previous ISDM techniques failed to involve domain scientists throughout the whole process, resulting in low software usability. Conducting user experience and usability studies requires a different set of skills that not every computer scientist possesses. Future ISDM projects should consider involving usability researchers.

While the concepts of usability and sustainability are intertwined, some specific research directions emerge when considering sustainable ISDM software. We need a deeper understanding of what specific factors will be important for an ISDM software ecosystem to be sustainable for 5, 10, or 20 years into the future. The HEP community, for example, is concerned about the long-term sustainability of its key software infrastructure through the multidecade lifespan of the Large Hadron Collider [The HEP Software Foundation, 2019]. We need to cultivate a deeper understanding of how deployment alternatives for package and delivering ISDM software to the scientific community will increase usability and sustainability.

The idea of sustainability and usability is something that is, or should be, pervasive across the entire ecosystem. In other words, if one area, such as ISDM software, focuses on sustainability and usability but other related areas do not, then the ISDM software sustainability and usability may not be as impactful as desired or hoped. Needed are resources for the R&D community to preserve curated software, sample data, and results; for example, while GitHub is ideal for source code, it may not be the right answer for long-term preservation and dissemination of digital artifacts.

**Potential Scientific Impact**

Since software is a central part of virtually all modern science projects [Bethel et al., 2016], efforts that make software more usable and long-lived will have several different potential impacts. Cost savings result when science projects are able to quickly bring in and use third-party software tools for any number of operations, ranging from modeling and simulation, to data-intensive activities such as statistical analysis or visualization. A related benefit is an increase in productivity when science teams can “buy” rather than “build” software. Those teams are spending more time doing science and less time doing software development and maintenance. Assuming that third-party tools have undergone rigorous testing and validation, then science projects may have increased confidence in the results of analysis. By taking a longer-term and holistic view of ISDM software, science benefits by having more stability and predictability in critical infrastructure.

Although leveraging third-party software can help with costs and productivity, it is only a part of a larger picture. The larger picture includes the idea that software projects have long lifetimes, and therefore sustainability needs to be part of the consideration. Careful attention to ISDM software design, which includes close and ongoing engagement with science stakeholders, fosters conditions favorable to the emergence of a sustainable ISDM software ecosystem.

### 3.7.3 Software Tool Interoperability

**Key Challenges and Opportunities**

Increasing the interoperability of software tools will increase productivity for science code teams and ISDM researchers and developers alike. Of the many software design and architecture issues pertaining to interoperability, those pertaining to interface design and data model are foundational. The issues of interoperability and usability are closely related: to be usable, the software components need to be able to work together with a minimum amount of effort (e.g., when converting between data models).
State of the Art

Interoperability requires finding the right balance in community practices between being too prescriptive and insufficiently precise. At the most fundamental level is the notion of data file interoperability. An example is the climate forecast conventions [Lawrence et al., 2005] for NetCDF files, where the climate community has agreed on metadata guidelines for specifying factors such as date conventions and units of measure. Here, the metadata conventions are what enable the data file interoperability: without them, the data files cannot be written and read by different codes. This idea of metadata conventions as the basis for interoperability applies as well to HDF5 files [Folk et al., 2011]. For example, the VisIt software provides a number of different file loaders [Whitlock, 2010]: Exodus and Silo are two popular data models, and both use the HDF5 data format. However, an Exodus loader cannot read Silo files, and vice versa, even though both are HDF5 files, because they use different metadata conventions within the HDF5 file.

At the next level is tool interoperability. A good example is Python. A NumPy array is the central data model, and Python-based tools that use or produce NumPy arrays may be combined: they are interoperable if there is agreement in how data are laid out in the array (e.g., interleaved, row-major), in the semantic meaning of data, and so forth.

Another dimension of interoperability concerns interface semantics and composition. For simple data type conversion, the C++ language standards and compilers provide operator overloading, so that a developer may provide data of one type as input to a method that expects data of a different type, such as type mismatches between arrays of integers and arrays of doubles. Beyond this simple data type conversion, it is up to the developer to rectify differences in data models and parameter ordering and semantics. For examples, software tools such as SENSEI [Ayachit et al., 2016b], ASCENT [Larsen et al., 2017], and ADIOS [Liu et al., 2014] map native data models to “bridge” data models as a way to rectify potential data differences between producer and consumer data.

New Research Directions

Establishing community guidelines for data models and interfaces would foster the growth of composable, interoperable software tools. Related research efforts would help with migration toward community guidelines. Such efforts might provide the means to identify incompatibilities, perhaps at the interface and data model level, and suggest (or automatically create) software solutions to such incompatibilities.

Interoperability should be considered in a broad sense that includes heterogeneity [Vetter et al., 2019] of design and execution patterns in computational platforms, in computational and data-centric workload characteristics, in data lifecycle requirements, and in temporal constraints and characteristics; considering use by computational, experimental, and observational science communities across a spectrum of science user facilities [Bethel et al., 2016].

Modularity and interoperability are highly dependent on the existence of a common data model (§3.2), on the ability to combine multiple components or tools in a way that facilitates tracking of provenance (§3.5), and on being able to take advantage of computational platform capabilities (§3.3).

Potential Scientific Impact

When science projects are able to “buy rather than build” the critical software infrastructure they need for their projects, they save time and money. Additionally, with careful planning, the investments made in key software infrastructure may reasonably be expected to be sustainable for many years.

With careful attention to community guidelines, there is a path forward to deployment for methods produced by the ISDM R&D community. This path forward helps facilitate the longevity of an R&D investment and helps lower barriers to use by science code teams and other third-party users.
3.7.4 User Confidence and ISDM Software

Key Challenges and Opportunities

One of the major impediments standing in the way of a science code team adopting third-party software tools, such as ISDM, is the issue of trust. Specifically, there is concern that third-party software might not be well engineered, could break in unexpected and potentially catastrophic ways, and may not actually work on the desired computational platform.

Another factor is concern about the sustainability of software over time. Quoting from the Roadmap for HEP Software and Computing R&D report [The HEP Software Foundation, 2019], “there is considerable anxiety in the experiments that much of this software is not sustainable, with the original authors no longer in the field and much of the code itself in a poorly maintained state, ill-documented, and lacking tests.” This concern is not limited to HEP: it is present in a diversity of DOE mission science programs [Bethel et al., 2016]. A related concern is ongoing support and maintenance: a greater likelihood of trust exists when a tool, technology creator, or provider has a history of providing timely user support, responding to bug reports, and providing updates or fixes over time.

Addressing these challenges requires careful attention to software architecture and design, implementation, release engineering, testing, deployment, and sustainment. These factors contribute to software robustness, correctness, and longevity, and in turn foster increased trust and adoption by the science stakeholder community.

State of the Art

Modern software practices that help with both sustainability and user confidence include use of community-based development workflows based on services such as GitHub and GitLab, automatic and manual documentation generation through services such as readthedocs.org, and generalized cross-platform software building and testing capabilities through tools such as CMake. Such services encourage community participation; and by facilitating recording of code provenance and documentation, they also provide a “safety net” for future developers and users.

Another dimension of best practices that help with user confidence is well-demonstrated building and testing infrastructure. Software quality assurance practices may focus on different logical layers of testing such as unit tests or integrated tests. The notion of continuous integration refers to the idea of ongoing testing of software in different target deployment environments [Atlassian.com, 2019]. This approach helps quickly catch problems that inevitably occur when software elements in the target ecosystem change, as can happen with system software upgrades or changes to third-party libraries.

Within the DOE community several software engineering projects have adopted these best practices. DOE’s Exascale Computing Project uses the process of continuous integration [Montoya, 2019], to continually test and integrate software at the DOE facilities supporting ECP software products, applications, and associated software environments. The intention is to ensure a healthy software ecosystem through continuous testing. To facilitate ease of deployment, the ECP is increasing accessibility to its software products by using the Spack package manager [Gamblin et al., 2015] when doing software releases [Neely, 2019]. The vision for long-term sustainability includes careful testing, including continuous integration, and project-wide release engineering, e.g., in the form of software development kits (SDKs). Projects such as VTK [Schroeder et al., 2006], VisIt [Childs et al., 2012a], and ParaView [Ayachit, 2015], which are staples in the HPC visual data analysis and exploration community, all use best practices in software and release engineering. As a result, all have earned the confidence and trust of a worldwide community of users and developers.

New Research Directions

While component-level testing is reasonably well understood, testing of multistage ISDM tools and workflows, especially those involving experimental or observational facilities, is significantly more complex. To help limit complexity, one may wish to consider a set of common ISDM use cases (dwarfs, § 3.7.1). The research
question here is what to test and measure in order to increase confidence. For example, testing could measure scalability (for both problem size and level of concurrency) and include performance analysis and modeling. Multistage testing may be possible, from coarse grained to fine grained, and include measurements for data movement between components and to/from storage, memory footprint, and compute operations.

While continuous integration testing is useful, unique complexities and challenges arise in HPC environments. Performing frequent continuous integration testing on anything but small-scale tests is impractical, but ISDM tools must perform well at large scale. So, although a small-scale test will likely catch most correctness issues with the code, large-scale issues that do not occur at small scale are also known to occur. How to quickly catch and repair large-scale issues with continuous integration testing is unclear.

Successful software interoperability requires the existence of tools for multiscale performance analysis at the scale of individual ISDM methods, as well as combinations of methods.

Potential Scientific Impact

With an increased focus on testing and deployment strategies as part of the overall ISDM software strategy, the intention is to produce software that is more robust, more usable and accessible, and more sustainable. Since these factors contribute to trust in software, the ultimate objective is for increased adoption and use of ISDM software in DOE mission science.

3.7.5 Science Facilities Partnerships

Key Challenges and Opportunities

Meeting the challenges resulting from the increasing diversity of ISDM software tools and science use cases will entail deeper interactions and partnerships with science and computing facilities. The deployment and use at computational and scientific user facilities of ISDM involves several activities, entailing use of facility-level resources for resource discovery and marshaling, for specifying use of heterogeneous resources, and for managing complex multistage processing pipelines. One must marshal the resources (compute, storage, data movement) needed to execute a task. A task can be optimized by specifying heterogeneous resources, each targeting some stage of processing (e.g., GPU for a compute-intensive task, solid-stage storage for low-latency temporary storage of intermediate results). Additionally, the ISDM software will need to set up and monitor workflow execution, which may occur within a single HPC facility as well as across distributed locations.

State of the Art

In support of BES workloads from the Advanced Light Source, software tools such as Xi-cam [Pandolfi et al., 2018] and Spot Suite [Vu, 2014] provide the ability to perform processing close to the beamline, to move experimental data over ESnet to NERSC, to schedule additional processing at NERSC, and to prepare collections of data products accessible by individuals and groups of researchers. This use case is a glimpse of what is possible as HPC centers evolve from providing only floating-point operations to providing several types of resources for data-intensive science [Vetter et al., 2016]. As part of its growth to better support experimental and observational science efforts, NERSC has provided web-based APIs for accessing services [National Energy Research Scientific Computing Center, 2019b] as well as adding real-time queues to service time-critical workloads. When combined with networking infrastructure optimized for high-volume scientific data movement [Dart et al., 2014], the resulting set of resources is helpful for meeting the needs of on-demand, high-throughput processing.

Inside the HPC center, where an ISDM task may require multiple independent tasks to work together, early projects encountered obstacles whereby the system software would not permit two independent MPI-based applications to share an address space. The Henson system [Morozov and Lukić, 2016] works around this limitation by using cooperative multitasking, enabling multiple programs to run concurrently and interact in a coupled, in situ fashion.
New Research Directions

Leveraging a well-defined set of ISDM design patterns and use cases (e.g., ISDM dwarfs § 3.7.1) provides an opportunity to ask well-grounded questions about how to discover resources, how to specify and marshal their use, and how to manage execution across potentially heterogeneous and distributed resources. Similarly, while run-time systems (§ 3.6) for traditional HPC workloads have been well studied, run-time environments for ISDM workloads likely will have some similar requirements and also many new and different requirements. Having a better understanding of these similarities and differences in requirements will help facilitate a dialogue between science stakeholders with specific workloads, ISDM researchers, and facilities personnel.

Since ISDM workloads will need to run on heterogeneous resources, sometimes in a distributed fashion across multiple facilities, such workloads will require support for the discovery and use of heterogeneous resources: for example, being able to specify and run multiple parallel components on different classes of resources—some on GPUs, some on CPUs, and some on edge devices.

Potential Scientific Impact

Increasing the interactions and synergy between ISDM researchers, science stakeholders, and science user facilities will create new opportunities for science. For example, it will increase the number of options for science teams in terms of running data-centric ISDM-like workloads. It will help promote the standardization and interoperability of ISDM software tools. Moreover, it will encourage the evolution of HPC facilities, in terms of policies and offerings, to better support increasingly diverse scientific needs in data management and data understanding.

While “hero” ISDM workflows have been demonstrated to run successfully across multiple sites [Bashor, 2015], the ultimate objective is for such endeavors to be routine and reliable. The impact will be improved scientific productivity—which results from less time spent engineering custom, bespoke solutions in favor of approaches that are based on services provided by the science user facilities—and use of well-engineered ISDM software. An added impact and benefit of “routine” efforts is that they are more likely to be reproducible. Therefore, the complex data-intensive pipelines that are characteristics of current science projects will be increasingly reproducible, and sharable across projects.
4 Conclusions

Scientific computing will increasingly incorporate a number of different tasks that need to be managed along with the main simulation or experimental tasks—ensemble analysis, data-driven science, artificial intelligence, machine learning, surrogate modeling, and graph analytics—all nontraditional applications unheard of in HPC just a few years ago. Many of these tasks will need to execute concurrently, that is, in situ, with simulations and experiments sharing the same computing resources.

The workshop revealed two primary, interdependent motivations for processing and managing data in situ. The first motivation is that the in situ methodology enables scientific discovery from a broad range of data sources—HPC simulations, experiments, scientific instruments, and sensor networks—over a wide scale of computing platforms including leadership-class HPC, clusters, clouds, workstations, and embedded devices at the edge. The successful development of ISDM capabilities will benefit real-time decision-making, design optimization, and data-driven scientific discovery. The second motivation is the need to decrease data volumes. ISDM can make critical contributions to managing large data volumes from computations and experiments, with the aim of minimizing data movement, saving storage space, and boosting resource efficiency—often while simultaneously increasing scientific precision.

A fundamental finding of this workshop is that the methodologies used to manage data among a variety of tasks in situ can be used to facilitate scientific discovery from a variety of different data sources—simulation, experiment, and sensors, for example—and that being able to do so at a variety of computing scales will benefit real-time decision-making, design optimization, and data-driven scientific discovery across the Office of Science mission space. Applications wanting to use the in situ capabilities include those where data analysis feeds back to the simulation, decisions are made autonomously, big data or machine learning is among the tasks to be coordinated, and computations need to be completed in real time.

The workshop identified six PRDs that highlight the components and capabilities needed for ISDM to be successful for the wide variety of applications discussed: making ISDM capabilities more pervasive, controllable, composable, and transparent, with a focus on greater coordination with the software stack, and a diversity of fundamentally new data algorithms.

- **Pervasive ISDM**: Apply ISDM methodologies and in situ workflows on a variety of platforms and scales.
- **Co-designed ISDM**: Coordinate the development of ISDM with the underlying system software so that it is part of the software stack.
- **In Situ Algorithms**: Redesign data analysis algorithms for the in situ paradigm.
- **Controllable ISDM**: Understand the design space of autonomous decision-making and control of in situ workflows.
- **Composable ISDM**: Develop interoperable ISDM components and capabilities for an agile and sustainable programming paradigm.
- **Transparent ISDM**: Increase confidence in reproducible science, deliver repeatable performance, and discover new data features through the provenance of ISDM.
References


78
ENABLING SCIENTIFIC DISCOVERY FROM DIVERSE DATA SOURCES


Page left intentionally blank.
### Appendix 1: List of Abbreviations

**Table 2: List of Abbreviations**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>artificial intelligence</td>
</tr>
<tr>
<td>AMR</td>
<td>adaptive mesh refinement</td>
</tr>
<tr>
<td>API</td>
<td>application program interface</td>
</tr>
<tr>
<td>ASCR</td>
<td>Advanced Scientific Computing Research</td>
</tr>
<tr>
<td>BES</td>
<td>Basic Energy Sciences</td>
</tr>
<tr>
<td>CEP</td>
<td>complex event processing</td>
</tr>
<tr>
<td>CP</td>
<td>Exascale Computing Project</td>
</tr>
<tr>
<td>DNN</td>
<td>deep neural network</td>
</tr>
<tr>
<td>DOE</td>
<td>Department of Energy</td>
</tr>
<tr>
<td>ECP</td>
<td>Exascale Computing Project</td>
</tr>
<tr>
<td>EH</td>
<td>extreme heterogeneity</td>
</tr>
<tr>
<td>EOS</td>
<td>experimental and observational science</td>
</tr>
<tr>
<td>FPGA</td>
<td>field-programmable gate array</td>
</tr>
<tr>
<td>GPU</td>
<td>graphics processing unit</td>
</tr>
<tr>
<td>HEP</td>
<td>High Energy Physics</td>
</tr>
<tr>
<td>HPC</td>
<td>high-performance computing</td>
</tr>
<tr>
<td>I/O</td>
<td>input/output</td>
</tr>
<tr>
<td>ISDM</td>
<td>in situ data management</td>
</tr>
<tr>
<td>ML</td>
<td>machine learning</td>
</tr>
<tr>
<td>MPI</td>
<td>Message Passing Interface</td>
</tr>
<tr>
<td>NERSC</td>
<td>National Energy Research Scientific Computing Center</td>
</tr>
<tr>
<td>NVRAM</td>
<td>nonvolatile random access memory</td>
</tr>
<tr>
<td>OS/R</td>
<td>operating system/run-time</td>
</tr>
<tr>
<td>PEM</td>
<td>programming and execution model</td>
</tr>
<tr>
<td>PRD</td>
<td>priority research direction</td>
</tr>
<tr>
<td>QA</td>
<td>quality assurance</td>
</tr>
<tr>
<td>QoS</td>
<td>quality of service</td>
</tr>
<tr>
<td>SDK</td>
<td>software development kit</td>
</tr>
<tr>
<td>SLA</td>
<td>service-level agreement</td>
</tr>
<tr>
<td>SSIO</td>
<td>storage systems and input/output</td>
</tr>
<tr>
<td>UQ</td>
<td>uncertainty quantification</td>
</tr>
<tr>
<td>WAN</td>
<td>wide area network</td>
</tr>
<tr>
<td>WMS</td>
<td>workflow management system</td>
</tr>
</tbody>
</table>
Page left intentionally blank.
**Appendix 2: Workshop Agenda**

**Workshop on In Situ Data Management (ISDM)**  
Bethesda North Marriott Hotel and Conference Center  
5701 Marinelli Road, Rockville, MD 20852  
January 28 – 29, 2019  
Workshop website: [https://www.orau.gov/insitudata2019/](https://www.orau.gov/insitudata2019/)

### MONDAY, JANUARY 28, 2019

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:30 – 8:30 a.m.</td>
<td>Registration and breakfast available</td>
<td></td>
</tr>
<tr>
<td>8:30 – 9:15 a.m.</td>
<td>Opening remarks</td>
<td></td>
</tr>
<tr>
<td>9:15 – 10:45 a.m.</td>
<td>Plenary session: Science applications</td>
<td></td>
</tr>
<tr>
<td>10:45 – 11:00 a.m.</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>11:00 a.m. – 12:30 p.m.</td>
<td>Breakout session 1A: Data models: connection and communication</td>
<td>Breakout session 1B: Computational platforms and environments</td>
</tr>
<tr>
<td>12:30 p.m. – 1:30 p.m.</td>
<td>Lunch</td>
<td></td>
</tr>
<tr>
<td>1:30 – 3:00 p.m.</td>
<td>Breakout session 2A: Analysis algorithms</td>
<td>Breakout session 2B: Provenance &amp; reproducibility</td>
</tr>
<tr>
<td>3:00 – 3:30 p.m.</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>3:30 – 5:00 p.m.</td>
<td>Breakout session 3A: Programming &amp; execution models</td>
<td>Breakout session 3B: Software architecture for usability and sustainability</td>
</tr>
</tbody>
</table>

### TUESDAY, JANUARY 29, 2019

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:30 – 8:30 a.m.</td>
<td>Registration and breakfast available</td>
</tr>
<tr>
<td>8:30 – 9:15 a.m.</td>
<td>Summaries of related workshop activities</td>
</tr>
<tr>
<td>9:15 – 10:00 a.m.</td>
<td>Report back from breakout sessions 1A and 1B</td>
</tr>
<tr>
<td>10:00 – 10:45 a.m.</td>
<td>Report back from breakout sessions 2A and 2B</td>
</tr>
<tr>
<td>10:45 – 11:15 a.m.</td>
<td>Break</td>
</tr>
<tr>
<td>11:15 a.m. – 12:00 p.m.</td>
<td>Report back from breakout sessions 3A and 3B</td>
</tr>
<tr>
<td>12:00 – 1:00 p.m.</td>
<td>Lunch</td>
</tr>
<tr>
<td>1:00 – 2:30 p.m.</td>
<td>Prioritizing research directions</td>
</tr>
<tr>
<td>2:30 p.m.</td>
<td>Workshop adjourns</td>
</tr>
</tbody>
</table>

*Figure 14: Workshop agenda.*
Appendix 3: Workshop Participants

Table 3: Workshop Participants

<table>
<thead>
<tr>
<th>Name</th>
<th>Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jim Ahrens</td>
<td>LANL</td>
</tr>
<tr>
<td>Ann Almgren</td>
<td>LBNL</td>
</tr>
<tr>
<td>Katie Antypas</td>
<td>NERSC</td>
</tr>
<tr>
<td>Debbie Bard</td>
<td>NERSC</td>
</tr>
<tr>
<td>Edmun Begoli</td>
<td>ORNL</td>
</tr>
<tr>
<td>Janine Bennett</td>
<td>SNL</td>
</tr>
<tr>
<td>Wes Bethel</td>
<td>LBNL</td>
</tr>
<tr>
<td>Laura Biven</td>
<td>DOE</td>
</tr>
<tr>
<td>Amber Boehnlein</td>
<td>Jlab</td>
</tr>
<tr>
<td>Ron Brightwell</td>
<td>SNL</td>
</tr>
<tr>
<td>Benjamin Brown</td>
<td>DOE</td>
</tr>
<tr>
<td>Rich Carlson</td>
<td>DOE</td>
</tr>
<tr>
<td>Jacqueline Chen</td>
<td>SNL</td>
</tr>
<tr>
<td>Hank Childs</td>
<td>UO</td>
</tr>
<tr>
<td>Warren Davis</td>
<td>SNL</td>
</tr>
<tr>
<td>Marc Day</td>
<td>LBNL</td>
</tr>
<tr>
<td>Jai Dayal</td>
<td>Intel</td>
</tr>
<tr>
<td>Ewa Deelman</td>
<td>USC</td>
</tr>
<tr>
<td>Lei Ding</td>
<td>Accenture Labs</td>
</tr>
<tr>
<td>Nicola Ferrier</td>
<td>ANL</td>
</tr>
<tr>
<td>Fernanda Foertter</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Berk Geveci</td>
<td>Kitware</td>
</tr>
<tr>
<td>Katrin Heitmann</td>
<td>ANL</td>
</tr>
<tr>
<td>Barb Helland</td>
<td>DOE</td>
</tr>
<tr>
<td>Bruce Hendrickson</td>
<td>LLNL</td>
</tr>
<tr>
<td>Jay Hnilo</td>
<td>DOE</td>
</tr>
<tr>
<td>Adolphy Hoisie</td>
<td>BNL</td>
</tr>
<tr>
<td>Dan Jacobson</td>
<td>ORNL</td>
</tr>
<tr>
<td>Shantenu Jha</td>
<td>BNL</td>
</tr>
<tr>
<td>Ming Jiang</td>
<td>LLNL</td>
</tr>
<tr>
<td>Kirk Jordan</td>
<td>IBM</td>
</tr>
<tr>
<td>Scott Klasky</td>
<td>ORNL</td>
</tr>
<tr>
<td>Kerstin Kleese van Dam</td>
<td>BNL</td>
</tr>
<tr>
<td>Eric Lancon</td>
<td>BNL</td>
</tr>
<tr>
<td>Earl Lawrence</td>
<td>LANL</td>
</tr>
<tr>
<td>Steve Legensky</td>
<td>Intelligent Light</td>
</tr>
<tr>
<td>Jay Lofstead</td>
<td>SNL</td>
</tr>
<tr>
<td>Andrew Lumsdaine</td>
<td>PNNL</td>
</tr>
<tr>
<td>Kwan-Liu Ma</td>
<td>UCDavis</td>
</tr>
<tr>
<td>Pat McCormick</td>
<td>LANL</td>
</tr>
<tr>
<td>Ken Moreland</td>
<td>SNL</td>
</tr>
<tr>
<td>Dmitriy Morozov</td>
<td>LBNL</td>
</tr>
<tr>
<td>Ron Oldfield</td>
<td>SNL</td>
</tr>
<tr>
<td>H. Sarp Oral</td>
<td>OLCF</td>
</tr>
<tr>
<td>Manish Parashar</td>
<td>Rutgers University</td>
</tr>
<tr>
<td>Valerio Pascucci</td>
<td>SCI</td>
</tr>
<tr>
<td>Amedeo Perazzo</td>
<td>SLAC/Stanford</td>
</tr>
<tr>
<td>Tom Peterka</td>
<td>ANL</td>
</tr>
<tr>
<td>Robinson Pino</td>
<td>DOE</td>
</tr>
<tr>
<td>Beth Plale</td>
<td>Indiana University</td>
</tr>
<tr>
<td>Line Pouchard</td>
<td>BNL</td>
</tr>
<tr>
<td>Thomas Proffen</td>
<td>ORNL</td>
</tr>
<tr>
<td>Mike Ringenburg</td>
<td>Cray</td>
</tr>
<tr>
<td>Silvio Rizzi</td>
<td>ALCF</td>
</tr>
<tr>
<td>Rob Ross</td>
<td>ANL</td>
</tr>
<tr>
<td>Tim Scheibe</td>
<td>PNNL</td>
</tr>
<tr>
<td>David Schissel</td>
<td>General Atomics</td>
</tr>
<tr>
<td>Malachi Schram</td>
<td>PNNL</td>
</tr>
<tr>
<td>Katie Schuman</td>
<td>ORNL</td>
</tr>
<tr>
<td>Nicholas Schwarz</td>
<td>ANL</td>
</tr>
<tr>
<td>Han-Wei Shen</td>
<td>OSU</td>
</tr>
<tr>
<td>Eric Stephan</td>
<td>PNNL</td>
</tr>
<tr>
<td>Victoria Stodden</td>
<td>GSLIS/U Illinois</td>
</tr>
<tr>
<td>Ceren Susut</td>
<td>DOE</td>
</tr>
<tr>
<td>Christine Sweeney</td>
<td>LANL</td>
</tr>
<tr>
<td>Michela Taufer</td>
<td>UTK</td>
</tr>
<tr>
<td>Matthew Wolf</td>
<td>ORNL</td>
</tr>
<tr>
<td>Justin Wozniak</td>
<td>ANL</td>
</tr>
<tr>
<td>John Wu</td>
<td>LBNL</td>
</tr>
<tr>
<td>Hongfeng Yu</td>
<td>U Nebraska</td>
</tr>
</tbody>
</table>
DISCLAIMER: This report was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government.

Image credits:
3 High-energy X-ray test specimen. Image courtesy Y. Nashed.
5 Nucleated crystals in molecular dynamics simulation. Image courtesy O. Yildiz.
8 Supersampled high-energy coherent diffraction. Image courtesy Y. Nashed.
9 Phenotype vectors in genome wide association studies. Image courtesy D. Jacobson.
10 Rayleigh-Taylor instability. Image courtesy B. Loring.