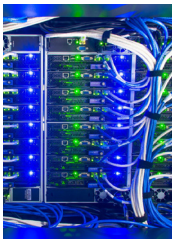
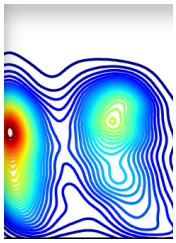
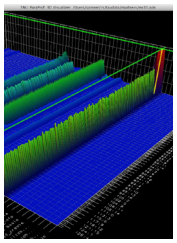
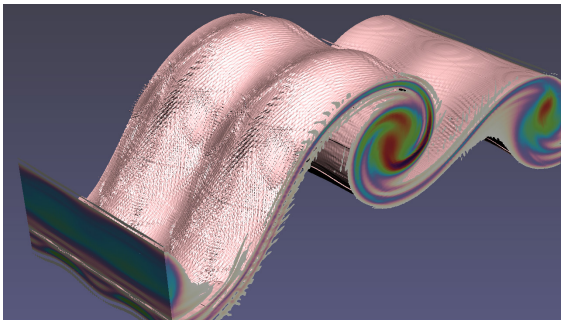
ASCR

ADVANCED SCIENTIFIC COMPUTING RESEARCH

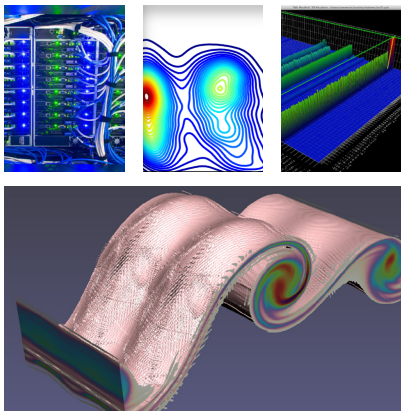
EXASCALE REQUIREMENTS REVIEW

An Office of Science review sponsored by
Advanced Scientific Computing Research

SEPTEMBER 27–29, 2016

ROCKVILLE, MARYLAND

U.S. DEPARTMENT OF ENERGY



On the cover:

Top left: Researchers and application teams need to explore hardware and software tradeoffs on early systems to determine which configuration best matches their respective codes (Image courtesy of Richard Coffey, Argonne National Laboratory).

Top middle: Part of the SciDAC partnership, a QUEST and WastePD EFRC project was aimed at enabling the design of multicomponent alloys with more parameters. The project team constructed a high-throughput framework to allow direct control over parameters Cu-Ni and Cu-Zr to illuminate model construction failures and tune optimization techniques. The project revealed that multi-component alloys have an Si parameter for each element (image) and that Sa, Sb only minimally impact properties in Cu-Ni (Image courtesy of Habib Najm, Cosmin Safta, Michael Eldred, and Gianluca Geraci, Sandia National Laboratories; Gerald Frankel, Wolfgang Windl, David Riegner, and N. Antonlin, Ohio State University).

Top right: TAU's ParaProf 3D browser reports the load imbalance in the profile and the time spent waiting in a condition variable on a subset of MPI ranks in a hybrid MPI and OpenMP program. The image shows the shape of the profile across multiple ranks and threads for the MADNESS application on an IBM BlueGene/Q system in TAU's ParaProf 3D browser, where we can observe the excessive time spent in a synchronization operation (Image courtesy of Sameer Shende, University of Oregon).

Bottom: Engineering analysis of combustion processes can be impractical at the exascale due to massive data storage and processing requirements. This issue can be solved using *in situ* data processing to calculate vorticity magnitude, expose data array slices, and evaluate the evolution of a turbulence mixing layer (Image courtesy of Brad Whitlock and Earl Duque, Intelligent Light).

Disclaimer

ADVANCED SCIENTIFIC COMPUTING RESEARCH

U.S. DEPARTMENT OF ENERGY

An Office of Science review sponsored by Advanced Scientific Computing Research

Meeting Co-Chairs
Jeffrey Vetter, Oak Ridge National Laboratory
Ann Almgren, Lawrence Berkeley National Laboratory
Phil DeMar, Fermi National Accelerator Laboratory

Meeting Organizers
Richard Coffey, Argonne Leadership Computing Facility
Katherine Riley, Argonne Leadership Computing Facility

Report Section Leads
High-Performance Computing (HPC) Architecture
Emerging Systems
Franck Cappello, Argonne National Laboratory

Early Delivery Systems
Samuel Williams, Lawrence Berkeley National Laboratory
Sameer Shende, University of Oregon

Production Systems
Galen Shipman, Los Alamos National Laboratory

Data, Visualization, Analytics, and Storage
Emerging Systems
Ken Moreland, Sandia National Laboratories
Wes Bethel, Lawrence Berkeley National Laboratory

Early Delivery Systems
Ken Moreland, Sandia National Laboratories
Wes Bethel, Lawrence Berkeley National Laboratory

Production Systems
Wes Bethel, Lawrence Berkeley National Laboratory
Ken Moreland, Sandia National Laboratories

Distributed Computing and Networking (HPDC)
Early Delivery Systems
Rob Roser, Fermi National Accelerator Laboratory

Production Systems
Salman Habib, Argonne National Laboratory

Software Development
Early Delivery Systems
David E. Bernholdt, Oak Ridge National Laboratory

Production Systems
Lois Curfman McInnes, Argonne National Laboratory
Judy Hill, Oak Ridge Leadership Computing Facility

Systems Software
Emerging Systems
Kathryn Mohror, Lawrence Livermore National Laboratory
Sameer Shende, University of Oregon
George Bosilca, University of Tennessee

Early Delivery Systems
Jeffrey K. Hollingsworth, University of Maryland

Software Deployment and Support on Production Systems
Todd Gamblin, Lawrence Livermore National Laboratory

Operational Data and Policies on Production Systems
Shirley Moore, Oak Ridge National Laboratory

Sponsors and Representatives
Carolyn Lauzon, DOE Office of Advanced Scientific Computing Research
Lucy Nowell, DOE Office of Advanced Scientific Computing Research
Ceren Susut, DOE Office of Advanced Scientific Computing Research

Additional Contributors
Mary Fitzpatrick, Carolyn Steele, Michele Nelson, and Vicki Skonicki, Communications, Education, and Public Affairs; Beth Cerny and Richard Coffey, Argonne Leadership Computing Facility

TABLE OF CONTENTS

FIGURES

TABLES

EXECUTIVE SUMMARY

Abstract

The widespread use of computing in the American economy would not be possible without a thoughtful, exploratory research and development (R&D) community pushing the performance edge of operating systems, computer languages, and software libraries. These are the tools and building blocks — the hammers, chisels, bricks, and mortar — of the smartphone, the cloud, and the computing services on which we rely. Engineers and scientists need ever-more specialized computing tools to discover new material properties for manufacturing, make energy generation safer and more efficient, and provide insight into the fundamentals of the universe, for example. The research division of the U.S. Department of Energy's (DOE's) Office of Advanced Scientific Computing and Research (ASCR Research) ensures that these tools and building blocks are being developed and honed to meet the extreme needs of modern science.

ES.1 Summary and Key Findings

The ASCR Research Requirements Review brought together scientific and computational experts with interests in ASCR-supported computer science, applied mathematics, and next-generation network research programs. The objectives of the review were to (1) assess the ability of the ASCR high-performance computing (HPC) facilities to support the needs of these researchers in an exascale computing environment; and (2) identify areas where that support could be enhanced, potentially opening new opportunity areas for computational research. Given the broad spectrum of ASCR Research activities, the organizing committee surveyed potential attendees to identify discussion topics. On the basis of the survey results and organizing committee discussions, the breakout sessions were structured to cover a diverse set of HPC technology areas: HPC architectures, data management/visualization/analysis/storage, high-performance distributed computing, software development, systems software, deployment, and operations.

The following broadly grouped areas directly affect the mission need of the DOE ASCR Research program to pursue exploratory R&D in programming systems, operating systems, architectures, data management systems, and performance-edge hardware and software. Section ES.3 (ASCR Computing Needs) describes the identified areas of need in more detail.

- In contrast to the domain science areas, the ASCR Research focus on computer science, applied mathematics, and next-generation networking means that current and future HPC computing systems themselves are frequently the subject of the research, and thus, researchers need *frequent, comprehensive access* to these systems.
- Close collaboration and effective communication between HPC facilities and ASCR researchers are critical to facilitating researcher access to HPC resources, as well as enabling researchers to *study and test HPC systems in detail.*
- Because HPC facilities are often the venue for ASCR researchers to develop, test, and deploy their tools, these facilities need to provide a computing ecosystem that supports a *development environment for ASCR research efforts* and timely deployment of ASCR research products, as well as an environment for HPC users.

In aggregate, participants in the breakout sessions identified 77 needs, each framed by specific science driver(s). Of these, 52 were categorized as "high-priority." Because many of the needs identified were similar, the organizing committee combined logically related needs into higher-level areas of need. The intention was to provide a holistic framework to address areas of need, rather than seeking point-by-point solutions for the large number of individual needs.

The HPC environment is a computational ecosystem with diverse, complex, and tightly inter-related components — both for production use of the facilities and for the ASCR-funded computational R&D activities that rely on those facilities. Enhancing that ecosystem benefits all aspects of the ASCR Research program: computer science, applied mathematics, and next-generation networking programs. Thus, our findings are best viewed in the context of enhancing the system as a whole, rather than just its individual components.

ES.2 ASCR Mission and Community

Mission

The mission of the ASCR program is to advance applied mathematics and computer science; deliver the most advanced computational scientific applications in partnership with disciplinary science; advance computing and networking capabilities; and develop future generations of computing hardware and software tools for science, in partnership with the research community, including U.S. industry. The strategy to accomplish this mission has two thrusts: (1) developing and maintaining world-class computing and network facilities for science; and (2) advancing research in applied mathematics, computer science, and advanced networking.

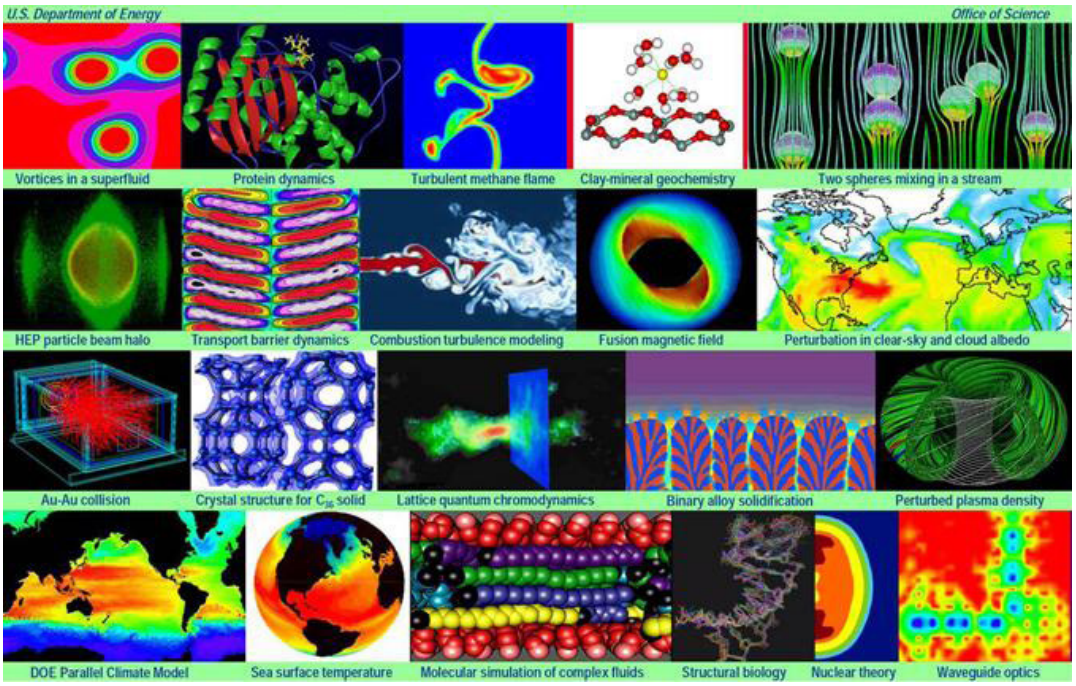
The ASCR Research Division underpins DOE's world leadership in scientific computation by supporting research in applied mathematics, computer science, high-performance networks, and computational partnerships (SciDAC).

The ***Computer Science*** program pursues innovative advances in a broad range of topics, including programming systems, system software, architectures, performance and productivity tools, and many others. In particular, the program focuses on effective use of very large-scale computers and networks, many of which contain thousands of multi-core processors with complex interconnections and data movement.

The ***Applied Mathematics*** program supports mathematical and computational research that facilitates the use of the latest HPC systems to advance our understanding of science and technology. More specifically, this program develops mathematical descriptions, models, methods, and algorithms to describe and understand complex systems, often involving processes that span a wide range of time and/or length scales.

The ***Next-Generation Network for Science*** program in ASCR conducts R&D activities to support distributed high-end science. It focuses on end-to-end operation of high-performance, high-capacity, and middleware network technologies needed to provide secure access to distributed science facilities, HPC resources, and large-scale scientific collaborations.

Scientific Discovery through Advanced Computing (SciDAC) brings together computational scientists, applied mathematicians, and computer scientists from across application domains and from universities and national laboratories to ensure that scientists are using state-of-the-art technologies to solve their increasingly complex computational and data science challenges.



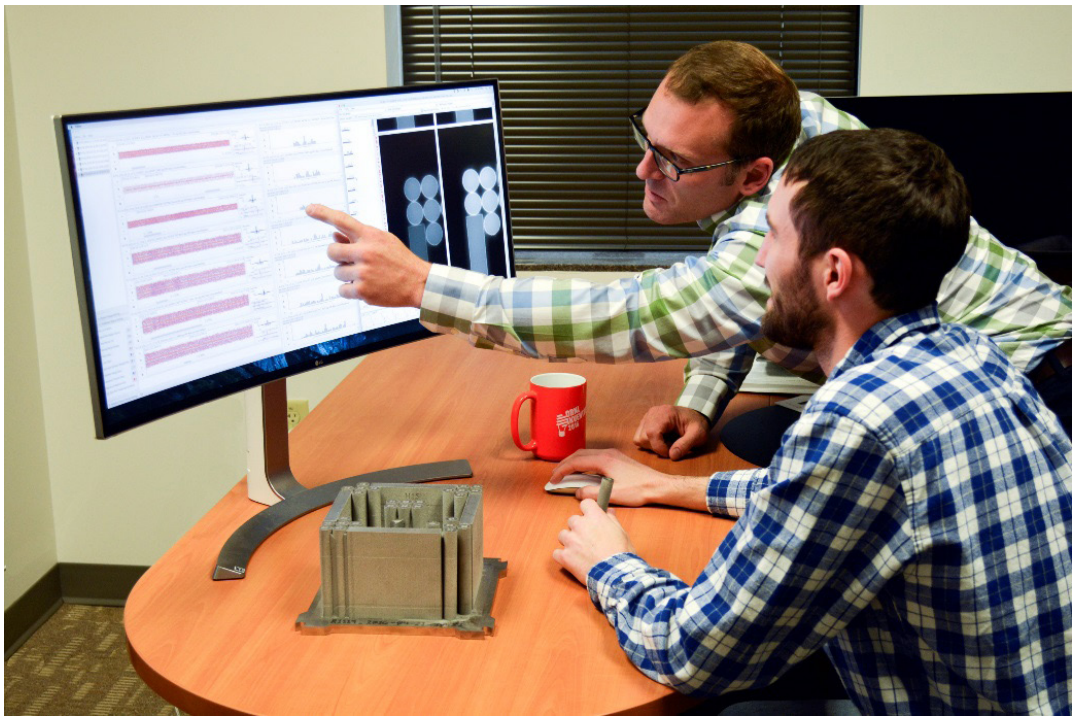
Community

ASCR researchers are often software developers, as well as software users. They require low-level access to the hardware to study the platform itself, and they may need access to computing systems in a variety of modes that domain scientists — who employ computing resources for large-scale production runs and require access to many nodes for many hours — may not need. ASCR research explicitly supports the development of software that introduces cutting-edge advances in applied mathematics and computer science, which in turn supports domain science applications. Software tools produced by ASCR computer science and applied math researchers enable physicists, biologists, and other domain scientists to exploit the full power of machines at the ASCR facilities and thereby increase their scientific output.

The information captured by our survey and in our workshop revealed that the needs of the ASCR research community differ drastically from those of other DOE Office of Science programs because of ASCR's focus on experimental computer science, applied mathematics, and networking research.

ASCR researchers' output ranges from a fundamental understanding of hardware systems, software and programming systems, and numerical algorithms to the design and development of fully functioning production software that enables cutting-edge domain science. In addition, because ASCR researchers work so "close to the metal," there are significant opportunities to realize mutual benefits for the ASCR research and computing facility missions.

For the ASCR community, facilities need to provide support for a true developer ecosystem in addition to supporting HPC *users*. Although facilities have application readiness programs to ensure that applications can make effective use of new hardware when it goes into production, there is currently no similar program that targets programming systems, libraries, or tools.



ES.3 ASCR Computing Needs

Because the *mission and goals* of the ASCR Research community differ significantly from those of other DOE Office of Science researchers, the computing *needs* of ASCR researchers in computer science, networking, and applied mathematics also differ substantially from those in other DOE domain science areas. Computational biologists, chemists, and other domain scientists rely on HPC to perform high-fidelity simulations, analyze observational and experimental data, and connect simulation and data. The computing facilities provide the means by which they perform their research.

For computer scientists, however, the computing facilities are not just a means to an end — they are, in fact, the subject of the research. Similarly, while other Office of Science researchers rely on high-performing networks to carry their data, networking researchers study the detailed qualities and performance of the network itself. Finally, while scientists in many areas of study rely on efficient, scalable algorithms embedded in their simulation codes or available at the facilities, applied mathematicians are at the cutting edge of defining new algorithms to run on new machines. Thus, the computing ecosystem is not just a tool to be used in their research, but often the subject of their research, which presents an entirely different set of computing needs.

The needs identified by ASCR researchers varied across the spectrum from emerging architectures to early delivery systems to production systems, but a number of common themes emerged.

For the ASCR Research Exascale Requirements Review, the available computing environment was decomposed into three tiers to clarify the differences in platform readiness and provide a common language to address the different types of systems and their associated needs. We employed the Technology Readiness Level (TRL) scale developed and used by several U.S. Government agencies: the U.S. Department of Defense (DOD), DOE, and National Aeronautics and Space Administration (DOE 2009). These agencies have applied TRLs to a broad range of technologies, including aircraft, electronics, and computing. Section 2 provides further detail about TRLs. For our review, the TRLs were calibrated to several important computing metrics and elements, including components (e.g., central processing unit [CPU], graphics processing unit [GPU], field-programmable gate array [FPGA]), level of integration, scale, software

readiness, and others. On the basis of the attendee survey and committee discussions, ASCR Research Exascale Review attendees were asked to categorize their findings and recommendations using the following three categories:

- ***Emerging Systems*** (TRLs 1–6) — innovative computing technologies that are currently not deployed to the HPC community at scale (e.g., FPGAs, quantum, neuromorphic, and experimental devices like carbon nanotube [CNT]-based integrated circuits).
- ***Early Delivery Systems*** (TRLs 7–8) — computing technologies scheduled to be deployed in a production system in the near future (e.g., Cori Phase 2, including early science period).
- ***Production Systems*** (TRL 9) — computing technologies deployed and operated at scale in the HPC community as a production resource (e.g., Titan, Edison, and Mira).

To organize the breakout sessions at the ASCR Research review, we combined (1) the three tiers of the computing environment with (2) the topics identified from the survey responses (HPC architectures, data management/visualization/analysis/storage, high-performance distributed computing, software development, systems software, deployment, and operations). Thus, each session focused on a particular topic or set of topics in the context of one of the three computing environment tiers (emerging, early delivery, and production).

The review found that the following broadly grouped areas relevant to ASCR facilities would directly affect the ASCR Research mission.

- The model of facilities providing compute cycles and researchers using those cycles to perform their research breaks down for ASCR Research. Computer scientists need access to emerging architectures and early delivery systems not just to use the systems, but to study them.
- Researchers benefit from availability of and access to operational data about the systems and the computing infrastructure with which they work. To allow computer scientists and networking researchers to study the systems themselves, they need improved system instrumentation and monitoring, access to system operational and performance data, network infrastructure counter data, usage statistics, running jobs data, data movement logs, and tools that provide performance insights. To enable applied mathematicians to develop new algorithms and implementations, they need to understand in detail how the current algorithms perform on current machines. In some cases (particularly emerging architectures), they also need the ability to change the system in order to study it (e.g., system reconfiguration agility and facilitation of hardware and software configuration changes).
- ASCR researchers need access to systems to perform testing and development; they often need a true development environment in addition to a user facility. This need includes accessibility and availability of development systems and small test bed facilities, end-to-end test and development capabilities up through the application layer, system availability for at-scale testing, routine testing of system software, and vendor participation in testbed facilities.
- Computer scientists, mathematicians, and distributed computing researchers must contend with a wide spectrum of HPC facility and resource access issues to conduct aggressive, relevant research in their areas of expertise. Simplification and uniformity of access to HPC resources across the different DOE-funded computing centers (including standardized authentication technologies) would enhance the R&D environment for those researchers, enabling them to be more efficient and productive.
- Researchers across the ASCR spectrum spoke about the need for improved information sharing about available systems (particularly emerging and early delivery systems) and clear policies regarding who can get access to these systems and how. Such communication can take the form of online documentation, outreach, and staff support. Users expressed frustration about the

difficulty in finding relevant information; in some cases, the information was there but hard to find, and in other cases (particularly for emerging and early systems), the information was not available. ASCR researchers would like consolidated information about access and available resources, staff training and training materials for developers, access to system hardware and software information, and improved communication and coordination among developers and system users.

- ASCR researchers, including applied mathematicians, computer scientists, and networking researchers, require a broad spectrum of deployment capabilities to conduct their research effectively. At HPC facilities, such deployment capabilities should include flexible software deployment processes, deployment of next-generation network technologies and data movement tools, software portability to run common application code easily across HPC facilities, easier on-ramp capabilities for new system software, and improved strategies for transitioning successful research artifacts into deployed software.
- The ability to modify a computing environment, either through software/hardware reconfiguration or upgrade, is essential to the research activities necessary to advance computing technology. ASCR researchers will be more successful — both in terms of their R&D efforts and in advancing their technologies to production — with greater software/hardware deployment and reconfiguration agility at HPC facilities.
- ASCR researchers need better data repository capabilities to support the wide spectrum of research activities they conduct at HPC facilities. Their storage needs range from readily accessible results from previous computations to short-term caching capacity for research involving data-intensive workflows.
- To address the increasing importance of data in all its forms, ASCR researchers need facilities for data collection and preservation, modern HPC-enabled data repositories, storage capacity at HPC facilities for data-intensive workflows, and long-term archiving capabilities.

The areas of need listed above are not presented in order of priority — every one of them includes needs identified as "high-priority" by at least four different breakout groups. For that reason, we have chosen not to attempt to prioritize them. We encourage readers of this report to view them as an interrelated set of needs that combine to form the key findings of the report.

ES.4 Path Forward

Because HPC systems are the object of research for ASCR researchers, the compute ecosystem needs identified in the ASCR Research Exascale Review are very different from those identified in other reviews. Primarily, ASCR researchers identified the following as crucial cross-cutting needs: increased access to all phases of hardware, increased agility in hardware and software configuration and deployment, and increased access to test and development hardware and system data. These needs require substantial new resources, new collaboration, and new workforce effort to explore how computer science and applied mathematics research can be better enabled in production and early delivery systems.

1 INTRODUCTION

1.1 Goal of the DOE Exascale Requirements Reviews

During fiscal years (FYs) 2015 and 2016, the Exascale Requirements Reviews brought key computational domain scientists, U.S. Department of Energy (DOE) planners and administrators, and experts in computer science and applied mathematics together. Meetings were held for each of the DOE's six Office of Science (SC) program offices, as follows:

- **The High-Energy Physics (HEP) review was held in June 2015.**
- **The Basic Energy Sciences (BES) review was held in November 2015.**
- **The Fusion Energy Sciences (FES) review was held in January 2016.**
- **The Biological and Environmental Science (BER) review was held in March 2016.**
- **The Nuclear Physics (NP) review was held in June 2016.**
- **The Advanced Scientific Computing Research (ASCR) was held in September 2016.**

The overarching goal was to determine the requirements for an exascale ecosystem that includes computation, data analysis, software, workflows, high-performance computing (HPC) services, and other programmatic or technological elements that may be needed to support forefront scientific research.

Each Exascale Requirements Review resulted in a report prepared by DOE for wide distribution to subject matter experts and stakeholders at DOE's ASCR facilities, including the Argonne and Oak Ridge Leadership Computing Facility centers (ALCF and OLCF) and the National Energy Research Scientific Computing Center (NERSC).

1.1.1 Previous DOE Requirements-Gathering Efforts: "Lead with the Science"

DOE has experienced definite value in implementing its previous requirements-gathering efforts. Such review meetings have served to:

- Establish requirements, capabilities, and services;
- Enable scientists, programs offices, and the facilities to have the same conversation;
- Provide a solid, fact-based foundation for service and capability investments; and
- Address DOE mission goals by ensuring that DOE science is supported effectively.

1.1.2 National Strategic Computing Initiative

Dovetailing with the current Exascale Computing Program (ECP) is establishment of the National Strategic Computing Initiative (NSCI) by Executive Order on July 30, 2015. The initiative has the following four guiding principles:

1. The United States must deploy and apply new HPC technologies broadly for economic competitiveness and scientific discovery.
2. The United States must foster public-private collaboration, relying on the respective strengths of government, industry, and academia to maximize the benefits of HPC.

3. The United States must adopt a whole-of-government approach that draws upon the strengths of and seeks cooperation among all executive departments and agencies with significant expertise or equities in HPC while also collaborating with industry and academia.
4. The United States must develop a comprehensive technical and scientific approach to transition HPC research on hardware, system software, development tools, and applications efficiently into development and, ultimately, operations.

Many of the NSCI's objectives echo plans already under way in DOE's current exascale computing initiatives. In fact, DOE is among the NSCI's three lead agencies (along with the U.S. Department of Defense and the National Science Foundation [NSF]), which recognizes these agencies' historical roles in pushing the frontiers of HPC and in helping to keep the United States at the forefront of this strategically important field.

1.2 Office of Advanced Scientific Computing Research Exascale Requirements Review

The DOE Office of Science (DOE SC) convened an Exascale Requirements Review for the ASCR Research program, which took place September 27–29, 2016, in Rockville, Maryland. The review brought together leading ASCR researchers and program managers, scientific and HPC experts from the ASCR facilities and scientific computing research areas, and DOE ASCR staff (see Appendix A for the list of participants). Participants addressed the following topics:

- Identify ASCR research that could benefit from exascale computing over the next decade;
- Establish the specifics of how and why new HPC capability will address issues at various ASCR frontiers; and
- Promote the exchange of ideas among application scientists, computer scientists, and applied mathematicians to maximize the potential for use of exascale computing.

ASCR Organizing Committee chairs guided the discussions in general sessions and topical breakouts. Committee members and review participants collaborated at the meeting to identify research directions and computing requirements. This report therefore reflects extensive and varied forms of input from many voices in the ASCR Research community regarding HPC requirements for ASCR's world-class initiatives.

The review afforded a rare opportunity for the nearly 100 participants to interact and learn about each other's areas of expertise, challenges they face, and the exciting opportunities made possible by the exascale computing environment.

Exascale Requirements Reports Will Meet Multiple Needs

DOE managers will use the Exascale Requirements Review reports to guide investments and budgeting, complete their strategic planning, and respond to inquiries, including specifically, their efforts to:

- Articulate the case for future computing upgrades to DOE and SC management, the Office of Management and Budget, and Congress;
- Identify emerging hardware and software needs for SC, including for research; and
- Develop a strategic roadmap for the facilities based on scientific needs.

ASCR program managers may also use the reports to inform their work. Although balancing such varied end uses can be a challenge, the reports are intended to serve as an informational tool that can be used by many stakeholders.

1.3 Report Organization

In the balance of this Exascale Requirements Review, Section 2 provides a description of the ASCR vision, characterization of the ASCR community, and a description of the preparations for and structure of the ASCR review. Section 3 addresses seven areas of scientific challenge for ASCR (organized by technology readiness level) and the needs identified in the breakout session for each area. Section 4 outlines a path forward for successful collaboration among DOE's ASCR facilities (i.e., the Leadership Computing Facility [LCF] centers and NERSC). References and acronyms/abbreviations used in the report are listed in Sections 5 and 6, respectively. Appendix A provides the list of meeting organizers and participants, Appendix B provides the meeting agenda, and Appendix C presents the results of the survey of the ASCR community conducted prior to the review.

This case study demonstrates the challenges and solutions ASCR researchers face. The science team needed to detect and quantify cracks and fiber breaks from micro-CT images to assess the resilience of new materials (Figure 1-1). The challenges: there are no automated methods for this type of analysis; existing software tools could not meet throughput requirements or scale to full resolution of the experiment (raw ~ 60 GB); and several copies of the data need to be maintained at different resolutions. The ASCR Research team

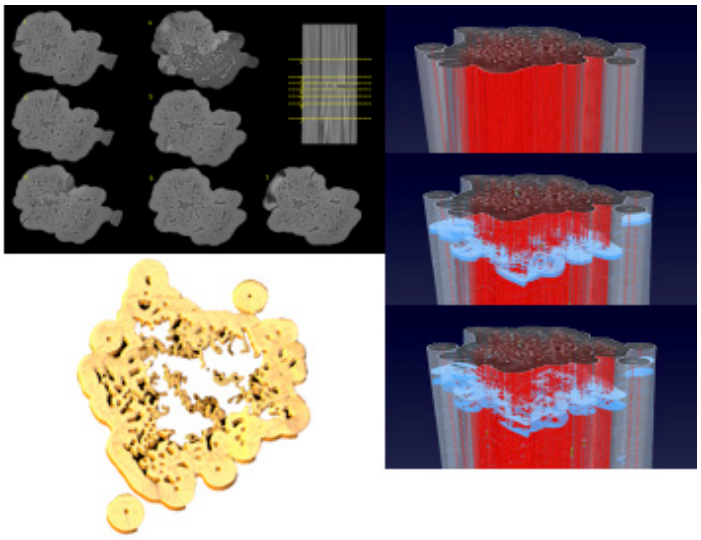


Figure 1-1. High-resolution data obtained at DOE imaging facilities will improve aviation manufacturing but depends on analytical schemes to recover material phases and track mechanical deformations of new ceramic matrix composites imaged during tensile tests. Images show processing steps of micro-tomography acquired at the Lawrence Berkeley National Laboratory (LBNL) Advanced Light Source (Image courtesy of Dani Ushizima et al., LBNL).

1. Developed platform-portable implementation of key data-parallel filtering and analysis algorithms, promoting applicability to other problems on a diversity of platforms;
2. Created a scheme to compare local patterns of voxel intensities to deliver total variation curves of SSIM and detect micro-crack volumes;
3. Laid out a processing construct to explore parallelism to perform required computations; and
4. Calculated tiled multi-resolution pyramids at four different scales and stored them in HDF5, chunked multi-dimensional arrays through Big-DataViewer.

The scientists can now process in a few minutes what would otherwise take hours or not be possible at all. The ASCR Research solution combines new mathematical methods with high-performance, platform-portable implementations to keep pace with increasing data size and complexity.

This page is intentionally left blank.

2 ASCR MISSION, CHARACTERIZATION OF RESEARCH COMMUNITY, AND STRUCTURE OF REVIEW

2.1 ASCR Mission

The mission of the ASCR program is to advance applied mathematics and computer science; deliver the most advanced computational scientific applications in partnership with disciplinary science; advance computing and networking capabilities; and develop future generations of computing hardware and software tools for science, in partnership with the research community, including U.S. industry. The strategy to accomplish this has two thrusts: (1) developing and maintaining world-class computing and network facilities for science; and (2) advancing research in applied mathematics, computer science, and advanced networking.

The ASCR Research Division underpins DOE's world leadership in scientific computation by supporting research in applied mathematics, computer science, high-performance networks, and computational partnerships such as Scientific Discovery through Advanced Computing (SciDAC).

The ***Applied Mathematics*** program supports mathematical and computational research that facilitates the use of the latest HPC systems in advancing our understanding of science and technology. More specifically, this program develops mathematical descriptions, models, methods, and algorithms to describe and understand complex systems, often involving processes that span a wide range of time and/or length scales.

The ***Computer Science*** program allows innovative advancement in a broad range of topics, including programming systems, system software, architectures, performance and productivity tools, and many others. In particular, the program focuses on effective use of very large-scale computers and networks, many of which contain thousands of multi-core processors with complex interconnections and data movement.

The ***Next Generation Network for Science*** program in ASCR conducts research and development (R&D) activities to support distributed high-end science. It focuses on end-to-end operation of high-performance, high-capacity, and middleware network technologies needed to provide secure access to distributed science facilities, HPC resources, and large-scale scientific collaborations.

SciDAC brings together computational scientists, applied mathematicians, and computer scientists from across application domains and from universities and national laboratories to ensure that scientists are using state-of-the-art technologies to solve their increasingly complex computational and data science challenges.

2.2 Characterization of Research Community

The needs of the ASCR research community differ from those of other DOE SC programs because of ASCR's focus on computer science, applied mathematics, and networking research. ASCR researchers are often software developers, as well as software users. They require low-level access to the hardware to study the platform itself, and they may need access to computing systems in a variety of modes that domain scientists — who employ computing resources for large-scale production runs and require access to many nodes for many hours — may not need. ASCR research explicitly supports the development of software that encapsulates cutting-edge advances in applied mathematics and computer science, which in turn supports domain science applications. Software tools produced by ASCR computer science and applied math researchers enable physicists,

biologists, and other domain scientists to exploit the full power of machines at the ASCR facilities and thereby increase their scientific output.

ASCR researchers' output ranges from a fundamental understanding of hardware systems, software and programming systems, and numerical algorithms to fully functioning production software that enables cutting-edge domain science applications. In addition, because ASCR researchers work so "close to the metal," there are significant opportunities to realize mutual benefits for the ASCR research and computing facility missions.

For the ASCR community, facilities need to provide support for a true developer ecosystem in addition to supporting HPC *users*. Although facilities have application readiness programs to ensure that applications can make effective use of new hardware when it goes into production, there is currently no similar program that targets the libraries and tools.

2.3 Requirements Review Structure

In order to define the goals of the ASCR Exascale Requirements Review, the organizing committee surveyed potential attendees (the results of this survey are listed in Appendix C.) Response was overwhelming, with more than 70 individual responses. Figures 2-1 through 2-3 provide sample charts showing the survey questions and attendee responses.

Q9: What breakout topics would you like to discuss at our September workshop? (Select up to four [4] topics)

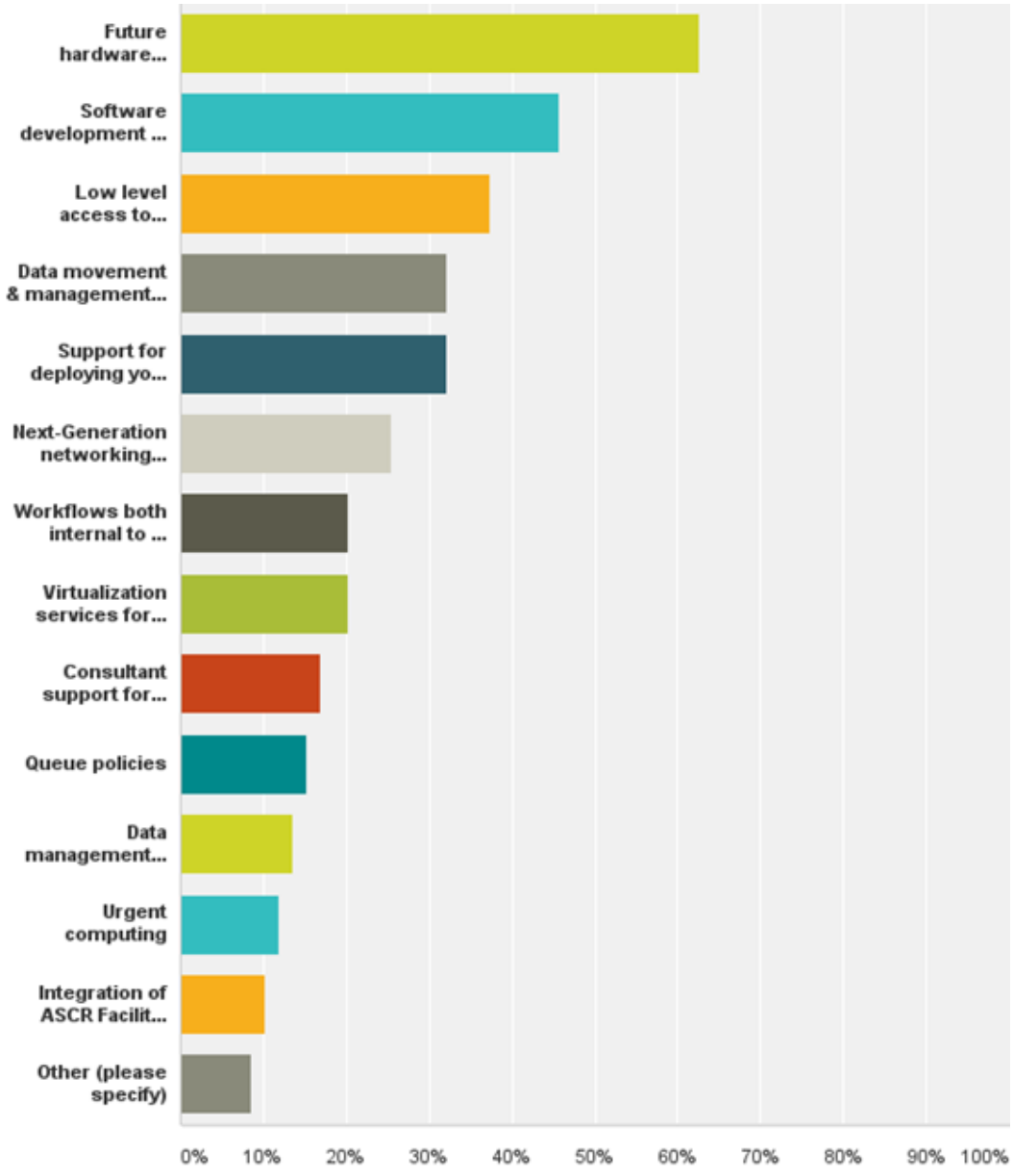


Figure 2-1. Survey question and responses regarding potential discussion topics for ASCR Exascale Requirements Review.

Q12: Are your ASCR-funded products deployed at ASCR computing facilities?

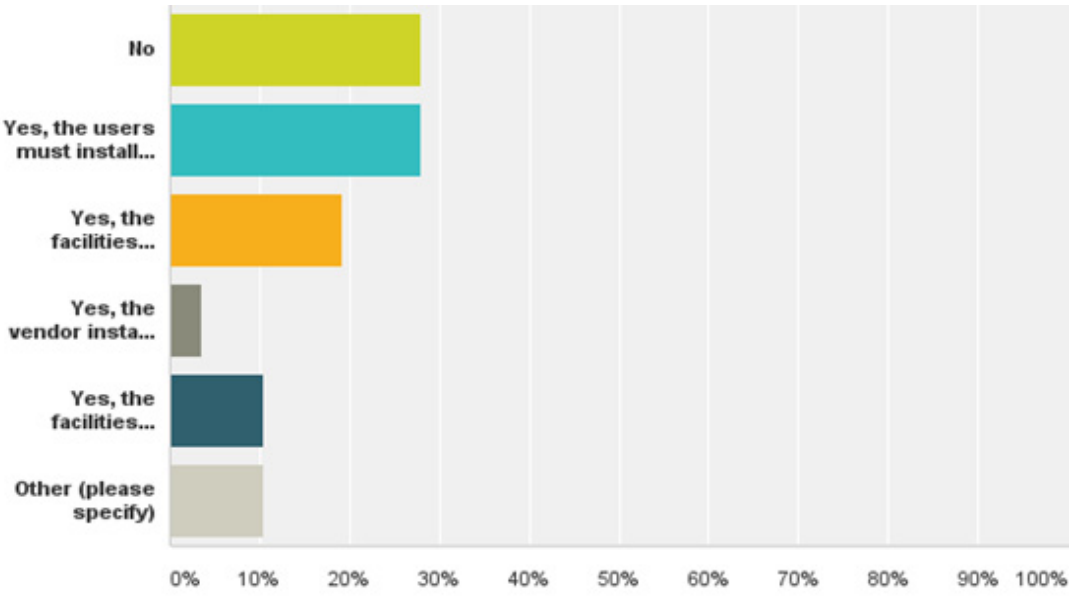


Figure 2-2. Survey question and responses regarding percentage of ASCR-funded products deployed at ASCR computing facilities.

Q13: How frequently do you communicate with the ASCR computing facilities about their research challenges in order to inform your ASCR-funded research objectives and directions?

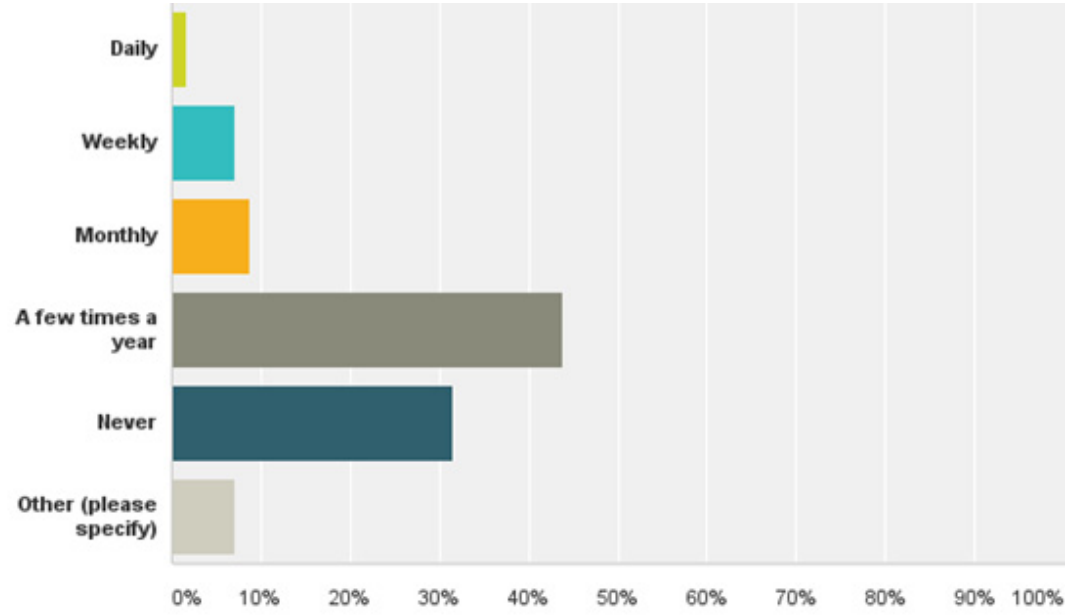


Figure 2-3. Survey question and responses regarding frequency of communication with ASCR computing facilities.

Survey respondents were also given the opportunity to provide open-ended comments. These comments were provided to the organizing committee and used in planning the breakout groups for the review. Samples of those open-ended comments (combined and distilled for brevity) are as follows:

1. System support for controlled computer science experiments is needed.
2. ASCR facilities are useless for system software R&D.
3. Viable development, deployment, and support strategies are needed for ASCR research tools on ASCR facility systems.
4. Computer science (CS) researchers need early access to systems before they hit the production stage (potentially to impact procurements, etc.).
5. Access to richer performance and reliability logs is needed, along with better availability of system operational data, including up-time and failure rates and types, to support development of math and software for diagnosis and resilience.
6. ASCR researchers desire interactive partitions and quick turnaround queues.
7. Operational policy adjustments should be made for experimental/observational data projects.
8. ASCR researchers need access to vendor tools and information to make tools more capable and effective.
9. Input is needed for ASCR requests for proposal (RFPs) regarding desirable hardware and software capabilities.
10. The line between facility and research is far too blurry; ASCR facilities seem to be increasingly competing with ASCR research.
11. A research pipeline (i.e., graduate students migrating into laboratory) should be developed.
12. The role of academic CS and math research and software in the ASCR facilities ecosystem should be defined.

Platform Readiness Level

On the basis of the survey responses, the organizing committee developed a structure with two dimensions: platform readiness level and technical topic. The platform readiness level (i.e., emerging, early delivery, production) (see Figure 2-4) addresses the highest-priority request of the survey respondents: that they have access to emerging and early delivery platforms in addition to production systems. Over 60% of the attendees listed this item as one of their four highest priorities. In order to clarify the differences in platform readiness, we relied on the Technology Readiness Level (TRL) scale developed by the U.S. Department of Defense (DOD)/ DOE/National Aeronautics and Space Administration (DOE 2009). These TRLs have been applied to a broad range of technologies, including aircraft, electronics, and computing. For our review, the TRLs were calibrated to the following important metrics/elements associated with HPC systems:

- Components (central processing unit [CPU], graphics processing unit [GPU], memory);
- Integration (CPU+GPU+memory+network operations center);
- Scale (1, 10, 100, 1000, 10000, 100000);
- Application preparedness (kernels, mini apps, mission);
- Performance (measured, simulated, analytically modeled);
- Time scales (computing technology changes very rapidly relative to many other technologies); and
- User productivity.

Although many composite technologies can span more than one level, most scientists can converge relatively quickly using this scale. The scale was provided to ASCR exascale review attendees to assist them in clarifying and categorizing their comments about access to future technologies (the most frequently cited request of the survey).

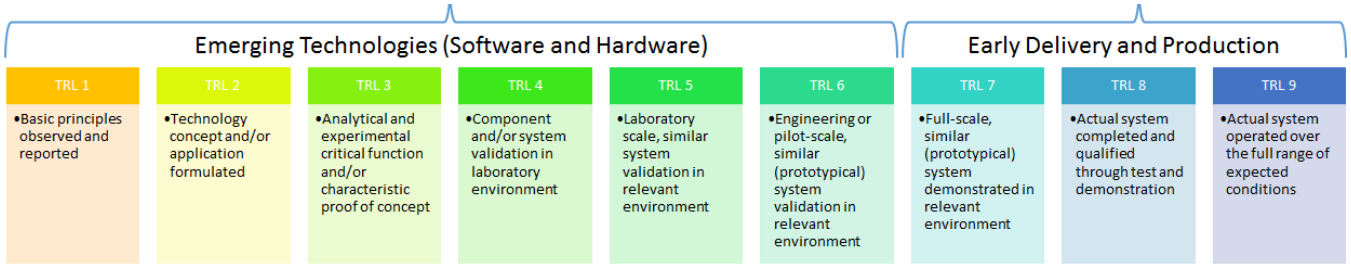


Figure 2-4. Technology Readiness Levels (DOE 2009).

For this workshop, the terms emerging system, early delivery system, and production system were defined as described in more detail below.

Emerging Systems (TRLs 1–6) — innovative computing technologies that are currently not deployed to the HPC community at scale (e.g., field-programmable gate arrays [FPGAs], TrueNorth, D-Wave)

Emerging architectures are defined as prototypes for evaluation and research that are experimental. Although these systems can be purchased, they are not ready for HPC for one of the following reasons:

- The technology is not mature enough (e.g., quantum computing),
- The technology is mature but some key software/hardware components are missing (e.g., high-level, parallel programming for FPGAs),[1] or
- The technology and software are available but their cost is too high.

Emerging architectures are important for application, CS, and applied math researchers, as well as facility personnel, in exploring the potential of new technologies; understanding the challenges that they pose; and developing algorithms, software, and tools to enable and facilitate the efficient use of technologies that are deemed relevant.

Early Delivery Systems (TRLs 7–8) — computing technologies scheduled to be deployed in a production system in the near future (e.g., Cori Phase 2, including early science period)

Broadly speaking, "early" systems and HPC architectures are production and procurement-track hardware that includes pre-acceptance systems (the system that will become a production system when accepted), phase 1 hardware (can be the same hardware by the same independent software vendor [ISV], but could be an earlier generation of hardware (Haswell for Broadwell, Sandy Bridge for Ivy Bridge, Fermi for Kepler, etc.), and white boxes (generic machines of similar hardware configurations). These systems are typically available for about a year, during which time final system configurations and performance of key applications, numerical libraries, and software can be evaluated and a best practices guide can be developed.

[1] There may be other reasons why FPGAs have not been widely adopted in HPC systems, such as complex programming, longer compile time, application-dependent design, and/or inflexibility.

Early delivery systems usually come in the form of a large-scale computer in its pre-production state. That is, the machine has been delivered but has not yet completed its acceptance testing. A limited group of users is given access to the machine to conduct acceptance testing, software preparation for production mode, and research on new hardware features. Early delivery systems may also include smaller test systems that have hardware similar to that of larger systems expected to be constructed in the near future.

Early delivery systems are used in a variety of ways. Software needs to be tested and often improved to ensure that it works predictably and efficiently before a machine goes into production. Early delivery systems often provide the first real access to new hardware and software features for many researchers. The research on this new hardware and software could range from a narrowly focused "mini-app" style project up through complex workflow-based applications that could benefit significantly from some new architectural feature. Finally, early delivery systems help inform decisions that affect how systems are put into production. Software often requires modifications to policies, configuration, and other features of production systems that are generally considered for independent science applications.

Production Systems (TRL 9) — computing technologies (e.g., Titan, Edison, Mira) deployed and operated at scale in the HPC community as a production resource

Production systems are those that are presently in production use and available to the SC science community. They differ from early delivery systems in that there is an expectation of high reliability, stability, and access that is marshaled through some form of resource allocation request process (for computational production systems, as opposed to production wide area network [WAN] networking production systems operated by ESnet).

This page is intentionally left blank.

3 RESEARCH DIRECTIONS AND COMPUTING NEEDS/REQUIREMENTS

This section presents the needs identified by each breakout group, organized first by technical topic (HPC architectures; data management, visualization, analytics, and storage; distributed computing and networking; software development; systems software; deployment and support; and operational data and policies) and then by platform readiness level (emerging, early delivery, and production [if applicable]).

3.1 HPC Architectures

Experimental computing research demands availability of and access to a wide range of HPC architectures and platforms across the spectrum of technical readiness — from emerging to production systems.

3.1.1 Emerging Systems

During the breakout session for HPC architectures for emerging systems (defined in Section 2), participants raised the following discussion topics.

- *Information/Communication/Access:* How do researchers and users learn that an emerging system is available in an ASCR facility? How do they get information about the system, its access, and operational mode? How do users access the emerging system that may have different infrastructure constraints and user needs than early or production systems?
- *Measuring/Monitoring/Tools for Insights:* What monitoring and measuring tools are available or needed on the emerging architectures?
- *Performing Experiments:* How do users of an emerging system actually perform experiments? Are the experiments performed through a classic batch scheduler or is a specific modality required (e.g., for a quantum computer)?
- *Connection to other Local Systems:* Does the emerging system need to be connected to another system to access specific data sets? For example, deep learning or data analytics systems may need to access very large data sets.
- *Local Sharing Policies:* Performing experiments and capturing accurate measurements in a reproducible way may require specific local (to the facility) sharing policies. For example, access to the emerging system could be time shared, as it is for scientific instruments. Users would access a dedicated portion (or the full system) of the emerging system, on which they would run their experiments at a high level of isolation.
- *Sharing Emerging Systems between Laboratories:* Emerging systems may be expensive to purchase and maintain. The research community may benefit from sharing access to rare systems among laboratories. For example, the D-Wave quantum computer is currently available only at Los Alamos National Laboratory (LANL). Even if a technology is available at multiple laboratories, there might be only one large-scale version of it. For example, some laboratories have a large diversity of single FPGA boards, while others have clusters equipped with the same FPGA board. Researchers in applications, CS, and applied mathematics are interested in accessing both.
- *Support/Expertise from the Facility (hosting, staff):* Emerging architectures are unique in that each different architecture requires the facility staff to (1) develop expertise on how to use it, (2) train users and researchers, and (3) maintain its potentially specific software stack. For example, the D-Wave system at LANL requires specific expertise. Also, FPGA systems require tool chains that are significantly different than classic compilation tools used in current early delivery and production systems.

Emerging architectures cover a very large diversity of technologies with different levels of maturity (or TRLs [see Section 2]) for scientific computing and data analytics. Three types of particularly relevant emerging technologies were identified on the TRL1–6 scale (see Section 2 and Table 3-1): ***exotic*** systems (quantum computer, neuromorphic, and approximate computing), ***near-term*** technologies (FPGAs, computing in memory, and specific hardware), and existing or future non-exotic technologies with potential relevance for scientific computing (***desired exploration***).

The distinction between exotic technologies and near-term technologies resides in the application programmers because significantly — sometimes drastically — changing algorithms and numerical methods are required. This is particularly obvious for quantum computer and neuromorphic circuits that are radical departures from the Von Neumann model. Approximate computing hardware, while implemented in classic circuit technologies as specific hardware or FPGA and following the Von Neumann model, falls into this category of technology because the understanding of its applicability to scientific computing is very limited. Near-term systems are evolutions of the current technologies, conforming to the Von Neumann model and considered as optimizations. Several researchers expressed the desire to explore new systems or existing, planned, or proposed infrastructures for which the suitability and performance in scientific computing context are unknown. These are categorized as "desired exploration" systems.

Each class covers several types of system: compute, memory, networking (internal and WAN), storage, operating system (OS), and infrastructure.

Table 3-1 presents the classes and types of emerging architectures that application, CS, and applied mathematics researchers identified during the breakout session. Table 3-2 lists the needs identified for these HPC architectures in emerging systems, the scientific drivers associated with those needs (by application [Apps], computer science and applied mathematics [CS], and facility [Fac] research), and the potential for impact of each identified need (low [L], medium [M], high [H]).

Figure 3-1. Experimental computing research demands availability of and access to a wide range of HPC architectures and platforms across the spectrum of technical readiness — from early to production systems. Images show an NVIDIA DGX-1 Deep Learning testbed; an Intel Lustre file system testbed; and the backside of a Cray Urika-GX Analytics Platform (Images courtesy of Richard Coffey, Argonne National Laboratory [Argonne]).

Table 3-1. Classes and Types of Early Architectures

Emerging Systems	Class	Type
Neuromorphic (e.g., IBM TrueNorth)	Exotic	Compute
Quantum (e.g., D-Wave)	Exotic	Compute
Specific processors (e.g., Google Tensor Processing Unit)	Desired exploration	Compute
Hardware support for switching jobs	Desired exploration	Operating System
Should we go beyond the roadmap Institute of Electrical and Electronics Engineers (IEEE) has defined concerning WAN?	Desired exploration	Networking (WAN)
Special-purpose link between major facilities (dedicated link: binding multiple ~100 G links)	Desired exploration	Networking (WAN)
Hardware for storage bandwidth provisioning (end-to-end quality of service)	Desired exploration	Storage
Super facility	Desired exploration	Infrastructure
Reconfigurable (FPGA, System On a Chip [SOC], Dark Silicon)	Near term	Compute
Processing in memory	Near term	Compute
ARM64	Near term	Compute
Persistent memory (nonvolatile memory [NVM] in the memory hierarchy)	Near term	Memory
Interconnect fabric (Power9 architecture, non-volatile memory express [NVMe] fabric to get around the peripheral component interconnect [PCI])	Near term	Network (internal)

Table 3-2. Needs: HPC Architectures of Emerging Systems

Number	Need	Scientific Drivers	Potential for Impact (L, M, H)
1	A more regimented access (as opposed to *ad hoc*) to emerging systems	**Apps:** Identify the potential of emerging systems **CS:** Identify challenges and new research directions **Fac:** Understand and potentially influence performance and reliability through co-design	H
2	Access to emulators	**Apps:** Use emulators as proxies for scalability study **CS:** Design space exploration **Fac:** Allows for early decisions	H
3	Access to exotic technologies	**Apps:** Enable new application areas and potentially solve new problems **CS:** Design and develop new software stack and operational modes **Fac:** Identify specific infrastructure and access requirements	H (for applications not currently solvable) May be M/L for other classes of applications
4	Access to near-term technologies (e.g., reconfigurable, computing in memory)	**Apps:** Optimize the performance of the traditional HPC portfolio **CS:** Research potentially at all layers of software stack **Fac:** Evaluate existing workload on new technologies	H
5	Facility personnel training and maintenance of the software stack	**Apps/CS:** Be able to use the system efficiently in its operational mode **Fac:** Assess the readiness of the facility to provide access to this system	H

3.1.1.1 Needs Identified from Breakout Session

More Regimented Access to Emerging Systems (1)

Description

In most instances, emerging systems are currently managed at facilities in an *ad hoc*, case-by-case way. A researcher may have acquired an emerging system and have unique, dedicated access to that system. In other cases, an emerging system is available to multiple researchers but information about it and its access modalities is not widely known. In still other situations, a laboratory has an emerging system that researchers and potential users from other laboratories are interested in accessing, but there is no common mechanism to access it.

Emerging systems are, in general, of interest for a subset of users, applications developers, and CS and applied mathematics researchers. They are also of interest for facility personnel. Information about the acquisition, availability, and access modality of emerging systems should be disseminated within and between the laboratories as soon as possible and, ideally, as soon as the emerging system is purchased or installed. When possible, access should be granted to researchers and system administrators of the laboratory acquiring the system, as well as other laboratories. Easier access to emerging systems would (1) significantly improve the community's understanding of the new systems; (2) accelerate the acquisition of expertise, assessment of the technology and its fitness for scientific computing and data analytics, and adoption of a promising technology; and (3) increase research productivity.

One approach to addressing this need is to provide a centralized place to share information and access to emerging technologies hardware between laboratories within the ASCR community.

Scientific Drivers

Impact for Applications

Impacts include the ability to identify and assess the potential of emerging systems. Emerging systems generally raise the following questions for potential users and application developers: Does a promising technology have a benefit today? What are the gaps for its effective use in a production context? What change in the technology or its interface/modality could be of benefit in the future (e.g., hardware, programming model)? What optimizations (efficiency) are needed before the technology becomes relevant?

Impact for CS and Applied Mathematics

Impacts include identifying challenges in the product as it is delivered and turning these challenges into opportunities and new research directions. When a technology becomes relevant for scientific computing and data analytics, it often demands the development of new algorithms and software to optimize its performance (operations per second, power consumption, reliability) and turn it into a production-ready system.

Impact for Facilities

Facility personnel have a major role in emerging technologies. Facility personnel typically install the system, make it available to a broad community of potential users, and disseminate information about it. They can

- Help users understand how their applications perform on these new technologies;
- Assess the new technology with respect to access modality and develop new access modalities and policies when needed;
- Assess the readiness of the facility to support an emerging system;
- Assess the reliability of the system;
- Identify, detect, and isolate any novel fault modes;
- Work with vendors to co-design the emerging system; and
- When the system is deemed relevant for scientific computing and data analytics, accelerate its time to production (if the system is procured).

Access to Emulators, Exotic Technologies, and Near-Term Technologies (2, 3, 4)

Description

Potential users, application developers, computer scientists and applied mathematicians, and facility personnel expressed a critical need to study and assess emerging technologies. In all cases, accessing a real system is preferred. However, accessing emulators is considered very important when the real system is not available or when a system large enough to perform experiments at scale does not exist.

Scientific Drivers for Applications

Access to exotic technologies is considered fundamental for discovering methods to accelerate scientific computing by orders of magnitude or for solving problems that are several orders of magnitude larger than the current ones. These exotic technologies have the potential to solve new problems that were not tractable using the Von Neumann model. They may also enable new application areas. Access to near-term technologies is essential to optimize the performance of traditional scientific computing portfolios. It enables application developers to understand which applications and data analytics will benefit from near-term technologies. For users and application developers, emulators are considered acceptable proxies for scalability studies, at least to understand the behavior of the emerging system at scale and obtain rough estimates of its performance.

Impact for CS and Applied Mathematics
Access to exotic technologies is critical for CS and applied mathematics researchers to design and develop operational modes (including workflow [i.e., the way we use the system]), algorithms, and software stacks for these technologies. For example, the adoption of quantum computers by a broad community of users will require the development of a new software stack, including new programming models, libraries, and system software. Early access to near-term technology is also very important for CS and applied mathematics researchers. This is an important vector to develop innovative research in programming models, compilers, optimization technologies, and OSs. For example, significant efforts are currently under way in the CS community and industry to develop parallel programming models (such as OpenMP, OpenACC, and OpenCL) for FPGA and to accelerate communication libraries with FPGA. The adoption of FPGA for scientific computing also requires porting or developing numerical and data analytics libraries. Without these libraries, application programmers will face serious performance optimization challenges because of the complexity and latency of the FPGA tool chains. For CS and applied mathematics researchers, emulators allow design space exploration when the system offers multiple configurations. In some cases, such exploration is the only way to move the research forward. For example, currently, a common way to explore approximate computing is through simulators and emulators, although reconfigurable hardware is a platform of choice for this research.

Impact for Facilities
From the facility perspective, exotic systems may have significant infrastructure needs. Only early access to these systems will enable facility personnel to anticipate and identify the particular infrastructure and access requirements, which other technologies do not need. Access to near-term systems is also important to allow facility personnel to evaluate their performance (operations per second, energy consumption, and reliability) on existing workloads. For facility personnel, emulators enable early evaluation of the fitness of the technology for the production context (at a later stage), especially when the facility is unlikely to obtain the real system for assessment.

From the facility perspective, exotic systems may have significant infrastructure needs. Only early access to these systems will enable facility personnel to anticipate and identify the particular infrastructure and access requirements, which other technologies do not need.

Facility Personnel Training/Maintenance of the Software Stack (5)

Facility personnel have a critical role to play regarding emerging systems for users, application developers, and CS and mathematics researchers in terms of training and maintenance of the software stack. This role is consistent with the role of training/maintenance of the software stack for early delivery and production systems.

Training of users, application developers, and CS and applied mathematics researchers and maintenance of the software stack are critical to (1) efficient use of the system in its operational mode and (2) ensuring that researchers have well-maintained software available for use. By installing and running the emerging systems, facility personnel are in the best position to perform these two important activities.

3.1.2 Early Delivery Systems

This traditional model described for early delivery systems in Section 2 (i.e., production or procurement track hardware) has been challenged on several fronts. First, when procured architectures are more revolutionary or novel (which will likely happen with increasing frequency in the exascale time frame), rather than evolutionary and incremental (the trend that dominated the last decade), performance and performance optimization can be nonintuitive to the uninitiated. Second, documentation and best practices for previous architectures are an effective basis for

evolutionary or incremental changes in the architecture. When faced with revolutionary or novel architectures, documentation (in all its forms) can be extremely scarce and, in many cases, restricted under various nondisclosure agreement (NDA) umbrellas. Third, most early-access programs have favored or encouraged access for either applications (often from outside of ASCR) or the occasional ASCR CS insider. Overall, the process tends to discourage or restrict access to such systems for the broader CS and applied mathematics communities. As a result, software, tools, libraries, and applications that are not part of the early testing phase will incur substantial delays when they are ported to the ultimate production architecture. Finally, although such machines may form an effective test bed for architectural research, their associated restrictions, impediments, and personnel limitations may preclude researchers from answering some of the most pertinent questions in computer architecture research in the context of HPC.

Table 3-3 lists five pressing ASCR research needs identified for HPC architectures in early delivery systems, the scientific drivers associated with those needs, and the potential for impact of each identified need (low, medium, high).

Table 3-3. Needs: HPC Architectures of Early Delivery Systems

Number	Need	Scientific Drivers	Potential for Impact (L, M, H)
1	**Early access for ASCR CS and applied mathematics researchers:** provide a common vehicle across centers/vendors for access to early systems, vendors, and dungeon session	Eliminates 6–18 months of delay on delivery of ASCR SW technologies, techniques, and best practices to facilities and applications teams	H
2	**Access to documentation:** architecture documentation, user guides, and best practices; common, well-maintained, inter-institutional wiki (historically, Google docs have fallen short)	Boosts productivity; avoids wasted/ duplicated work	H
3	**Facilitation of changes to hardware (HW) (basic input/ output system [BIOS]) and software (SW) configurations:** vehicle/method for requesting changes to non-uniform memory access (NUMA) interleaving, high-bandwidth memory (HBM) cache, enabled L3 caches, frequencies, emerging architectural features, SW development/software development kit (SDK), etc.	Determines best practices; identifies pathological cases; enables research into SW tech to mitigate expensive HW solutions	H
4	**Facilitation of HW setup exploration (nominally white boxes rather than pre-acceptance HW):** nodes need to be interconnected (mini-clusters vs. one-off nodes); differing CPU:GPU ratios, differing memory capacities/types, etc..	Sets realistic expectations on distributed memory performance; informs late binding and future procurement decisions	M
5	**Access to performance counter infrastructure:** includes both OS and SW setup, as well as documentation and training	Facilitates application/solver performance optimization; enables insights into novel/complex/ nonintuitive/emerging architectural features	M

3.1.2.1 Needs Identified from Breakout Session

Early Access for ASCR CS and Applied Mathematics Researchers (1)

Broadly speaking, the ASCR research community (outside of those focused on applications) has not been sufficiently encouraged to gain access to the early delivery systems being evaluated by the computing facilities. Researchers are further discouraged by the fact that each center provides a spectrum of access solutions, ranging from center-specific proposals (often perceived as application-centric) to *ad hoc* access request solutions often facilitated by "who you know." Access to such systems is accompanied by access to NDA information and to vendor software and architectural teams. Without such access, ASCR researchers are cut off from these resources and information. Combined, such restrictions could add 6 to 18 months of additional latency between machine acceptance and when ASCR research-funded libraries, tools, and technologies are effective on any new architecture. With the lifetime of production machines often being only 4 years, this delay reduces the timeframe for effective usage of the system by perhaps 25%.

ASCR facilities should provide a common, CS/applied mathematics-friendly access program to their early machines (pre-acceptance, phase 1, and white boxes) and include ASCR research teams in inter-institutional and vendor discussions. Such a process should be more widely advertised within the CS and applied mathematics communities, with clearly defined motivations that note the challenges and potential benefits that accompany each early access program. Because early hardware will inevitably be in short supply, early access to hardware could be provided in tiers or priorities in order to match the needs and availability of the research, applications, and facilities. Nevertheless, some access to white boxes or phase 1 hardware well before the pre-acceptance hardware is available can greatly accelerate the research process, while ensuring the value and relevance of its ultimate deliverables.

Nevertheless, some access to white boxes or phase 1 hardware well before the pre-acceptance hardware is available can greatly accelerate the research process, while ensuring the value and relevance of its ultimate deliverables.

Access to Documentation (2)

Although the DOE LCFs have traditionally provided excellent documentation for effective use of their production systems, documentation on early systems is inherently limited, late, and often incomplete. The process is further obscured by the fact that pre-general-availability (pre-GA) hardware is almost invariably under some form of NDA. Ultimately, a great deal of know-how is passed by word of mouth or by email; such information is not made available to the broader ASCR CS, data science, and applied mathematics communities.

ASCR facilities should provide an effective means of centralizing, disseminating, and maintaining documentation (under access control) to the early access application, CS, and applied mathematics communities regarding architecture, performance characteristics (latencies, bandwidths, and concurrency limitations), and execution guides. Historically, online, collaborative documents (Google docs) and wikis have been used to codify this information. Unfortunately, these approaches have failed because (1) information can quickly become stale, (2) the impediments to adding information to a wiki can be deemed too high, and (3) collaborative documents can degenerate into an ersatz ticketing system where work items accumulate. We believe that online question forms (e.g., stackoverflow.com) would provide a better model for documenting quickly evolving and potentially erroneous information in a manner that could be easily queried by users, with misinformation and stale documentation deprecated via a low-overhead (thumbs up/down) voting system. The benefit of this inter-institutional documentation is that it will enhance the productivity of ASCR CS and applied mathematics researchers focused on porting libraries, tools, and technologies to forthcoming production systems by eliminating spurious experiments or designs that are unlikely to be viable.

Facilitation of Changes to HW (BIOS) and SW Configurations (3)

Over the next 10 years (into the exascale timeframe), we expect vendors to continue to innovate their respective processor architectures. Although vendors are not omniscient, they are increasingly willing to expose the configurability of their architectures. For example, Intel's Haswell has three different NUMA modalities, while Intel's Knights Landing (KNL) has six different NUMA modalities. These modalities trade uniformity (desirable in highly threaded or unstructured environments) for performance/locality (desirable in flat message parsing interface [MPI] environments). Researchers and application teams should be enabled to explore these tradeoffs on early systems to determine which configuration best matches their respective codes. Similar issues exist for caching modes (e.g., HBM cache on Knights Landing or disabling an L3 on a Haswell to gauge the value of its bandwidth-filtering and latency-hiding capabilities), as well as frequency and power capping effects and potential heterogeneity or dark silicon effects — even extending to features enabled by the latest programming environments (e.g., differences in unified virtual memory [UVM] support in CUDA 5 through 8 on NVIDIA's Fermi, Kepler, and Pascal GPUs).

Researchers and application teams should be enabled to explore these tradeoffs on early systems to determine which configuration best matches their respective codes.

Computing centers should provide a mechanism by which users may change the hardware (BIOS) or software configurations on subsets of the available early-access nodes (e.g., changed within a partition). Doing so will not only enable ASCR application and applied mathematics teams to determine the best configuration for their respective libraries and applications, but it will also enable ASCR architecture researchers to understand the effects of these architectural features. Ultimately, these performance insights may enable ASCR computer science researchers to identify and prototype alternate software, compiler, runtime, and OS solutions for these potentially expensive hardware technologies.

Figure 3-2. Researchers and application teams need to explore hardware and software tradeoffs on early systems to determine which configuration best matches their respective codes (Image courtesy of Richard Coffey, Argonne).

Facilitation of Hardware Setup Exploration (4)

Although pre-acceptance hardware has likely been procured under contract for evaluation and is thus generally immutable, white boxes using production-track processors or the previous generation of processors have become an effective means of exploring some variation in hardware setup. Historically, such experiments have involved varying core counts, memory capacities or, more recently, the ratio of CPUs to GPUs on a node. Nevertheless, in the future, there may be more flexibility at both the socket and node level. Single-socket solutions may allow for a range of integrated core architectures, accelerators, and other internet protocol (IP) blocks that may be powered in either a static, dynamic, or dark silicon model. Moreover, the memory architecture at the node level may include a number of memory types whose performance, capacity, resilience, and durability (longevity) will vary. With potential late-binding options for these architectures, some design space exploration may be required to determine what solution best matches a center's application target requirements.

Regardless of architecture, it is essential in all such experiments that multiple nodes of a design be available and networked together in a performance-representative manner to ensure that researchers can create a reasonable proxy for distributed memory performance. Recent history has shown that interconnecting emerging and early delivery systems with second-class networks (or no networks at all) can lead to erroneous conclusions on scalability, memory requirements, the benefits of hybrid programming models, and alternate communication paradigms.

By procuring an architecturally broad (or easily configurable) suite of early hardware, facilities can enable ASCR research in a number of areas not generally possible on production hardware. For example, recent supercomputing systems worldwide have been deployed with varying ratios of CPUs to accelerators. Although the breadth of systems allows researchers to evaluate the implication of having a single process that drives multiple accelerators on programming models, it requires that they have access to multiple, geographically dispersed (possibly international) systems with some variation in setup. Centralized access, coupled with consistent configuration, eliminates some degree of variance from their experiments and can strengthen their conclusions. Similarly, any software-managed data locality, coherency, and consistency mechanisms on systems with multiple collaborating accelerators (GPUs) will face a distinct set of challenges altogether — far different from those observed on systems with a single CPU and GPU. Such challenges only multiply when researchers are confronted with the diverse capabilities and challenges associated with various emerging memory technologies, non-volatile or not.

Provide Access to Performance Counter Infrastructure (5)

Regimented, well-planned, and user-accessible performance and energy infrastructure is an integral component in system design and must be included as a priority from the earliest architectural planning, through OS kernel configuration, and throughout the lifetime of a production system. Performance infrastructure has become increasingly essential over the last decade as computer systems rapidly evolved to include numerous speculative and autonomous agents (runtimes or hardware) whose undocumented behavior and performance effects can be counterintuitive to even experienced programmers.

Recently, such performance infrastructure technologies have proved essential in understanding performance on many-core and GPU-accelerated early systems and white boxes — far beyond inscrutable wall-clock time measurements — particularly when the application performance departed from the throughput-oriented (Roofline) performance bound. Ultimately, early access is transient, and the other identified needs may be higher priority. Nevertheless, the non-production status of early delivery systems may facilitate installation, configuration, and user-mode access to performance instrumentation software.

Research into these activities (and commercialization of research tools) would be accelerated by perhaps 1 year through performance infrastructure enabled in early access and would increase the lifetime of production systems by perhaps 25%.

In the longer term (production systems), user-accessible performance infrastructure will facilitate a number of aspects of CS, data science, and applied mathematics research within ASCR. For example, all performance analysis, modeling, and architectural simulation must be validated with empirical performance data extracted from real applications running on real hardware. Moreover, establishment of best practices for early architectures is facilitative and quantitatively explained through the use of performance infrastructure. Finally, in the future, research into dynamic, execution-aware runtimes (threading, hierarchical memory, task scheduling, or power) must be capable of examining low-level performance, data, and energy counters in user mode with minimal overhead. Research into these activities (and commercialization of research tools) would be accelerated by perhaps 1 year through performance infrastructure enabled in early access and would increase the lifetime of production systems by perhaps 25%.

Fundamentally, effective performance counter infrastructure requires long-term collaboration among vendors, ISVs, and tool developers to ensure the functionality is present in hardware, accessible in user mode, understandable without decades of computer architecture training, and scalable to the largest DOE applications.

3.1.2.2 Relationships among Needs and Conclusions

Access to early systems goes hand-in-hand with access to training and documentation. Early systems, particularly when they represent a departure from the historical architectural trend, can be difficult to program, utilize, and analyze without sufficient documentation of the architecture, programming model, or execution environment. Although limiting documentation to the early access team can address any NDA restrictions, it risks accumulated knowledge not being disseminated to the broader user or research communities. Ultimately, any vendor-provided documentation must be provided early, and any center-generated documentation should be maintained to reflect the best practices of using the early systems. When a pre-acceptance system becomes production, any accumulated documentation created by the early access team should be polished and made available to the center's users.

Realistically, access to OS configuration may be required for user-level access to the performance infrastructure on early-access machines (i.e., counters may require privileged access). Enabling configuration changes in the highly regimented and restricted environment of early-access machines may be more palatable than enabling such changes in a production machine, and may thus be the only means of providing an empirical and instrumented approach to performance analysis. If so, it is imperative that such a system be maintained for testing (once again in a constrained-access environment), with its configuration preserved, over the lifetime of the production computing phase.

3.1.3 Production Systems

Table 3-4 lists five ASCR research needs identified for HPC architectures in production systems, the scientific drivers associated with those needs (by application, CS and applied mathematics, and facility research), and the potential for impact of each identified need (low, medium, high).

Table 3-4. Needs: HPC Architectures of Production Systems

Number	Need	Scientific Drivers	Potential for Impact (L, M, H)
1	Improved system monitoring at all levels (logs, machine-specific registers [MSRs], perf counters, and a top-down model)	**Apps:** Complicated architectures require more fine-grained performance monitoring. **CS:** Develop analysis capabilities to understand application performance. Provide actionable advice for future system architectures. Model checking/ validation. **Fac:** Perform debug of system performance. Identify hardware problems that impact performance.	H
2	System reconfiguration	**Apps:** How should applications run/ configure for production runs to achieve optimal performance of these runs? **CS:** Enable research into software mitigation for missing/suboptimal features, enable design space exploration for future system architecture based on existing systems, model qualification, and development **Data intensive:** Provide better support for complex workflows, new applications.	H
3	Interactive/batch scheduled resource balance	**Apps:** Higher productivity for application developers/users. Particular need for interactive access for resources. **CS:** Again, a productivity argument, particularly for compilation/ recompilation during design space exploration.	M
4	Dedicated system time for at-scale testing (large-scale testing, potentially with custom software/ configuration)	**CS:** Design evaluation of the software stack and hardware configurations.	M (high for ECP)
5	Improved access to small-scale test systems	**CS:** Design evaluation of the software stack and hardware configurations, qualification of software and hardware configurations.	M (high for ECP)

3.1.3.1 Needs Identified from Breakout Session

ASCR's CS and mathematics research communities place unique requirements on HPC facilities because of the nature of their research. Performance modeling and tuning researchers often require significantly more information from the system (counters) and about the system (a top-down model). System software developers must be able to customize the software stack at multiple levels, potentially all the way to the OS and lower-level firmware. Math researchers must be able to assess the performance and scalability of novel algorithms and to quickly prototype and access their changes. The following sections detail specific requirements of HPC production facilities identified by these research communities.

Improved System Monitoring at All Levels (1)

Today's HPC production systems are capable of providing a wealth of information that is critically important to CS and applied mathematics researchers. Because many in the CS and mathematics research community focus on the performance and scalability of their techniques, accurate measurement of the impact of their techniques on today's production systems is required. As management guru Peter Drucker is often quoted as saying, "you cannot improve what you cannot measure." At each level of the system, from CPU/GPU, memory, network, storage, and even at the aggregate system level, critical information for these activities is often present but not exposed. As an example, many processors offer MSRs that can provide important information such as floating point performance, cache efficiency, NUMA efficiency, and data movement through the memory hierarchy. GPUs can similarly provide detailed information on the efficiency of data movement across UVM, thread divergence, and efficiency of the memory hierarchy within the device (thread local, warp local, etc.). Networks can often provide information about placement of queues (on host or on device), spilling of these queues from device to host, head-of-line blocking counts, and other important information. Focusing on runtime monitoring schemes can include performance monitoring, as well as power, thermal, and reliability data.

At a system level, system logs routinely collected on production systems can provide important information on system events such as correctable and uncorrectable bit errors, CPU or GPU failures, network endpoint failures, router failures, routing topology information, job placement within the system, and application failures and their resultant error codes. This system-level information is rarely available outside of a select group, inhibiting the research in resiliency, performance, and scalability that can be performed in the CS and mathematics research communities. Finally, recent efforts aimed at providing more application-specific information (such as input/output [I/O] patterns and use, as captured by tools such as Darshan) can provide researchers with historic application behavior and utilization of system resources, as well as potential interactions across multiple applications using a shared resource. Again, this information could accelerate ASCR R&D efforts but is rarely made available outside of a select few.

Finally, often the most useful information that can be obtained from HPC systems is encumbered by NDAs, thereby limiting access to those who are parties to the NDA. Procedures should be developed to allow members of the ASCR R&D community to become parties to these NDAs, with clear guidelines for publishability of their research results based on this information.

There are some issues that must be addressed to facilitate broader access to system-level information. Some systems may require privileged access for certain MSRs, others can provide this information in a read-only mode if configured to do so. The interpretation of many performance counters requires a top-down model of the CPU/GPU/node architecture to properly interpret these measurements. One solution to this issue is for facilities to require top-down models and unprivileged access to performance and utilization counters as part of their statement of work with the vendor. Early access to this information would then facilitate readiness of performance modeling/analysis tools, instrumentation of math and CS algorithms, and experimentation plans for the ASCR R&D community. Similarly, the broad release (or even procedural release) of system logs and other application usage information may require anonymization of key information to ensure product privacy and security. Techniques have been developed to enable this anonymization, but they would need to be adopted by facilities. Finally, often the most useful information that can be obtained from HPC systems is encumbered by NDAs, thereby limiting access to those who are parties to the NDA. Procedures should be developed to allow members of the ASCR R&D community to become parties to these NDAs, with clear guidelines for publishability of their research results based on this information.

System Reconfiguration (2)

Many of today's HPC production systems have some capabilities for reconfiguration (at the system software and BIOS levels) that would be beneficial for many CS and mathematics researchers. Examples of these include the ability to conduct system architecture studies or math solver optimizations by changing (or emulating) different balances between node performance and network performance, varying the performance of different memories, or changing the NUMA/caching configurations of CPUs and GPUs. In addition, with Intel KNL for instance, hybrid memory configurations are available that can be partitioned into pure memory, pure cache, and LLC+memory modes. Also, core frequency, core voltage, memory controller frequency, LLC on/off, etc., can be part of the system reconfiguration.

Past studies in system architecture have been made possible by changing the available network bandwidth within a system (i.e., constraining the system to use only a few tapered virtual channels), thereby allowing CS researchers to experiment with different balances of CPU, memory, and network. These studies include the impact to different applications, providing a basis for projection to future system architectures that may have a significantly different balance (bytes/flop, bandwidth/flop) than today's systems.

Similarly, future systems are likely to have deeper memory hierarchies with very different bandwidth/capacity tradeoffs; the ability to reconfigure today's production systems to emulate these future systems is critical to developing meaningful scalability studies today that will influence design decisions in future (exascale) systems.

Finally, today's CPU and GPU technologies can often be reconfigured to support different caching and NUMA policies. The impact to applications can be significant, in some cases resulting in orders-of-magnitude performance differences. While some progress can be made on smaller-scale testbeds to explore this impact, some studies require larger-scale experiments to ascertain optimal configurations. The ability of CS and mathematics researchers to reconfigure different aspects of today's production HPC systems will allow them to conduct these and other studies.

HPC facilities have made progress in providing some level of reconfiguration, particularly in the areas of NUMA and caching configurations. Other system-level reconfiguration remains opaque and accessible to only a small number of researchers. A broader awareness of aspects of reconfigurability in today's systems is critically needed. Processes by which CS and mathematics researchers can request access to these capabilities are needed, as are methodologies by which different configurations can be validated on smaller-scale test systems prior to testing on production systems. These requirements will become more critical as software technologies projects begin work in earnest toward an exascale ecosystem.

Interactive/Batch Scheduled Resource Balance (3)

Today, researchers can be thwarted by scheduling policies that place significantly higher priority on large-scale simulations or by policies that otherwise reduce their priority in terms of allocation of interactive resources.

Many in the CS and mathematics research community require significant computational resources for interactive tasks. Frequent recompilation of source code, such as can occur when new system software, middleware, and algorithms are being developed and evaluated, requires timely access to interactive infrastructure. Other tasks — such as interactive analysis of performance data and subsequent reconfiguration and execution of an application, algorithm, or middleware — are commonplace in the performance-tuning communities. Productive development and prototyping of complex workflows that often require interactive resources further necessitate improvements in the allocation of interactive and batch resources on today's production systems. Today, researchers can be thwarted by scheduling policies that place significantly higher priority on large-scale simulations or by policies that otherwise reduce their priority in terms of allocation of interactive resources. Software development and performance tuning can be highly interactive processes, incorporating rapid prototyping; ideally, policies that enable rapid evaluation and the ability to rapidly acquire interactive resources can significantly improve productivity in these communities. Similarly, the ability to utilize full-featured OS/programming environments can often be limited to a relatively small number of "head nodes." This approach is particularly problematic for the CS and mathematics research communities because these head nodes are often over-utilized, resulting in long compilation times. As software stacks become increasingly complex, these compilation times can span many hours, and, in some cases, days on a heavily utilized system. This aspect of the research process must be improved to increase productivity.

HPC facilities have made some progress in this area; NERSC has made available "real-time" queues that could better support CS and mathematics researchers' workflow. An examination and re-thinking of the scheduling policies at the facilities may be necessary to better support projects that require rapid turnaround of experiments as part of the software development and performance-tuning process. However, because of figures of merit (FOMs) associated with the LCFs that reinforce a preference for large jobs, this requirement may prove problematic. A careful analysis of these FOMs and their unintended consequences may be in order. It may be beneficial to consider applying these FOMs to a subset of the total system, thereby making the remaining subset of the LCF systems available for alternative scheduling policies (e.g., interactive queues, software development queues). Addressing the resource constraints of head nodes (for compilation and other activities) may prove more difficult because most machine architectures have a fixed number of head nodes that is significantly smaller than available compute nodes. Given that these head nodes are a finite resource, a short-term solution may be allowing some projects to request dedicated head node resources through a reservation process. In the long term, it may be beneficial to include members of the CS and mathematics research community in system procurement discussions. This would be helpful because future architectures could blur the lines between head nodes and compute nodes, or the balance of head nodes and compute nodes could be changed based on a broader evaluation of use cases.

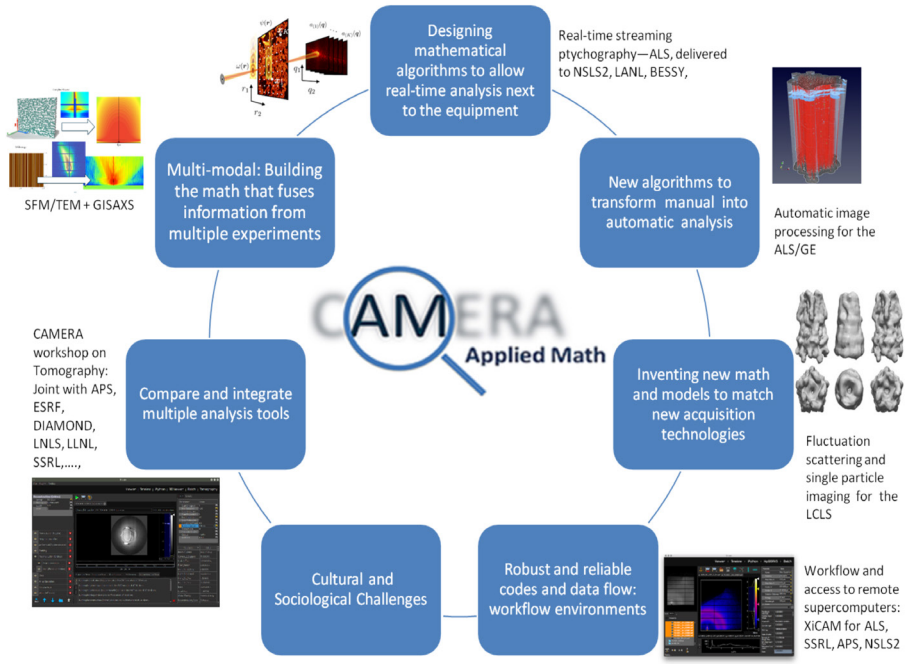


Figure 3-3. Experimental science and advanced scientific computing will continue to evolve in the exascale era. With the advent of new technology, scientific facilities are collecting data at increasing rates and higher resolution. Advances in computing architectures offer new opportunities to keep up with the data-processing demands. However, making sense of these data is becoming a major bottleneck. New mathematics and algorithms are needed to extract useful information from these experiments. The joint ASCR-BES Center for Advanced Mathematics for Energy Research Applications (CAMERA) is focused on building the mathematics, algorithms, and software to meet these challenges (Image courtesy of James Sethian, LBNL).

Dedicated System Time for At-scale Testing (4)

CS and mathematics researchers have some unique needs in testing at scale on production HPC systems. Many CS research projects focus on system software (OS/runtime/storage systems and I/O), tools (performance/debugging), and programming models. Their work requires custom software stacks, up to and including a wholesale replacement of the entire OS. Storage system researchers can require even more invasive changes to the system, including changing the underlying parallel file system. Acquiring access for these types of studies is difficult given the production requirements of ASCR facilities; supporting these studies can significantly disrupt the system. Nevertheless, these studies are critical to the advancement of the field. Many mechanisms developed in lower levels of the software stack to address the scalability requirements of future systems cannot be adequately evaluated without full-scale testing. The only systems of sufficient scale to test these technologies are those available at ASCR facilities.

Math researchers also require large-scale (including full-scale) testing during dedicated system times to accurately evaluate new solver algorithms in isolation from other users running on the system. HPC systems have shared components (network, storage) that can be sources of extreme variation in performance; accurate evaluation of new solver technologies requires dedicated access or other mechanisms of resource isolation to conduct experiments.

Supporting studies that require significant changes in the software stack can be challenging for ASCR facilities. Some systems, such as the IBM BlueGene, provided good support for these studies because individual racks could be provisioned in bare-metal mode, allowing completely custom system software configurations. This scenario is the exception, however; recent systems have fallen

backward in this regard, no longer supporting bare-metal provisioning. Facilities should consider elevating the priority of this requirement or consider alternative mechanisms for supporting custom software configurations. Including representatives of the CS and mathematics research communities in procurement discussions could help facilities meet this requirement on future systems.

Improved Access to Small-Scale Test Systems (5)

Improving communication about the availability of these resources, the process by which to gain access, and careful coordination with facilities staff could be beneficial to the CS and mathematics communities.

A common requirement expressed by the CS and mathematics research communities is improved access to small-scale test systems that mirror the large-scale HPC systems at NERSC and the LCFs. This requirement is particularly important to support studies that involve custom software and hardware configurations that may otherwise require dedicated system time on large-scale production systems. When conducting performance tuning experiments, these smaller-scale systems would be useful to first verify that the performance information and the interpretation of it (relative to a top-down model of performance) are valid. These steps would alleviate the need to conduct these experiments on production environments during the initial development phase. Similarly, the ability to reconfigure aspects of the system (e.g., changing caching modes, network virtual channels) would likely be easier on smaller-scale systems and could prove out configurations prior to large-scale testing.

All facilities maintain a smaller-scale "replica" of their large-scale systems and often provide access to these systems to support such studies. One challenge, however, is that the availability of these resources is not broadly known, nor is the process used to request access. Furthermore, access to these systems can impact facility staff who require access for testing future releases of vendor software stacks or debugging issues observed on the large-scale system in a controlled environment. Improving communication about the availability of these resources, the process by which to gain access, and careful coordination with facilities staff could be beneficial to the CS and mathematics communities.

3.2 Data Management, Visualization, Analytics, and Storage

3.2.1 Emerging Systems

This section is focused on future systems in data management, visualization, analytics, and storage. Table 3-5 lists five ASCR research needs identified for data management, visualization, analytics, and storage in emerging systems; the scientific drivers associated with those needs; and the potential for impact of each identified need (low, medium, high).

Table 3-5. Needs: Data Management, Visualization, Analytics, and Storage of Emerging Systems

Number	Need	Scientific Drivers	Potential for Impact (L, M, H)
1	Modern, HPC-enabled data repositories/centers (i.e., a superfacility)	New science enabled by the merger of simulation and data analysis	H
2	Pivot from "all flops all the time" to a more data-centric view of allocations and resources	Exascale simulations will produce tremendously large data sets whose analysis is key to maximizing the scientific return	M
3	R&D for data movement across the DOE complex in a variety of ways	Simulations, experiments, and analysis will take place across several facilities — the ability to seamlessly move data will increase the scientific turnaround time.	M

3.2.1.1 Needs Identified from Breakout Session

This breakout session, coupled with the superfacility special session, focused on the role of data (analytics, movement, storage, and visualization) in emerging architectures. The main finding of these sessions can be summarized simply as the overwhelming desire to create modern, HPC-enabled data centers as we approach the era of exascale computing. Today's petascale simulations produce scores of terabyte (TB)- to petabyte (PB)-sized data sets. Today's experimental facilities, including light sources, genomics centers, and telescopes, produce equally large data sets. With the strong scientific desire to confront observational and experimental data sets with those produced by simulations, and because each of these will grow by more than an order of magnitude over the coming decade, it is readily apparent that investments in both new types of centers, as well as R&D in a variety of data-centric efforts, will be required to handle this deluge of data.

Modern, HPC-Enabled Data Repositories/Centers — Superfacility (1)

Description and Drivers

A 'superfacility' is a network of connected computing and other scientific facilities (e.g., NERSC, Spallation Neutron Source [SNS]), software, and expertise that will enable new modes of discovery. Each of DOE's experimental and observational facilities is a unique national resource. Seamlessly connecting these resources to HPC facilities has already enabled new discoveries and will be transformative in the future, offering the potential to benefit many more fields in the coming decade. As we move to an era when exascale simulations are directly confronted by, and analyzed in conjunction with, multi-PB experimental and observational data sets, a superfacility will be needed to handle this analysis and thus is our highest priority.

The idea behind having a modern, HPC-enabled data repository or center has emerged already on a per-domain basis, but domain centers are based on (arguably) old technology and are not able to meet the expanding needs of the scientist in terms of capacity, speed, coupled analysis and other data-centric operations, fine-grained-access policies for the compute facilities and data, etc. Several experiments have created their own high-throughput computing (HTC) and HPC "pipelines" to meet their *current* needs; however, these pipelines are often constricting the potential science that can be done. Consider for instance the two very different science cases of a light-source experiment and a cosmological telescope survey to highlight this point. Both are constructed to answer a specific question, and a pipeline is developed to address this point. Confrontation of the data generated from these pipelines with results from a simulation is completed after the fact; new science questions can only be posed *ex post facto*, with no ability to steer the experiment while it is collecting data or enable real-time follow-up resources.

Thus, alleviating many of these constraints for the domain scientists could have a large impact on future science. And while here we provide examples from joint experimental and simulation-based analysis, the same holds true for the simulation-only analysis more typically encountered by ASCR scientists. Simulations are reaching the point, even now, where it is infeasible for a given HPC facility to store all the data generated. Thus, analysis needs to be performed *in situ* in order to stay ahead of the deluge of data, or the data must be moved from one facility to another for subsequent analysis. In addition, researchers are just now asking the question: what could be gained by "steering" a simulation while it is running?

What do users want in terms of a "superfacility"? They would like to bring together aspects of HPC, distributed computing, networking, and facilities personnel in a secure environment, where HPC and data are on equal footing. They would like to see multiple facilities (including compute and experimental) joined together and operated together by means of a high-performance network where geography is not a constraint, with ubiquitous access and a common software stack at all locations. In this superfacility, the network is explicitly part of the whole, not "just there" or invisible, and federated identity is a requirement to access all resources.

Because certain simulations or experimental runs and subsequent analysis will often generate "bursty" (i.e., non-smooth) resource allocation requirements, the facility must be able to handle elastic compute and storage requirements. In addition, coordination and co-scheduling will often have to be carried out at multiple-component facilities (e.g., a large simulation is run at OLCF and/or ALCF with data streamed to NERSC, where a scientist can perform analysis using one of the real-time queues). In addition, what is "good" or "right" to run at a given location on one day may be different from the best scenario tomorrow. The superfacility would allow the flexibility to make those decisions in a way that is useful for the scientist.

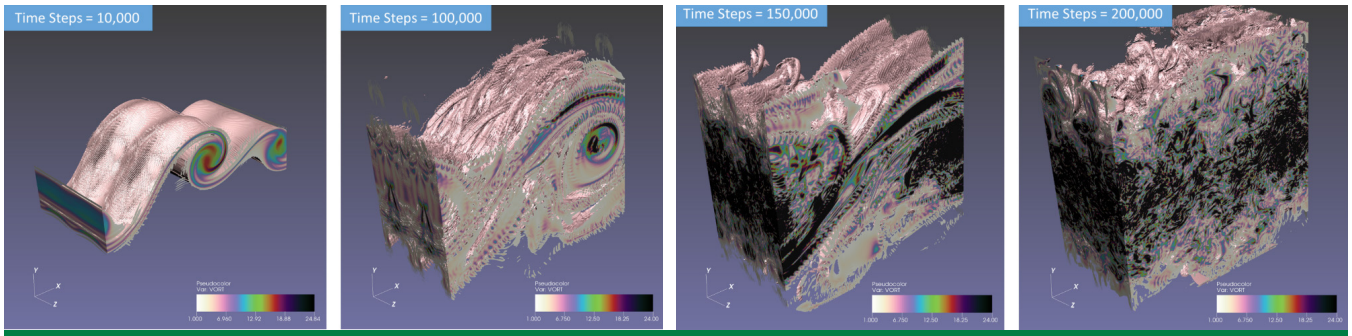


Figure 3-4. Engineering analysis of combustion processes can be impractical at the exascale due to massive data storage and processing requirements. This issue can be solved using *in situ* data processing: an instrumented AVF-LESLIE with a SENSEI adaptor that calculates vorticity magnitude, exposes data array slices, and evaluates the evolution of a turbulence mixing layer. The research team conducted benchmark studies on Titan that resulted in a four-fold increase in temporal resolution for visualization and analysis compared with volume-based, post-hoc processing. This strategy could have wide impact for manufacturing, engineering, science, and commercial software stakeholders for gas turbines and rocket engine design (Images courtesy of Brad Whitlock and Earl Duque, Intelligent Light).

Persistence is a key component of a superfacility. In order to support experiments, the compute, storage, and other resources necessary for its success must be available for the lifetime of the experiment (e.g., decade-long experiment run). This does not imply that an individual center needs to persist for the length of an experiment or be available 24/7. However, the superfacility itself needs to continue to run over this timeframe. Thus the resources that constitute a superfacility should, by necessity, have a long-term funding model.

Discussion

There are many interesting challenges (and accompanying R&D) that need further effort and discussion at the institutional, scientific community, and facility levels. Such challenges include the following.

- Providing high-performance data repositories for data products so that it is faster to access a dataset from a high-performance repository over the network than it is to access it from a local disk (DOE 2015).
- Data properties need to be expanded to include additional attributes, such as value, lifetime, etc., and a policy concerning data retention and "value" needs to be developed.

The time is ripe to create a call for white papers from scientists on how they would make use of a superfacility in the exascale era (or before). Subsequently, we could imagine partnering domain and ASCR scientists with NERSC, ALCF, OLCF, and ESnet to develop demonstrations of a superfacility and discover the bottlenecks. This would likely lead to potential R&D efforts in workflow management, edge services, math/analytics, HTC on HPC, containers, and compression, among other many others.

Pivot from "All Flops All the Time" to More Data-Centric View of Allocations and Resources (2)

Description and Drivers
As we move toward the era of exascale computing, it is obvious to all that the real cost of the system, financially as well as in terms of minimizing the wall-clock time of a calculation, is not in the flops but rather in pushing the data to the flops — be it from memory, non-volatile random-access memory (NVRAM), disk, etc. This reality, coupled with the fact that future, large experimental data sets are targeting joint analysis and simulation at the HPC facilities, will place a premium on the data resources available to scientists at a facility. Thus, it is critical that ASCR begins the process of reevaluating the way it provisions resources at its facilities to include both data and compute. The potential for impact assigned to this requirement is medium; by-and-large, it is focused on the need for a change in policy and focus at the HPC facilities.

Of course the next-generation experimental analyses and simulations, while producing multi-PB data sets, are often carried out by large collaborations and involve hundreds — if not thousands — of scientists. While not everyone is needed to run a given pipeline or set of simulation codes, almost everyone in a collaboration plays the role of a data-scientist, mining the resultant simulation data sets in variety of ways that span a number of scientific goals. Determining the best way to handle this new type of "user" will be critical to maximizing the scientific return on these large-scale runs. In addition, these analyses may often benefit from a variety of different hardware platforms. The ability to make these available to the user, either locally or through high-speed networking to another facility, will be critical in dramatically decreasing the turnaround time on the analysis.

Discussion
The typical, future workflow for large-scale simulations is that a computational scientist runs a simulation code to generate data, then data scientists consume the data and perform analysis, preparing data products for the rest of the collaboration. An example from cosmology nicely highlights this type of effort. Simulations generated at multiple HPC facilities must be confronted by the observational data sets in order to reduce systematic uncertainties in the data. The data can come from a variety of observational facilities, each with its corresponding analysis tool chains — thus, data movement plays a key role. Because both the simulation and observational data sets are applicable to a wide variety of science goals, there are issues pertaining to access, sharing, and file formats that make large-scale analysis easier. Finally, new mathematical techniques involving uncertainty quantification and simulation-based, inference-incorporating emulators are needed to confront the observations with the simulations in a statistically robust way. These emulators can be different across experiments, as well as within an experiment. The Dark Energy Survey (DES) currently has four major focus areas (cosmology from supernova, weak lensing, clusters, and photometric baryon acoustic oscillations), each of which has multiple sets of analysis codes. DES is an international project with more than 400 scientists from 25 institutions in 7 countries — small by the standards of many future experiments. The data sets for DES are relatively small — scores of TBs for both the simulations and corresponding observational data sets — but they will grow considerably as we move to surveys such as those that will be generated using the Large Synoptic Survey Telescope.

Another area highlighting the need for a specialized data-centric focus would be for the analysis of "large data" projects. For example, brain-centric activities right now would strain any DOE HPC facility — and likely will for several years to come. Whole-brain studies can easily require 10s–100s of PBs of storage for full-resolution, multi-modal data that are then used/analyzed/studied in different ways by members of a wide variety of collaborative teams. Just managing one such dataset at a facility would be challenging today, and yet several such datasets will likely exist in many different science domains in the near future (including light sources, climate modeling, cosmology, and genomics).

Similarly, the need to have specialized storage resources for "unusual" scientific data (e.g., graph-based data) is also evident. Classical DOE applications typically do not focus on this topic, although some are starting to do so. Many challenging science problems use or process this type of data (i.e., precision medicine).

Imagine moving analysis to data that are geographically located elsewhere, so that parts of the job may run locally while other parts may run remotely.

Finally, the computational scientist may want to have access to dedicated co-processors for analysis as data come off a machine. In this case, the ability to schedule and place data-centric operations at various locations in the storage and memory hierarchy will be critical. The capability to enable/disable/activate/deactivate "dark hardware" that may be present on nodes for particular types of analysis could improve the speed at which *in situ* analysis can be performed. This idea can be extended to include trans-center migration of data-centric tasks: imagine moving analysis to data that are geographically located elsewhere, so that parts of the job may run locally while other parts may run remotely.

R&D for Data Movement and Processing across the DOE Complex (3)

Description and Drivers
While we have categorized this requirement as medium with regard to potential for impact, in many ways, the following R&D efforts can be seen as the low-hanging fruit on the path to the creation of a superfacility. These efforts span a huge range across CS — from queuing and scheduling to cold storage to workflows. The main driver for this research is to fabricate a base of activities that can enable a future superfacility.

Discussion
The first set of activities we will explore can be summarized as those that enable efficient workflows across facilities through advanced networking. The focus of this research is effectively on cost modeling. We wish to answer the question, "Should I move the data or should I move the compute?" Examples of this dilemma include running on a GPU machine versus a Knights Landing and understanding whether it is more economical to move the data to the fast machine or do it locally on the slow machine. Users would also need to understand when reducing data at a given location is more valuable than moving it as-is. Through this effort, it is clear that the ability to generate high-confidence performance estimates of complex, distributed workflows that may run on multiple, complex resources will be an invaluable tool for estimating such costs. This is especially true if the complexity involves, for example, cross-facility mounting of file systems, data movement around the globe from memory to memory with minimal latency, and an optimized use of the memory hierarchy at each facility.

Given the sheer number of large data sets across several science domains, a new area for ASCR research should likely be cold (or "cool") storage. Here we are talking about storage that does not require power (see Balakrishnan et al. 2014) as a much faster and more energy-efficient alternative to tape. The specific focus of this work should be on determining the relative cost-benefit ratios for access times to the data.

Then there is work centered on the hardware configurations and software stacks available at the facilities. The focus of this work is on determining the right mix of hardware configurations to support different classes of experiments, as well as the right software stacks and programming environments. Addressing queuing and scheduling, and the interplay between real-time queues and batch queues, is mandatory because an effective facility for data and HPC will require both. Automation of workflows, as well as federated identity management, is paramount to getting the most out of these systems. In addition, a process by which thousands of users can access these resources or, at the very least, the data sets they process or generate will need to be developed. Finally, the key question — what is the right set of tools to tie all this together and make it work? — will need to be answered.

3.2.1.2 Relationships among Needs and Conclusions

Our group prioritized our needs based on the vision of creating a superfacility and the assumption that such a facility will be vital to the success of exascale data analysis, the analysis of future experimental and observational data sets, and the joint analysis of both. While we classified them as medium priorities, taking a more data-centric view of the HPC facilities — as well as R&D into data movement across the DOE complex (from experimental to HPC facilities as well as from one HPC facility to another) — should be seen as the baseline effort needed to achieve our vision of a superfacility. Both of these requirements involve meaty computer science, networking, and mathematical R&D efforts, as well as more policy-oriented decisions about allocation of resources (storage, compute, and networking) at the HPC facilities, long-term maintenance and provenance of computational and experimental data, and federated identity management across the DOE complex, among other needs.

3.2.2 Early Delivery Systems

The uses of early delivery systems for data management, analysis, and visualization give rise to special needs for the system. Table 3-6 lists the highest-priority ASCR research needs identified for data management, visualization, analytics, and storage in early delivery systems; the scientific drivers associated with those needs; and the potential for impact of each identified need (low, medium, high).

Table 3-6. Needs: Data Management, Visualization, Analytics, and Storage of Early Delivery Systems

Number	Need	Scientific Drivers	Potential for Impact (L, M, H)
1	**Promotion of early access with timed grants:** Taking advantage of early delivery requires a confluence of events (funds, resources, and people) that happen to be ready. If early delivery is tied to a funding opportunity, resources will be provided for R&D on that platform.	Short window of opportunity to conduct R&D on new architectural features during early delivery. Early delivery and relevant research projects are often misaligned.	H
2	**Less restrictive access:** Facilities could provide dedicated system time on a regular basis to those who request it and more relaxed policies for experiments that would otherwise break the system for other users.	Many scientific data management, analysis, and visualization at extreme scale (SDMAV) research needs (e.g., experimental I/O, job coupling, human-in-the-loop, time-constrained) do not mesh well with batch-queue.	Really High
3	**Engagement with system software and vendors about application program interface (API):** APIs are not well established, documentation is incomplete/underwhelming, incomplete access is provided for new features so that users do not "break things."	Early delivery requires a reasonable prototype API but also flexibility to make changes before production.	Super High
4	**Improved coordination/communication among early delivery users:** Often need to know what other users are doing on early delivery systems. Often this information is internal to the facility.	Much SDMAV software layered on other software. Need coordination (or knowledge) among software dependencies.	Totally High

3.2.2.1 Needs Identified from Breakout Session

Promotion of Early Access with Timed Grants (1)

Description

Early delivery systems generally provide an opportunity to test the readiness of existing software and to explore new ideas. However, a common issue, in addition to struggling with a machine in its pre-production state, is that researchers often have trouble providing resources to address problems while a machine is available in its early delivery state. The early delivery must coincide with researcher availability and funding that are consistent with the opportunities offered by new machine features. Research opportunities are missed when this confluence of events does not occur.

The concurrence of early delivery and researcher availability can be greatly facilitated by ensuring that delivery of the early system is accompanied by grant funds to support research activities that specifically take advantage of this narrow window of opportunity on new features. Organizing such grants necessitates careful coordination between the facility team and program managers to time the availability of the grant with the availability of the system.

Drivers

The drivers of this need are that much data management, analysis, and visualization R&D is focused on taking advantage of new architecture features, but these are often inaccessible to researchers. The early delivery systems often have these features, and a window of opportunity exists to conduct R&D using these features prior to the systems going into production. In some cases, R&D can only be performed on pre-production systems because of the excessive disruption that would result to the system and production users.

Discussion

Research funding specifically tied to the availability of an early delivery system depends on a confluence of events to allow researchers to be ready to take advantage of the system. For example, several breakout participants could remember seeing an email announcing the availability of an early delivery of burst buffers at NERSC, but they were unable to act upon the announcement because of the numerous competing needs of current projects. Grant proposals rarely specifically design work that relies on early delivery systems. Even if a facility's roadmap is planned, it is a leap of faith to rely on delivery, features, and availability of early systems for project success. If early delivery systems were tied to grant opportunities, such as a Funding Opportunity Announcement (FOA), resources would be made available for R&D specific to that platform and the new opportunities it provides. Implementing this feature would require facilities to work and coordinate with ASCR program managers. Over time, this model could prove to be useful in promoting new R&D work on early delivery systems and could promote better integration and interaction between ASCR facilities and researchers.

Grant money for research on early delivery systems can come from multiple sources. Ideally, the planning for the acquisition of a computer will include grant money set aside for research while the machine in its early delivery phase. However, grant money can also come from less formal sources. There is a possibility for program managers to provide "plus-ups" on existing grants that have value in tying their ongoing research to a new architecture. Program managers can also be receptive to unsolicited proposals related to early delivery systems.

Less Restrictive Access (2)

Open policies on new system features are better because they give researchers a chance to explore a broader range of ideas, and the fruitfulness of these ideas is a much better driver for policies than preconceived notions.

Description

Choosing policies is a critical aspect of new hardware and software features, and early delivery systems can play a crucial role. New features are often introduced with a particular use case in mind, which rationally influences the policies. These policies, in turn, affect the type of research involving the new system features. As such, research ideas may be dismissed early as infeasible, simply because the policy is too restrictive to implement the research idea. A better approach is to flip the relationship around. Open policies on new system features are better because they give researchers a chance to explore a broader range of ideas, and the fruitfulness of these ideas is a much better driver for policies than preconceived notions.

In general, it is best to keep policies as relaxed as possible on the early delivery systems. Because the system itself is in pre-production, it is a good opportunity to relax policies that otherwise stand in the way of potential research and features. An example of such a rational, but restrictive, policy is the common scheduling policy that prevents multiple jobs from accessing the same resources, which prevents unwanted resource contention between unrelated jobs. Another example is a common security policy that restricts communication between separately scheduled jobs, which prevents accidental or malicious interference of a running job. Although sensible for traditional batch-queue processing, such limitations inhibit many data management, analysis, and research ideas. For example, allowing multiple jobs to be scheduled on the same nodes and to directly share memory could simplify the coupling and scheduling of analysis and visualization services. Another restriction to modern workflow systems is the difficulty in providing a long-running workflow service that manages and communicates with multiple jobs that start and stop over the lifetime of the workflow. Relaxing policies for early delivery systems provides an opportunity to explore these ideas.

Likewise, early delivery systems, with the help of receptive administrators, can provide a unique opportunity for modifying or replacing key system components. For example, parallel file systems are a critical component of high-performance computers and the jobs that run on them. Thus, the administrators of a production system do everything in their power to ensure stability and restrict all access to well-defined file-system protocols. No user is allowed to directly access file-system nodes, and running custom software on them during production use is out of the question. This makes R&D of parallel file systems, particularly at scale and for radically new designs, extremely difficult. Experimenting on a production machine during system time, when no standard users have access, is feasible, but difficult and expensive. In contrast, early delivery systems provide a better and potentially less costly solution. Experiments on critical components are more reasonable before the system goes into production. Such use cases could be better supported by regularly scheduled (e.g., weekly) system time that can be easily allotted to such experiments.

Drivers

The driver for this request is that many data management, analysis, and visualization needs involve complex use of multiple components, as well as time-constrained use scenarios that do not mesh well with traditional batch-queue-mediated system access. Such R&D activities include user interactions, human-in-the-loop operations, inter-job/coupled-code communication and interaction, and elastic computing and resource needs.

Discussion

Data management, analysis, and visualization may have needs in accessing early delivery systems that go beyond the carefully marshaled batch-queue access. Examples of such R&D activities include interactive visualization, distributed multi-stage style data-intensive pipelines, and coupling via *in situ* and in-transit methods to codes running on other machines.

Facilities could provide such access through a blend of user reservation and free-for-all access at prescribed times when the system is closed to other users. These dedicated times could be at regular intervals (e.g., every N days). Likewise, the system could flip between modes with safer, more production-like policies at some part of the interval and open access at others.

Engagement with System Software and Vendors about API (3)

Description and Drivers

One of the most challenging issues with software development on pre-production, early delivery systems is that the API for new and developing subsystems is often in its infancy. Frequently, there are multiple issues with these APIs and their associated drivers. There might be performance issues in the initial implementations. Documentation is often incomplete, incomprehensible, or incorrect.

The API might have intentional feature gaps to prevent users from "breaking" things. Often these APIs are inconsistent between HPC systems. For example, there is no consistent API for burst buffer technology, which continues to change into second-generation systems. These restrictions make early adoption difficult, but they also provide the opportunity to shape the API and these features before production.

Discussion
The discussion on using pre-production APIs was a contentious one because of the conflicting needs researchers have for these features. On one hand there is a need to have stable, consistent, well-designed, and well-documented access to new features. On the other hand, there is a need to have early access to these systems (before production-readiness) and a need to influence and change the abilities of the API as its use becomes better understood. These needs cannot coexist. Pre-production access, by its nature, means that undiscovered bugs might make the system less stable, and documentation might not be complete. Likewise, if research is going to potentially change the API, then the researchers must contend with the fact that the API might change.

This contention of needs for pre-production system hardware and software necessitates a close engagement between researchers and vendors. Researchers need such an engagement to navigate the use of a system whose documentation and implementation might not be error free or complete. Researchers also need vendor engagement to help them through problems, which may require changes in the vendor's hardware or software. Vendors also benefit from such engagement by learning about their customers' needs and ultimately improving the performance of DOE applications on their systems.

Improved Coordination/Communication among Early Delivery Users (4)

Description and Drivers
Data management, analysis, and visualization R&D must continue to leverage early delivery systems to test readiness, performance, and functionality limits, as well as to construct new techniques and software for the next-generation system. In fact, the earlier these can be performed, the better. Thus, it is critical that calls for early access be as widely disseminated as possible to researchers, particularly those in ASCR.

Software products typically have dependencies with other software products; this is particularly true for data management, analysis, and visualization. For example, simulation software is likely to depend on an *in situ* visualization library. Likewise, an in situ visualization library is likely to have dependencies on rendering and I/O software systems. Those involved with these software products on the early delivery system need better knowledge of, and communication with, other software teams also using the machine. Such communication can help coordinate the development and release of software with deep groups of interdependencies. Such information may only be informally known through user group attendance, and that information itself may only be internal to the facility.

Discussion
The example of burst buffers came up a lot during this discussion. The access mode to burst buffers is still in a state of flux and varies greatly from one machine to the next. The API for burst buffers is still in a state of contention for a variety of reasons. First, burst buffers are installed in a range of places: (1) as a resource on a dedicated set of I/O nodes, (2) as a storage drive located in each compute node, or (3) as a bank of special NVM. Furthermore, there is contention as to how the buffer is used. For example, it could be used as a high-speed temporary cache for the file system (the traditional "burst buffer" idea), as a communication tool across jobs, as part of an implementation for resiliency, or as extra memory available in the memory hierarchy.


Figure 3-5. ASCR researchers require engagement with system software and vendors about APIs: APIs are not well established, documentation is incomplete/underwhelming, and incomplete access is provided for new features so that users do not "break things" (Images courtesy of Argonne).

The management of the APIs for features in an early delivery system is contentious, and the group has no clear direction. Researchers want and need a stable, functional, and well-documented interface to features on an early delivery system. However, we also recognize that the software and drivers for early features are likely immature, and researchers also want opportunities to shape the APIs to support the work and results of their research.

The NERSC burst buffer early delivery was mentioned as a model of how an API rollout could work. The early burst buffer API was available; however, users of the early delivery system were also able to contribute in ways that resulted in changes when the system was moved to production.

3.2.2.2 Relationships among Needs and Conclusions

Our group made no real effort to prioritize the needs that we identified. Rather, the four needs presented here represent a conglomeration of topics that are grouped around a common theme.

First, the need for less restrictive access came up repeatedly in several different contexts, both in this session and in the Data, Visualization, Analytics, and Storage on Production Systems session. The idea often came up in two different contexts. The first — and the one more passionately discussed — is the ability to get "close to the metal" in ways that circumvent the regular protections of the system. The common example in this case was the design of a new parallel file system, which requires access to a section of the machine that is generally considered off limits. Another theme along this topic was the relaxation of policies. A common example of a relaxed policy is to allow multiple jobs to share resources and tightly communicate in support of coupling of services and advanced workflows.

Second, the need for timed grants started as a deep sub-bullet during a long conversation on the availability of access to early delivery systems, but percolated to the top of the discussion as one of the more novel and useful ideas. Third, communication was also a common theme that was eventually grouped into two categories: engagement with vendors and communication among users.

3.2.3 Production Systems

Table 3-7 lists the ASCR research needs identified for data management, visualization, analytics, and storage in production systems; the scientific drivers associated with those needs; and the potential for impact of each identified need (low, medium, high). There are two primary drivers for the needs identified. The first is the evolving needs of our science stakeholders/collaborators who wish to make use of HPC centers to do science. Their use models diverge, to varying degrees, from the batch-oriented and computationally centric workloads of the past. The second is the need to better support the ongoing needs of research efforts in data management, visualization, analytics, and storage.

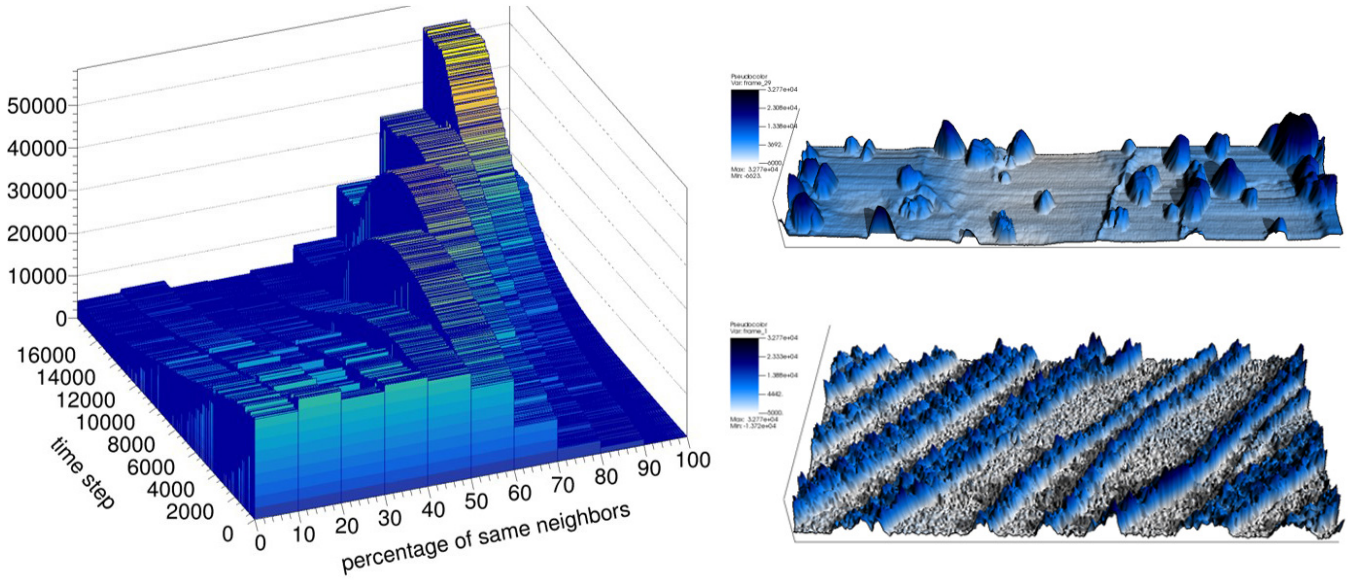


Figure 3-6. Left Image: The SciDAC SDAV Institute-WastePD EFRC visualization and analysis project allowed development and deployment of algorithms to detect locally correlated events in metallic glass materials. The image shows the evolution of atoms with the percentage of neighbors remaining the same (Image courtesy of John Wu, LBNL; and Wolfgang Windl, Ohio State University). Right images: the SDAV-IDREAM visualization and analysis project facilitated improved understanding of particle interactions to characterize the roles of interfacial chemistry and structure in particle-particle and particle-solvent interactions in complex chemical environments. The top image shows aluminum oxyhydroxide clusters on mica and the bottom image shows interfaces of collagen on mica (Images courtesy of Oliver Rüebel, LBNL; and Benjamin Legg and Jinhui Tao, Pacific Northwest National Laboratory [PNNL]).

Table 3-7. Needs: Data Management, Visualization, Analytics, and Storage of Production Systems

Number	Need	Scientific Drivers	Potential for Impact
1	Community-centric data archives, access, distribution	Sharing data, preserving data, fulfilling data management plan	H
2	Better support for low-latency, high-throughput workflows/data pipelines	Science research increasingly data driven; CS/math research focusing on solutions	H
3	Persistent data-centric services	Scientific process automation, next-generation services focus on data	M
4	Increased role of ASCR facilities in creating better software; regression testing, nightly compiles for user-supplied software	Facilitate software transition from research to production, foster facilities' confidence in user software	H
5	Knowledge of available resources across the ASCR facilities complex; better/consolidated/simplified information about gaining access to ASCR facilities	Unclear understanding of what resources are available across the complex	M
6	Access to testbeds at scale	Comprehensive scalability studies	M
7	Policy changes in support of ASCR computing research	To accommodate data-centric processing, gracefully accommodate uneven system performance, group-level access to resources	M
8	Access to operational data	New research in methods for SDMAV focusing on systems	M
9	Project-centric services	General support services that would be broadly beneficial to ASCR research programs	M

3.2.3.1 Needs Identified from Breakout Session

Community-Centric Data Archives, Access, Distribution (1)

Description and Drivers

Many projects — research efforts within ASCR, as well as numerous computational, experimental, and observational science projects — need to quickly stand up data archives and repositories that are accessible to groups ranging from individual project teams to worldwide communities. The scope of needs in this space is diverse, ranging from long-term archival storage accessible to a single principal investigator (PI) or project group, to collections of high-quality curated data that are accessible to the broader scientific community. In between these two extremes are collections that are made available on a selective basis. The idea here goes well beyond passive collections of data to include additional capabilities that enable data to be useful. Data needs to be discoverable, which means that it needs to be searchable in ways beyond filename-based methods.

Discussion

In terms of how this set of capabilities is presented, many session members spoke about the usefulness of being able to easily set up such collections, along with the ability to administer access to data. This idea is not the same as privileged access to system resources — that is, "root-level" access — but is more about having the ability to set up, configure, and conduct ongoing administration and management of data collections.

Also within scope for this topic is the idea of being able to set up and run pipelines of operations on data collections. Examples include on-demand preparation of derived data products, such as data subsets, or the results of computations performed on collections of data. Many types of simulation sciences require access to data collections for setting initial conditions and comparing results for accuracy/convergence over the course of the simulation run.

All science projects — particularly experimental and observational projects that wish to make use of ASCR HPC facilities — have as part of their core mission the generation of data products. These products are shared with potentially diverse groups of stakeholders and consumers. These consumers could be an individual PI, a project team, a community, or "the world."

There is a federal mandate to make the results of federally funded research publicly available. In particular, a memorandum from the Office of Science and Technology Policy (OSTP 2013) mandates that the data from all federally funded research, where possible, become public. There is currently no established infrastructure for making these data available, particularly large data sets.

Results from the 2015 ASCR workshop on experimental and observational data (DOE 2015) included several observations about the central role such capabilities could play in experimental and observational science projects. One such science project team indicated that the only long-term archival capability available to them, which includes long-term public access to data, was that provided by the journal when they publish a paper. These project teams also mentioned that they are left to deal with this problem on their own; there is no program-wide perspective on this problem.

To some extent, this condition exists across ASCR research projects: we are left to our own devices when it comes to long-term data archiving, curation, and dissemination. Individual researchers or projects try to find quick and easy solutions, which often include project-centric websites. However, there is no program-wide view of problems in this area, which is the subject of the subsection on need 9 (Project-Centric Services).

Discussion
Opportunities to enable ASCR research efforts in data science may be obstructed by the lack of suitable infrastructure. Examples include advanced search, collaboration, and sharing. Opportunities are needed to transform ASCR computing facilities from compute-only operations into models for enabling knowledge discovery in data-driven science (which encompasses the majority of current science).

Better Support for Low-Latency, High-Throughput Workflows/Data Pipelines (2)

Description
Historically, a significant portion of the ASCR research portfolio focused on traditional HPC problems and workloads, namely high-concurrency applications that are managed by batch queues. Increasingly, a new class of challenges is emerging from the science community: experimental and observational science, in which computational activities must be performed quickly and involve movement and processing of large amounts of data. Emerging ASCR research topics in the areas of workflow management, approximate (fast) methods, distributed computation architecture and optimization, optimization of data movement and placement, etc., all diverge from the traditional uses of ASCR HPC computational facilities. In the subsection for need 7, we describe several different types of policy changes in support of ASCR computing research that would benefit such new types of usage patterns and needs.

Drivers

The primary drivers stem from how many science projects want to make use of ASCR HPC facilities. For example, the 2015 ASCR workshop on experimental and observational data (DOE 2015) concluded that HPC resources must be available at the time experiments are run to support specific types of science use needs.

A secondary set of drivers follows the first: some ASCR CS and mathematics research projects focus on developing new methods and approaches to respond to the science drivers, and these new methods and approaches often involve the use of ASCR HPC facilities as a vehicle for delivering the new capabilities to science users. One approach for optimizing throughput in data-intensive workloads could involve staging data on on-node, local, persistent storage for use by consecutive stages in a processing pipeline.

Discussion

One significant challenge to supporting this need is the tension between current ASCR facilities' metrics, which include a focus on level of utilization, and the way low-latency, high-throughput workloads make use of resources. Those types of workloads and use patterns can be characterized as "bursty." Enabling facilities to provision resources for such workloads may entail allowing some resources to be "idled" so that they may quickly be provisioned as needed. Creative approaches to revising policies and metrics may yield a workable solution.

Persistent Data-Centric Services (3)

Description and Drivers

The traditional view of "data services" at HPC facilities centers on use patterns for computational workloads: I/O libraries, persistent file-based storage, and optimizing these software and hardware components to perform reasonably well for common use patterns. As science becomes increasingly driven by data, the way that data are used, processed, and managed in support of science is rapidly evolving.

The term "persistent data services" is broad in that it encompasses a diverse set of topics. These topics include the combination of data and processing infrastructure to support accessing data, creating data products, and performing on-demand analysis by a potentially large collection of users (see subsection for need 1 on community-centric data archives, access, and distribution). Also within this scope are classes of services accessible for intra-center use, such as the ability to perform rapid, large-scale data indexing in support of database-like operations (e.g., queries, subsetting based on potentially complex and fuzzy criteria). The topics also include the building blocks needed to implement complex distributed workflows, which are tasked with responding in the presence of new data from internal or external sources, or requests for data or processing from internal or external sources. In addition, beyond the building blocks themselves is the need for easy-to-use configuration and management of such services in a way that does not require privileged system access and that is consistent with site policies and practices.

Currently, however, each project member would have to load all the data into the LCF system for every job, making such an approach prohibitively costly.

A wide variety of scientific projects regularly use workflow management systems, but they experience tremendous challenges when they want to move these systems to the LCF systems because there is no opportunity to run these types of services permanently at the facilities so that many members of a community can use them.

Another emerging need is the community-wide ability to analyze large data collections, a materials properties archive, power grid data, climate data, and so forth. Currently, however, each project member would have to load all the data into the LCF system for every job, making such an approach prohibitively costly.

Discussion
Climate, materials, cosmology, chemistry, and biology all require workflows for complex numerical modeling and analytical jobs or large-scale combinatorial or ensemble runs. The availability of workflow management systems from the LCF computers would help to optimize system use and accelerate scientific discovery.

The longer-term availability of community data archives on the LCF system would enable these communities to run a large-scale analytical analysis across massive data collections, accelerating scientific discovery, but also enabling new discoveries. These discoveries would only be possible if large volumes of information can be correlated and explored in a responsive mode enabled by extreme-scale compute resources.

Increased Role of ASCR Facilities in Creating Better Software (4)

Description
A significant amount of software in use by the science community on ASCR platforms is open source: distributed teams of developers contribute to and make use of software tools and applications. Many ASCR-funded research projects engage in this exact model of collaborative software design and development: research ideas are converted to practice through collaborative, open-source software projects. Open-source software efforts that strive for "production quality" status include software and release engineering efforts that consist of nightly builds on multiple platforms, along with ongoing regression testing, bug tracking systems, etc. Such approaches increase software quality, as well as user confidence in the software.

The specific need identified at the review is the ability to set up and manage this type of software and release engineering at ASCR computational facilities. Participants pointed out the need to set up nightly regression tests, compilations on multiple platforms, and related capabilities (e.g., bug tracking; see discussion for need 9, project-centric services) for user-supplied software, which would be the product of efforts funded by ASCR research.

Drivers
One of the primary drivers behind this idea is to foster an improved ability to deploy ASCR software products on ASCR facilities. Having this capability would help to refine the process for preparing software from ASCR research efforts for deployment, and it would also create closer, more cooperative relationships between ASCR R&D activities and ASCR facilities.

Discussion
ASCR facilities may have more confidence in installing software that is known to meet some minimum threshold of quality assurance, which would be demonstrated by regular testing.

Some ASCR facilities already maintain such services internally as part of their operations, while others do not. Normalizing the presence of these services at facilities that do not yet operate them may be challenging, but the ultimate benefit would be more uniform quality of service across the centers.

The opportunity exists to enable new types of CS research, such as fault injection, into software testing, which would result in better software. The test harness could simulate code runs in a "hostile" or "resource-constrained" environment (e.g., low memory).

Realizing a successful deployment of this kind of capability would likely require attention to operational details, such as authentication, authorization, estimates of resources needed, and quotas. In addition, emerging forms of software products may not fit neatly into a single-application model, such as those that are workflow-based or that use a distributed-computing pattern.

Knowledge of Available Resources across the ASCR Facilities Complex (5)

Description and Drivers
A diverse array of resources exists across the ASCR facilities complex. While information about the primary resources at each facility is available on the respective facility's websites, information about additional resources is often not readily available. Instead, knowledge of the existence of such resources (e.g., testbed systems) is often spread via word-of-mouth among colleagues.

There are at least two dimensions to this issue. The first is "broadcasting" information about available resources at ASCR facilities to the ASCR research community. Several participants suggested that such information accompany a funding award letter, thereby notifying researchers of the existence of resources at facilities. Second is the notion of making available information about "secondary resources," like testbed systems, in a way that goes beyond word of mouth. These ideas were echoed across other areas of the ASCR Exascale Requirements Review.

The primary driver here is to enable ASCR researchers to make effective use of computational and data resources that exist at facilities to support their research efforts. One side benefit is the potential for reduced costs: researchers can avoid purchasing hardware when such hardware is already available at facilities.

Discussion
One potential challenge is finding the right balance between operational and research concerns. For example, facilities often have small-scale versions of production systems they use for testing new software releases (e.g., a new OS) before rolling it out on the larger-scale machine. Enabling researchers to access such resources may result in an increased workload for facility staff, but may be of tremendous benefit to the research program.

The potential opportunity is the acceleration and advancement of research by users who would have access to unique resources that may be out of reach of their individual projects.

Figure 3-7. Productive development and prototyping of complex workflows often require interactive resources, necessitating improvements in the allocation of interactive and batch resources on today's production systems, like Argonne's Intel/KNL Theta system.

Access to Testbeds at Scale (6)

Description and Drivers
Several project teams indicated the need for access to systems to conduct at-scale testing of new methods and applications. In many cases, the new methods or applications could require configurations that are different from those in production use (e.g., new kernel modules or new kernels) or are designed for a diverse set of uses (e.g., file-system optimizations, interacting with full-scale runs on a production system, and streaming data methods). To minimize the impact to production systems and users, workshop participants felt that using large-scale testbed systems would be a good approach for such activities.

Many ASCR data-centric research programs are focused on problems of scale; studying the behavior of different approaches at scale is an integral part of the research effort. In some types of research, scientists can pursue approaches that could potentially be disruptive to production systems — approaches like changes to the file system that target performance optimization. Using a testbed system of significant scale would be of benefit to both research efforts, as well as to the ASCR facilities, in minimizing or eliminating the risk of disrupting production systems.

Discussion
Many research efforts (e.g., storage systems) require the ability to conduct testing and evaluation at full system scale. New ideas that may seem promising on small-scale research systems require study at scale and, where possible, under the types of full-load conditions that exist on production systems. These types of projects provide the next generation of technologies that are then deployed at ASCR facilities.

It is beyond the reach of a single research project to purchase such a system for these types of studies, so there is some economy of scale possible when a large testbed system at an ASCR facility can serve multiple purposes, including for use by research projects in conducting at-scale studies.

Policy Changes in Support of ASCR Computing Research (7)

Description and Drivers
The representatives of ASCR research projects present at the review suggested a number of changes at ASCR facilities that would benefit their research efforts. In some cases, these suggested changes would be of broad benefit to existing production workloads. Some of these suggestions may involve policy changes, while others may involve configuration changes or new capabilities from system software vendors.

- Support should be provided for multiple executables running on different core sets within a single node, where the different executables would communicate through shared memory.
- Creative job launching options are needed to support data-centric operations, ranging from *in situ*/in-transit workflows to distributed workflows. Whereas traditional job launchers provision cores on nodes, more complex use scenarios may entail provisioning additional resources (e.g., I/O capacity, storage, and network circuits).
- Relaxing of batch-queue limits is needed. Traditionally, part of the batch job launch configuration includes a hard wall-clock limit; some batch queues are configured for long-running jobs, while others serve short-running workloads. When a given job exceeds its wall-clock limit, it is terminated. The idea here is to relax the limits. The example that came up in discussion is a job that is taking too long because of I/O, where I/O is slowed down because of contention resulting from many users. Such contention is unpredictable, and it is impractical to take into account *a priori* when estimating job run time. In cases like this, it would be useful if the batch queue would allow the job to continue under certain circumstances.
- I/O should be available as a schedulable resource or quality of service, in the same way that cores are schedulable as resources.
- At present, we can only schedule resources on HPC systems on a per-user basis. In the case of data-centric projects, which often involve a collaborative effort by numerous persons/users, it would be useful to be able to schedule resources for whole groups, so that they can flexibly use such resources during the allocated time.

Drivers
Generally speaking, the drivers for these types of policy/configuration changes reflect the changing way in which users employ HPC facilities to do research and science.

Discussion
While some of these changes may be relatively straightforward, others may be more complex and require evolution in the infrastructure used at HPC facilities. Because these changes reflect needs arising from an evolution in how science users and ASCR researchers want to make use of these facilities, an opportunity exists for better support for science needs by implementing such changes.

Access to Operational Data (8)

Description
Several different elements within the ASCR research portfolio can benefit from access to system operational data, both live and historical. For example, visualization and analysis efforts that focus on high-dimensional exploration and analysis methods need this type of data both to develop new methods for understanding the operational data and working with and managing the HPC system effectively. Other uses include building better models aimed at performance prediction under varying circumstances.

Drivers
A significant driver for obtaining access to operation data is the opportunity to conduct new types of research that ultimately benefit both ASCR facilities and users. Examples include the following:

- Dynamically adapting application/service behavior to the system state (e.g., autonomics);
- Providing insight into the performance of applications, including the root causes of performance degradation or variability;
- Providing insight into common patterns of data service use at the facility;
- Guiding development of new services to better serve ASCR science needs; and
- For related applications such as power grid monitoring and management, in which a large collection of sensors gathers information in real time, optimizing overall system operation in light of present and future predicted conditions.

Discussion
The likely primary challenges to provision of operation data are the complexity and cost of collecting such data, curating it, and making it available to the ASCR research community.

Project-Centric Services (9)

Description
Project-centric services are those things that "every project needs," but that may be out of reach or not readily available at the institution where the project resides or elsewhere. Several different types of services were mentioned during the breakout session:

- The ability to set up persistent data repositories to serve individuals, project groups, or communities (see discussion for need 1, community-centric data archives, access, distribution);
- The ability to have a web presence for projects (that includes custom domain names), along with suitable content management infrastructure that is accessible and usable by individual users, as well as project groups; and
- Code/data revision control systems for use by individual users or project groups.

Drivers
One significant driver here is the 2013 OSTP memo (OSTP 2013) that mandates that the results of all federally funded research be made publicly available. Ideally, all ASCR-funded research projects should have a web presence of some sort that supports dissemination of information about the project; however, there is uneven capability across institutions to provide this kind of service. The reasons vary and include cost, lack of institutional will to provide such services, and insufficient or inadequately trained staff. The same factors apply to the other project-centric services.

Discussion

Hosting websites and wikis where users/researchers are primarily responsible for the content presents a security challenge, involving risks beyond the loss of scientific data. Users and facility staff alike must be educated on, and follow, best practices for securing websites.

The issue of whether these kinds of services are within the scope of responsibility for ASCR facilities is an open question. It seems clear that some types of services — such as publicly accessible scientific data repositories, along with the means to invoke data-centric services (e.g., analysis, data subsetting, and on-demand creation of derived data products) — are needed but that they present unique opportunities and challenges.

Other types of services, like project web portals/presence, would be of broad benefit. Such services would take advantage of the highly skilled staff already in place at ASCR facilities, would eliminate redundant efforts and costs that are presently borne across the entire complex in an *ad hoc* fashion, and would help ASCR to make significant strides toward meeting the intent of the 2013 OSTP memo.

3.2.3.2 Relationships among Needs and Conclusions

The set of needs outlined in the preceding sections reflects two primary sets of drivers. The first is the fact that the way the scientific community wants to make use of HPC computing facilities is evolving. That evolution reflects an increase in sensor/instrument resolution, which in turn generates an ever-increasing amount of increasingly complex data. It also reflects the increasingly important role that data play in scientific research, along with the changing ways in which data are shared and used by broad scientific communities. The inability to make effective use of HPC facilities is viewed as a significant barrier/bottleneck to many science projects (Bethel et al. 2016; Bethel and Greenwald 2016). The second driver is that a significant amount of research activity in SMDAV focuses on a family of topics that involve the use of (or optimizing the use of) production HPC facilities in support of scientific missions.

3.3 Distributed Computing and Networking

3.3.1 Early Delivery and Production Systems

The high-performance distributed computing (HPDC) environment envisions HPC facilities as key computing elements in a heterogeneous distributed processing environment that includes storage, computation, and analysis components — all tied together by high-performance networks. HPDC compute elements include not only HPC facilities, but also scientific grid computing services, commercial cloud services, and other local computing facilities. The ability to effectively move science data into, and sometimes out of, any of these HPDC compute elements is fundamental to the efficiency of an HPDC system. The requirement for effective data movement makes local facility network infrastructure a critical component in the ability of HPC facilities to meet the needs of science collaborations and experiments that adopt HPDC models.

Requirements for HPC facilities to satisfy these HPDC computing models, however, extend well beyond simple local network infrastructure. HPDC applications are truly end-to-end at the application level, putting HPC facility storage systems, file systems, access mechanisms, scheduling capabilities, etc., within the scope of requirement analysis for HPDC needs.

Very early in the breakout session, the HPDC group arrived at a consensus that there would be no real differences between the early delivery and production requirements for HPC facilities. As a consequence, the two breakout sessions were functionally merged into a single breakout session. Table 3-8 lists the research needs identified for HPDC in early delivery and production systems, the scientific drivers associated with those needs, and the potential for impact of each identified need (low, medium, high).

Table 3-8. Needs: Distributed Computing and Networking for Early Delivery and Production Systems

Number	Need	Scientific Drivers	Potential for Impact (L, M, H)
1	**End-to-end test and development environment:** Integrated test network facilities, including data transfer nodes (DTNs) dedicated for testing and development, disruptable Science DMZ infrastructure, and storage	Development of federated distributed computing environment without disruption to production HPC science data movement	H
2	**Next-generation network technology support:** Deployment and support for new/emerging network technologies (capacity steps increases, software-defined network [SDN], etc.)	Development of end-to-end, next-generation network (NGN) technology capability for large-scale science	H
3	**Ease of access:** Federated authentication layer and common base-level HPC cyber security policies	Network R&D developer productivity; framework for HPDC security mechanism(s)	H
4	**Operational data:** Availability of network data and data movement logs	Analysis/analytic resource for developing data movement optimization techniques/tools	H
5	**Storage facilities at HPC facilities**	Development of workflows for data-intensive applications in use or in future plans of large-scale science	H
6	**Co-scheduling capability:** Provide a capability that allows storage systems, file systems, computers, and networks to be co-scheduled to facilitate multi-facility workflows (e.g., superfacility model)	Superfacility — BES (light and neutron sources), HEP (distributed computing)	H
7	**Software portability:** Portability of HPDC applications into HPC environment, ability to run a common application code on the HPC platforms at multiple facilities	Enable scientific communities to put together codes that cross into the HPC hardware domain	H

Number	Need	Scientific Drivers	Potential for Impact (L, M, H)
8	**Data movement software tools support:** Agile software adoption process for new data movement tools/middleware; preferably common across HPC facilities	Availability of new/improved data movement tools for network researcher R&D	H
9	**Opportunistic computing support**	Development of most efficient computation models for HPDC science environments	M

3.3.3.1 Needs Identified from Breakout Session

The ASCR research community has shown extreme interest in the topic of HPDC. The discussions all came down to a common theme: researchers want access to the latest hardware, with opportunities to test "at scale." Some just want access to a portion of the facility, others want to insert a piece of hardware or install their latest OS or monitoring software and see how things respond.

Essentially, researchers want the facility to be able to adapt itself in all respects to be part of a coherent computing, storage, and networking ecosystem. This means interacting with other national and, later, university-based systems that are the archival storage sites. Such capabilities would have implications, for example, in development of edge systems and services; high-throughput, generally used applications; resolving security issues to support such interaction; and engagement with the discipline science and domain-specific systems and software teams.

Over time, obtaining such access would also mean having testbed facilities, because the ecosystem needs to evolve while leveraging developments in storage, networking, and software that were not foreseen when the system was first acquired but that are important when considering the configuration, code and data architecture, and operational modes of the next-generation system. The following sections provide the specifics of how ASCR scientists want to interact with the facilities to perform their research.

End-to-End Test and Development Environment (1)

Description and Drivers

Data-intensive sciences in the exascale era are expected to adopt HPDC models in order to satisfy their extreme computing requirements. HPC facilities will be core components within those HPDC systems. NGN services will need to be developed and deployed to provide the network foundation on which to build these HPDC systems, including at HPC facilities. The commercial marketplace will not provide these NGN services. Rather, they will need to be developed by network and CS researchers. There are two critical elements in that development path. First, the network infrastructure will have to provide the advanced network services and technologies needed for HPDC. Unlike current-generation network services, NGN services are expected to be configurable and even customizable, to meet the needs of the application(s) driving the data movement. ASCR research efforts[2] are already underway to develop some of these NGN services, albeit in very isolated and controlled network environments. However, deployment of NGN infrastructure and services *per se* will not be sufficient to meet emerging HPDC needs. The workflows of data-intensive, extreme-scale science will need to make optimal use of these NGN services in order to

2 Active ASCR SDN research projects include the following:
SDN for End-to-end Networked Science at the Exascale (SENSE) — I. Monga, ESnet (lead PI)
BigData Express — W. Wu, FNAL (lead PI)
SDN Science Flows — N. Rao, ORNL (lead PI)
SDN NGenIA — H. Newman, Caltech (lead PI)
Optical SDN — D. Kilper, University of Arizona (lead PI)
SDN-Enabled Interconnects — M. Lang, LANL (lead PI)
SDN Flexgrid — S.J. Ben Yoo, UC-Davis (lead PI)

achieve the performance levels required within these HPDC systems. Put another way, the NGN services will need to be controllable and schedulable resources for use by HPDC applications. A discussion of these two distinct areas of technical development follows.

As previously noted, HPDC systems are inherently based on highly performant network services. The Science DMZ architecture, a concept developed and pioneered by ESnet and the national laboratories, has greatly enhanced data movement performance for large-scale science. Science DMZs include systems called DTNs that are dedicated to and optimized for wide-area data transfers. As NGN technologies emerge, it will be necessary for network researchers to test NGN services with high-impact science data movement between DTNs located on Science DMZs, including those at HPC facilities. SDN technologies, in particular, hold enormous promise to provide the types of NGN services that will be needed by large-scale science in the coming decade. SDN technologies are expected to provide application-customizable network services (e.g., bandwidth and latency guarantees), as well as logical isolation from general network traffic. SDN-based network services are expected to be on-demand and dynamic. They will be truly end-to-end in scope, where end-to-end no longer refers to site boundary to site boundary. Instead, end-to-end network services will be between the DTN(s) sourcing and sinking the data. The next-generation DMZ concept would provide the capabilities to extend the connectivity deeper into site networks to connect instruments and supercomputers and implement advanced services such as computational and instrument steering.

R&D in NGN services is no longer merely an issue of providing advanced network services; it is now also an issue of enabling the science applications to control and customize them.

The second technical HPDC development area for network researchers is application control and orchestration of the computing resources that are involved in science workflows, including network resources. Advances in data management applications and workflow technologies are enabling those applications to have more control over the computing resources they must utilize in executing their workflows. Dynamic resource management of storage and compute elements to optimize workflow execution has been an emerging trend. Predictable network performance needs to become a third core element in workflow optimization if NGN services are to be effectively used within HPDC systems. In addition, it is not sufficient for the NGN infrastructure and services to be highly performant in a general context; they must satisfy the specific needs of individual science application(s). HPDC systems anticipate that science applications will customize their network service(s) to suit their individual purposes. It must be emphasized that the capability to customize, schedule, and control NGN services for HPDC workflows needs to be end-to-end in scope. R&D in NGN services is no longer merely an issue of providing advanced network services; it is now also an issue of enabling the science applications to control and customize them.

Discussion

HPC facilities currently support state-of-the-art network technologies for access to their resources. However, the HPC facilities do not support a test and development environment for network and CS researchers to develop base NGN services or to enhance data movement applications to support network control and orchestration capabilities. For example, HPC facilities have deployed Science DMZ architectures, including DTNs for moving experiment data into (and occasionally out of) HPC computation environments. Today, however, these HPC DTNs are production use systems. There are no DTNs for development and testing of the types of NGN services that will be needed to enable HPC facilities to become core components within emerging HPDC systems. Such test and development DTNs would need to be configurable and perturbable by the network researchers, and therefore isolated from the production service. Similarly, the underlying Science DMZ infrastructure supporting such test and development DTNs would also need to be configurable and perturbable. In order for such test and development environments at the HPC facilities to be fully useful for NGN R&D activities, configurable and perturbable storage resources would also be required. A small-scale, shared storage system among Science DMZ DTNs would help facilitate

development of orchestration (storage/network) middleware for HPDC systems. Finally, HPC computation resources themselves would need to be accessible as well, in order to fully test and evaluate HPDC systems under production-like workflow conditions.

In an HPDC model, an HPC facility is just one component of a distributed computing ecosystem involving potentially many computation and storage facilities. The NGN services that would provide the foundation for that distributed computing ecosystem need to be end-to-end, and therefore distributed across the entire system. As network and CS researchers build the innovative network technologies and new science DMZ architectures that will constitute the NGN services, they need to be robustly testing science workflows under environments that closely match DOE's production network environment. The HPC facilities need to deploy and support a local test and development infrastructure that mimics the HPC production distributed computing environment, and federates that infrastructure with emerging test and development infrastructures with other DOE computing facilities and sites.

As network and CS researchers build the innovative network technologies and new science DMZ architectures that will constitute the NGN services, they need to be robustly testing science workflows under environments that closely match DOE's production network environment.

Next-Generation Network Technology Support (2)

Description and Drivers

One of the challenges in the exascale environment is effectively dealing with the volume of the data. The scientific community is interested in leveraging exascale machines not just for simulation, but also for data processing and analytics. Thus, being able to feed these computational engines with the data in a timely fashion is a significant challenge.

Discussion

Researchers want the computational facilities to have, in addition to their nominal network connections, a separate test network that they can control. This test environment could host next-generation hardware prototypes or commodity hardware running different software/firmware. As networking heads to the realm of terabit links, finding methods to fills these "pipes" and get the data off the network and properly distributed, so as to not create bottlenecks, is a significant challenge. Thus, there are times when researchers will want to use the full facility with these latest network technologies to see how they respond.

Ease of Access (3)

Description and Drivers

Currently, each of the HPC centers is essentially an isolated "island" of computing. Each has its own rules, ways to submit jobs, and cyber security requirements. Scientists who want to access multiple compute platforms must use different authentication mechanisms in order to engage. Providing a single, federated authentication and common base-level cyber security policy would greatly simplify the user experience and facilitate access.

Discussion

There are a number of reasons why a common authentication layer is necessary for an effective HPDC environment. It promotes productivity not only for the scientist who wants to submit jobs, but also for the ASCR researchers. Network R&D developers, for instance, would benefit greatly by having a federated access model. The technical barrier to accomplishing this is low: it requires only that the three HPC centers come together and establish a standard with which all three are comfortable.

Operational Data (4)

Description and Drivers

ESNet moves immense amounts of scientific data around every day as scientists and scientific collaborations execute their science programs. With the high bandwidth available to them, scientists are often moving data to the compute resource(s) rather than the more traditional way of locating the compute function with the data. Each science workflow network "transaction" is recorded in network logs. These logs provide a treasure trove of data for a variety of use cases that are extremely valuable to the network and workflow researcher. Operational network data is also readily available about network usage and performance within the ESnet environment. NGN services are projected to be able to provide science applications with information about conditions along the network path(s) being utilized by the application. The availability and continuity of that network performance information, however, needs to be end-to-end in scope, not just across the WAN service providers, such as ESnet.

Discussion

Network data logs capture valuable information about how scientists leverage networks and computing to accomplish their goals. Making these data available to researchers would help them better understand how the systems are being used, as well as develop data movement optimization tools and techniques that would ultimately better serve the scientific community. A second aspect of operational data availability is the desire to develop capabilities for science applications to react to network conditions and, potentially, modify their network service(s) in reaction to current conditions. This adaptive/reactive network service capability is one of the major drivers for SDN. Because an end-to-end perspective on network conditions is required for such a capability, it is critical for HPC facilities to provide operational network data for their component of the end-to-end network paths utilized by HPDC applications. PerfSONAR servers can provide a service framework for making such operational network information available (perfSONAR undated). Today, perfSONAR services at HPC sites are typically oriented around active end-to-end network measurements. For NGN services, a richer suite of network data will be required, including counter information on local network infrastructure devices. HPC facilities need to begin to develop this richer operational network data framework.

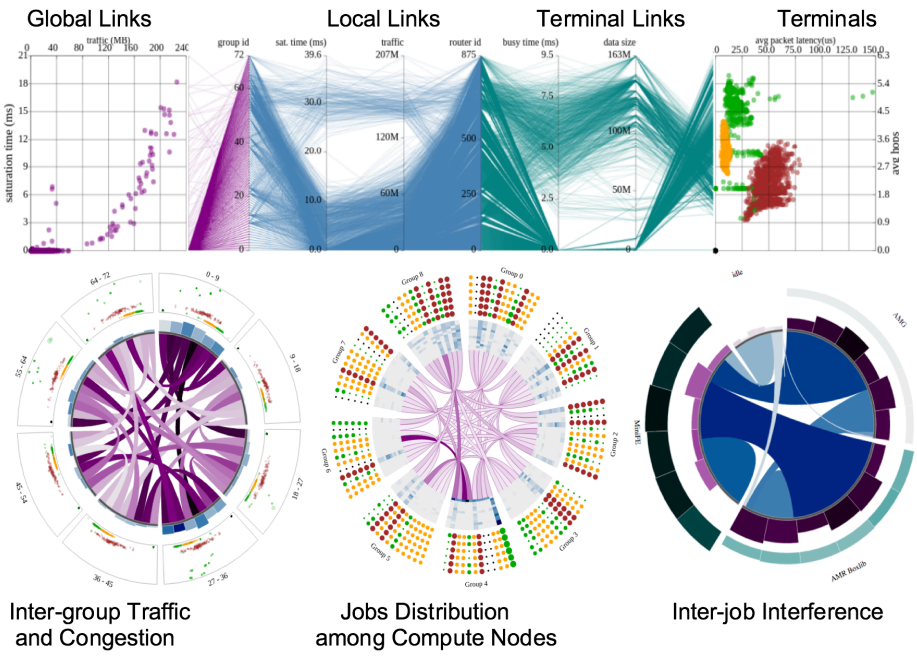


Figure 3-8. For a project supporting co-design of extreme-scale systems using *in situ* visual analysis of event-driven simulations, the project team developed a tool to study the behavior and performance of the simulated network. The tool provides different views of the data: top shows the selected performance metrics of the network simulation; bottom shows different aspects of the network. The visualization examples here show a simulated Dragonfly network with ~5K nodes running three jobs (AMR Boxlib, AMG, and MiniFE) (Images courtesy of Kwan-Liu Ma, UC Davis; Christopher Carothers, Rensselaer Polytechnic Institute; and Robert Ross, Argonne).

Storage at HPC Facilities for Test/Development Data Movement (5)

Description and Drivers
Development of workflows for data-intensive applications in use or planned for large-scale science was the overarching theme behind this specific request. While the use of HPC resources is well established in the compute-intensive arena, the data-intensive set of use cases is an emerging application for HPC systems that offers its own unique set of challenges. The evolution of data-intensive computational streams, and their interaction, will function as an important driver for a future exascale environment that encompasses elements of computing, data transfer, and data storage. Facilitating the evolution to this new environment will require more than just a discussion of computational requirements; the discussions must address how the entire computational fabric (e.g., networking, data movement and storage, and cyber security) needs to evolve to best meet science requirements. A common concern related to these plans is whether the current scientific computing and data infrastructure will be able to handle the impending demand for computational resources.

Discussion
Future ASCR HPC facilities will play a new role as scientists seek to leverage HPC machines for data-intensive applications. ASCR researchers will need a testbed to develop workflows for these data-intensive applications, design edge services that are the bridge between the domain scientist and the HPC facility, and provide insight into how facilities need to be provisioned in order to execute these applications.

Co-Scheduling Capability (6)

Description and Drivers
It is important that ASCR researchers have the ability to do "co-scheduling" — the ability to schedule research time on any combination of storage systems, file systems, computers, and networks that can be used concurrently. This type of capability allows researchers to test complicated multi-facility workflows. While it is currently possible to do this, it is not easy. Co-scheduling is typically done through a series of phone calls and emails with a given facility. There is no mechanism at any of the facilities to schedule an "ecosystem" as a complete test bench.

Discussion
As the computing ecosystem becomes increasingly complicated and scientists want to push the limits of what is possible with these large computational engines, it is important for researchers to be able to schedule an entire vertical slice of the facility. Data sets continue to reach unprecedented scales, and the ability to move large data sets not only from exascale facilities to analysis centers but also to exascale facilities to perform various reconstruction or analytic tasks requires new workflows, different types of capabilities, and a different approach and attitude at the large facilities. Tests at- or near-scale will be imperative to fully exercise these complicated systems and be ready to provide production capability for these new computing paradigms at HPC facilities.

Software Portability (7)

Description and Drivers
A common problem that both the compute- and data-intensive communities face is the possible proliferation of "swim lanes" in future computational architectures and the difficulty with writing portable code for these systems. Currently, and in the next generation of large ASCR systems ("pre-exascale"), there are only two types of computational architectures available: CPU/GPU (accelerated) and many-core (non-accelerated). While HPC users can imagine running codes on these two types of architectures — and even this capability is limited to only a few teams — data-intensive users have a much more difficult planning decision to make. Disruptive changes cannot be made often to a scientific software stack, and even then, only with considerable difficulty. This

means that future evolution of this software will very likely follow conservative trends, which for now, appear to lead down the many-core path.

Although we cannot predict the future of the exascale environment with precision, from what is currently known, this strategy would appear to make sense. The above argument suggests that a parallel effort in developing portable programming models for the exascale would be particularly beneficial for data-intensive applications. Such applications are not typically characterized by a few highly tuned kernels, but by a number of chained subprograms that may not be individually tuned for performance (nor is there usually an attempt to apply global optimization). Therefore, in this case, portability may not necessarily be accompanied by an unavoidable loss in performance, as is the case for the vast majority of HPC applications.

Discussion
Software portability gets to the heart of how efficient or practical is it for the various scientific communities to leverage these computational resources. There is no crystal ball to accurately predict how vendors are going to provision the next few generations of computational resources. Thus, developing portable software is imperative — especially considering that many of the domain sciences have software stacks that comprise millions of lines of code. Scientists do not have the skill or time to refactor the code as a new machine hits the market.

Data Movement Software Tools Support (8)

Description and Drivers
As science programs thirst for data increases, it is important to develop and maintain a suite of data movement tools and middleware. Providing an agile software adoption process for testing and deploying these tools is critical for timely progress. Researchers are constantly optimizing existing data-movement software tools, as well as developing next-generation data movement tools, for the emerging exascale computing environment. Both scientists and researchers benefit when these enhanced or new tools are pushed aggressively into production use. The scientists benefit through enhanced performance in data movement. The researchers benefit from practical use of their products and the feedback loop generated from that use.

Discussion
Given the importance of keeping the exascale machines fully utilized in order to take full advantage of their computational capability, it is critical to develop the computational models, tools, schedulers, etc., that would enable full use of each facility. For HPDC applications, highly efficient data movement into and out of HPC systems is a critical element in fully developing that computational capability. Agility at HPC sites in terms of deployment of new or enhanced data movement software will be necessary for HPDC applications to make the most efficient use of HPC resources. Both scientists and researchers would benefit greatly if this agile process was common across the different HPC centers. Such an environment would provide new and improved data movement tools for the scientist in a much more timely fashion than what is currently in place. In addition, the gap between ASCR-funded research development and its production use for science would be narrowed.

Opportunistic Computing Support (9)

Description and Drivers
Five percent of an exascale machine provides 50 petaflops of computing. While 5% sounds like a small number, in today's terms, 5% of the power of this new machine is substantially more than the power of today's NERSC+TITAN+MIRA facilities combined. It is a sizeable amount of computing, and thus the facilities should strive to keep these exascale machines at 100% utilization. There are many scientific applications in which a small percentage of the full machine capability is more than sufficient. Thus, these cycles need to be captured and fully exploited.

Discussion

Given the importance of keeping the exascale machines fully utilized in order to take advantage of their full computational capability, it is critical to develop the computational models, tools, schedulers, and so forth that would enable full use of the facilities.

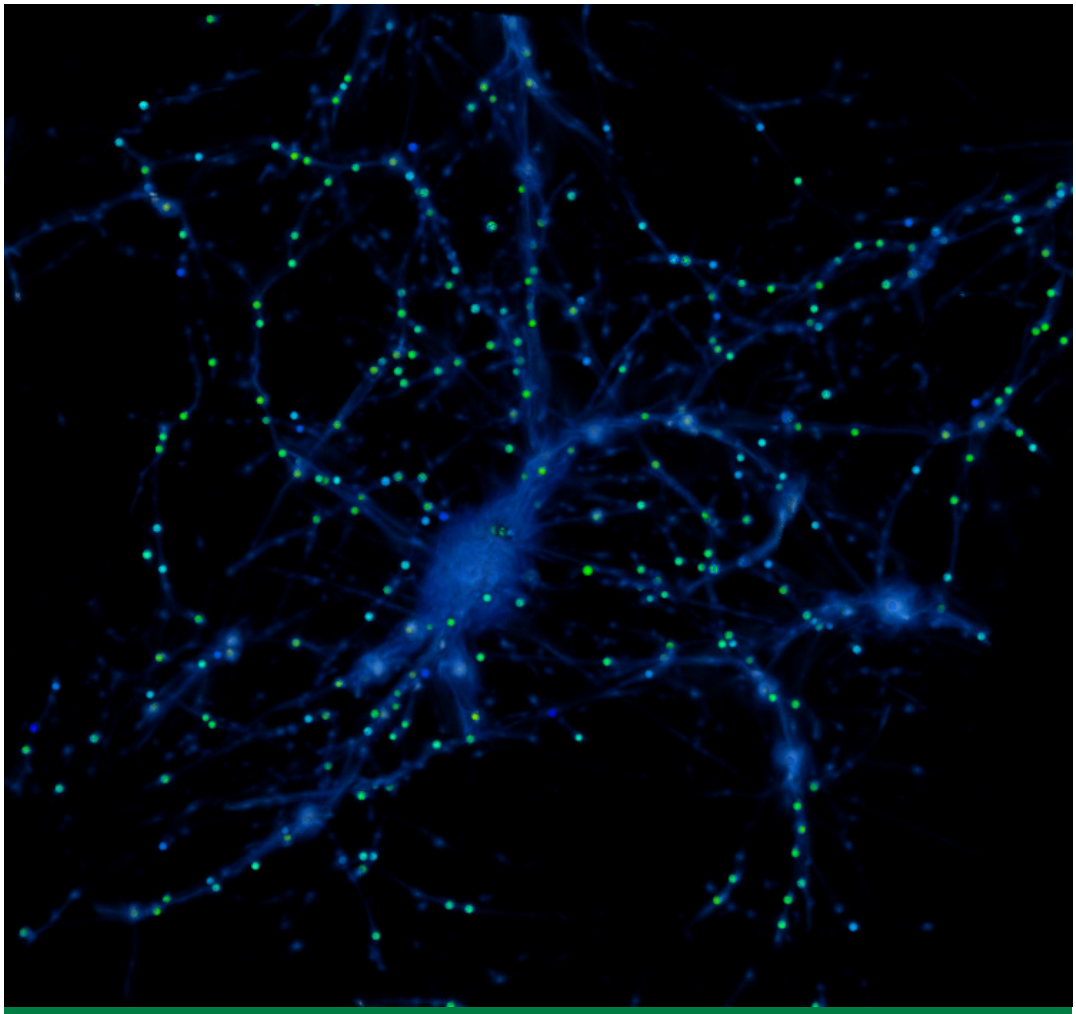


Figure 3-9. Valuable science can be lost because of the widening gap between our ability to compute and save data and the different scaling behavior of simulation and analyses. One solution is to perform analysis while the simulation is running and use in-transit analysis to optimize the partition of number of cores dedicated to simulation and analysis. By saving only the analysis results, rather than full-resolution simulation output, we can reduce the size of output by 1000x, access all simulation time steps, facilitate high-temporal analysis fidelity, and ensure optimal use of all compute cores. The image shows the use of *in situ* and in-transit halo findings in cosmological simulations; halos are regions of locally higher density shown as spheres (Image courtesy of Brian Friesen, Ann Almgren, Zarija Lukić, Gunther Weber, Dmitriy Morozov, Vincent Beckner, and Marcus Day, LBNL [Friesen et al. 2016]).

3.4 Software Development

The scientific community has broadly recognized the key role of software as a crosscutting technology that connects advances in applied mathematics, computer science, and domain sciences to enable long-term collaboration and scientific discovery (Rűde et al. 2016). However, urgent challenges in software development — especially issues in productivity, quality, and sustainability — must be addressed in order to fully meet the needs of extreme-scale science (Johansen et al. 2014; Heroux and Allen 2016); work on many of these issues requires partnerships between ASCR-supported researchers and DOE computing facilities.

3.4.1 Early Delivery Systems

In many respects, the software development needs of researchers are the same whether they are working on production systems or other system types. The majority of participants in this breakout also participated in the complementary breakout on Software Development on Production Systems (Section 3.4.2). We spent some time at the beginning discussing the definition of "early delivery" systems, and the fact that there was no additional breakout to cover "emerging" systems. We decided that our scope was essentially any system that would not be considered "production," and in our elicitation of requirements, we tried to focus on issues that arise for non-production systems, with the understanding that nearly all of the requirements identified in Section 3.4.2 would also be applicable here. In retrospect, some of the requirements discussed in this section (Table 3-10) would also be applicable to production systems. There are also some issues that came up in recent large-system procurements, but were exposed during the pre-production, non-recurring engineering (NRE) process, and so are included here.

Table 3-9 lists the research needs identified for software development in early delivery systems, the scientific drivers associated with those needs, and the potential for impact of each identified need (low, medium, high).

Table 3-9. Needs: Software Development on Early Delivery Systems

Number	Need	Scientific Drivers	Potential for Impact (L, M, H)
1	Access to testbeds with staffing and vendor engagement	Broad access and vendor software stabilization	H
2	Multi-party NDAs, including both labs and universities	Communication among researchers	M
3	Tools that give performance insights into new hardware features	Programmer productivity	H
4	Collaboration with vendors to ensure low-level software components required to construct the rest of the software stack needed by the community	Tool developer productivity	H
5	Collaboration with vendors to ensure that low-level software components are available under licensing terms suitable to support creating and distributing open source tools on top of them	Tool availability	H
6	Maximized availability of systems for development (e.g., by providing redundant systems)	Developer productivity	L
7	Access to performance tuning best practices used by the vendors in acceptance benchmarks	Programmer productivity	M

3.4.1.1 Needs Identified from Breakout Session

Access to Testbed and Prototype Platforms with Staffing and Vendor Engagement (1)

ASCR researchers need access as early as possible to the hardware and software environments of forthcoming large-scale production systems to help prepare their software products, and implicitly the applications that rely on them, for the new environments. ASCR researchers are also interested in obtaining access to a wide range of other systems (different architectures or other distinctive features), at various states of maturity, in order to understand how the different environments might impact their work.

Past experience suggests that researchers get the best value when time commitment expectations are realistically established (working on testbeds is time consuming due to instabilities and uncertain sources of error, etc.) and when testbed systems are supported by knowledgeable staff members who are invested in deeply understanding the capabilities and limitations of the systems through insightful interactions with both users and the system vendor. Historically, researchers often find that obtaining access to testbed systems is a haphazard process. Testbed systems are often hosted in research organizations rather than facilities, or in academic research groups that may have particular vendor relationships. Finding and obtaining access to such systems tends to be strongly dependent on hearing about their availability, and knowing people who are willing and able to provide access. Further, support for these systems varies widely. Testbed systems are often not stable or full-featured. Porting, debugging, and performance tuning can be very labor intensive; the skills required to complete these tasks may not be in the ASCR research staff skill sets. Such tasks can also require dedicated hours, which makes it difficult to obtain staff commitments. Because of this inherent complexity, ASCR research staff should be made aware of these challenges, and their research expectations should reflect the time commitments involved in working on testbeds. This is also why dedicated testbed staff are extremely important. Participants in the breakout session expressed a strong desire to make the process of identifying and obtaining access to testbed systems(in addition to those systems provided with appropriate levels of support) more straightforward to maximize their value to the research programs.

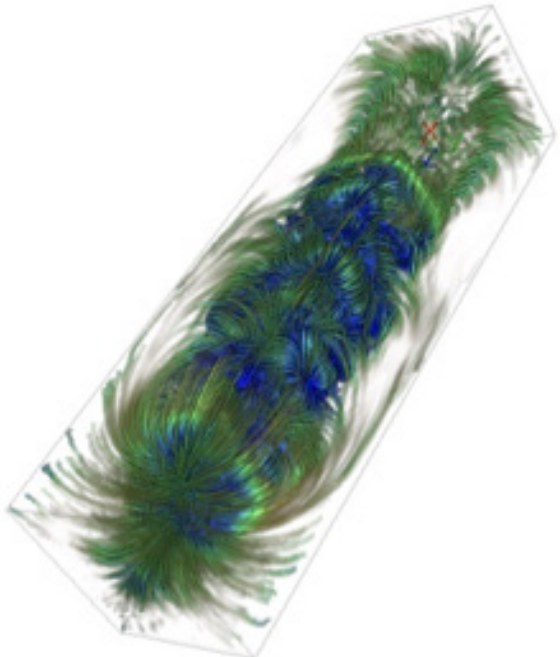


Figure 3-10. As scientists anticipate the benefits of extreme-scale computing, roadblocks — including the disparity between computing and storing information and the gap between stored information and understanding — threaten to impede their progress. The Information-Theoretic Framework for Enabling Extreme-Scale Science Discovery project addresses two difficulties faced by computational scientists: deciding what data are the most essential for analysis and transforming these data into meaningful visual representations. ASCR research will (1) quantify the amount of information in data using information-theoretic approaches, allowing decisions about how data should be stored and analyzed, and (2) construct a data analysis and visualization framework based on information theory. This image was generated using the information-theoretic streamline placement algorithm (Image courtesy of Han-Wei Shen, Ohio State University; Rob Ross and Tom Peterka, Argonne; and Yi-Jen Chiang, Polytechnic Institute of New York University).

The ASCR research programs would benefit from easier and better-supported access to a broader variety of systems in order to ensure that research products worked as well as possible across the spectrum and to identify limitations of the research approach. ASCR facilities benefit, in particular, from precursors to production systems, and from better, earlier platform support for ASCR research products that scientific applications depend upon. Another significant benefit is that broader, earlier usage tends to help shake out issues in the vendor-provided software stack, which allows them to be addressed sooner.

Multi-Party NDAs, Including Both Laboratories and Universities (2)

Access to software and computer systems prior to their general availability to the public, and even discussions of them, often requires the execution of NDAs. Typically, NDAs are between a vendor and a single institution. Researchers within the institution are covered by the NDA and may talk to each other, as well as the vendor, without hindrance. However, researchers at different institutions are legally not allowed to talk to each other, even when both institutions have an NDA in place with the same vendor. In the increasingly collaborative environment of DOE HPC, the inability to share information, experiences, and code across institutions can limit our ability to work together for a common purpose.

In recent years, the DOE laboratories have arranged multi-party NDAs on a number of topics. As the name suggests, multi-party NDAs provide a larger umbrella for the sharing of information than the usual two-party NDA. Multi-party NDAs were put in place some years ago to facilitate discussions of exascale systems across the seven labs then participating in the discussions. More recently, multi-party NDAs have been established for the CORAL (Collaboration of Oak Ridge, Argonne, and Livermore) procurements. Although we can cite these specific examples, and others probably exist, it is not yet standard procedure to establish multi-party NDAs when pre-GA hardware and software are brought into the laboratories.

Further, when multi-party NDAs have been established, they typically cover only the DOE national laboratories. In practice, there are many cases in which it would be useful, or even necessary, to bring non-laboratory personnel into the NDA umbrella. Such useful personnel include teams involved in application readiness activities for a forthcoming system and tool developers from the broader ASCR research community. Such individuals are as likely to be employed by universities as by the labs. Once again, there are specific examples where multi-party NDAs have been executed, but this is not yet a standard procedure.

Therefore, we believe that it would benefit both ASCR researchers and facilities to operate under the expectation that many NDAs will need to be inclusive, and that they should work with their local legal counsel and vendors to facilitate and accelerate the necessary processes. This should apply to long-running procurement processes that will *eventually* lead to the deployment of non-NDA production systems, as well as to non-production systems.

Tools That Give Performance Insights into New Hardware Features (3)

One of the key reasons ASCR researchers are interested in access to non-production systems is to help them understand how the novel features of such systems impact their work and the approaches and tools they are using, or, conversely, how well their approaches and tools support the novel system features. However, it is often challenging to gain the insights necessary to achieve this understanding because adequate tools are not available to expose the workings of the new features (usually from a performance perspective). To give just one example, as memory systems grow deeper and more complex, tools are needed to expose details of memory traffic between levels, between coherency domains, across busses, and so forth.

As the hardware features mature and go into production, tooling typically catches up. However, it is at the earliest stages where access to such information would be most useful and offer the greatest productivity benefits. Early in the process, the features themselves (and the tooling the vendor might provide to support them) are not likely to be ready for direct incorporation into mainstream tools; most users would be satisfied initially with rough-and-ready vendor-provided tools as long as these tools can provide the necessary basic information. This would also facilitate a broader conversation among the vendor, facility staff, and users regarding how to evolve both the hardware and the tooling to make it most effective and useful.

As such, we recommend that the facilities engage strongly with vendors as early as possible about providing the information and tools necessary to gain insights into novel system features that are not covered by commodity tools. From the tooling standpoint, vendors may find that many ASCR researchers would be eager to engage with them on supporting novel features in their commodity tools if the vendors are willing to provide access to the necessary information. These concerns also typically apply to the NRE process that leads to the deployment of large production systems.

Vendor-Provided Software Components to Support Community Needs (4)

For various reasons, vendors often provide proprietary software components to help users leverage a system's hardware capabilities. Although such components are often critical to the success of the system, they can also pose challenges for integrating system support into the tools widely used within the community. If the proprietary components do not expose reasonable interfaces at the appropriate levels, they may hamper the community's development of software. This issue has been particularly problematic for tools that support performance measurement, analysis, and attribution, but it is not limited to them. In addition, as researchers try to develop tools to support emerging programming models, the capabilities of vendor-proprietary components are often designed with a particular use case in mind and may not adequately support new directions.

We request that through discussions with vendors, and procurement processes, facilities stress the importance of their strong engagement in and support of the larger tools ecosystem, particularly when it comes to proprietary software components.

Compatibility of Licensing Arrangements for Vendor Software Components with Community Needs (5)

Related to, but distinct from, the previous issue is the question of how the proprietary software components are licensed. Most community tools are distributed under open-source licenses. This does not necessarily prevent the use or calling of proprietary components at lower levels. However, the terms under which the proprietary components are licensed can effectively block the ability of widely used open-source tools to support a platform that relies on such components. In the CORAL procurements, for example, tool developers have encountered a number of issues. One component is available only under an NDA. Another component is available, but it cannot be redistributed in full. Both situations hamper the use of these components in an open-source tool. For technical reasons, it is not always feasible for the build of the open-source tool to simply reference a completely separate distribution of the proprietary component. Furthermore, when components are under an NDA, the tool developer and the user of the tool are legally prohibited from talking to each other unless the NDA is multi-party, covering both of their employers. In addition to cases like this, there are lesser issues of concern for tool developers and users who wish to avoid possible legal complications. For example, cases have also been encountered in which a component containing a library, header files, and a license also requires a "click-wrap" license agreement to be acknowledged before the component can be downloaded. Unfortunately, the two licenses are not merely different, but differ in important ways that affect their redistribution. Experience has shown that interacting with vendors to adjust licensing after the fact can drag on for months, wasting researcher time, reducing the availability of tools, and adversely impacting productivity.

We request that the facilities help with these issues by incorporating appropriate requirements for licensing of proprietary software components into their procurements and vendor contracts.

Maximized System Availability for Development Purposes (6)

Downtime for computing resources is frustrating for all users. For production usage involving large batch jobs, which often take some time to get through the queues, the net impact on productivity may not be severe. For users doing interactive development and debugging, however, having to work around system outages can have a more significant impact on productivity. The same is probably true for at least some analysis and visualization workloads, depending on the extent to which they require interactive attention. Longer periods of downtime are particularly common early in the lifecycle of production systems, and in early access and emerging systems. This downtime can hamper early progress on porting and tuning of applications at times when such work is most valuable.

The group expressed widespread interest in having higher levels of availability for computing resources, at least for development and similar activities. One way this could be achieved is by having smaller "companion" systems that match the architecture of primary systems. Such systems could be managed to ensure that they are available to users during scheduled outages for the primary systems (and vice versa). This approach was considered to be desirable not only for the large production systems, but also for the smaller early-access and emerging systems that might be made available.

Documentation and Sharing of Performance Tuning Experiences (7)

A key concern for the developers and users of applications and libraries on HPC systems is their performance because of the direct link between maximizing performance and maximizing scientific productivity. Understanding the achievable levels of performance and how they can be obtained can consume significant amounts of time, particularly as machines become more and more specialized. Maximizing performance helps maximize overall scientific productivity. In most cases, having access to information about the experiences others have had with the system can significantly improve the productivity of the developer who is trying to maximize the performance of an application or a library.

For most large systems, at least, experiences start early; the vendors work to tune the acceptance benchmarks, and the facilities make applications ready for their early science programs. The same level of effort may not be invested initially for smaller systems, but any information can be helpful. Our group expressed a desire for a wide variety of artifacts from such activities to be captured and made available to users to assist them with their own tuning efforts. The artifacts of interest included the following:

- Written documentation of processes and experiences, including a history of the performance tuning process;
- Code, makefiles, and associated scripts associated with the benchmark;
- Reproducibility information, including versions of key components of the software stack, compiler flags, etc.;
- Raw output of both the benchmark and the performance measurements (e.g., performance application programming interface [PAPI] counters);
- If appropriate, known parameters for which the performance optimizations are valid (e.g., problem sizes, software stack dependencies); and
- Metrics for success of performance improvement (e.g., node performance, scalability, memory throughput, and I/O performance).

It was also noted that negative experiences (what not to do) can be as useful as positive ones and that a historical perspective of how the tuning parameters change as the machine parameters change would also be useful.

Although it was not discussed in the breakout session, an obvious corollary is that encouraging and facilitating the sharing of code tuning experiences on an ongoing basis would be useful to the community.

3.4.1.2 Relationships among Needs and Conclusions

The effectiveness of documentation and the sharing of performance tuning experiences would be enhanced if the output of performance tools could be provided in a form that is compatible with whatever archiving methodology is adopted.

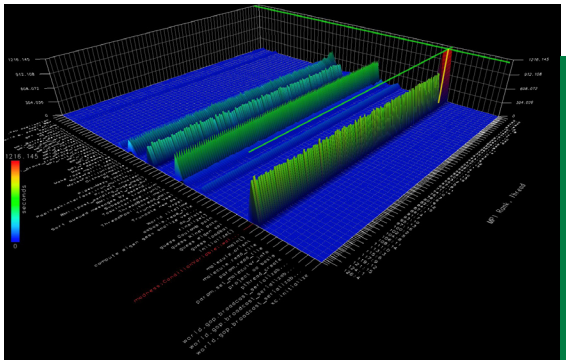


Figure 3-11. TAU's ParaProf 3D browser reports the load imbalance in the profile and the time spent waiting in a condition variable on a subset of MPI ranks in a hybrid MPI and OpenMP program. The image shows the shape of the profile across multiple ranks and threads for the MADNESS application on an IBM BlueGene/Q system in TAU's ParaProf 3D browser, where we can observe the excessive time spent in a synchronization operation (Image courtesy of Sameer Shende, University of Oregon).

3.4.2 Production Systems

Session participants included a broad range of ASCR-supported researchers who develop software throughout various levels of the stack (ranging from low-level system software through intermediate layers of numerical packages and higher-level scientific application codes), as well as tools for profiling and analysis. The group identified two overarching themes in the needs of ASCR facilities that could support software development on production systems:

1. A holistic approach to computing systems provided by DOE facilities, including access to a variety of systems at each facility (to support both small-scale testing and at-scale debugging and tuning) and coordination across facilities; and
2. A cohesive, sustainable cross-facility software ecosystem that enables developer productivity and promotes encapsulation of ASCR research for use by scientific application teams.

Table 3-10 lists the research needs identified for software development in production systems, the scientific drivers associated with those needs, and the potential for impact of each identified need (low, medium, high). The table and the sections that follow elaborate on these themes from the perspective of seven fundamental needs. Most of these needs also apply to early delivery systems. Likewise, many of the needs discussed in Section 3.4.1 (Software Development on Early Delivery Systems) also apply to production systems, although the relative priorities for various topics shift depending on machine and software lifecycles. Several of the needs related to software deployment are only briefly introduced here, because they are discussed in more detail in Section 3.6 (Software Deployment and Support).

Table 3-10. Needs: Software Development on Production Systems

Number	Need	Scientific Drivers	Potential for Impact (L, M, H)
1	**Access to a variety of systems:** Fast compilation nodes, small-scale test and development systems, large-scale production systems for testing at scale	Software developer productivity	H
2	**Software ecosystem:** Strategy/path to production for ASCR research activities (on-ramp and lifecycle)	Broad application space using ASCR-research software projects	H
3	**Coordination across facilities:** Federated ID, common job launch, managed software stack (common tools/libraries/compilers/etc.)	Software developer productivity, application user productivity	H
4	**Automated software testing on HPC facilities:** Unit, verification, regression, continuous integration, performance, etc.	Software developer productivity, application developer productivity, verification of correctness	H
5	**Debugging capabilities:** Variety of debugging tools, debugging at scale	Software developer productivity	M
6	**Containers for portability from laptops to computing centers:** Common software environment to improve workflow	Software portability for both developers and application users	H
7	**Requirements for relevant operational data from the facilities:** Facility monitoring data, usage statistics to support software tool development	Informs impact of ASCR-research tools	M

3.4.2.1 Needs Identified from Breakout Session

Access to a Variety of Systems (1)

Description and Drivers

In contrast to domain scientists, who need computing resources for large-scale production runs (i.e., access to many nodes for many hours), ASCR-funded researchers need access to computing systems in a variety of modes that explicitly support the development of software that encapsulates cutting-edge advances in applied mathematics and CS, which in turn supports domain science applications. Because DOE facility policies traditionally tend to favor large-scale production runs, ASCR researchers often encounter difficulties in obtaining consistent access to systems for software development. ASCR researchers need DOE computing facilities to balance allocation of computing resources to support both production runs and software development, including software installation, testing, debugging, performance analysis, and enhancement. Development nodes should be expected to be "more available" (short wait time) in comparison to production nodes.

Developers often need interactive access to hardware (as opposed to batch access for production runs). Programmers are more productive when they can debug and profile in real time rather than wait for machine availability. Batch access typically is not sufficiently fast for "cognitive turnaround" needed in many development activities, such as debugging.

Achieving robust, efficient, and scalable performance is more difficult than ever before due to the complexities of emerging architectures; ensuring good performance at scale requires resources for debugging and performance tuning at scale. Short-term access to many nodes (as opposed to long-term access for production runs) is a very different (but important) usage model. Drivers include debugging (may need many nodes for bugs that appear only at scale), profiling, performance optimization, and scalability testing.

In addition, developers may need special privileges on machines for testing different machine configurations, such as setting cache/memory mode, and other changes that require root access or kernel changes.

Discussion

These needs primarily span the later stages of software development (during performance analysis and optimization); earlier phases of development are typically done on laptops and small clusters before transitioning to DOE facilities. An important consideration is that ASCR researchers often need to debug software that has already been deployed (and is already supporting application users) at DOE facilities and elsewhere.

We advocate at least two approaches to address these needs: (1) provide developer access to small clusters that match the environment of production machines, and (2) provide developer access to the full machine for scalability testing. Because these needs for software development span all scientific research areas and apply to early delivery systems, we believe that addressing these issues will increase programmer and research productivity for both ASCR investigators and domain scientists.

We advocate at least two approaches to address these needs: (1) provide developer access to small clusters that match the environment of production machines, and (2) provide developer access to the full machine for scalability testing.

Software Ecosystem (2)

Description and Drivers

ASCR research supports science programs across DOE and has the potential to provide a broad range of new math and CS functionality needed for next-generation science, as well as to improve the performance, programmability, reliability, and power efficiency of production applications. However, the impact of research done in ASCR is limited because ASCR's funding model and reward structure do not scale to the large user base of ASCR facilities. Many ASCR research products never see use in real applications because of lack of interest or awareness from application developers, lack of clearly defined guidelines for deploying and supporting software at LCFs, and lack of funding for researchers to make their tools usable in production applications. ASCR researchers need a clear path for transitioning research software to production usage, as well as clear models for maintaining and extending software products that are valuable to application codes. Without these resources, it will be impossible to build a sustainable exascale software ecosystem for use by the broader community.

Discussion

ASCR research software is currently fragmented — many research artifacts are produced by separate groups, but few are used widely in production at facilities. The basic software ecosystem on HPC systems today has remained unchanged for many years. For ASCR software to broadly impact DOE science discoveries, it must be used in production codes. However, application teams are wary of relying on research software. Many have had bad experiences in the past when the software on which they relied ceased to work; some teams opt to implement their own versions of complex techniques and algorithms to avoid relying on external groups. With exascale machines arriving soon, and with the increased architectural diversity they are expected to bring, it will be critical for application developers to leverage research in advanced algorithms, performance portability, tuning, resilience, and power efficiency, to make the best use of future machines. At exascale, application teams will not have the resources to do all algorithmic development and tuning themselves. The working group identified two areas where the facilities could help to build a sustainable software ecosystem:

1. Identify key research products for facility users and help to sustain and grow their development; and
2. Provide a clear set of requirements and processes for software deployment on production systems.

These areas, and their associated drivers, are discussed below.

Identify Key Research Products for Facility Users and Help to Sustain and Grow their Development

Description and Drivers

In the startup world, companies budget for growth, and the success of a product is measured in terms of the number of users it attracts. Within ASCR, the incentive structure is heavily biased toward paper publications, and researchers have little incentive to develop tools for widespread use. Indeed, maintenance is 80% of software development, but makes up only a tiny fraction of ASCR funding, and research project budgets are typically flat from year to year. For application developers to adopt software, they need to know that it is stable, reliable, and has long-term support.

Discussion

Software sustainability is an issue that extends beyond the facilities, and productizing research is a very resource-intensive process. The working group recommends that a larger discussion be initiated to establish a viable path for software as it moves from research to production. ASCR and the facilities can help by aggressively lowering barriers to good software practices and by helping research teams transition to production roles. Facilities should help researchers make connections with production application teams from the beginning of their projects. ASCR and the facilities should also help by identifying products with growing adoption and by allocating funding, staff, and hosted collaboration tools so that these projects can easily establish release processes, issue tracking, automated regression testing, version control, and other best practices. Funding and staff allocated to such support efforts should be commensurate with uptake by application developers, so that projects can grow with demand. That is, ASCR and the facilities should plan for success, growth, and long-term sustainability from the start. Facilities should view themselves as supporting a true developer ecosystem in addition to HPC users.

Provide a Clear Set of Requirements and Processes for Software Deployment on Production Systems

Description and Drivers

For research products to be taken up by application developers, the developers must be made aware of the research projects, and the research products must be made available for easy use. Currently, the majority of publicly installed software on facility systems is installed by the facilities, and the procedures through which users can deploy their software and advertise it to other facility users are inconsistent and poorly documented. Researchers have expressed frustration at not being able to find ways to reach potential users at these sites.

Discussion

To remedy this situation, facilities should clearly document requirements for deploying software for public use on HPC systems, and they should provide mechanisms that allow users to advertise their research tools. ASCR researchers should be encouraged to deploy their software widely from the start of application projects and to seek out application users with help from the facilities. Lowering the barriers for software deployment will allow researchers to more easily update and support software for facilities. These topics, as well as automation issues associated with deployment, are covered in more detail in Section 3.6.

Coordinate across Facilities (3)

Description and Drivers

Access to computing systems and installation of research software on these systems are currently handled independently (and typically differently) by each DOE computing facility. In addition, policies for access have traditionally been formulated primarily to serve the needs of science teams that are conducting simulation campaigns. For example, an approach frequently used by developers

of numerical libraries is to apply for access to a specific facility for purposes of algorithmic research, testing, and performance tuning. However, access to systems at one DOE computing facility does not extend to other DOE computing facilities; in addition, substantial paperwork is required to re-apply for access during subsequent allocation cycles. Consequently, facility policies and procedures do not fully address the needs of researchers who develop reusable software that is employed by applications teams on systems at multiple facilities. Unlike science teams, who may naturally focus on just one machine at just one facility at a given time, and who can justify machine usage by the results of simulation campaigns, developers of software packages and CS tools need sustained access to all relevant computing systems at all ASCR computing facilities in order to fully debug and test software so that it is robust, performs well, and is ready for use by applications teams wherever they work.

Additional challenges arise because of the different processes and infrastructure each facility uses. *Coordination of access and policies across facilities*, including a holistic perspective on machines as a coordinated set of systems, would help to simplify work for ASCR-funded researchers. Session participants expressed the desire for a single federated login name and one-time password token that are valid across all facilities. Also desired are compilers installed in a consistent manner across facilities, common environment variables across facilities (e.g., $SCRATCH_HOME, $ARCHIVE_HOME), common queue names, common commands to view allocations and disk quota usage, and common job launchers. Researchers also want tools to be installed and accessible using common environment variables and modules, including portable tools for performance analysis, testing, and debugging.

Discussion
Steps toward coordinated and sustained access to computing facilities, as well as common environments across systems, would lead to improved productivity and tool usage for both software developers and applications users. A possible metric to evaluate the impact of supporting such coordinated access is the time required to transition software between sites.

Automated Software Testing on HPC Facilities (4)

Description and Drivers
Regular and extensive testing helps to ensure the correctness of scientific software, which tends to change frequently in order to address research needs (e.g., refactoring for better performance on emerging architectures and the incorporation of new functionality as needed by new science frontiers) (Bartlett et al. 2016). It is imperative that code changes do not inadvertently introduce new errors or reintroduce old errors, particularly in ASCR research software that supports a broad range of applications teams (Bartlett et al. 2016).

While software testing is time-consuming and challenging at any scale and in any computing environment, testing is especially difficult across DOE computing facilities due to the need to maintain portability across a wide variety of (ever-changing) systems and compilers. Each software package has a broad test space to adequately cover its functionality, and this space is multiplied across different platforms and their corresponding software stacks. Difficulties are compounded because computing resources at facilities are limited, while software researchers need frequent testing access and a relatively quick turnaround time. Two particularly difficult tasks are testing system software and handing performance variability (for performance testing).

Although automated testing is well recognized as a best practice, DOE facility policies do not allow this. ASCR-funded researchers need support for automated software testing on all systems at all computing facilities, including unit, regression, verification, integration, and performance testing. As HPC software packages have become increasingly complex, with multiple developers, support is also needed for continuous integration-style testing (to identify errors as soon as possible after

they are introduced) and result aggregation (to help simplify the examination and understanding of test results).

Discussion
DOE computing facilities could consider trying to consolidate testing infrastructure in order to provide better and more consistent testing support across facilities and systems. A metric to consider is the time developers of a software package require to establish automated testing capabilities on a new system.

Debugging Capabilities: Variety of Tools, Debugging at Scale (5)

Description and Drivers
Debugging — the process of isolating and correcting errors or abnormalities in code to allow proper program operation — is part of the software testing process and is an integral aspect of the entire software development lifecycle. A variety of debugging tools are available on conventional systems to help identify and fix code errors, but debugging in parallel is very difficult due to the complexities of parallel programming. Debugging at DOE computing facilities is even more difficult because of challenges with machine access, limited parallel debugger functionality at scale, and the growing complexity of both hardware and software. More types of parallelism, deeper memory hierarchies, and more complex data structures contribute to numerous types of possible errors, including race conditions, memory leaks and access errors, implementation errors, degraded performance, and system errors or faults. While as a community we should be able to do better than printing information about variables (e.g., using 'print f' statements), this low-tech approach is the most common way of tracking down errors at scale. ASCR researchers need access to a variety of tools that support debugging at scale, across all systems at all computing facilities, for both static and dynamic analysis of code execution, so that we can identify programming and system errors rapidly and ensure robustness and correctness of software. We also need to capture the system state during execution to accelerate debugging.

Discussion
The opportunity cost of time spent debugging for users of DOE computing facilities is huge. The more effective the tools we have, the more effective we can be (and research and developer staff time is very expensive).

Containers for Portability from Laptops to Computing Centers, Common Software Environment to Improve Workflow (6)

Description and Drivers
ASCR researchers need (1) improvement in portability across the many systems our software uses, (2) pre-built software environments that reduce compilation issues and provide rapid access to many software capabilities, and (3) workflow improvements that are compatible with the broader computing ecosystem. ASCR researchers face these problems in their role as infrastructure providers for scientific applications that often use a variety of third-party software.

Container technologies have emerged as a promising technology to address these needs. Containers allow an application, its dependencies, and their associated run environment to be packaged into a binary image. Investing in container technologies can ease integration, help with portability and regressions, and encourage increased adoption of component-based application development.

Discussion
Container technologies with proven performance and portability are being adopted rapidly in the broader software ecosystem. Containers promise to revolutionize workflows by simplifying the use of many third-party software components, improving portability, and providing a common software

environment for developers and users. Standardizing on containers would also prepare LCF efforts to blend more easily with those of data sciences communities, where these technologies are already being deployed.

Challenges with containers center on the unique needs for DOE computing facilities relative to the mainstream computing community. DOE facilities need to address security issues, and DOE facility users need to access specialized LCF hardware. Industry standards such as Docker do not support these capabilities. Efforts such as Shifter should continue in order to satisfy these additional HPC requirements.

Requirements for Relevant Operational Data from the Facilities (Facility Monitoring Data, Usage Statistics) to Support Software Tool Development (7)

Description and Drivers

In order to effectively focus efforts for software R&D and to facilitate cross-layer performance analysis at scale, ASCR researchers need access to system-wide monitoring and logging capabilities. For example, we need to understand shared resource activity (I/O, parallel file system, and network), power/energy usage, and failure rates for hardware and software. The relevance of various types of data will evolve over time (e.g., data movement, fault tolerance, and power management will become increasingly important); this information will help to ensure that research addresses the most urgent issues. ASCR researchers also need detailed information about tools, compilers, languages, and libraries used on all systems at all facilities. Section 3.7 (Operational Data and Policies) provides further discussion.

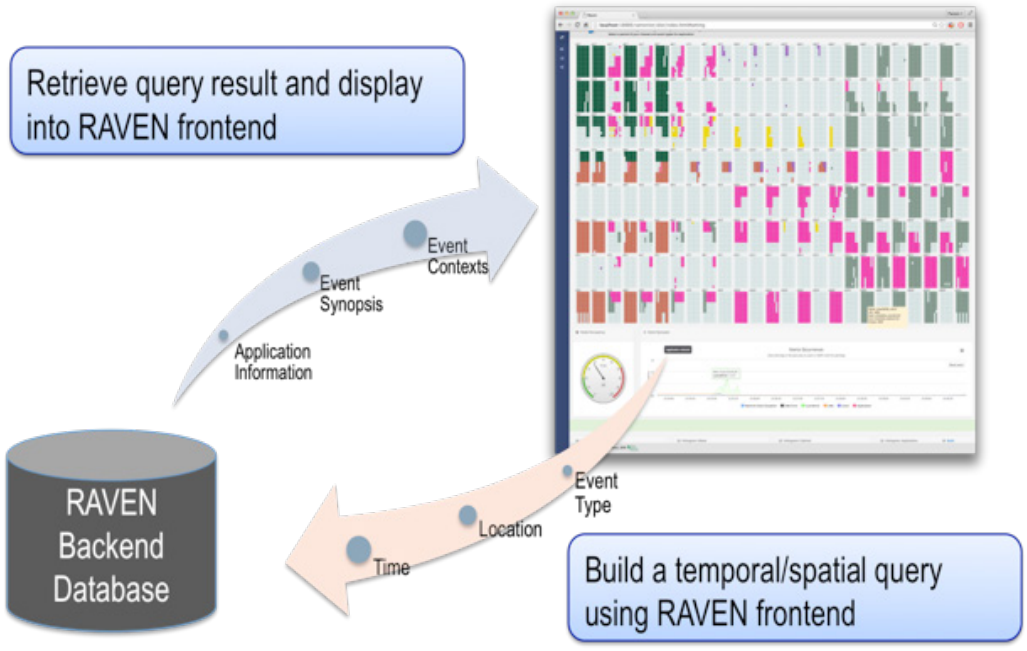


Figure 3-12. RAVEN consists of a back-end database and a front-end (the graphical user interface). The RAVEN back-end database is designed for efficient context-driven retrieval. Here a context is time, location, user application, event type, or any combination of these. The RAVEN front-end is running in a Web browser and is designed to facilitate the creation of context-driven queries by the user and to render the query results on a physical layout of the analyzed computer system for visual inspection of patterns (Image courtesy of Christian Engelmann and Byung-Hoon Park, Oak Ridge National Laboratory).

3.4.2.2 Relationships among Needs and Conclusions

Progress is urgently needed in these areas to support cutting-edge ASCR research and software development, which in turn provide cross-cutting capabilities that enable science teams to fully leverage LCF computing power for scientific discovery. Additional community perspectives on scientific software development challenges are provided in Johansen et al. (2014) and Heroux and Allen (2016). Addressing some of these needs will require partnerships between the computing facilities and ASCR. Dialogue and collaboration among the leadership and members of both communities are imperative in order to create a cohesive, sustainable, cross-facility software ecosystem that enables developer productivity and promotes sustained encapsulation of ASCR research for use by application research teams.

3.5 Systems Software

3.5.1 Emerging Systems

Table 3-11 lists the research needs identified for system software development on emerging systems, the scientific drivers associated with those needs, and the potential for impact of each identified need (low, medium, high).

We identified access to systems and system information as our primary need, followed by a need for increased support on emerging systems. Both of these key needs are seen as barriers to research on emerging systems today. We also identified a need to define new metrics for success in evaluation of emerging systems, because traditional metrics are typically not applicable and prevent researchers from conveying the value of their work to ASCR. We also noted that some emerging systems require integration with data sources for adequate evaluation, and without the data sources, the evaluations can be misleading. Finally, we found a need for integration of research software at different levels of the stack. This is currently an issue primarily because of the inability to share intellectual property among research groups.

Table 3-11. Needs: Systems Software Development on Emerging Systems

Number	Need	Scientific Drivers	Impact (L, M, H)
1	Access to the technology and information: who gets it, how do they get it, classes of access.	Enable fundamental understanding of architecture	H
2	Metrics for success: to inform future system investments, usefulness for science. Fast failure okay.	Assessment of emerging systems impact/usefulness for scientific community.	M
3	Support for research: from facilities or a PI model? Hackathon/dungeon model. Lease model.	Productivity of research efforts	H
4	System integration: internal data sources and external sensors/data sources.	Converged systems research	H/M
5	Software ecosystem: integration of software layers that exist (IP facilitation).	Collaborative research, facilitate transfer to production/identify gaps	M

3.5.1.1 Needs Identified from Breakout Session

Access to the Technology and Information (1)

Description and Drivers

Access to emerging systems is critical for enabling fundamental understanding of these new architectures. Here, the term *access* has several different, but equally important, meanings to consider — access to system information, physical or login access to the systems, and levels of privileged access to systems and information.

Access to system information about emerging systems has several aspects, as described below.

1. Primarily, researchers need to be able to access information about what systems are available or will be available for research purposes. Access to information essentially determines who will be able to acquire or make use of emerging systems or devices. Many times, information regarding these systems is limited for intellectual property reasons. While that is a necessary precaution for vendors to take, this makes access to information about emerging systems especially difficult for university researchers to obtain. Researchers at national labs generally have broad NDAs in place; this is not generally true for university researchers, however. There should be a way for researchers to apply for general NDA access to information, or they

could be given access to high-level information and then apply for more detailed information as part of an access or acquisition proposal process. Reduced access to information about systems means that fewer investigators are able to perform their research and evaluations on the systems, which leads to a reduced understanding of the broader applicability of emerging systems.

2. Researchers need to have access to information about emerging systems, including documentation and vendors' long-term plans for a particular system or technology. Getting access to up-to-date and complete documentation for emerging systems is typically extremely challenging. While it is understood that emerging systems are prototypes, vendors should do their best to provide documentation that is as complete as possible, and to provide a mechanism for dispersing updates to the documentation to interested parties. Currently, this tends to be an *ad hoc* process where users pass information to each other, or they must explicitly ask for updated information from facility staff or vendors. In addition, researchers need to know vendors' long-term plans for emerging systems. For example, for a storage device, a vendor could provide a high-level description of how the vendor expects this device to be incorporated into a larger system (e.g., node attached or perhaps as a shared storage resource). Without having this sort of idea of the roadmap for a particular technology, it is impossible for researchers to know how to best evaluate the technology to provide feedback on how well suited the technology is for the intended use. Again, this sort of information is usually protected as intellectual property; thus, there needs to be a way for a broad range of researchers to access it so that emerging systems are fully evaluated.

Physical or login access to emerging systems by a broad spectrum of experts is critical for enabling fundamental understanding of these systems. However, generally speaking, access to emerging systems is limited to researchers at the facility that has acquired the technology. This limitation of access can be due to lack of information about the system, as discussed above, or due to facility policies that block access to the technology from external researchers. Facilities tend to limit access to emerging technologies due to the special needs of these systems. Emerging systems require special handling due to intellectual property requirements, thus access to them is limited. In addition, emerging systems tend to lack full or up-to-date documentation and software, which makes them inherently difficult for facility staff to support. Thus, limiting access to known "friendly" researchers at the site reduces the support burden on facility staff. In addition, the availability of an emerging system at a facility tends to be short-lived, so there is an additional burden on facility staff to enable access to a broad range of internal and external researchers for a device or system that may only exist for a short time. We need to broaden the availability of access to emerging systems by enabling sharing across laboratories and university partners when possible. We need to provide resources to facility staff to be able to better support emerging systems to a larger number of users potentially from other sites.

Depending on the kind of system software research being performed, different levels of privileged access to systems and information need to be available for users of emerging systems. For example, researchers developing OS and other low-level software need to be able to have root access privileges to perform their work. However, researchers who develop system software that runs at the user level would not require root access privileges in most cases. The level of access privilege and type of system software being developed determines the kind of system documentation and information required by researchers. Researchers with root privilege developing low-level system software need access to detailed, low-level documentation and technology roadmap information that could potentially have higher availability restrictions due to intellectual property concerns. In contrast, researchers developing user-level system software can have varying needs with respect to documentation and information. While some of these researchers may only require basic

documentation for their work, other researchers working on user-level system software may require the same level of information as researchers with root privilege access. For example, performance tool developers may need low-level information about hardware performance counters and their meaning to complete their research. Thus, vendors and facilities need to support different classes of users on emerging systems, both in terms of privileged access, and in terms of access to different levels of information, and researchers should be able to apply for access levels based on their needs.

Discussion
Broad access to emerging systems can enable a fundamental understanding of architectures. However, due to the special nature of emerging systems, there are many barriers to accessing these systems. In general, researchers lack access in several ways. First, researchers lack access to information about the availability of emerging systems, as well as long-term roadmap information about the systems. Second, researchers lack physical and login access to emerging systems at different facilities because of the difficulties associated with intellectual property management and the special support needs of emerging systems borne by facility staff. Finally, researchers lack access privileges at the different levels required by systems software development.

Without early and relatively easy access to emerging systems, system software will not be mature enough to fully evaluate emerging systems for eventual production system acquisition. Researchers who work at all levels of system software layers/levels need broad access to emerging systems to ensure interoperability evaluation. We need to facilitate access mechanisms by providing facility staff the resources to support a larger number of researchers having access from different sites.

Metrics for Success (2)

Description and Drivers
Although there are established metrics for deciding the success of production systems that focus on science outcomes, there is a lack of metrics to inform decisions on future systems' usefulness for science. We need to develop metrics for time-to-insight so that we can "fail fast" on speculative research on emerging systems, which will minimize effort toward technologies that are not a fit for the needs of ASCR scientists. There is also a need for softer metrics for emerging systems evaluation, such as outreach efforts (e.g., user workshops), rather than traditional metrics such as system utilization or application performance. Finally, researchers cannot typically publish their results from emerging systems due to intellectual property concerns. We need to develop metrics to quantify the success of work on emerging systems that cannot be published, and funding agencies need to value this work with a merit similar to publication count.

Discussion
It is critical to define new metrics for emerging systems evaluation, because we need a scientific approach to exploring the expanding space of architectural possibilities in the Moore's Law End Game and Post-Moore's Law eras. We also need to expand the budget for research on experimental testbeds over that of previous years when the future hardware roadmap was more predictable.

The key challenge in developing metrics for success of emerging systems is that we need to promote the idea of failure being a positive outcome. In general, researchers are rewarded for reporting successful outcomes. For emerging systems, however, we need to view "fast failure" of an emerging technology — with respect to its viability for HPC or for a specific purpose — as a successful outcome suitable for reporting purposes or for publication. We need to define metrics that reward efforts that prevent funding of or eventual procurement of emerging systems that are unsuitable for production-class HPC systems.

Support for Research (3)

Description and Drivers
As described in previous sections, we need to increase researchers' access to emerging systems. However, increasing access to these emerging systems will increase the burden of support. Currently, access to these systems is limited by both vendors and facilities because of the complexities involved in supporting them, such as dissemination of information and documentation that may be updated relatively frequently, protecting intellectual property, and managing different levels of access privileges needed by system software researchers. We need to identify support models for systems software research on these emerging systems that enable access by a broader range of researchers without overwhelming facility staff. The support model would benefit by moving closer to a "user facilities" model — as is done with more mature systems (production, early access) — than to the very *ad hoc* support model in use now. We will need to increase resources at facilities to adequately achieve this kind of support.

Discussion
An important facet of this support is the need to facilitate communication between researchers and vendor staff in order to address and debug difficult and undocumented issues as they arise. Without this direct connection, researchers are left to fend for themselves as they try to understand (if possible) system details, resulting in longer times to obtain research results, difficulty in determining the viability of emerging systems, or possibly inadequate results if researchers do not understand system details. Appropriate communication could be achieved by a "chain of command" system, in which a researcher or facility staff member assumes leadership related to technical issues for a particular emerging system, assumes responsibility for distributing updated documentation, and provides appropriate vendor contacts to other researchers on the system.

Researchers will also benefit from intensive work sessions with vendors and/or knowledgeable facilities staff (e.g., hackathon or dungeon sessions), where users can get hands-on experience with the technology in close interaction with support.

Researchers will also benefit from intensive work sessions with vendors and/or knowledgeable facilities staff (e.g., hackathon or dungeon sessions), where users can get hands-on experience with the technology in close interaction with support. Recently, these types of sessions have proven to be quite beneficial in DOE/vendor interactions, and we believe that making them standard practice will benefit ASCR research on emerging systems.

System Integration (4)

Description and Drivers
In particular, for emerging systems designed for data analytics applications (e.g., neuromorphic systems), it is critical that there be machine data sources for both offline analysis and online introspection in order to develop advanced system software such as dynamic compilers, schedulers, and power controls. If these data sources are unavailable, research on the system is effectively contrived, and the results may not be applicable to real-world problems. In contrast, having access to real or realistic data sources is not seen as a problem for emerging systems that are being evaluated solely for traditional simulation-based applications. For systems that may support both data analytics and traditional simulations, we will need to support data analytics applications by providing appropriate data sources.

Discussion
Some emerging architectures will be particularly suitable for research in data analytics that use online sources — an example is a neuromorphic architecture analyzing a video stream or streaming data from a scientific experiment, possibly computing in a closed-loop or enactive manner driving external controls. Without access to data sources, it will be impossible to perform this research. Access to streaming data is important because the volume of such sensor data can, in some cases, be much larger than what can be held in storage.

Software Ecosystem (5)

Description and Drivers
There is a need to research development of a rich ecosystem of system software and tools for emerging systems. These software components are likely to be nontraditional and unique to the emerging architecture (e.g., for quantum or neuromorphic architecture). There will likely be a mix of open-source and proprietary solutions. The proprietary solutions will come from the hardware vendor and third parties (e.g., Small Business Innovation Research/Small Business Technology Transfer research). Intellectual property facilitation is through model multi-party intellectual property agreements and encouragement of both open-source and proprietary solutions. In order to fully evaluate software layer integration, we need to facilitate interactions between vendors and researcher groups who are developing software for these systems. Without this integration, we cannot adequately identify gaps in the software stack or determine the viability of the emerging system for its intended use cases. Currently, most research efforts are conducted independently in a "vacuum," and software layer integration is left as future work. However, to quickly and accurately identify candidate emerging systems for DOE use cases, we need to understand the software integration issues early in the process.

Without a collaboration model that encourages integration of system software layers of components and that allows for transfer of (selected) intellectual property across research groups, each research effort must write all supporting software from scratch. Utilization of commercial tools and products is beneficial because they come with support and continued development from the suppliers.

Discussion
Integration of system software layers at the stage of evaluating emerging systems will enable collaborative research, identify gaps in the software stack, and facilitate transfer to production of the software. This also reduces duplicated effort, as one vendor or research group's product can be used by another research group.

3.5.1.2 Relationships among Needs and Conclusions

Emerging systems and systems software development both have special needs that were drawn out in our discussions. Emerging systems can vary widely — from the technologies they use (e.g., storage devices or networking hardware) to their full system configuration (e.g., quantum and neuromorphic systems). Further, there is increasing interest in developing scientific computing systems for converged missions, both for traditional scientific computing (e.g., simulation) and for data analytics. The requirements for these converged-mission systems increase the need for collaborative scrutiny, far beyond the current state of research on emerging systems, which — up until recently — supported only traditional simulation-based applications. In addition, system software can vary in terms of tools, runtime libraries, OS, and file systems, among other characteristics. Each of these requires different levels of access to systems and different kinds of information/support to perform research work.

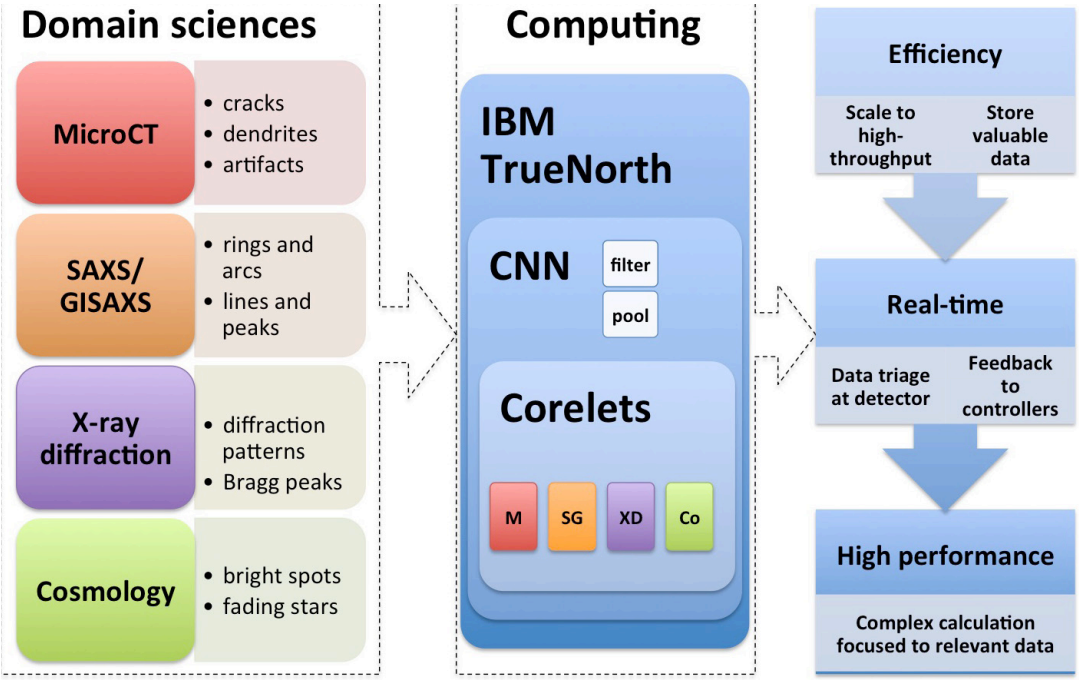


Figure 3-13. Neuromorphic computing is a new paradigm for cognitive computing, machine learning, and data analytics. It can be used to solve a wide variety of scientific problems, including image-based problems relevant to industry processes (Image courtesy of Daniela Ushizima, LBNL [Ushizima et. al 2016]).

Above all, supporting ASCR research at the intersection of emerging systems and systems software development requires increased access to information regarding the availability of emerging systems for research, to system documentation, and to intellectual property that potentially can be shared across groups working on the same system. ASCR research also requires increased support from facilities, so that researchers efficiently perform their studies and not spend time searching for the right contact for the information they need. In general, because emerging systems are typically temporary, access to information and facilities support are *ad hoc*. However, in order to ensure that researchers can quickly evaluate systems for ASCR applicability (i.e., "fail fast"), they need systematic access to systems and information. In addition, ASCR facilities need additional resources to be able to provide support closer to that available for production or early systems.

Above all, supporting ASCR research at the intersection of emerging systems and systems software development requires increased access to information regarding the availability of emerging systems for research, to system documentation, and to intellectual property that potentially can be shared across groups working on the same system.

3.5.2 Early Delivery Systems

Table 3-12 lists the research needs identified for systems software development on early delivery systems, the scientific drivers associated with those needs, and the potential for impact of each identified need (low, medium, high).

Table 3-12. Needs: Systems Software Development on Early Delivery Systems

Number	Need	Scientific Drivers	Potential for Impact (L, M, H)
1	Environment for routine testing of system software replacing any or all parts of the stack (all the way down to bare metal)	Only way to study systems software at scale. Supports applications and programming models with nontraditional software stacks.	H
2	Data collection and preservation	Data are necessary to analyze experiments, to identify correlations and test causality.	H
3	Library and tool readiness programs to complement application readiness programs	Making sure tools, libraries, and systems software are ready when the machine goes into production.	H
4	Modest-sized INCITE-like allocations and placement in queue for conducting research (on production systems)	Systems software research needs resources at full scale, but does not fit process (readiness issues, format geared around science).	M
5	On-ramping for systems software (e.g., new scheduler)	Bridge the gap from research systems software to deployment in production. Part of how ASCR fulfills its mission.	M
6	Emulators for exploring impact of faults, component performance, etc.	Reduces barriers to entry, promotes parameter studies.	L

3.5.2.1 Needs Identified from Breakout Session

Because there was not a session on system software needs for production systems, the breakout group noted several places where a requirement exists for system software on production systems (generally in addition to the same requirement existing for early delivery).

Environment for Routine Testing of System Software Replacing Any or All Parts of the Stack (1)

Description

The ability to replace any and all parts of the software stack with experimental versions is critical in enabling many types of ASCR CS and mathematics research. In particular, it is necessary for researchers to be able to load new OS kernels, networking drivers, file systems, and runtime libraries on a select set of nodes as part of the normal operation of the machine. In order to allow scaling studies, it must be possible to isolate (large) portions of the system (physically or for short durations) in order to conduct research on *both* production and early delivery systems.

The breakout group did not want to prescribe how this capability could be provided, but did note the following important technical requirements.

- The overhead of using such a mechanism must be lightweight enough to permit meaningful performance (time, energy, and other metrics) studies. This means that any overhead be limited to a small percentage of what would be possible should the components be run as the "normal" software stack.
- The use of this capability to load new components of the software stack should be part of a normal job submission process and not be restricted to special windows of time such as when a machine goes into and out of maintenance.

- The use of this capability should be isolated from other jobs on the machine to the same extent that two scientific applications are isolated. Specifically, this means that a crash of a custom stack job could crash that job, but should not be able to crash other jobs on the machine. In addition, any performance interference should be consistent with that possible between two applications jobs.

Drivers
The fundamental driver of this requirement is that it is the only way to study systems software at scale. If the community is going to make advances in system software, this capability is critical. Specific ASCR research that would benefit from this capability includes kernel architectures, programming model research, communication protocol research, file-system research, resiliency, and fault tolerance.

In addition to innovating in traditional HPC programming models, such a capability would be helpful to support applications/programming models with nontraditional software stacks. For example, it would allow a single machine to support a blending of ecosystems (e.g., traditional HPC with data analysis/workflows).

Discussion
The breakout session participants recognized that there are significant technical and administrative challenges to providing a mechanism to replace all levels of the software stack. However, the unique capabilities and broad utility of doing so make it worth pursuing. The following observations were made about the challenges and opportunities of such a mechanism:

- Fat-node systems may be easier to partition than smaller thinner nodes.
- It will likely be harder to physically isolate a network than to isolate CPUs.
- Replacing system software with new versions might expose hardware limitations (such as protection or performance isolation) that other users of the machine might not see.
- Facilities might have significant concerns such as security ramifications and long job launch times if a job launch required re-booting nodes.
- Potential opportunities exist to leverage bare-metal testbeds from NSF (e.g., the Chameleon testbed) and other agencies.
- Commercial networks (Ethernet) supported the ability to isolate the network to prevent security problems, and IBM Blue Gene machines provided isolated networks resulting in consistent performance for applications, so it is possible. Likewise, such capabilities are a standard part of the offering of cloud services such as Amazon Web Services (AWS) and Azure. Like a decade ago when node kernel matured to be full featured to match the capabilities of commercial computing, capabilities to replace the stack are required now to keep HPC in pace with the state of the art in commercial systems.

Data Collection and Preservation (2)

Description
There is a critical need to be able to gather, store, and analyze the data coming out of the systems software (e.g., networking data to evaluate routing policies). In addition to this monitoring capability, it is critical to have (correlated) data from disparate sources and metadata (how data was captured, the environment in which it was captured). Ideally, this capability would be consistent across facilities.

Specific needs identified in this category include the following:

- Hardware counter and other performance data and policies making the data available (e.g., through user agreements that permit monitoring of production runs) are needed.
- Monitoring information (real-time and archival) should be available to users and not just systems administrators; APIs, and not just a web interface or proprietary graphical user interface (GUI), should be provided.
- The need to collect, store, and analyze the data from systems software applies to both early and production systems (probably even more important in production).

Drivers
Data are necessary to analyze experiments and to identify correlations and causality within them. Providing insights will allow the identification of new research needs (i.e., find out what is not working well), as well as allow the evaluation of ASCR research artifacts when they are being tried out.

This information is also critical to some newer types of system software such as adaptive feedback (i.e., online auto-tuning).

The data researchers need will also likely be useful to facilities and vendors.

Discussion
Facilities currently collect a large amount of data. However, several factors limit its widespread use:

- Data privacy (e.g., hardware monitors expose the behavior of other apps sharing a resource),
- Proprietary APIs, and
- Keeping sufficient data for a long enough time.

The session participants also discussed concerns facilities might have in making these data available:

- The concern that ASCR researchers might misinterpret the data and thus come to incorrect conclusions, and
- Facility staff could devote significant time to help researchers correctly interpret the data and context.

Library and Tool Readiness Programs to Complement Application Readiness Programs (3)

Description
Currently, facilities have application readiness programs to ensure that applications can make effective use of new hardware when it goes into production. There is no similar program today that targets libraries and tools. Although vendors are contractually required to make some tools and libraries available, a significant component of the HPC ecosystem exists in open-source tools and libraries that are developed and maintained by ASCR researchers. A readiness program for this type of software is required to ensure that applications can use these tools and libraries as soon as a new machine is deployed.

Specific needs identified for such a program include the following:

- A streamlined process for documentation of proprietary interfaces. Access to proprietary interfaces is often critical in creating tools (e.g., hardware counters) or libraries (e.g., networking information for communications libraries).
- A way for researchers to communicate with vendors. When problems are encountered in porting to new platforms, ASCR researchers need to be able to communicate with the vendors' appropriate technical personnel.
- Providing both small-scale and full machine access to early access systems is critical to allow this type of effort to proceed.

Generally, this requirement only applies to early systems and not production systems.

Drivers
The specific drivers for this item include the following:

- Making sure tools, libraries, and systems software are ready when the machine goes into production.
- Ensuring that applications can rely on the libraries and tools.
- Ensuring that a necessary precursor to tuning applications is available to measure and improve performance. Much of the work of library and tool early access programs needs to be done before many parts of an applications readiness program can begin.

Discussion
Several challenges to creating such a readiness program were identified:

- This additional item could complicate the existing readiness programs.
- There could be an increased demand for scarce resources (people and hardware).
- Unlike application readiness programs, which often have specific performance or throughput goals for science applications (e.g., doubling of years of simulations of a climate code per hour on the machine), the success of a tools and libraries readiness program is harder to define and measure.
- Intrinsic in such a readiness program would be the measurement of the performance of early hardware. It will be important to manage the political sensitivities of reporting early systems that likely do not exactly match production systems.

Modest-Sized INCITE-Like Allocations and Place in Queue for Conducting Research (4)

Description
Modest-sized INCITE-like allocations are needed to conduct ASCR research. The total node hours required by such a program are fewer than those required for applications studies. However, such allocations are critical in order to conduct systems research at scale on production systems.

Drivers
- Full-scale access to productions systems is required for scaling studies.
- Systems software research needs resources at full scale, but does not fit the INCITE process very well (readiness issues, format geared around science).

Discussion
The biggest obstacle to such a program is that the needs of ASCR researchers are somewhat of a square peg in a round hole compared to applications. Specifically, requirements about science goals and performance readiness developed for applications do not fit CS and mathematics research. It is reasonable to have requirements to show that tools and libraries are ready for such an allocation, but the metrics will be somewhat different.

On-Ramping for Systems Software (5)

Description
Getting new software onto systems can be a challenge. For example, if an ASCR researcher developed a new scheduling algorithm, there is a challenge inherent in transitioning from the initial successes demonstrated in research papers to using the algorithm as the default production scheduler on a system. In order to bridge this gap, a specific "on-ramping" program should be developed for new systems software. Specific goals would include the following:

- Published/common policies across facilities for what systems software requirements must be met for deployment.
- Mechanisms to test systems software with real user workloads.

Drivers
The drivers for this need include the following:

- Bridging the gap from research systems software to deployment in production.
- Providing a way for new systems software to prove its readiness for production use. Systems administrators and users are naturally skeptical of new software that could reduce the performance or availability of production systems. However, without the ability to eventually get this software onto systems, the entire software stack will become calcified as it exists today.
- Getting new system software into production is a key part of how ASCR fulfills its mission to provide CS and mathematics research that is relevant and useful to the broader DOE mission.

Discussion
Several specific ideas were developed to help get such a program working, including the following:

- Provide discounts for applications running on pre-production systems software. For example, applications that agree to try new software could be charged a discounted rate. Alternatively, the share for running such a job could be split (not necessarily 50/50) between an application allocation and one provided to the ASCR researchers who created the new software.
- It is also useful to provide dedicated test and development systems (smaller versions of the production hardware) as an intermediate step to further test new system software.

Emulators to Provide a Controlled Environment for Exploring the Impact of Faults, Component Performance, Etc. (6)

Description
A common software layer that emulates computers with different properties than the current hardware is a key enabling technology for ASCR research. Such a system makes the machine appear to have different properties (e.g., more nodes, greater probabilities of hardware faults, or a faster network). This need applies to both early systems and production environments.

Drivers
Several scientific drivers make such an emulator desirable:

- Such a system would reduce the entry barrier to trying out new research ideas. Currently, an ASCR research team can spend a significant fraction of its time constructing simulators, emulators, or other harnesses to evaluate its core idea. Having such a system in place would reduce this overhead and allow the more rapid exploration of new ideas.
- Without such a system, researchers often are limited in how they are able to evaluate a new idea. To fully demonstrate the utility of ASCR research, it is important to conduct parameter studies to understand how the ideas perform under a variety of real-world and extreme conditions. Such a facility would make it easier to do this type of "stress" testing of new systems software.
- Using a common emulator would be more realistic than *ad hoc* models. In addition, a common framework would simplify direct comparison of the various software developed by different research groups.

Discussion
The major challenge of such an effort would be to construct an emulator that covers all salient features. A likely strategy is to create a "best of breed" emulator by combining software and techniques previously developed by ASCR researchers. In this way, the goal would be to unify and harden the best of what is already being done individually.

3.5.2.2 Relationships among Needs and Conclusions

The need that has the highest possible impact on ASCR research is "environment for routine testing of system software replacing any or all parts of the stack (all the way down to bare metal)." It is also the one that requires the most resources. Supporting this requirement will likely require additional hardware support as well as significant OS changes. However, it is also the one that has the highest chance of resulting in major new breakthroughs in OS, runtimes, and libraries. If DOE is going to move beyond slow incremental changes to the existing software stack, this key technology could make it happen.

3.6 Software Deployment and Support

3.6.1 Production Systems

Lowering barriers to software deployment at ASCR HPC facilities is critical for DOE's mission and the success of U.S. exascale computing. Software tools produced by ASCR CS and applied mathematics researchers enable physicists, biologists, and other domain scientists to exploit the full power of machines at ASCR facilities and thereby increase scientific output. Often, scientists cannot reap these benefits because research software is not deployed for general use at ASCR facilities. Facilities have limited resources dedicated to software deployment, typically supporting only a core software stack of compilers, programming models, tools, and math libraries. There is currently no well-defined path for researchers (or any facility users) to make their tools available to scientists. Facilities are the nexus of the HPC community. Barriers to software deployment must be eliminated so that facility users can scale beyond the limited resources of facility staff and reach the broader community with their software.

The exascale software ecosystem will comprise a wide array of tools, libraries, programming models, and performance portability frameworks, all of which are expected to be used by DOE application developers. If application developers are to build their simulations using this software, it must be portable and reliably deployed at ASCR facilities. Without better support for software deployment, application scientists will not benefit from the exascale software stack, and scientific productivity will be sacrificed as scientists struggle to deploy and scale their codes on new platforms.

Table 3-13 lists the research needs identified for software deployment and support for production systems, the scientific drivers associated with those needs, and the potential for impact of each identified need (low, medium, high).

Table 3-13. Needs: Deployment and Support for Production Systems

Number	Need	Scientific Drivers	Potential for Impact (L, M, H)
1	Strategy for transitioning research projects/products to deployed software	Exascale complexity, mission science	H
2	Automated, flexible software distribution process: (a) source, (b) binaries, (c) containers	Rapid development Share/reuse science libraries So many packages in stack!	(a) H (b) M (c) H
3	Common testing and continuous integration capabilities	Developer productivity, app developer confidence	H
4	A community to address cross-facility software ecosystem and deployment	Productivity of everyone, coordination, reduce redundant effort	H
5	Training materials for developer technologies	Initiate users to new technology	M
6	Post-deployment feedback process from the facilities: (a) to facilities, (b) to vendors	Facilitate communication, reward mechanism for code maintenance	M

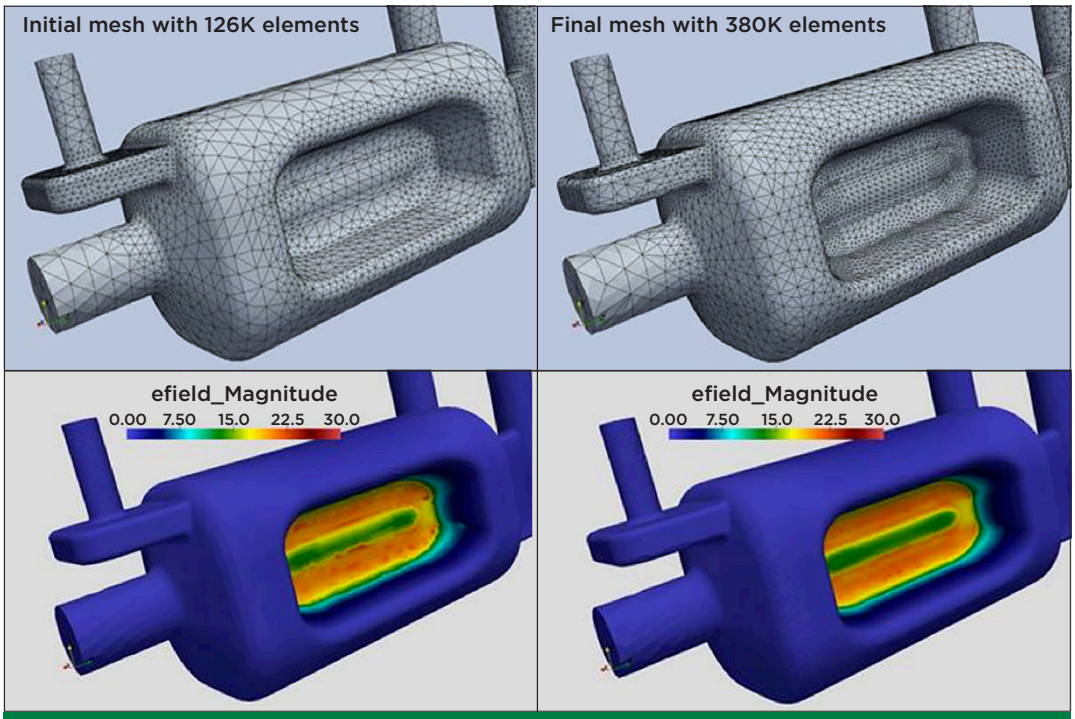


Figure 3-14. This figure illustrates the application of SciDAC Institute FASTMath-developed curved mesh adaptation to ensure accurate eigenmode predictions when applying the SLAC ACE3P high-order, finite-element eigen solver to an accelerator design problem. The example problem is the CERN HL-LHC crab-deflecting cavity, which contains geometrically complex internal components. The images on the left show the initial mesh with ~126K elements (top) and first eigenmode for the electric field, while the images on the right show the adapted mesh with ~380K elements (top) and the converged first eigenmode of the electric field (bottom). These procedures have been applied to problems with up to eight five-cavity sets, which required several million finite elements (Images courtesy of Lixin Ge, SLAC National Accelerator Laboratory, Stanford University; and Cameron Smith, Rensselaer Polytechnic Institute).

3.6.1.1 Needs Identified from Breakout Session

The software deployment working group identified six key needs for successful exascale deployment, as discussed below.

Strategy for Transitioning Research Products to Deployed Software (1)

ASCR researchers in the deployment breakout came from many different fields. Some were numerical library developers, others developed CS tools and libraries, and others were application developers. Researchers in this working group agreed that at the exascale, hardware challenges would demand a more rapid path to deploying research software so that application developers can make their code portable across diverse hardware.

Facilities currently draw a clear line between facility staff and users, but many facility users are also software developers. The group agreed that it is currently too difficult for researchers and other facility users to easily share software. ASCR facilities are not merely hosts for simulation, they are *development platforms* for HPC software, and this group believes it should be just as easy for developers to deploy software products on these systems as it is for facility staff.

The group identified several obstacles to software deployment:

1. Procedures for deploying software at ASCR facilities are not well documented.
2. Procedures for deploying software at ASCR facilities vary from facility to facility.

3. There is no central location or web page for users to learn about software deployed at facilities, and software is deployed in different locations at different facilities.
4. Users have no means of providing feedback to developers of deployed software.

In addition to the above issues, the researchers noted that developing and deploying software products are not the focus of most research grants, and PIs are not incentivized to deploy software for others to use. PIs are reluctant to spend scarce research dollars on deployment, which reduces the potential impact of their work.

More support is needed to migrate research software into production use. We recommend the following:

1. Facilities should work together to establish consistent guides for software deployment, which would describe how to make software available for all users of facilities.
2. Facilities should promote research software and develop websites or wikis where researchers can document how to use the software they install.
3. Facilities should work with ASCR to define and report clear metrics of success for deployed software (e.g., number of applications using the software).

In addition to the above recommendations for facilities, the group believes that ASCR program managers should provide incentives to deploy software early, when research programs begin. PIs should be encouraged to use their resources for deployment, and ASCR should work with facilities to recognize and reward metrics for software success.

Automated, Flexible Software Distribution Process (2)

In the previous section, we recommended that facilities develop a well-defined process for users to deploy their software. However, merely documenting the process is not sufficient for a sustainable software ecosystem. There is an acute need for a flexible, *automated* software distribution mechanism, so that users and facilities can reliably repeat a build of the same software configuration at different facilities.

Simulations and tools for HPC machines are no longer single, monolithic packages. Rather, they are a complex network of interdependent physics packages, math libraries, and tools. The exascale software stack is expected to be *more* complex. Modern simulation codes can depend on tens or hundreds of dependency libraries, and often they must be built with multiple MPI implementations, compilers, and configurations. This leads to a combinatorial explosion of required libraries. There are more than 100 packages. Installing all of these libraries by hand is a daunting task even for facility staff.

Currently, facilities deploy most software by hand, which limits the number of packages that can be supported. There are historical and practical reasons for this; ASCR facilities typically host unique, bleeding-edge architectures, and full, well-supported software stacks are not always available on these systems. Moreover, even if software is available, optimized builds for HPC compilers and MPI implementations are typically not. These must be ported, painstakingly configured, and built from source code.

To address the challenges of deploying an exascale software stack, the group recommends developing a common mechanism for facilities *and* users to deploy software for themselves and other users automatically. This should work in a repeatable way across facilities, in both global installations and user home directories. In essence, if a user has deployed software at one facility, he or she should be able to expect that a similar configuration of the same software can be deployed reliably and automatically at another facility.

The working group considered the above requirements for software deployed from source code, binary code, and in containers. They discussed available and emerging technologies and made the following recommendations.

1. *Source Distribution (High Priority)*. Source code is the de facto distribution mechanism for HPC software and arguably the most important distribution mechanism to support. Recently, tools such as Spack and EasyBuild have emerged that allow complex software with many dependencies to be deployed in an automated way and in many different configurations. In addition to automating much of the deployment process, these tools also document and formalize the task of creating a software *package* for reuse by facilities and users. These tools are promising, but they require maintenance to ensure that the source packages they contain continue to build and run reliably on ASCR systems.

   For source deployment, the working group recommendations are as follows:

   a. Facilities should contribute to package managers like Spack, and coordinate and collaborate to share common package recipes. This will leverage deployment effort across facilities and reduce time and maintenance required for software deployments.

   b. Facilities should document the process of making and installing packages so that users can easily create packages recipes and share them with facilities.

   c. Facilities should frequently test builds of their own software and of user-contributed software to ensure that the software remains robust and that it can be built reliably.

2. *Binary Distribution (Medium Priority)*. Traditional OS package managers distribute software in binary form, but this is not currently typical for HPC systems where users prefer to generate optimized builds and tune applications for the host architecture. Tools like Spack and projects like OpenHPC are developing binary deployment techniques for HPC architectures.

   Facilities should consider running build farms and caching binary versions of source packages generated in (1). This would increase the speed with which users could deploy complex software, as well as the reliability of deployment. We view this as a secondary priority to (1), because it is effectively an optimization on source packaging, and an initial source packaging capability is more important.

3. *Containers (High Priority)*. Containers are a widely used deployment tool in the cloud. They allow an application, its dependencies, and its associated run environment to be packaged into a binary image. Similar to virtual machines, but lighter weight and with few of the performance drawbacks, containers can be used to reliably reproduce an entire software stack from machine to machine, as long as the machines are binary compatible at the kernel level. This means that a container produced for one OS can run unmodified on a host machine running an entirely different OS.

   Containers have great potential for reproducibility of research results and for easily deploying large-scale HPC applications. Application developers can create a single binary image and distribute it to many remote sites, thus allowing users to skip the software installation step entirely and run directly in the same environment the application developer created. Containers also allow for easy deployment and orchestration of distributed applications, such as data analysis workflows and web portal front-ends used frequently in the big data world. As HPC grows closer to big data and users increasingly need to use big data tools, containers will become increasingly relevant for HPC centers.

   Containers have not been widely vetted across HPC centers, as there remain some security, usability, and low-level hardware access challenges. Container solutions for HPC have just begun to emerge, such as Shifter (NERSC) and Singularity (Lawrence Berkeley National

Laboratory [LBL]). Although containers do not solve the complexity of building HPC applications the same way source-based package managers do, they solve a critical problem of reproducing an existing scientific simulation environment, and they are a critical piece in the merger between HPC and big data. There is thus a clear need for them at HPC facilities.

Our recommendations:

- **a.** Facilities should ensure that secure, performant container technology is available in the HPC environment, either by working together on solutions such as Singularity or Shifter, or by requiring container capabilities from HPC system vendors.
- **b.** Facilities should begin to distribute lightweight OS images that can be used in containers to emulate the facility's environment (e.g., on a laptop or other development machine). This would greatly increase developer productivity.

Common Testing and Continuous Integration Capabilities (3)

Automated unit testing, regression testing, and continuous integration provide quick feedback to software developers and help them to quickly discover bugs. Good testing practices also give users confidence that software has been verified and validated, and that it will deliver correct results if they use it in a production simulation. However, no ASCR facility offers continuous integration, automated builds, or testing as a service to users. The deployment working group identified this as a significant gap.

Exascale simulation codes are expected to rely on a wide variety of CS tools, mathematics libraries, and physics components to best exploit the performance of future machines. For this software ecosystem to function, it must be tested and validated both on development machines like laptops and workstations, and in the production environment of ASCR facilities. Although software is typically developed on laptops and workstations, the build and runtime environments of an HPC center are significantly different. In addition, without automated testing, it is extremely difficult for researchers to verify that their code will work reliably and correctly in the facility environment. The current lack of support for automated testing makes most tools brittle at best in these environments, and scaling tools and applications is a difficult, time-consuming, and manual process.

Automated regression testing is not new; most codes have some form of test suite, and many teams run tests on a regular basis at their home facilities or on their own machine. Continuous integration (CI) is a more modern practice where every change to a code is tested automatically before it is committed to the repository and deployed, ensuring that even extremely active projects continue to function correctly as many developers make changes. Web-hosted CI tools such as Jenkins and Bamboo are widely deployed in industry development environments, where teams have resources to set up web servers and extra servers for testing. For open-source and cloud projects, tools like Travis CI and Appveyor offer testing on Linux, Mac OS, and Windows platforms free of charge, and integrate with popular source hosting sites like GitHub and BitBucket. These tools are second nature to cloud developers, but in HPC they are typically used only by large teams that have the resources to set them up, and only by teams with connections to HPC facility staff who allow them to run services in the facility environment. No site currently offers basic testing tools as a service to all users, despite the well-understood benefits for software reliability.

In our group discussion, application developers identified reliability and ongoing support as key factors in their decision to either adopt or not adopt tools and libraries. Some simulation developers, such as the Hardware/Hybrid Accelerated Cosmology Code (HACC) team at Argonne and the University of Chicago, intentionally eschew the use of external dependencies because they do not believe that they can rely on code from other teams to work for them in production. This is a crisis for exascale, as the hardware and software environments will be too complex for

any one team to manage all the components themselves. For the exascale software environment to succeed, researchers and developer require support from the facilities to lower the barriers to automated testing.

The working group recommends the following:

1. Facilities should provide tools for automated unit testing, regression testing, and continuous integration (e.g., Jenkins, Travis, and Bamboo).
2. If these tools cannot be deployed securely for users, the facilities should work together on an open-source CI tool and modify it to provide the missing functionality.
3. Facilities should coordinate to adopt common technologies for automated testing so that the barrier to use for software developers is low. If developers have learned a tool at one facility, they should be able to easily use the offerings of another facility.
4. Facilities should document their automated testing tools and provide training to software developers on their use.
5. Users of facilities should be able to expect the same or similar tooling to what an open-source or cloud developer would expect in automation and deployment tools.

Community to Address Cross-Facility Software Deployment (4)

A common theme across all the topics discussed by this working group was the need for a more cohesive community for HPC software development. Extreme-scale science requires the combined use of software developed by diverse groups, and software developers would benefit from increased consistency across platforms at ASCR facilities and easier, more effective ways to share software. Moreover, the facilities should recognize that many of their users are *software developers*, and that ASCR facilities support more than simply running simulations.

The HPC community includes developers of tools, simulations, and numerical libraries, as well as facility staff and vendors, and all of these parties are interested in deploying software at ALCF, NERSC, OLCF, and other common platforms like laptops and commodity clusters. There is currently little standardization across these environments. Compilers and MPI libraries are deployed in different locations and with different usage models at different facilities. Even the commands used to run parallel applications are different at different facilities, and scripts for parallel software testing must be customized for each site. Support for deployed software is handled on a per-facility basis, and there is little communication among facilities about best practices or common deployment methodologies.

In the exascale timeframe, developers will be expected to deploy on *many* diverse architectures and in multiple software environments at ASCR facilities. Collaboration among facilities and developers could reduce the burden of software deployment and increase commonality in facility environments.

The working group recommends that facilities:

1. Start a Council for Scientific Software composed of developers who work with facilities to define a process for coherent cross-facility software deployment. The council should:
    a. Define and document the requirements that a software package should satisfy in order to be used in a cross-facility software ecosystem;
    b. Define and document an on-ramp process to transition software into production;
    c. Define topics for training researchers and developers to deploy at ASCR facilities; and
    d. Address the needs of developers for software sustainability.

2. Communicate across disciplines and promote dialogue on ASCR's software ecosystem and deployment tools. Do this by:
    a. Hosting knowledge bases, user forums, and lessons-learned documents; and
    b. Hosting training on deployment for users.
3. Recognize the importance of supporting a modern software development environment and support grassroots efforts to provide developer tools to all facility users, including tools and services for testing, debugging, building, deploying, and interoperability.

HEP software foundation (HSF undated) facilitates coordination and common efforts in the HEP community. The group supports working groups and publishes software-related knowledge bases and white papers for developers' member institutions.

Training Materials for Developer Technologies (5)

ASCR facilities currently provide documentation and training materials aimed at scientific application developers and their users, but little documentation is aimed at library and tool developers. Ways to make software available at facilities are often poorly documented. Often, only the facilities can deploy modules and shared directories, and application developers cannot use these mechanisms to make software available for all users. Further, if the facilities do deploy HPC-specific packaging, containerization, and testing systems, developers may not have experience with them and will need documentation.

The working group recommends that once the gaps identified above are addressed, that the facility thoroughly document procedures and provide training for HPC developers.

Post-Deployment Feedback Process: (a) to Facilities, (b) to Vendors (6)

Application developers, tool developers, and library developers in the working group would like to be able to more easily deploy their software at ASCR facilities. They would also like to be able to receive feedback from facility users on deployed software. The working group noted that there is currently no mechanism for users to provide feedback or ratings on installed software, and few mechanisms for developers of packages not deployed by the facilities to get feedback on usage of their installations. Moreover, there is also no direct way for developers to rate or send feedback on vendor software (e.g., compilers and MPI implementations). Such requests have to go through the facility, and turnaround on these requests can be very slow.

The working group recommends that:

1. Facilities provide a common web page with usage statistics and rankings of installed software, including the following:
    a. How many people have used each installed package, and
    b. Ability for users to rank packages (e.g., with "likes" or "stars").
2. Facilities expand the web presence of ASCR software packages to include links to the following:
    a. Software home pages,
    b. Points of contact for user-installed software,
    c. Issue trackers,
    d. Continuous integration and testing results, and
    e. Rankings and usage statistics.

Taking these steps would facilitate communication among application scientists, facilities, and ASCR researchers. It would also enable application scientists to explore new technologies available at the centers. Centers could use web statistics like these to inform facility software council decisions on which software packages the facilities should help to maintain and deploy, and which should remain user supported. Documenting the basis for such decisions could help users understand the steps needed to gain facility support and increase their user base. Vendors could also use these statistics to track usage of their tools at facilities and to better understand users' issues with deployed software. Statistics like these would also encourage vendors to become more active members of the ASCR HPC community. Finally, web statistics and feedback would incentivize ASCR researchers to devote more maintenance time to widely used software.

3.6.1.2 Relationships among Needs and Conclusions

The common theme running through all topics addressed by the deployment working group was that the HPC software developers community spans all ASCR sites, and facilities should work more closely with each other and with developers to reduce the barriers preventing ASCR software from being deployed on ASCR facility machines. This involves fostering community working groups; documenting processes; and providing tools for software packaging, deployment, containers, issue tracking, and distributed collaboration. Facilities should provide clear guidelines for developers that enable them to build, test, deploy, track, and publicize their software through all stages of its lifecycle, for a broad range of ASCR users. Facility staff should also collaborate to establish best practices and to provide open-source tools that provide common development and deployment environments across the facilities. This will enable all users to easily deploy software across the full range of exascale machines.

3.7 Operational Data and Policies

3.7.1 Production Systems

Participants in the breakout session on operational data and policies for production systems included both facilities personnel and researchers interested in operational data. In contrast to emerging and early systems, research on production systems focuses on improving the system software infrastructure and optimizing systems software and applications to make efficient use of the established hardware.

Table 3-14 lists the research needs identified for operational data and policies for production systems, the scientific drivers associated with those needs, and the potential for impact of each identified need (low, medium, high). The most important requirements are access to the relevant data and the context necessary to properly interpret that data. Comprehensive and consistent data collection and privacy policies across centers are also desirable, but they are less critical.

Table 3-14. Needs: Operational Data and Policies for Production Systems

Number	Need	Scientific Drivers	Potential for Impact (L, M, H)
1	Post-mortem and real-time data of different types (performance, power, faults) about private and shared (I/O, network) resources	End-to-end and cross-layer optimization; retrospective and real-time analysis	H
2	Documentation and context for data; ability to track evolving context	Improved analysis; realistic and actionable results	H
3	Usage statistics (e.g., for compilers, libraries, programming models, and tools)	Requirements-based infrastructure	H
4	API-driven access to consistent data formats	Automation of data search, input, and filtering; real-time adaptation; tool interoperability	H
5	Complete data for all jobs running on system	System workload analysis; co-scheduling research for throughput and power optimization	M
6	Consistent cross-facility logging, monitoring, sharing, and privacy policy	Distributed system design, monitoring, and optimization	M

3.7.1.1 Needs Identified from Breakout Session

The core finding of this session was that there are significant opportunities for mutual benefits to the ASCR research and computing facility missions. These opportunities exist within an environment of potentially divergent aims. In the absence of well-grounded data, research activities can employ unrealistic and possibly incorrect assumptions, resulting in unrealizable recommendations. Without context-aware research, an unsurmountable lag will exist between production systems and the state of the art. The needs identified in this session acknowledge this reality and reflect our attempts at narrowing the gap between systems operations and applicable research.

Post-Mortem/Real-Time Data of Different Types about Private and Shared Resources (1)

Description and Drivers

Many performance, power, and fault analysis studies are based on historical data about application executions. Sources of relevant data range from private resources, such as the processor and memory on a node, to shared resources, such as secondary storage and networks. Analysis may be for a single application or for a workload consisting of multiple applications. For example, a power management algorithm for a single job running under a power cap would allocate power among

the nodes allocated for that job, while a system-wide power management algorithm would allocate a global power bound among scheduled jobs. Because performance of a single application can depend on competition for shared resources with other applications, data from shared resources is needed even for performance analysis of a single application. Because data for shared resources is often collected external to an application and data for different shared resources may be collected independently, some means (e.g., system-consistent timestamps) of correlating the data from different sources is needed.

A growing area of research is real-time adaptation to optimize performance, power consumption, resilience, or some combination of these (Bailey et al. 2014; Balaprakash et al. 2014; Huck et al. 2013). For example, a power management algorithm may shift power among nodes or within a node to compensate for load imbalance or noise, or it may reduce processor frequency or power in memory or communication-intensive portions of an application to save energy (Hoffman and Maggio 2014; Porterfield et al. 2015). In some cases, real-time data are needed to perform such adaptations. Adaptation to faults is another research area where real-time data about failures are needed (Balaprakash et al. 2015).

Discussion
Individual applications can be instrumented to collect performance and power data using currently available performance analysis tools such as PAPI, Tuning and Analysis Utilities (TAU), and HPCToolkit, as well as some vendor tools. However, instrumenting these applications requires considerable effort on the part of application developers and users. Facility support for loading appropriate modules that would automate such instrumentation (to the extent possible) would help ease the burden.

Post-mortem data about shared resources may already be collected at some facilities (Carns et al. 2011), but information about what data is collected and how it is accessed are not always readily available to researchers. We recommend requesting input from researchers about what data is needed, augmenting the collection of such data if needed, and documenting policies and methods for accessing the data.

Real-time monitoring data are generally less available to researchers than post-mortem data on current facilities.

Documentation and Context for Data (2)

Description and Drivers
Without sufficient context, at a minimum, this comprises what is colloquially called metadata; operational data can be misinterpreted and lead to erroneous and/or unactionable conclusions. Context, which can include the specifics of the hardware platform and its configuration, systems software versions, job queuing and control policies (in effect), and evolving mission priorities can have a significant impact on operational data despite being peripheral to the measured data. For example, application optimizations and queueing policy improvements can both result in higher job throughput, but given only job log data — without the additional context of software versions and policy changes — users would be unable to distinguish between the two.

At a minimum, documentation on the format of data files and the semantics of data fields is necessary to properly interpret operational data. The format of operational data, both structured and unstructured, varies widely (Brandt et al. 2014) and is often not self-describing, which can lead to various false assumptions. Common problems that arise from lack of documentation range from the simple difference between base-2 and base-10 units (e.g., gibibyte vs. gigabyte [GiB vs. GB]) to semantic differences (e.g., a field labeled as GB actually representing a bandwidth GB/second), all of which can render analysis impossible or inaccurate.

Tool drivers for context and documentation include comprehensive performance analysis tools such as TAU and its TAUdb database (Huck et al. 2005), as well as workflow analysis tools (Zhang et al. 2016). The TAUdb schema provides for collection of metadata and correlation of the metadata with performance data, provided that the metadata can be obtained. Science drivers for documentation and context include analysis of job scheduling systems; end-to-end application analysis (Sreepathi et al. 2016); and workflow performance analysis (van Dam 2015) for performance, throughput, and power/energy.

Discussion
HPC systems are extremely complex, and both the hardware and the systems software are continuously evolving. Most HPC centers collect system diagnostics such as usage data, processing loads, jobs run, and network and storage utilization. Tracking the evolving system context over time and relating it to application performance data would enable longitudinal studies of trends in efficient use of resources. In addition, such studies — which reveal areas for system and application performance optimization — would ideally be fed back to facilities for both short-term resource optimization and long-term resource planning. It follows that establishing clear documentation is necessary to minimize the burden of supporting operational data for facilities. Allowing researchers to self-serve and analyze data provides the additional possibility for high-frequency or low-latency analysis.

Usage Statistics for Systems/Library Software (3)

Description and Drivers
Usage statistics for mathematics and communication libraries can help researchers and vendors focus library/tool development and code optimization effort on the more frequently used algorithms and routines, allowing them to better prepare applications to run on next-generation systems (Zhao 2014; Brightwell et al. 2006). The information can also inform procurement decisions and acceptance criteria concerning what software technologies are needed. Data about what programming models and languages are being used by applications can yield insight into how applications are using the system and how they are exploiting parallelism, especially if tracked over time. Historical usage data may reveal performance bugs caused by using certain libraries or suboptimal use of parallel programming models. Correlating programming models with performance can help with evaluation of portability versus performance.

Discussion
System logs are often insufficient for tracking library usage, especially when applications are linked statically. Use of the Automatic Library Tracking Database (ALTD) (Fahey et al. 2010) at NERSC (Zhao 2014) has shown that library usage data can be recorded automatically with low overhead. On Cray systems, ALTD can also capture which compiler was used to build an executable and the compiler used to build the MPI library, as well as what GPU programming model or library was used (Robinson and Stringfellow 2013). The data collection can be carried out automatically with low overhead, and users can be given the option of turning it off by simply unloading a module.

API-Driven Access to Data in Consistent Formats (4)

Description and Drivers
Accessing operational data today usually involves parsing log files to extract the desired information. An API would enable tools to make queries about what information is available and extract only what is needed. Consistent data formats, or at least well-documented formats that could interoperate, would facilitate combining data from multiple sources and interoperation between different analysis tools and across different facilities. An API would also be beneficial to researchers exploring adaptive runtime systems; for example, runtime systems for adaptive power management and checkpointing (Palmer et al. 2015).

Discussion
The Open XDMoD project uses a RESTful API for all interactions with core infrastructure components (e.g., data warehouse, report generator) and provides an abstraction that insulates clients from any changes to the underlying infrastructure, while ensuring that each request is authenticated and properly authorized based on the user's role (Palmer et al. 2015). NERSC has also demonstrated the utility of REST APIs for accessing operational data through the NERSC Web ToolKit (NEWT), which provides a REST interface that allows users to query system status and accounting information on NERSC systems (NERSC undated). While this API was designed to serve the needs of user portals and science gateways, researchers can interact with it directly through either REST or a JavaScript library if they choose. In this sense, NEWT demonstrates how facilities that already support user portals and science gateways may be able to repurpose existing back-end interfaces to provide ASCR researchers with APIs to access operational data. Similar to these efforts, an API with either an agreed-upon set of fields on self-documenting contents that could be consistently interpreted, would enable both researchers and facilities personnel to access and analyze relevant data quickly and easily.

Complete Data for All Jobs Running on a System (5)

Description and Drivers
Collecting performance and power consumption data for all jobs running on a computer system would enable system-wide workload analysis. System-wide data would benefit research on job co-scheduling for performance, throughput, and energy consumption optimization. Because it is increasingly expected that shared resources (e.g., memory and I/O [Behzad et al. 2014]) are on the critical path toward high performance, a more accurate understanding of real-time, application-external utilization is vital to understanding the distribution between ideal and worst-case performance. The operational and mission-informed decisions of facilities can have a considerable impact on realized performance, and there is growing demand among researchers to be able to take into account the system-wide implications and how these affect overall (i.e., system-wide) application performance.

Discussion

Low-overhead data collection for a job could be made the default on production systems, with the understanding by users that running on the system implies consent for such data collection. The default could be implemented by means of modules that are automatically loaded. A user could opt out if necessary (e.g., for a benchmark run where every ounce of performance is critical) by unloading the relevant module(s).

Consistent Cross-Facility Logging, Monitoring, Sharing, and Privacy Policy (6)

Description and Drivers
A primary concern for ASCR researchers is the ability to pull in comparable data from multiple sites to inform their work. Because sites tend to have different architectures, the only way a researcher can conduct analysis across multiple architectures is to go to multiple facilities. Consistency is needed across the facilities in terms of what data they are logging and making available, policies for what can be shared (and with whom), and issues related to user privacy/sanitization. The privacy policy selected by a facility has a significant impact on the ease of publishing operational data, as well as implications for correlation of data from multiple subsystems.

Discussion
Satisfying the need for cross-facility consistency would require discussion and ongoing collaboration between sites to implement consistent mechanisms and policies. A policy ensuring the privacy of users as it relates to captured operational data implies that an anonymization process

must be defined and performed prior to the release of any data. Unfortunately, in attempting to remove attribution, important facets of the data are often obscured; in particular, those that facilitate relating data from one subsystem to data in another. Care must be taken when performing anonymization, if it must be performed. Another option would be to establish a policy of no expectation of privacy for most users, with exemptions for specific cases.

3.7.1.2 Relationships among Needs and Conclusions

In addition to the needs described above, needs for operational data were identified by a number of other breakout groups, including the following:

- Breakout 1: Software Development on Production Systems — identified a need for facility monitoring data and usage statistics to inform software tool development.
- Breakout 3: Distributed Computing and Networking on Production Systems — identified a need for availability of network data and data movement logs to enable research in data movement optimization techniques and tools.
- Breakout 4: Data, Visualization, Analytics, and Storage on Production Systems — identified several needs for operational data.

Operational data are already being captured in some form on all DOE production systems. However, further steps to make sure that all data needed by ASCR researchers are made available in formats that can be interpreted accurately would increase the impact of research efforts and enable new research directions. Such steps would also facilitate applications to make more effective use of current and next-generation computing systems.

4 PATH FORWARD

For researchers to move forward in addressing ASCR research challenges, an evolving computing ecosystem must support them. While Section 3 describes detailed requirements in specific areas for ASCR research, common themes emerged across the different areas of research; these themes represent potential high-impact paths forward for ASCR research.

4.1 Cross-Cutting Need Areas

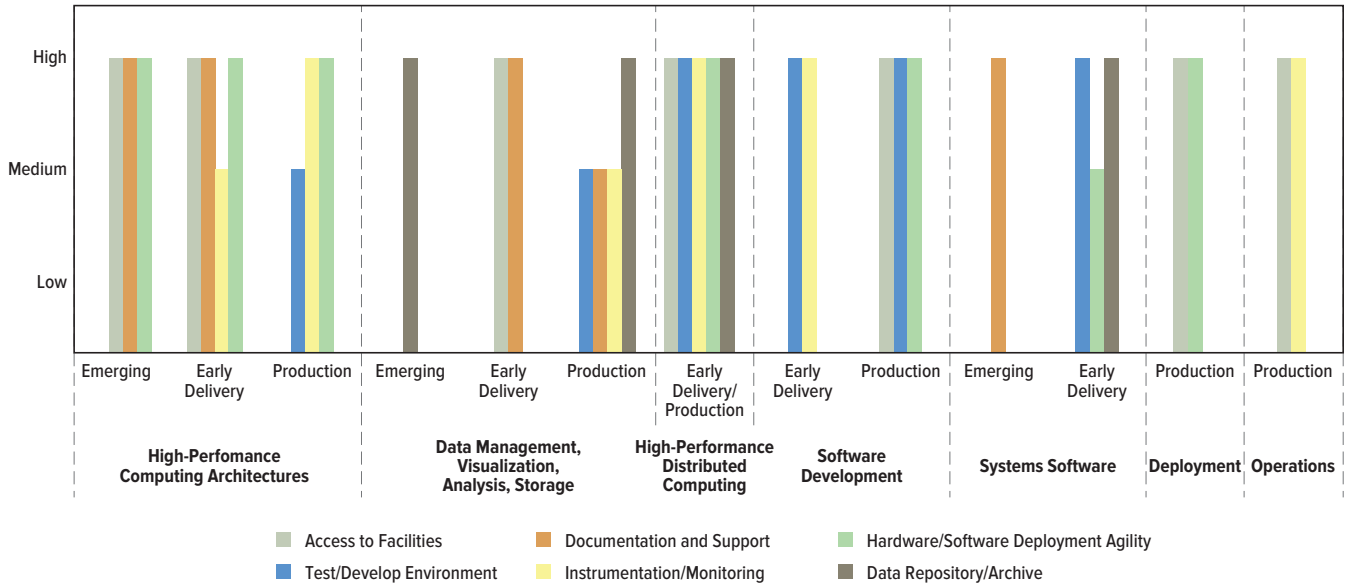
Many of the recurring or highest-priority requirements identified during the ASCR research exascale review address communication and collaboration between ASCR research and the ASCR facilities — from knowledge about the availability of resources to policy and process information sharing. Additional high-level requirements include improved access to hardware at all development stages and an ecosystem that is amenable to the ASCR researchers' requirements for development and testing of software. This section identifies impactful first steps to evolve the ecosystem based on ASCR research needs.

Table 4-1 summarizes the primary cross-cutting requirements identified for ASCR research. Most of the cross-cut topic areas fall into two larger areas: (1) hardware capabilities and (2) ecosystem access, capabilities, and policies. In Sections 4.1.1 and 4.1.2, we provide some specific examples of these two high-level, cross-cut areas.

Table 4-1 displays cross-cutting need areas, defined as general technology areas that encompass multiple needs identified by breakout groups in their individual requirements discussions. Cross-cutting need areas do not reflect a specific need common across different breakout sessions; rather, they represent a spectrum of needs falling under a common technology umbrella. Each breakout group targeted a specific aspect of ASCR research requirements analysis for HPC facilities within a specific phase of development/deployment (emerging, early delivery, production). Cross-cutting need areas represent an attempt to group together the logically related needs of the individual breakout groups to facilitate a more system-wide approach in responding to those needs.

Such a holistic approach to addressing identified needs could provide greater benefit to the HPC computing ecosystem as a whole than simply addressing point solutions for each specific need. The six cross-cutting need areas listed in Table 4-1 are not prioritized; however, the table does indicate by letter (H for high; M for medium) how an individual breakout group rated a need in its session that was subsequently included in one of the cross-cutting needs areas. Table 4-1 contains only a subset of the needs identified by individual breakout groups in Section 3. Specific needs identified by a breakout group that did not logically fall into one of the cross-cutting areas stand on their own merits, as detailed in Section 3.

Table 4-1. Cross-Cutting Need Areas



Because a cross-cutting needs area may cover a broad spectrum of identified needs, we provide a brief description of the scope of each cross-cutting need area below.

- ***Access to facilities:*** Includes access policies, authentication technologies, ease of access, availability of resources, federated authentication capabilities, and common authorization/ authentication frameworks across HPC facilities.
- ***Test and development environment:*** Includes accessibility and availability of development systems and small test bed facilities, end-to-end test and development capabilities up through the application layer, system availability for at-scale testing, routine testing of system software, and vendor participation in testbed facilities.
- ***Documentation, communication, and staff support:*** Includes consolidated information about access and available resources, staff training and training materials for developers, access to system hardware and software information, and improving communication and coordination among developers and system users.
- ***Instrumentation and monitoring:*** Includes improving system monitoring, access to system operational and performance data, network infrastructure counter data, usage statistics, running jobs data, data movement logs, and tools that provide performance insights.
- ***Hardware/software deployment agility:*** Includes flexible software deployment processes, system reconfiguration agility, facilitation of hardware and software configuration changes, deployment of next-generation network technologies and data movement tools, software portability to run common application code easily across HPC facilities, easier on-ramp capabilities for new system software, and improved strategies for transitioning research project successes into deployed software.
- ***Data repository and archiving capabilities:*** Includes data collection and preservation capabilities, modern HPC-enabled data repositories, storage capacity at HPC facilities for data-intensive workflows, and long-term archiving capabilities.

4.1.1 Hardware

To meet science needs, ASCR researchers need a data ecosystem not present today. This ecosystem includes long-term data storage solutions and effective software and hardware to move, manage, and discover (Section 3.2.3). In addition, discovery and computation on these data requires new scheduling capabilities (Section 3.3.1). These requirements apply, to some degree, across early delivery and production hardware deployments.

- Facility-style availability of emerging and early delivery systems to enable as diverse a research environment as possible and increase transparency about availability (Sections 3.1 and 3.2).
- Access to various architectures is not the sole need; tools to measure and monitor applications and hardware are necessary to provide insight.
- Growing data needs are driving the ASCR facilities to consider a different balance of capabilities for the future; the need for this new capabilities balance is echoed in this report (Section 3.2.1).
- Distributed computing support is needed at HPC facilities (Section 3.3).

4.1.2 Ecosystem Access, Capabilities, and Policies

ASCR researchers rely on access to emerging, pre-production, and production HPC resources to execute their mission. Hardware resources also require capabilities and policies that are welcoming to CS and applied mathematics research. The traditional use models of HPC resources are focused on domain science campaigns that might not need the level of system interaction that ASCR researchers need. As a result, different consideration is needed to address how the HPC facilities operate.

- ASCR researchers require scheduling, interaction, and allocations that are more in line with their research needs (Sections 3.1.1, 3.2.2, 3.3.1).
- Distributed computing support is needed at HPC facilities (Section 3.3).
- ASCR researchers must be able to interact, test, and interface with new software technologies on ASCR facility hardware. This need spans most research, including software development, data, and networking (Sections 3.3.1, 3.4, and 3.5).
- Growing data needs are driving the ASCR facilities to consider a different balance of capabilities for the future; the need for this new capabilities balance is echoed in this report (Section 3.2.1).
- Support — through policies and infrastructure for software development, data, and networking — and training are needed.
- Access to logs and operational data for research is required; also required are consistent cross-facility logging, monitoring, and sharing and privacy policies (Sections 3.1.3 and 3.7).
- Enhanced communication is needed:
  - Increased exposure to available facility resources and programs;
  - More regimented access to emerging systems, emulators, and early systems to facilitate research and reduce delays in software deployment (Sections 3.1 and 3.2.2); and
  - Communication among the facilities, vendors, and users of early delivery systems.

4.2 Conclusion

ASCR researchers identified requirements that are key to an evolving and more welcoming computing ecosystem in the areas of hardware resources and ecosystem capabilities, policies, and communication. These areas represent a collaborative opportunity across much of the computational ecosystem.

This page is intentionally left blank.

5 REFERENCES

Bailey, P.E., D.K. Lowenthal, V. Ravi, B. Rountree, M. Schulz, and B.R de Supinski, 2014, "Adaptive Configuration Selection for Power-Constrained Heterogeneous Systems," in *Proceedings of the 43rd International Conference on Parallel Processing (ICPP)*, September.

Balakrishnan, S., R. Black, A. Donnelly, P. England, A. Glass, D. Harper, S. Legchenko, A. Ogus, E. Peterson, and A. Rowstron, 2014, *Pelican: A Building Block for Exascale Cold Data Storage*, Microsoft, October 6, https://www.microsoft.com/en-us/research/publication/pelican-a-building-block-for-exascale-cold-data-storage/, accessed January 2017.

Balaprakash, P., L.A. Bautista Gomez, M.-S. Bouguerra, S.M. Wild, F. Cappello, and P.D. Hovland, 2015, "Analysis of the Tradeoffs between Energy and Run Time for Multilevel Checkpointing," in *Performance Modeling, Benchmarking, and Simulation, Vol. 8966 of LNCS*, 249–263. doi:10.1007/978-3-319-17248-4_13.

Balaprakash, P., A. Tiwari, and S.M. Wild, 2014, "Multi-objective Optimization of HPC Kernels for Performance, Power, and Energy," in *Performance Modeling, Benchmarking and Simulation, Vol 8551 of LNCS*, 239–260. doi:10.1007/978-3-319-10214-6_12.

Bartlett, R., A. Dubey, S. Li, J.D. Moulton, J. Willenbring, and U.M. Yang, 2016, "The Role of Automated Testing in Scientific Software: Impacts on Research Credibility, Development Productivity, Maturation, and Sustainability," in *Software Engineering for Science,* editors J. Carver, N.P. Chue Hong, and G. Thiruvathukal, Chapman and Hall/CRC.

Behzad, B., S. Byna, S.M. Wild, Prabhat, and M. Snir, 2014, "Improving Parallel I/O Autotuning with Performance Modeling," in *Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing (HPDC14)*, *2014.* doi:10.1145/2600212.2600708.

Bethel, E.W., and M. Greenwald (eds.), 2016, *Report of the DOE Workshop on Management, Analysis, and Visualization of Experimental and Observational Data – The Convergence of Data and Computing,* technical report, Lawrence Berkeley National Laboratory, Berkeley, Calif., May, https://science.energy.gov/~/media/ascr/pdf/programdocuments/docs/ascr-eod-workshop-2015-report_160524.pdf, accessed January 2017.

Bethel, E.W., M. Greenwald, K. Kleese van Dam, M. Parashar, S.M. Wild, and H.S. Wiley, 2016, "Management, Analysis, and Visualization of Experimental and Observational Data — The Convergence of Data and Computing," In *Proceedings of the 2016 IEEE 12th International Conference on eScience*, Baltimore, Maryland, October.

Brandt, J.M., A.C. Gentile, M.T. Showerman, J., Enos, J. Fullop, and G.H. Bauer, 2014,"Large-scale Persistent Numerical Data Source Monitoring System Experiences," in *Proceedings of the 2014 Cray User Group*.

Brightwell, R., S. Goudy, A. Rodrigues, and K.D. Underwood, 2006, "Implications of Application Usage Characteristics for Collective Communication Offload. IJHPCN 4(3/4): 104–116.

Carns, P., K. Harms, W. Allcock, C. Bacon, S. Lang, R. Latham, and R. Ross, 2011, "Understanding and Improving Computational Science Storage Access through Continuous Characterization," *ACM Transactions on Storage (TOS)* 7(3).

DOE (U.S. Department of Energy), 2009, *Technology Readiness Assessment Guide,* DOE G 413.3-4, https://www.directives.doe.gov/directives-documents/400-series/0413.3-EGuide-04, accessed January 2007.

DOE, 2015, "Data Management, Visualization, and Analysis of Experimental and Observational Data (EOD) Workshop," Extreme Scale Research, National Nuclear Security Administration, September 29, 2015, to Thursday, October 1, http://extremescaleresearch.labworks.org/events/data-management-visualization-and-analysis-experimental-and-observational-data-eod-workshop, accessed January 2017.

Fahey, M., N. Jones, and B. Hadri, 2010, "The Automatic Library Tracking Database," *Proceedings of the Cray User Group 2010*, Edinburgh, United Kingdom.

Friesen, B., A. Almgren, Z. Lukić, G. Weber, D. Morozov, V. Beckner, and M. Day, 2016, "In situ and In-transit Analysis of Cosmological Simulations," *Computational Astrophysics and Cosmology*, Simulations, Data Analysis and Algorithms, 3:4, DOI: 10.1186/s40668-016-0017-2.

Heroux, M.A., and G. Allen, 2016, *Computational Science and Engineering Software Sustainability and Productivity (CSESSP) Challenges Workshop Report,* Networking and Information Technology Research and Development (NITRD) Program, September, https://www.nitrd.gov/PUBS/CSESSPWorkshopReport.pdf., accessed January 2017.

Hoffmann, H., and M. Maggio, 2014, "Optimizing Performance under Power Constraints through Resource Management," *11th International Conference on Autonomic Computing (ICAC'14)*, Philadelphia, PA, June.

Huck, K.A., A.D. Malony, R. Bell, and A. Morris, 2005, "Design and Implementation of a Parallel Performance Data Management Framework," ICPP 2005:473–482.

Huck, K., S. Shende, A. Malony, H. Kaiser, A. Porterfield, R. Fowler, and R. Brightwell, 2013, "An Early Prototype of an Autonomic Performance Environment for Exascale," in *Proceedings of the 3rd International Workshop on Runtime and Operating Systems for Supercomputers (ROSS '13)*. doi:10.1145/2491661.2481434.

HSF (HEP Software Foundation), undated, http://hepsoftwarefoundation.org/, accessed January 2017.

Johansen, H., L.C. McInnes, D. Bernholdt, J. Carver, M. Heroux, P. Jones, B. Lucas, A. Siegel, and T. Ndousse-Fetter, 2014, *Software Productivity for Extreme-Scale Science,* Report on DOE Workshop, January 13–14, http://www.orau.gov/swproductivity2014/SoftwareProductivity WorkshopReport2014.pdf, accessed January 2017.

NERSC, undated, "NERSC Web ToolKit (NEWT)," https://newt.nersc.gov, accessed January 2017.

OSTP (Office of Science and Technology Policy), 2013, Memorandum for the Heads of Executive Departments and Agencies, Executive Office of the President, February 22, https://www.whitehouse.gov/sites/default/files/microsites/ostp/ostp_public_access_memo_2013.pdf, accessed January 2017.

Palmer, J.T., S.M. Gallo, T.R. Furlani, M.D. Jones, R.L. DeLeon, J.P. White, N. Simakov, A.K. Patra, J.M. Sperhac, T. Yearke, R. Rathsam, M. Innus, C.D. Cornelius, J.C. Browne, W.L. Barth, R.T. Evans, 2015, "Open XDMoD: A Tool for the Comprehensive Management of High-Performance Computing Resources," *Computing in Science and Engineering* 17(4):52–62.

PerfSONAR, undated, "What Is perfSONAR?" http://www.perfsonar.net/about/what-is-perfsonar/, accessed January 2017.

Porterfield, A., R. Fowler, S. Bhalachaundra, B. Rountree, D. Deb, R. Lewis, and B. Blanton, 2015, "Application Runtime Variability and Power Optimization for Exascale Computers," *International Workshop on Runtime and Operating Systems for Supercomputers*, Portland, Oregon, June.

Robinson, T.W., and N.D. Stringfellow, 2013, "Recent Enhancements to the Automatic Library Tracking Database Infrastructure at the Swiss National Supercomputing Centre," *Proceedings of the Cray User Group 2013*, https://cug.org/proceedings/cug2013_proceedings/includes/files/pap162.pdf.

Rűde, U., K. Willcox, L.C. McInnes, Hans De Sterck, et al., 2016, "Research and Education in Computational Science and Engineering," available via https://arxiv.org/abs/1610.02608, submitted to *SIAM Review*.

Sreepathi, S., E.F. D'Azevedo, B. Philip, and P.H. Worley, 2016, "Characterization and Optimization of Applications Using Topology-Aware Task Mapping on Large Supercomputers," *ICPE 2016*: 225–236.

Ushizima, D., H. Bale, W. Bethel, P. Ercius, B. Helms, H. Krishnam, L. Grinberg, M. Haranczyk, A. Macdowell, K. Odziomek, D. Parkinson, T. Perciano, R. Ritchie, and C. Yang, 2016, "IDEAL: Images across Domains, Experiments, Algorithms and Learning," *Journal of Minerals, Metals and Materials*, 68(11):2963–2972.

van Dam, K., E. Stephan, B. Raju, I. Altintas, T. Elsethagen, and S. Krishnamoorthy, 2015, "Enabling Structured Exploration of Workflow Performance Variability in Extreme-scale Environments," in *Proceedings of MTAGS15*, co-located with SC15, November.

Zhang, X., H. Abbasi, K.A. Huck, and A.D. Malony, 2016, "WOWMON: A Machine Learning-based Profiler for Self-adaptive Instrumentation of Scientific Workflows," ICCS 2016:1507–1518.

Zhao, Z., 2014, "Automatic Library Tracking Database at NERSC," First International Workshop on HPC User Support Tools (HUST 2014), New Orleans, Louisiana, USA, November, http://www.nersc.gov/assets/altdatNERSC.pdf.

This page is intentionally left blank.

6 ACRONYMS AND ABBREVIATIONS

ALCF	Argonne Leadership Computing Facility
ALTD	Automatic Library Tracking Database
API	application program interface
Argonne	Argonne National Laboratory
ASCR	Office of Advanced Scientific Computing Research
AWS	Amazon Web Services
BER	Biological and Environmental Science
BES	Basic Energy Sciences
BIOS	basic input/output system
CI	continuous integration
CORAL	Collaboration Argonne, Oak Ridge, Livermore
CPU	central processing unit
CS	computer science
DES	Dark Energy Survey
DOD	U.S. Department of Defense
DOE	U.S. Department of Energy
DOE SC	DOE Office of Science
DTN	data transfer node
ECP	Exascale Computing Program
FES	Fusion Energy Sciences
FOA	Funding Opportunity Announcement
FOM	figure of merit
FPGA	field-programmable gate array
FY	fiscal year
GA	general availability
GB	gigabyte
GPU	graphics processing unit
GUI	graphical user interface

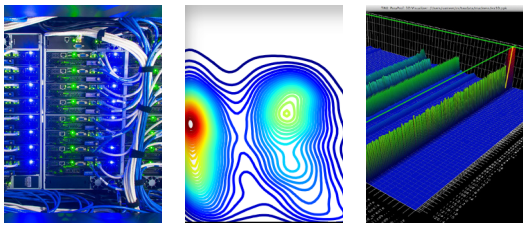
HACC	Hardware/Hybrid Accelerated Cosmology Code
HBM	high-bandwidth memory
HEP	High-Energy Physics
HPC	high-performance computing
HPDC	high-performance distributed computing
HSF	HEP Software Foundation
HTC	high-throughput computing
HW	hardware
IEEE	Institute of Electrical and Electronics Engineers
I/O	input/output
IP	internet protocol
ISV	independent software vendor
LANL	Los Alamos National Laboratory
LBL	Lawrence Berkeley National Laboratory
LCF	leadership computing facility
MPI	message parsing interface
MSR	machine-specific register
NDA	non-disclosure agreement
NERSC	National Energy Research Scientific Computing Center
NEWT	NERSC Web ToolKit
NGN	next-generation networking
NP	Nuclear Physics
NRE	non-recurring engineering
NSCI	National Strategic Computing Initiative
NSF	National Science Foundation
NUMA	non-uniform memory access
NVM	non-volatile memory
NVMe	non-volatile memory express
NVRAM	non-volatile random-access memory
OLCF	Oak Ridge Leadership Computing Facility
OS	operating system(s)
OSTP	Office of Science and Technology Policy

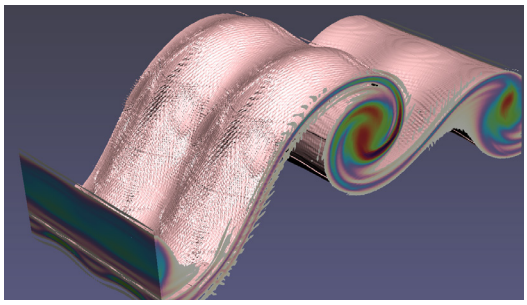
PAPI	performance application programming interface
PB	petabyte
PCI	peripheral component interconnect
PI	principal investigator
R&D	research and development
RFP	request for proposal
SC	DOE Office of Science
SciDAC	Scientific Discovery through Advanced Computing
SDK	software development kit
SDMAV	scientific data management, analysis, and visualization at extreme scale
SDN	software-defined networking
SENSE	SDN for End-to-end Networked Science at the Exascale
SNS	Spallation Neutron Source
SOC	Security Operations Center
SW	software
TAU	tuning and analysis utilities
TAUdb	TAU database
TB	terabyte
TRL	technology readiness level
UVM	unified virtual memory
WAN	wide area network

This page is intentionally left blank.

ASCR

ADVANCED SCIENTIFIC COMPUTING RESEARCH





APPENDICES: MEETING MATERIALS

An Office of Science review sponsored by
Advanced Scientific Computing Research

This page is intentionally left blank.

APPENDIX A: ADVANCED SCIENTIFIC COMPUTING RESEARCH ORGANIZING COMMITTEE AND MEETING PARTICIPANTS

A.1 ASCR ORGANIZING COMMITTEE

Jeffrey Vetter, Oak Ridge National Laboratory

Ann Almgren, Lawrence Berkeley National Laboratory

Phil DeMar, Fermi National Accelerator Laboratory

Richard Coffey, Argonne Leadership Computing Facility

Katherine Riley, Argonne Leadership Computing Facility

A.2 ASCR SPONSORS AND REPRESENTATIVES

Carolyn Lauzon, DOE Office of Advanced Scientific Computing Research

Lucy Nowell, DOE Office of Advanced Scientific Computing Research

Ceren Susut, DOE Office of Advanced Scientific Computing Research

A.3 ASCR MEETING PARTICIPANTS

Jim Ahrens	Los Alamos National Laboratory
Frank Alexander	Los Alamos National Laboratory
Ann Almgren	Lawrence Berkeley National Laboratory
James Ang	Sandia National Laboratories
Rick Archibald	Oak Ridge National Laboratory
Scott Baden	Lawrence Berkeley National Laboratory
Pavan Balaji	Argonne National Laboratory
Jeff Banks	Rensselaer Polytechnic Institute
Debbie Bard	National Energy Research Scientific Computing Center
Pete Beckman	Argonne National Laboratory
Janine Bennett	Sandia National Laboratories
David Bernholdt	Oak Ridge National Laboratory
Wes Bethel	Lawrence Berkeley National Laboratory
Steve Binkley	DOE
George Bosilca	University of Tennessee
David Brown	Lawrence Berkeley National Laboratory
Franck Cappello	Argonne National Laboratory
Laura Carrington	San Diego Supercomputer Center
Luis Chacon	Los Alamos National Laboratory
Christine Chalk	DOE

Edmond Chow	Georgia Tech
Richard Coffey	Argonne Leadership Computing Facility
Susan Coghlan	Argonne National Laboratory
Claire Cramer	DOE
Eli Dart	ESnet
Vince Dattoria	DOE
Phil Demar	Fermi National Accelerator Laboratory
Jack Deslippe	National Energy Research Scientific Computing Center
Lori Diachin	Lawrence Livermore National Laboratory
Lei Ding	State University of New York
Jeff Donatelli	Lawrence Berkeley National Laboratory
Milo Dorr	Lawrence Livermore National Laboratory
Mark Fahey	Argonne Leadership Computing Facility
Mary Fitzpatrick	Argonne National Laboratory
Robert Fowler	University of North Carolina-Chapel Hill
Todd Gamblin	Lawrence Livermore National Laboratory
Ada Gavrilovska	Georgia Tech
Al Geist	Oak Ridge National Laboratory
Richard Gerber	National Energy Research Scientific Computing Center
Tim Germann	Los Alamos National Laboratory
David Goodwin	DOE
Salman Habib	Argonne National Laboratory
James Hack	Oak Ridge National Laboratory
Mary Hall	University of Utah
Bill Harrod	DOE
Barb Helland	DOE
Michael Heroux	Sandia National Laboratories
Judy Hill	Oak Ridge Leadership Computing Facility
Jeff Hittinger	Lawrence Livermore National Laboratory
Adolfy Hoisie	Pacific Northwest National Laboratory
Jeffrey Hollingsworth	University of Maryland
Paul Hovland	Argonne National Laboratory
Travis Humble	Oak Ridge National Laboratory
Kate Keahey	Argonne National Laboratory
Darren Kerbyson	Pacific Northwest National Laboratory
Raj Kettimuthu	Argonne National Laboratory

Scott Klasky	Oak Ridge National Laboratory
Kerstin Kleese van Dam	Brookhaven National Laboratory
Doug Kothe	Oak Ridge National Laboratory
Dhireesha Kudithipudi	Rochester Institute of Technology
Milind Kulkarni	Purdue
Kalyan Kumaran	Argonne Leadership Computing Facility
Carolyn Lauzon	DOE
Randall Laviolette	DOE
Steven Lee	DOE
Stephen Lee	Los Alamos National Laboratory
Tom Lehman	University of Maryland
Richard Lethin	Yale, Reservoir Labs
Sven Leyffer	Argonne National Laboratory
Miron Livny	University of Wisconsin
Glenn Lockwood	Lawrence Berkeley National Laboratory
Arthur Maccabe	Oak Ridge National Laboratory
Sonia McCarthy	DOE
Pat McCormick	Los Alamos National Laboratory
Lois McInnes	Argonne National Laboratory
John Mellor-Crummey	Rice
Paul Messina	Argonne National Laboratory
Kathryn Mohror	Lawrence Livermore National Laboratory
Inder Monga	ESnet
Shirley Moore	Oak Ridge National Laboratory
Ken Moreland	Sandia National Laboratories
Reed Mosher	US Army/HPCMP
Klaus Mueller	Brookhaven National Laboratory
Todd Munson	Argonne National Laboratory
Thomas N'Dousse-Fetter	DOE
Harvey Newman	Caltech
Esmond Ng	Lawrence Berkeley National Laboratory
Lucy Nowell	DOE
Leonid Oliker	Lawrence Berkeley National Laboratory
Michael Papka	Argonne Leadership Computing Facility
Michael Parks	Sandia National Laboratories
Valerio Pascucci	University of Utah

Abani Patra	DOE
Robert Patton	Lawrence Berkeley National Laboratory
Keshav Pingali	University of Texas
Robinson Pino	DOE
Terry Quinn	Lawrence Livermore National Laboratory
Nagi Rao	Oak Ridge National Laboratory
Katherine Riley	Argonne Leadership Computing Facility
Betsy Riley	DOE
Robert Roser	Fermi National Acceleratory Laboratory
Rob Ross	Argonne National Laboratory
Lauren Rotman	ESnet/Lawrence Berkeley National Laboratory
Ponnuswamy Sadayappan	Ohio State University
Vivek Sarkar	Rice University
Catherine Schuman	Oak Ridge National Laboratory
John Shalf	Lawrence Berkeley National Laboratory
Sameer Shende	University of Oregon
Galen Shipman	Los Alamos National Laboratory
Maria Spiropulu	Caltech
Tjerk Straatsma	Oak Ridge Leadership Computing Facility
Ceren Susut	DOE
Pieter Swart	Los Alamos National Laboratory
Nathan Tallent	Pacific Northwest National Laboratory
Rajeev Thakur	Argonne National Laboratory
Rollin Thomas	National Energy Research Scientific Computing Center
John Turner	Oak Ridge National Laboratory
Jeffrey Vetter	Oak Ridge National Laboratory
Jack Wells	Oak Ridge Leadership Computing Facility
Julia White	Oak Ridge National Laboratory
Stefan Wild	Argonne National Laboratory
Samuel Williams	Lawrence Berkeley National Laboratory
Wenji Wu	Fermi National Acceleratory Laboratory
Asim YarKhan	University of Tennessee
Kathy Yelick	Lawrence Berkeley National Laboratory
Shin Yoo	Brookhaven National Laboratory
Dantong Yu	Stony Brook University, Brookhaven National Laboratory

APPENDIX B: ADVANCED SCIENTIFIC COMPUTING RESEARCH MEETING AGENDA

MONDAY, SEPTEMBER 26

8:00 pm	Chairs meet with breakout leads.

TUESDAY, SEPTEMBER 27

Time	Topic	Speaker
7:30	Registration, Refreshments	
8:30	Welcome & Introductions	Chairs
8:45	Genesis of this Meeting	Barb Helland, ASCR
9:00	View from ASCR Research	Bill Harrod, ASCR
9:50	Break	
10:15	ASCR Computing Facilities Presentation – Katherine Riley, ALCF Compute Facilities resources, plans, and activities, a brief overview of reviews ESNet resources, plans, and activities Followed by open Q&A including Facility Directors	
11:15	Survey Results and Breakout Group Organization, Chairs	
11:45	Working Lunch – Charge to Working Groups	
12:30	First Breakout Sessions (Day 1): PRODUCTION SYSTEMS ONLY Software Development – Lois McInnes HPC Architectures – Galen Shipman Distributed Computing and Networking (HPDC) – Salman Habib Data Management, Vis & Analytics, Storage – Wes Bethel	
2:30	Break (Refreshments)	
2:45	Breakout Sessions: Science Drivers	
	Second Breakout Sessions (Day 1): EARLY SYSTEMS ONLY Software Development – David Bernholdt HPC Architectures – Sam Williams Distributed Computing and Networking (HPDC) – Rob Roser Data Management, Vis & Analytics, Storage – Peter Nugent Systems Software – Kathryn Mohror	
4:45	Break	
5:00	ECP Update – Paul Messina	
5:45	ECP Q&A with Paul	

WEDNESDAY, SEPTEMBER 28

8:00	Refreshments
8:30	Plan for day and (all Tuesday breakouts) Summaries of the breakouts from the previous day. If previous day breakouts repeated, integrate the summaries perhaps.
9:30	Break and move to breakouts (see matrix)
9:45	First Breakout Sessions (Day 2): EMERGING SYSTEMS ONLY (First 4) HPC Architectures – Franck Cappello Data Management, Vis & Analytics, Storage – Ken Morel Systems Software – Jeffrey K. Hollingsworth Production Systems: Operational Data & Policies – Shirley Moore Production Systems: Systems Deployment & Support – Todd Gamblin
11:45	Prepare breakout summaries
12:00	Working Lunch including Breakout Summaries and Q&A – full group
1:30	Final Breakout Sessions
3:300	Break
3:45	Reports on Wednesday Breakouts, Breakout Leads Next steps Writing assignments
4:45	Summary and Thanks from Chairs

THURSDAY, SEPTEMBER 29

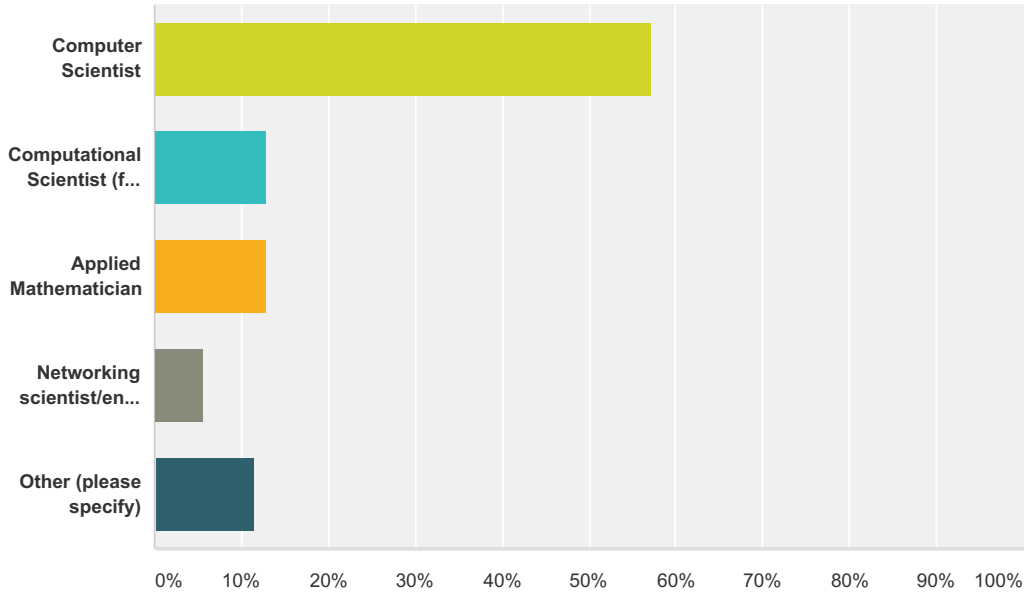
All Day	Co-chairs, Leads, Writers meet to continue working on report

APPENDIX C: PRE-REVIEW SURVEY RESULTS

This page is intentionally left blank.

Q1 What is your primary occupation?

Answered: 70 Skipped: 0

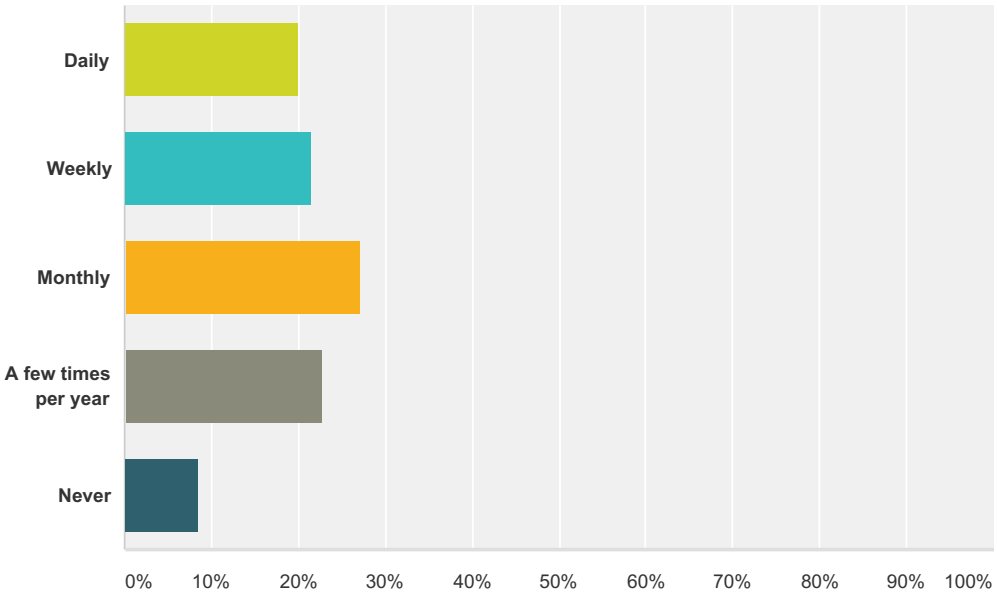


Answer Choices	Responses	
Computer Scientist	**57.14%**	40
Computational Scientist (for a specific Application or set of Applications)	**12.86%**	9
Applied Mathematician	**12.86%**	9
Networking scientist/engineer	**5.71%**	4
Other (please specify)	**11.43%**	8
Total		**70**

#	Other (please specify)	Date
1	Both Networking Scientist and Computer Scientist	8/15/2016 10:39 AM
2	Physicist, Network scientist/engineer, Distributed system developer and architect	8/15/2016 5:56 AM
3	Computer Systems Engineer	8/12/2016 12:12 PM
4	[redacted]	8/12/2016 11:03 AM
5	I am doing several different aspects of several of these categories for different projects	8/12/2016 7:25 AM
6	Computational Scientist, trained in applied math, working with many applications	8/12/2016 6:53 AM
7	Physicist and also network and systems architect	8/11/2016 11:54 PM
8	Manager	8/11/2016 11:00 PM

Q2 How frequently do you use at least one of the ASCR Computing Facilities (i.e., NERSC, OLCF, ALCF) for your ASCR-funded research?

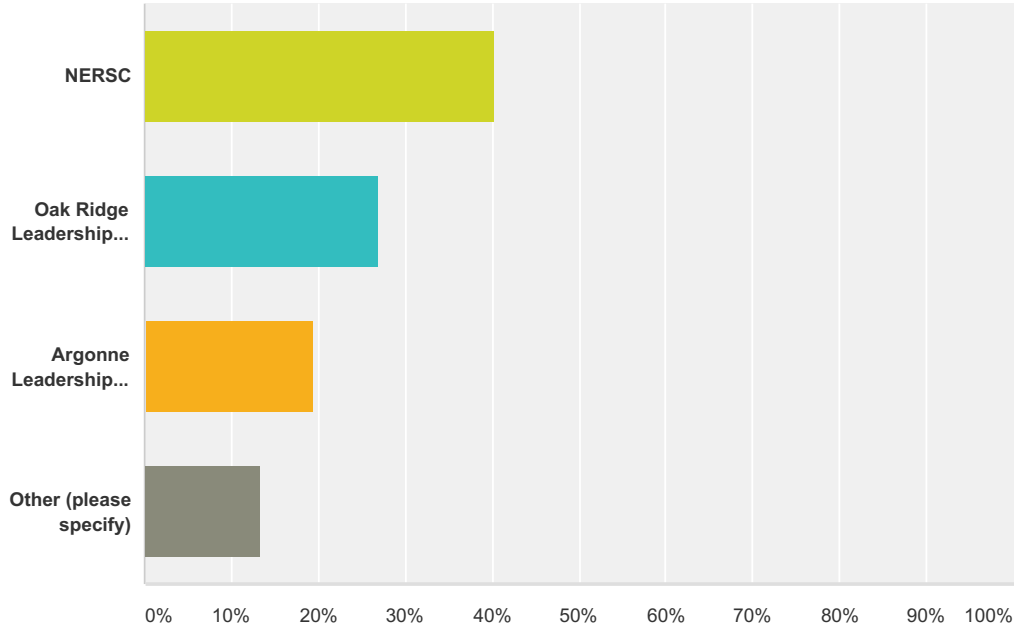
Answered: 70 Skipped: 0



Answer Choices	Responses	
Daily	**20.00%**	14
Weekly	**21.43%**	15
Monthly	**27.14%**	19
A few times per year	**22.86%**	16
Never	**8.57%**	6
Total		**70**

Q3 Which facility do you use most frequently for your ASCR-funded research?

Answered: 67 Skipped: 3

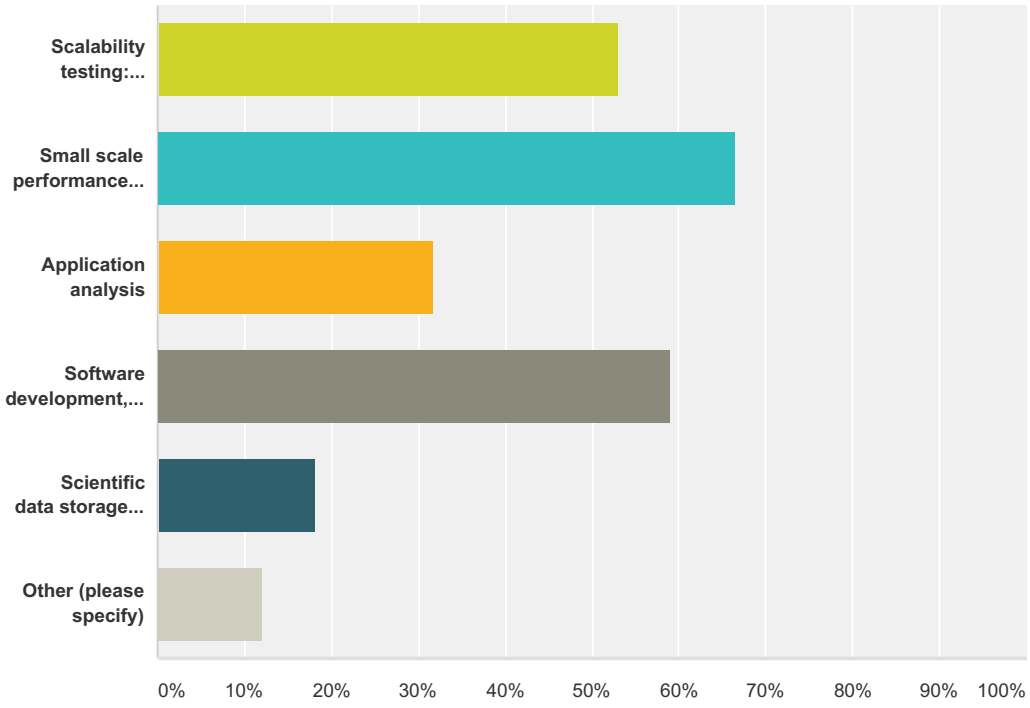


Answer Choices	Responses	
NERSC	**40.30%**	27
Oak Ridge Leadership Facility	**26.87%**	18
Argonne Leadership Facility	**19.40%**	13
Other (please specify)	**13.43%**	9
Total		**67**

#	Other (please specify)	Date
1	n/a	8/17/2016 6:08 AM
2	local computing resources and occasionally LLNL computers	8/15/2016 12:23 PM
3	ESNET 100G SDN testbed	8/15/2016 9:00 AM
4	Local departmental cluster	8/15/2016 8:16 AM
5	University resources	8/12/2016 2:55 PM
6	testing software at all facilities, in order to ensure it is ready for use by other teams who use our software	8/12/2016 6:53 AM
7	ESnet	8/12/2016 1:41 AM
8	We use ALCF and OLCF primarily as well as NERSC	8/11/2016 11:07 PM
9	50-50 NERSC and ALCF	8/11/2016 10:17 PM

Q4 During these occasions, how do you primarily use the ASCR Computing Facilities for your ASCR-funded research? (Check all that apply.)

Answered: 66 Skipped: 4



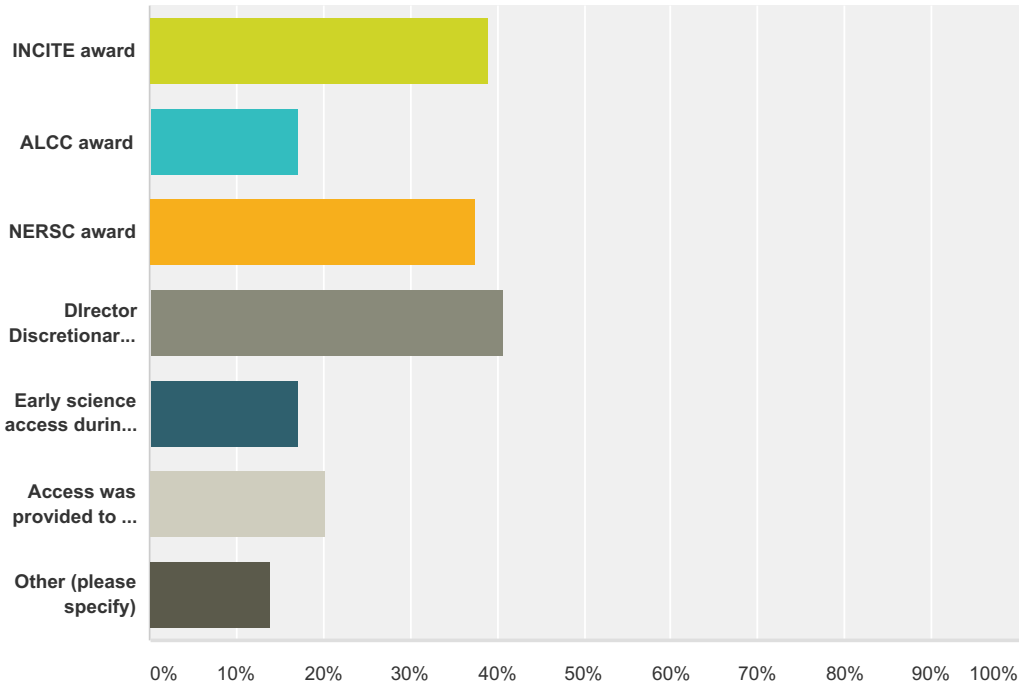
Answer Choices	Responses	
Scalability testing: Large-scale (significant fraction of the system)	**53.03%**	35
Small scale performance optimization and testing	**66.67%**	44
Application analysis	**31.82%**	21
Software development, debugging, and testing	**59.09%**	39
Scientific data storage and transfer	**18.18%**	12
Other (please specify)	**12.12%**	8
Total Respondents: 66		

#	Other (please specify)	Date
1	Typically use 512-1024 nodes for small-scale scalability testing	8/17/2016 11:56 AM
2	n/a	8/17/2016 6:08 AM
3	Production at OLCF and ALCF	8/15/2016 1:55 PM
4	System diagnosis software, and data transfers	8/15/2016 10:39 AM
5	Developing new modes of data intensive use of these Facilities	8/15/2016 5:56 AM
6	Science	8/12/2016 12:13 AM

7	distributed network and system development	8/11/2016 11:54 PM
8	Production runs as well ;-)	8/11/2016 11:07 PM

Q5 How was the ASCR Facilities system allocated to you for this specific ASCR-funded research? (Check all that apply.)

Answered: 64 Skipped: 6



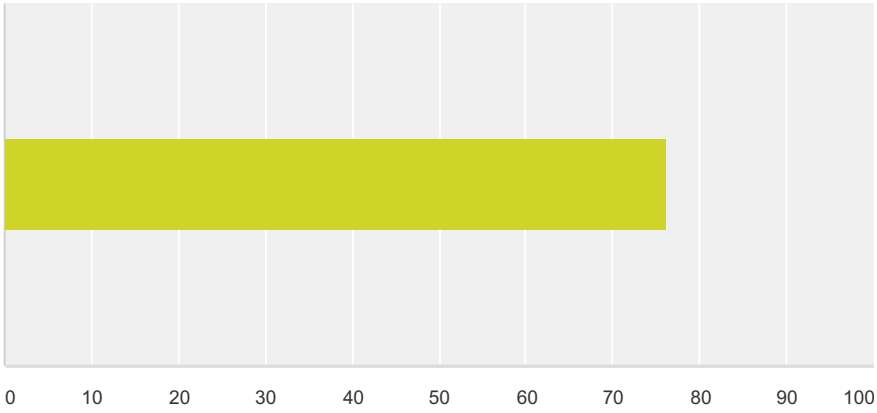
Answer Choices	Responses	
INCITE award	**39.06%**	25
ALCC award	**17.19%**	11
NERSC award	**37.50%**	24
DIrector Discretionary award	**40.63%**	26
Early science access during acceptance period	**17.19%**	11
Access was provided to a small platform not requiring an allocation	**20.31%**	13
Other (please specify)	**14.06%**	9
Total Respondents: 64		

#	Other (please specify)	Date
1	n/a	8/17/2016 6:08 AM
2	SCIDAC Partnership allocation	8/15/2016 1:45 PM
3	Latched on to an existing project	8/15/2016 10:42 AM
4	Provided by a collaborative DOD project	8/15/2016 10:39 AM
5	we use a web-based reservation system to reserve system/network resources	8/15/2016 9:00 AM
6	Runs on DOE systems were done by DOE project collaborators	8/15/2016 8:16 AM
7	Allocation through HEP for SciDAC project.	8/12/2016 11:03 AM

8	Benchmarking/testing of new platforms	8/11/2016 11:07 PM
9	staff	8/11/2016 10:21 PM

Q6 What is your overall satisfaction with the ability to perform your ASCR-funded research at ASCR Computing Facilities (i.e., NERSC, OLCF, ALCF)?

Answered: 52 Skipped: 18



Answer Choices	Average Number	Total Number	Responses
	76	3,965	52
Total Respondents: 52			

#		Date
1	85	8/17/2016 6:12 PM
2	95	8/17/2016 11:56 AM
3	20	8/17/2016 6:11 AM
4	75	8/16/2016 8:54 PM
5	80	8/16/2016 7:53 PM
6	75	8/16/2016 6:12 PM
7	80	8/16/2016 5:14 PM
8	90	8/16/2016 4:21 PM
9	75	8/16/2016 11:44 AM
10	70	8/16/2016 12:45 AM
11	50	8/15/2016 4:22 PM
12	70	8/15/2016 3:06 PM
13	76	8/15/2016 2:59 PM
14	78	8/15/2016 2:01 PM
15	85	8/15/2016 1:57 PM
16	90	8/15/2016 1:47 PM
17	75	8/15/2016 12:50 PM
18	80	8/15/2016 12:26 PM
19	63	8/15/2016 12:20 PM

20	95	8/15/2016 12:14 PM
21	90	8/15/2016 11:56 AM
22	67	8/15/2016 10:51 AM
23	90	8/15/2016 10:45 AM
24	70	8/15/2016 10:44 AM
25	85	8/15/2016 9:09 AM
26	90	8/15/2016 9:03 AM
27	50	8/15/2016 8:07 AM
28	65	8/15/2016 7:54 AM
29	73	8/15/2016 5:59 AM
30	90	8/13/2016 1:18 AM
31	90	8/12/2016 4:59 PM
32	90	8/12/2016 4:31 PM
33	90	8/12/2016 4:30 PM
34	80	8/12/2016 3:01 PM
35	75	8/12/2016 2:02 PM
36	90	8/12/2016 1:19 PM
37	91	8/12/2016 12:45 PM
38	10	8/12/2016 12:17 PM
39	90	8/12/2016 11:44 AM
40	60	8/12/2016 11:17 AM
41	90	8/12/2016 9:02 AM
42	56	8/12/2016 7:44 AM
43	80	8/12/2016 7:28 AM
44	66	8/12/2016 6:58 AM
45	67	8/12/2016 6:58 AM
46	40	8/12/2016 1:57 AM
47	87	8/12/2016 1:52 AM
48	85	8/12/2016 12:22 AM
49	70	8/11/2016 11:11 PM
50	85	8/11/2016 10:36 PM
51	96	8/11/2016 10:20 PM
52	100	8/11/2016 10:17 PM

Q7 For your ASCR-funded research, identify up to three (3) things about your experience with ASCR Facilities that you appreciate and want to stay the same.

Answered: 59 Skipped: 11

Answer Choices	Responses	
Item 1	**100.00%**	59
Item 2	**79.66%**	47
Item 3	**38.98%**	23

#	Item 1	Date
1	State of the art hardware systems	8/17/2016 6:12 PM
2	Timely availability of compute resources	8/17/2016 11:56 AM
3	Documentation	8/17/2016 6:11 AM
4	Good set of available software	8/16/2016 8:54 PM
5	Reliable operation	8/16/2016 7:53 PM
6	x	8/16/2016 6:12 PM
7	Access to computer science researchers	8/16/2016 5:14 PM
8	Responsiveness of staff	8/16/2016 4:21 PM
9	process for application for compute hours is reasonable	8/16/2016 11:44 AM
10	Linux API (almost on BG)	8/16/2016 12:45 AM
11	standard commodity machines are critical!	8/15/2016 4:22 PM
12	NX Remote Desktop (despite initial skepticism)	8/15/2016 3:06 PM
13	system availability	8/15/2016 2:59 PM
14	Hardware near the leading edge.	8/15/2016 2:01 PM
15	Phone support for accounts	8/15/2016 1:57 PM
16	state-of-the-art capabilities	8/15/2016 1:47 PM
17	ease of use of NERSC facilities	8/15/2016 12:50 PM
18	quick response by facilities staff	8/15/2016 12:26 PM
19	General sense of availability, no problem to get time.	8/15/2016 12:20 PM
20	Access to excellent facilities	8/15/2016 12:14 PM
21	help system	8/15/2016 11:56 AM
22	good customer service	8/15/2016 11:46 AM
23	Consulting services, responsiveness and helpfulness	8/15/2016 10:51 AM
24	Very good about arranging DAT access	8/15/2016 10:45 AM
25	Stable operation of computing and file systems	8/15/2016 10:44 AM
26	Excellent support, from the basics all the way to interfacing directly with the Dev/Ops staff to address complex issues	8/15/2016 10:25 AM
27	modules	8/15/2016 9:09 AM
28	System admins responded to questions/problems fast and in time	8/15/2016 9:03 AM

29	N/A	8/15/2016 8:17 AM
30	great support staff	8/15/2016 8:07 AM
31	Rapid and helpful user support.	8/15/2016 7:54 AM
32	One on one engagement with our application issues, and working towards a solution	8/15/2016 5:59 AM
33	Good user support for issue tracking	8/13/2016 1:18 AM
34	Helpline & workshops	8/12/2016 4:59 PM
35	hands-on workshops	8/12/2016 4:31 PM
36	User services groups are helpful and responsive	8/12/2016 4:30 PM
37	reliable access to machines (good uptime)	8/12/2016 3:38 PM
38	Responsive help desk re. account setup and resource allocations	8/12/2016 3:01 PM
39	Software support	8/12/2016 2:56 PM
40	Quality support	8/12/2016 2:02 PM
41	Short queue times	8/12/2016 1:19 PM
42	relatively easy access	8/12/2016 12:45 PM
43	Access to large compute resources is easy	8/12/2016 12:17 PM
44	Reliable system support	8/12/2016 12:01 PM
45	software robustness	8/12/2016 11:44 AM
46	responsiveness of support staff	8/12/2016 11:17 AM
47	System capability	8/12/2016 9:02 AM
48	stability	8/12/2016 7:44 AM
49	Ease of use	8/12/2016 7:28 AM
50	support staff are generally responsive to problems/questions	8/12/2016 6:58 AM
51	access to it, availability, reliability. Availability of a small system system for testing.	8/12/2016 6:58 AM
52	Good program = get time	8/12/2016 2:29 AM
53	Access to allocation that allows large scale runs	8/12/2016 1:57 AM
54	Accessibility	8/12/2016 1:52 AM
55	responsiveness of staff	8/12/2016 12:22 AM
56	Computing scale	8/11/2016 11:11 PM
57	High performance software availability	8/11/2016 10:36 PM
58	Performance evaluation tools (e.g., PAPI, TAU, Scalasca)	8/11/2016 10:20 PM
59	Support from facilities	8/11/2016 10:17 PM
#	**Item 2**	**Date**
1	Different versions of software libraries	8/17/2016 6:12 PM
2	Modern and reliable software installations	8/17/2016 11:56 AM
3	workshops	8/17/2016 6:11 AM
4	Regular schedule for maintenance and availability	8/16/2016 7:53 PM
5	x	8/16/2016 6:12 PM
6	User website documentation	8/16/2016 4:21 PM
7	reasonably accessible --- most of the time there is not a problem running jobs	8/16/2016 11:44 AM
8	Several compilers available	8/16/2016 12:45 AM
9	module packager (Lots of tools = good, I prefer not to build/install any software that is not mine)	8/15/2016 3:06 PM

10	software modules	8/15/2016 2:59 PM
11	Documentation and support.	8/15/2016 2:01 PM
12	Availability of technical staff	8/15/2016 1:57 PM
13	high availability	8/15/2016 1:47 PM
14	access to new technologies-- burst buffers	8/15/2016 12:50 PM
15	access to diverse architectures	8/15/2016 12:26 PM
16	Responsiveness of staff	8/15/2016 12:14 PM
17	software availability	8/15/2016 11:56 AM
18	good online documentation	8/15/2016 11:46 AM
19	Appearance of fair and equitable access to resources (ERCAP process at NERSC)	8/15/2016 10:51 AM
20	Status updates on computing and storage systems	8/15/2016 10:44 AM
21	Good availability of system resources at larger scale	8/15/2016 10:25 AM
22	support	8/15/2016 9:09 AM
23	Access to multiple sub-facilities	8/15/2016 5:59 AM
24	Good instructions on web site	8/13/2016 1:18 AM
25	Performance	8/12/2016 4:59 PM
26	applications engineers (e.g. catalysts)	8/12/2016 4:31 PM
27	Dedicated data transfer systems are much better than they were a few years ago (e.g. Globus capability)	8/12/2016 4:30 PM
28	generally reasonable queue lengths	8/12/2016 3:38 PM
29	Various resources for research activities	8/12/2016 3:01 PM
30	Compute power	8/12/2016 2:02 PM
31	Simple application process	8/12/2016 1:19 PM
32	good up times	8/12/2016 12:45 PM
33	Performance variability is relatively low	8/12/2016 12:17 PM
34	module systems	8/12/2016 11:44 AM
35	consulting for porting and optimization of codes	8/12/2016 11:17 AM
36	User Support	8/12/2016 9:02 AM
37	accessibility	8/12/2016 7:44 AM
38	people to discuss the issues with	8/12/2016 7:28 AM
39	testing at scale	8/12/2016 6:58 AM
40	NERSC can handle a lot of collaborators	8/12/2016 2:29 AM
41	Access to unique architectures	8/12/2016 1:57 AM
42	Availability	8/12/2016 1:52 AM
43	software environment	8/12/2016 12:22 AM
44	Size of allocation	8/11/2016 11:11 PM
45	Rapid responses from helpdesk	8/11/2016 10:36 PM
46	Debuggers	8/11/2016 10:20 PM
47	Overburn policy	8/11/2016 10:17 PM
#	**Item 3**	**Date**
1	Customer service	8/17/2016 6:12 PM
2	Helpdesk is very responsive via phone & email	8/17/2016 11:56 AM

3	Help available when I need it	8/16/2016 7:53 PM
4	x	8/16/2016 6:12 PM
5	Rich user environment	8/16/2016 4:21 PM
6	admins are reasonably responsive to issues	8/16/2016 11:44 AM
7	Interactive debug jobs	8/16/2016 12:45 AM
8	documentation (particularly getting started)	8/15/2016 3:06 PM
9	documentation	8/15/2016 2:59 PM
10	Reasonable turnaround times on smaller jobs.	8/15/2016 2:01 PM
11	Ability to get your software working there	8/15/2016 12:14 PM
12	Mid-year resource adjustments based upon need	8/15/2016 10:51 AM
13	Direct remote access to systems	8/15/2016 10:44 AM
14	Tools	8/12/2016 4:59 PM
15	Many events for user training	8/12/2016 3:01 PM
16	Stable resources	8/12/2016 2:02 PM
17	reliability (non-GPU systems)	8/12/2016 11:44 AM
18	availability of highly optimized libraries	8/12/2016 11:17 AM
19	Extensive, online documentation	8/12/2016 9:02 AM
20	help of staff	8/12/2016 6:58 AM
21	Stability	8/12/2016 1:52 AM
22	Machine availability	8/11/2016 10:36 PM
23	Wiki pages for tools	8/11/2016 10:20 PM

Q8 For your ASCR-funded research, identify up to three (3) things about your experience with ASCR Facilities that you want improved.

Answered: 59 Skipped: 11

Answer Choices	Responses	
Item 1	**100.00%**	59
Item 2	**66.10%**	39
Item 3	**33.90%**	20

#	Item 1	Date
1	Better availability of Interactive partitions	8/17/2016 6:12 PM
2	More accurate projections for when job will run (NERSC facilities are heavily used and can have unpredictable delays as a result)	8/17/2016 11:56 AM
3	system support for controlled computer science expeirments	8/17/2016 6:11 AM
4	Quick queues for shorter jobs	8/16/2016 8:54 PM
5	Turnaround time for small jobs	8/16/2016 7:53 PM
6	xx	8/16/2016 6:12 PM
7	More resources for computer science researchers	8/16/2016 5:14 PM
8	Better way to address login security: too complicated	8/16/2016 4:21 PM
9	policies seem to explicitly reward people who use all of their allocation, which encourages people to waste hours at the end of the year. More sensible policies would reward (or at least not punish) people for conserving compute hours.	8/16/2016 11:44 AM
10	Accelerators are difficult	8/16/2016 12:45 AM
11	I would like to lessen the emphasis on scalability	8/15/2016 4:22 PM
12	More installed profiling tools (e.g. HPCToolkit, Open|Speedshop)	8/15/2016 3:06 PM
13	None	8/15/2016 2:59 PM
14	Better environment for large scale workflows (100s of 10TF jobs.)	8/15/2016 2:01 PM
15	Better documentation of installed libraries	8/15/2016 1:57 PM
16	more development tools	8/15/2016 1:47 PM
17	remote access to systems	8/15/2016 12:50 PM
18	more timely software upgrades (CUDA at ORNL)	8/15/2016 12:26 PM
19	More opportunity to collaborate with applications scientists.	8/15/2016 12:20 PM
20	Queue responsiveness	8/15/2016 12:14 PM
21	job scheduling (long wait time)	8/15/2016 11:56 AM
22	better/more consistent file system performance.	8/15/2016 11:46 AM
23	Lower barrier of entry to use of ALCF, OLCF	8/15/2016 10:51 AM
24	Long queue times	8/15/2016 10:45 AM
25	up to date system configuration/failure information	8/15/2016 10:44 AM
26	Difficult to do software development and evaluation at smaller scale due to scheduling policies	8/15/2016 10:25 AM
27	incompatible architecture between compute and compile nodes (cross compiling)	8/15/2016 9:09 AM

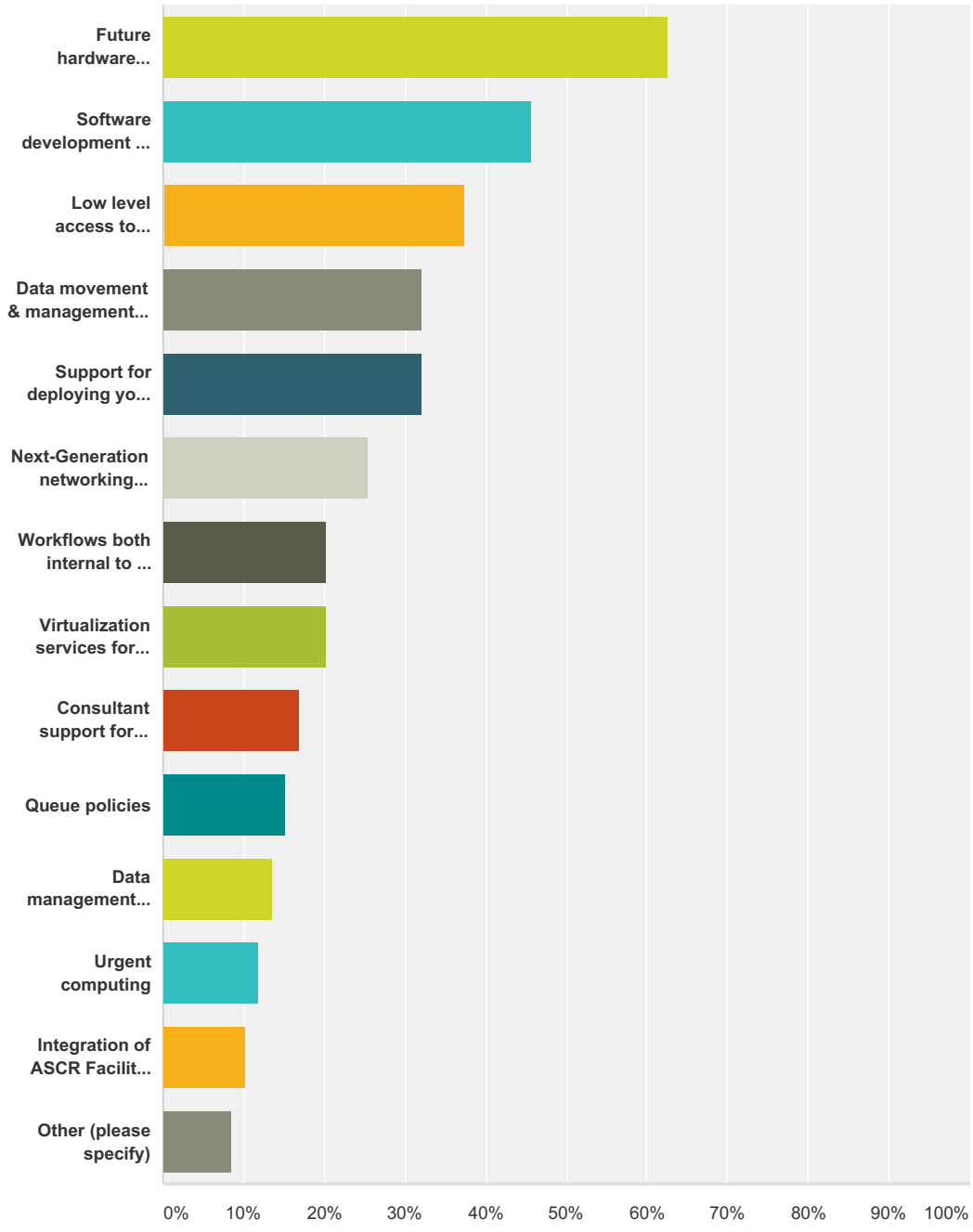
28	In busy seasons, the system/network resources we need are not readily available.	8/15/2016 9:03 AM
29	N/A	8/15/2016 8:17 AM
30	more memory per node	8/15/2016 8:07 AM
31	Better profiling and debugging tools. In particular, profiling performance of applications with GPUs.	8/15/2016 7:54 AM
32	Greater staff manpower and availability to help with integrating external software stacks and tools	8/15/2016 5:59 AM
33	NERSC's old phone consulting mode was better. Now when we make phone call, it doesn't go directly to a consultant, but transferred by someone, so we need to explain the problem at least twice. I would prefer the phone call goes directly to a consultant.	8/13/2016 1:18 AM
34	Tools	8/12/2016 4:59 PM
35	webinars to exchange best practices, e.g. code optimization, tools, etc.	8/12/2016 4:31 PM
36	Data transfer performance to/from tape using Globus is too slow	8/12/2016 4:30 PM
37	time to install new compilers and libraries	8/12/2016 3:38 PM
38	Too frequent and long maintenance	8/12/2016 3:01 PM
39	Availability	8/12/2016 2:56 PM
40	Flexible scheduling	8/12/2016 2:02 PM
41	Nothing, it's perfect	8/12/2016 1:19 PM
42	documentation	8/12/2016 12:45 PM
43	ASCR facilities are useless for system software development.	8/12/2016 12:17 PM
44	Access/availability of richer performance logs	8/12/2016 12:01 PM
45	performance tool productivity	8/12/2016 11:44 AM
46	workflow management infrastructure	8/12/2016 11:17 AM
47	More variety in allocation program	8/12/2016 9:02 AM
48	stability	8/12/2016 7:44 AM
49	Shared workspace that is on every facility, like the old AFS NERSC had, or dropbox.	8/12/2016 7:28 AM
50	sustained access over time (across multiple years) for testing software that is used by other applications	8/12/2016 6:58 AM
51	Access to logs of the system (all)	8/12/2016 6:58 AM
52	Improve queue turn-around-time	8/12/2016 2:29 AM
53	Lack of consistent software environment/well maintained and tested package manager for wide range of software	8/12/2016 1:57 AM
54	Time in the batch queue for large scale jobs	8/12/2016 1:52 AM
55	filesystem performance	8/12/2016 12:22 AM
56	File systems	8/11/2016 11:11 PM
57	Keeping software up to date	8/11/2016 10:36 PM
58	Licenses for commercial tracing tools (Vampir)	8/11/2016 10:20 PM
59	Multi-year allocations	8/11/2016 10:17 PM
#	**Item 2**	**Date**
1	Higher priority for jobs using smaller partitions	8/17/2016 6:12 PM
2	Allow submission of multiple jobs in debug queue (NERSC allows this, but ORNL does not)	8/17/2016 11:56 AM
3	staff that support computer science experimentation	8/17/2016 6:11 AM
4	Access to a development platform that may run an experimental software stack	8/16/2016 7:53 PM
5	xx	8/16/2016 6:12 PM
6	Automated regression testing capabilities	8/16/2016 4:21 PM

7	there is lack of support for bleeding edge efforts that use the machine in unusual ways; it is hard to get support for changes that benefit only one or a few users, even if the experiment may ultimately benefit everyone	8/16/2016 11:44 AM
8	I/O performance (highly variable latencies; metadata server response)	8/16/2016 12:45 AM
9	More tools on remote desktop (e.g. ParaView/VisIt, Vampir/JumpShot)	8/15/2016 3:06 PM
10	none	8/15/2016 2:59 PM
11	More even playing field between lab and academic researchers using the facilities.	8/15/2016 2:01 PM
12	Better support for shared libraries!!!	8/15/2016 1:57 PM
13	something different than 2 factor auth	8/15/2016 12:50 PM
14	support deployment of research software	8/15/2016 12:26 PM
15	Multi-year ALCC/INCITE awards	8/15/2016 12:14 PM
16	Operational policy adjustments to accommodate data-centric workloads (see Sep 2015 ASCR EOD report)	8/15/2016 10:51 AM
17	more/better tutotials on using different parts	8/15/2016 10:44 AM
18	automatic cleanup of /lustre/scratch too frequent	8/15/2016 9:09 AM
19	local storage on node to cache data that won't fit in memory	8/15/2016 8:07 AM
20	Better online knowledge-base, especially in effectively utilizing all of the machine's computing resources.	8/15/2016 7:54 AM
21	Simplified secure remote access and interaction with VO's environement via specified network ports	8/15/2016 5:59 AM
22	Data driven computing	8/12/2016 4:59 PM
23	Storage allocations on fast filesystems are too small	8/12/2016 4:30 PM
24	Shortage of personal home directory storages	8/12/2016 3:01 PM
25	Data oriented resources	8/12/2016 2:02 PM
26	I need turnaround times in seconds or minutes to do test-driven development.	8/12/2016 12:17 PM
27	Line between facility and research far too blurry, facilities seem to be increasingly competing (!) with ASCR research divisions	8/12/2016 12:01 PM
28	standard programming model support on GPUs	8/12/2016 11:44 AM
29	distributed data management infrastructure	8/12/2016 11:17 AM
30	Flexible workflow management systems	8/12/2016 9:02 AM
31	accessibility	8/12/2016 7:44 AM
32	A common data repository for DOE which then manages the data across the centers, for ease of use.	8/12/2016 7:28 AM
33	access for regular automated testing in addition to testing initiated by a person	8/12/2016 6:58 AM
34	Set expectations appropriately.	8/12/2016 2:29 AM
35	Lack of automated access to facilities for build/test/continuous integration and other software quality assurance tools	8/12/2016 1:57 AM
36	Possibility for dedicated time for large scale short jobs	8/12/2016 1:52 AM
37	differences between login and compute nodes	8/12/2016 12:22 AM
38	Storage allocations	8/11/2016 11:11 PM
39	More flexible allocation requests	8/11/2016 10:36 PM
#	**Item 3**	**Date**
1	Providing sample projects for beginners	8/17/2016 6:12 PM
2	access to emerging systems	8/17/2016 6:11 AM
3	x	8/16/2016 6:12 PM
4	Multiplicity of batch schedulers	8/16/2016 12:45 AM
5	module packager (mediocre UI, what config did I have when I built this?)	8/15/2016 3:06 PM

6	none	8/15/2016 2:59 PM
7	Awards tied to ongoing experimental programs	8/15/2016 12:14 PM
8	Restrictions on ASCR facilities staff competing for research dollars; they should be doing facilities work, not competitive research	8/15/2016 10:51 AM
9	simpler email/helpline for detailed technical areas	8/15/2016 10:44 AM
10	multiple GPUs per node	8/15/2016 8:07 AM
11	Platform Updates	8/12/2016 4:59 PM
12	Non-availability of basic benchmark results	8/12/2016 3:01 PM
13	More memory	8/12/2016 2:02 PM
14	My work requires both root-level software access and supervisor-mode hardware access.	8/12/2016 12:17 PM
15	reliability (GPU-accelerated systems)	8/12/2016 11:44 AM
16	commonality of tools, interfaces, and environment across facilities	8/12/2016 11:17 AM
17	Lack of support for containers and virtualization	8/12/2016 1:57 AM
18	IO	8/12/2016 1:52 AM
19	Queueing protocols	8/11/2016 11:11 PM
20	More small-scale computing options for initial development and debugging	8/11/2016 10:36 PM

Q9 What breakout topics would you like to discuss at our September workshop? (Select up to four (4) topics)

Answered: 59 Skipped: 11



Answer Choices	Responses
Future hardware requirements and testbeds	62.71% 37

Software development and testing environments for research software (e.g., regression testing, continuous integration, systems for rapid development cycles)	**45.76%**	27
Low level access to hardware and system software for experimentation	**37.29%**	22
Data movement & management considerations within HPDC ecosystems such as Characteristics of data movement & buffering for experimental/observational science within HPDC ecosystems. Modeling of experimental/observational science work flows	**32.20%**	19
Support for deploying your software on facility systems	**32.20%**	19
Next-Generation networking support for high-performance distributed computing (HPDC) ecosystems at HPC facilities, such as Next generation network infrastructure & services. Access & security policies to support those services. Testbed environments for development, prototyping, staging for production deployment	**25.42%**	15
Workflows both internal to a site and across distributed sites	**20.34%**	12
Virtualization services for deployment and experimentation	**20.34%**	12
Consultant support for CS/Math research issues	**16.95%**	10
Queue policies	**15.25%**	9
Data management including archival, scratch, streaming, provenance, etc.	**13.56%**	8
Urgent computing	**11.86%**	7
Integration of ASCR Facilities with other resources used by ASCR Scientists	**10.17%**	6
Other (please specify)	**8.47%**	5
Total Respondents: 59		

#	Other (please specify)	Date
1	Requirements for in situ visual analysis	8/17/2016 6:17 PM
2	ease of use/access to facility systems/data across labs	8/17/2016 6:11 AM
3	1. Operational policy adjustments for experimental/observational data projects (see ASCR 2015 EOD report); 2. Discussion about including within mission focus topics like long-term data dissemination, storage, archival (see ASCR 2015 EOD report); 3. Discuss parameters for engaging facilities staff in exploratory projects with non-facilities stakeholders, and limiting facilities personnel from leading research proposals	8/15/2016 10:54 AM
4	Data analytics and machine learning workflows	8/15/2016 8:10 AM
5	support for performance tools and programming models	8/12/2016 11:45 AM

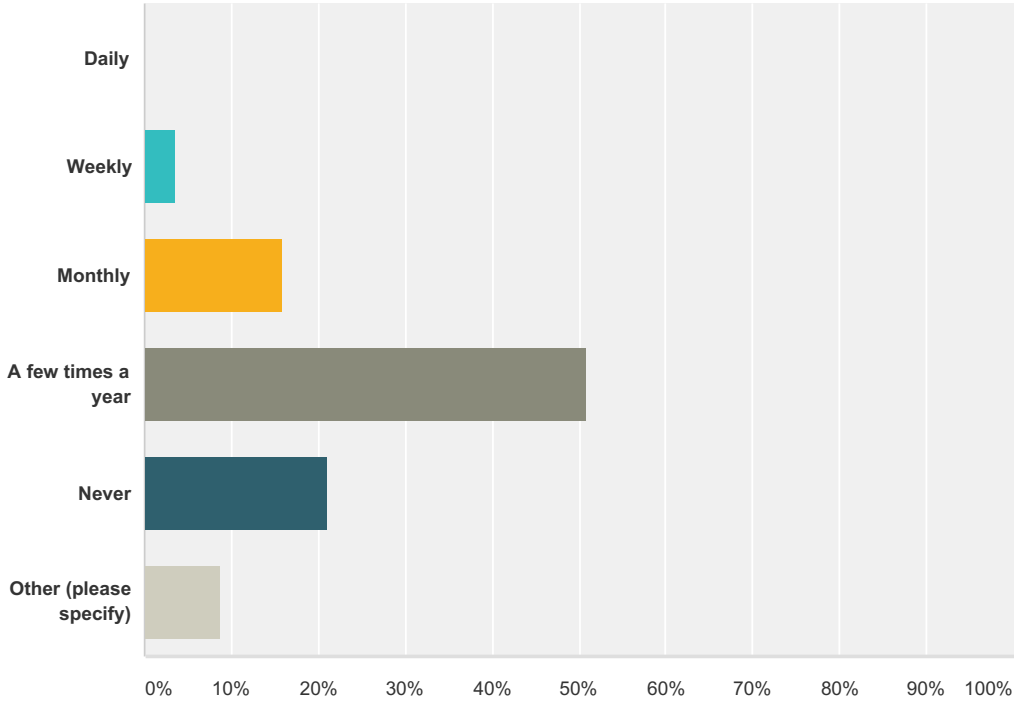
Q10 Do you have any other topics we should be sure to discuss at the meeting and possibly include in the final report?

Answered: 18 Skipped: 52

#	Responses	Date
1	Support for interactive simulation monitoring and steering	8/17/2016 6:17 PM
2	Opportunity to provide input to ASCR RFPs regarding desirable hardware and software capabilities. (We always seem to be missing critical hardware and/or software capabilities that might be avoidable.)	8/16/2016 7:59 PM
3	x	8/16/2016 6:12 PM
4	Overall application/workflow characterization and application sensitivity to hardware. If you're trying to determine your requirements then you need to fully characterize your applications in terms of sensitivity to hardware features to determine where to invest in the ecosystem to get the most out of your investment or where it is most needed.	8/15/2016 3:05 PM
5	The role of academic CS and Math research and software in the ASCR facilities ecosystem.	8/15/2016 2:05 PM
6	Research pipeline: role of academic partners, graduate students migrating into lab	8/15/2016 12:31 PM
7	Availability of system operational data, including up-time and failure rates and types, to support development of math and software for diagnosis and resilience.	8/15/2016 10:47 AM
8	Designing exascale systems to support data analytics and machine learning workflows in addition to modeling and simulation.	8/15/2016 8:10 AM
9	My main research focus is mathematics and HPC for DOE experimental science. Would like to see this discussed	8/12/2016 5:02 PM
10	Balance of high-speed persistent storage (what is today a scratch or project filesystem) with compute cycles. Current plans for Exascale put too much emphasis on compute, with too little emphasis on storage.	8/12/2016 4:33 PM
11	no	8/12/2016 3:39 PM
12	We need room (< $60K annually) in our project budget to support a small development system (< $250K total).	8/12/2016 12:20 PM
13	How does the facility support collaborations between users? Maybe this is covered by workflows and networking, but the point itself would be interesting to discuss.	8/12/2016 9:06 AM
14	Access to vendor failure injection tools (software) for resilience research. Broader access to (current and future) experimental platforms accross labs (FPGA, neurosynaptic chip, quantum computing chip, etc.)	8/12/2016 7:20 AM
15	[redacted] we are talking with staff at ALCF, OLCF, and NERSC about next steps in partnering with them on outreach/training topics related to software engineering and productivity (following partnership in a recent webinar series on 'Best Practices for HPC Software Developers', see https://www.olcf.ornl.gov/2016/05/31/olcf-alcf-nersc-co-host-hpc-software-webinar-series/). It would be useful to get in put from the broader community (possibly at this workshop, if that would fit with the broader agenda) on topics of interest for training.	8/12/2016 7:07 AM
16	Collaboration among facilities on software build, package management, reproducibility	8/12/2016 2:03 AM
17	ASCR facility policies for supporting "big science" experiments across multiple DOE/SC offices.	8/11/2016 11:16 PM
18	Performance evaluation tools.	8/11/2016 10:21 PM

Q11 How frequently do you disseminate information about your ASCR-funded research products/results to the ASCR Computing Facilities?

Answered: 57 Skipped: 13

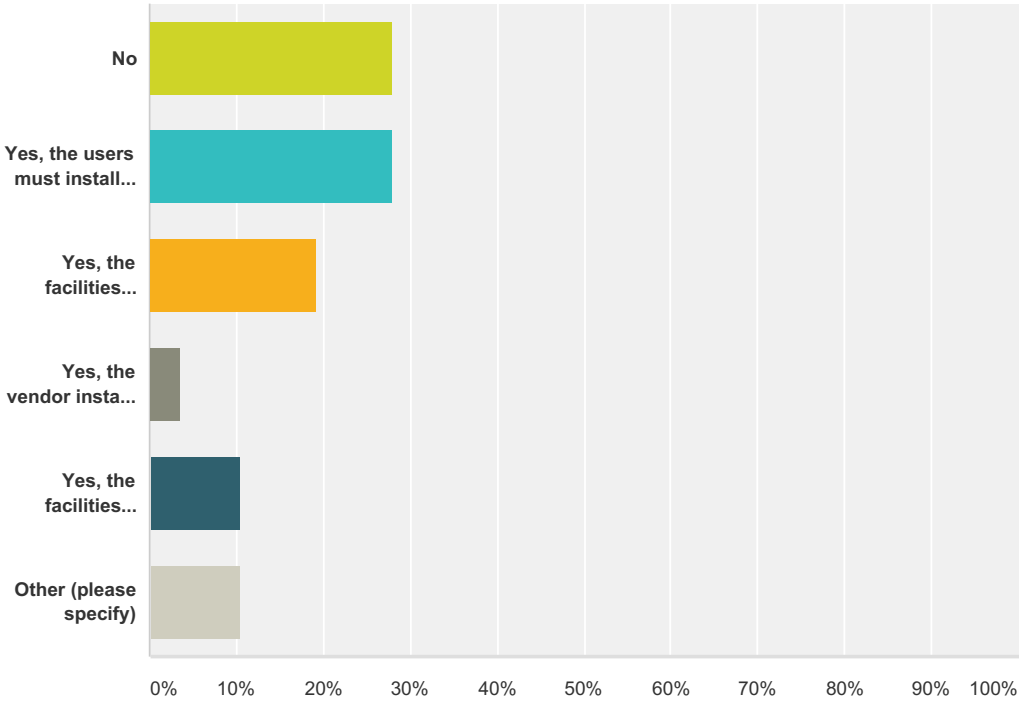


Answer Choices	Responses	
Daily	**0.00%**	0
Weekly	**3.51%**	2
Monthly	**15.79%**	9
A few times a year	**50.88%**	29
Never	**21.05%**	12
Other (please specify)	**8.77%**	5
Total		**57**

#	Other (please specify)	Date
1	We acknowledge the use of facilities in our CS publications	8/17/2016 11:58 AM
2	To some facilities regularly, others almost never	8/16/2016 8:03 PM
3	On occasion, when the opportunity arises. There are not many such opportunities.	8/15/2016 10:56 AM
4	publications	8/12/2016 12:48 PM
5	This question is vague. I share a building with a facility and chat with these folks all the time about our software.	8/12/2016 12:43 PM

Q12 Are your ASCR-funded products deployed at ASCR Computing Facilities?

Answered: 57 Skipped: 13

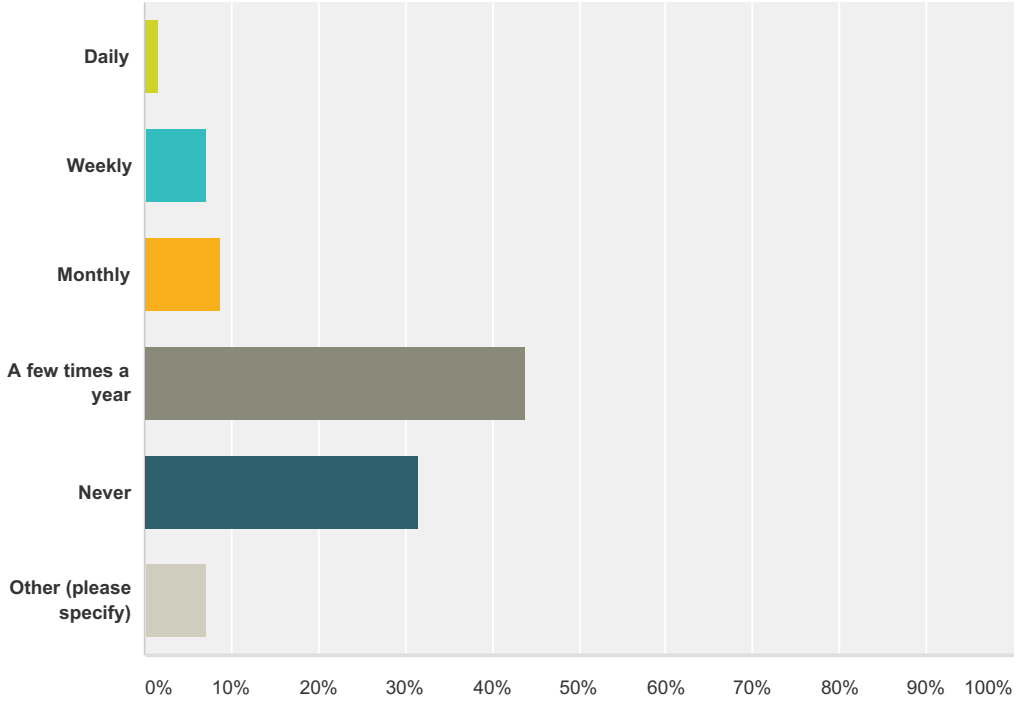


Answer Choices	Responses	
No	**28.07%**	16
Yes, the users must install it.	**28.07%**	16
Yes, the facilities staff installs it, but my team supports it.	**19.30%**	11
Yes, the vendor installs it and supports it.	**3.51%**	2
Yes, the facilities staff installs and supports it.	**10.53%**	6
Other (please specify)	**10.53%**	6
Total		**57**

#	Other (please specify)	Date
1	Occasional testing but no permanent installation yet	8/17/2016 6:21 PM
2	My team installs the software in user space, but gets help with the facilities staff for installing software dependencies e.g., Libxml	8/17/2016 11:58 AM
3	There is shared support between vendor and facilities staff	8/15/2016 3:12 PM
4	installation at NERSC for X-Stack, but not publicly accessible	8/15/2016 12:32 PM
5	Actually, it is a combination of facilities installing software and users also installing newer versions of software, and our team also installing software for regular testing	8/12/2016 7:09 AM
6	My NNSA-funded product is, and we jointly support it	8/12/2016 2:06 AM

Q13 How frequently do you communicate with the ASCR Computing Facilities about their research challenges in order to inform your ASCR-funded research objectives and directions?

Answered: 57 Skipped: 13

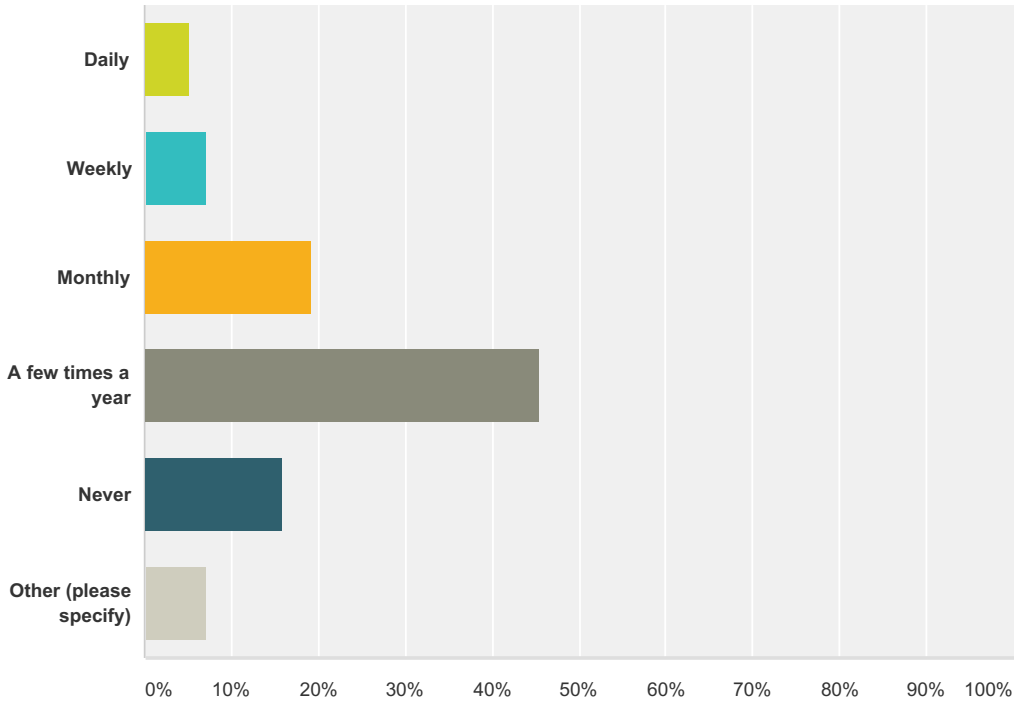


Answer Choices	Responses	
Daily	1.75%	1
Weekly	7.02%	4
Monthly	8.77%	5
A few times a year	43.86%	25
Never	31.58%	18
Other (please specify)	7.02%	4
Total		57

#	Other (please specify)	Date
1	We communicate indirectly with Facilities through our computer scientist collaborators at ORNL and LBNL	8/17/2016 11:58 AM
2	Occasionally, when the opportunity arises, which is infrequent	8/15/2016 10:56 AM
3	about once a year	8/12/2016 3:40 PM
4	The IDEAS software productivity project has regular communication with ALCF, OLCF, and NERSC about issues related to software productivity at the facilities	8/12/2016 7:09 AM

Q14 How frequently do you need access to data and information about the ASCR Computing Facilities in order to facilitate your ASCR-funded research objectives and plans? Examples of this information include usage statistics for software tools, platform reliability data, job scheduling traces, I/O profiles, and instances of performance anomalies.

Answered: 57 Skipped: 13



Answer Choices	Responses	
Daily	5.26%	3
Weekly	7.02%	4
Monthly	19.30%	11
A few times a year	45.61%	26
Never	15.79%	9
Other (please specify)	7.02%	4
Total		57

#	Other (please specify)	Date
1	Daily updates on status of pending jobs and projections of when they will run	8/17/2016 11:58 AM
2	To my knowledge, there has been a scarcity of such data.	8/16/2016 8:03 PM

3	More details like this would be very helpful to my research	8/12/2016 2:06 PM
4	I don't rely on this bc it's not available on a continuous basis. If it was, I would use it all the time	8/12/2016 2:06 AM

Q15 Other comments?

Answered: 7 Skipped: 63

#	Responses	Date
1	ASCR Computing facilities represent the trailing edge of technology and support for CS research. When the systems are made generally available through an ASCR Computing Facility, it is typically irrelevant for a competitive CS research agenda. Furthermore, there is limited information about upcoming systems provided to us so that we can prepare for these pending systems. In some cases, the facilities personnel are in direct competition with the research personnel, even though facilities personnel already have funding for their positions.	8/17/2016 6:17 AM
2	The PEAC INCITE award has been a great help to the computer science community. An obstacle to installing ASCR-funded software for broader use at ASCR facilities is that ASCR offers funding for research only. I have been explicitly told by DOE headquarters that ASCR does not fund software maintenance and/or user support, unless they consider that being funded exclusively through the SciDAC program, which was outside the scope of my discussion. Well-regarded software packages depend upon an unbroken sequence of research and development grants to remain available. I have been advised by DOE headquarters that any funds for maintaining or supporting software should come from the leadership computing facilities and that I should solicit funds directly from centers. As someone who has had annual one-year grants from some centers, sporadic one-year support from others, and none from others, I can honestly say that supporting long-term staff on a sequence of small, uncoordinated, one-year contracts---each with a SOW that includes some new development---is a difficult task. Tackling some of the big tasks, e.g. adding scalable I/O to an existing package, is more than offered support from any one organization will fund.	8/16/2016 8:17 PM
3	Additional potential discussion topics: What are viable paths/alternatives for deploying products from the research community at HPC facilities? Related, what is a viable support model?	8/15/2016 11:00 AM
4	There is a significant potential for developing applied math and computer science methods to assess the overall effectiveness (availability, resilience, etc.) of ASCR facilities; for example, fault detection methods, risk analysis methods using game theory, and machine learning based analytics of operations data, can be developed based on data that would help assess the overall performance of the complex of ASCR Facilities. This approach will tie R&D areas more closely with facilities, and in fact, lead to unique, important areas that are unlikely to be addressed by outside communities.	8/15/2016 10:57 AM
5	Several communities will just be learning to use the next generation of ASCR Facilities, and will encounter diverse challenges and some common challenges. Just identifying these and coming up with initial ideas as to how to meet the needs will be very valuable. But ongoing discussions will be needed (perhaps online) to help the communities' understanding and orientation towards solutions progress.	8/15/2016 6:10 AM
6	Thanks!	8/12/2016 4:34 PM
7	Thanks for your work in organizing the workshop!	8/12/2016 7:10 AM


U.S. DEPARTMENT OF
ENERGY