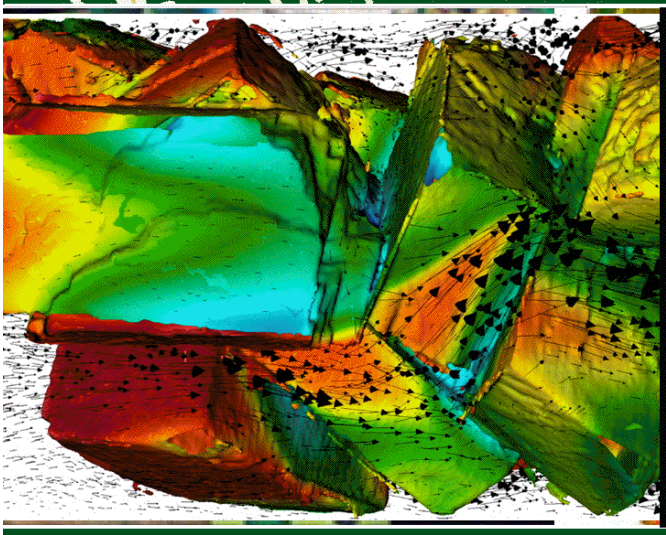
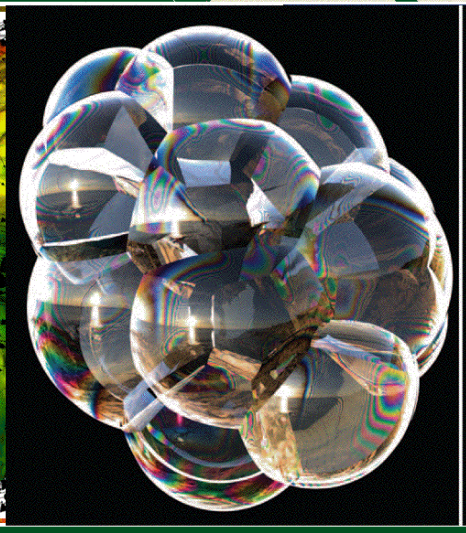
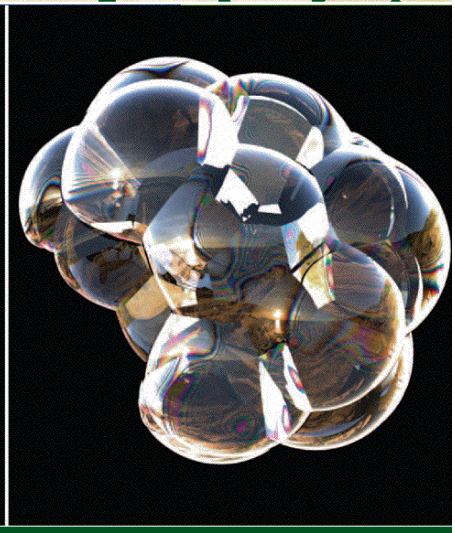
Large Scale Computing and Storage Requirements for Advanced Scientific Computing Research: Target 2017

Report of the NERSC Requirements Review
Conducted January 15, 2014







NERSC

BERKELEY LAB

U.S. DEPARTMENT OF ENERGY
Office of Science

DISCLAIMER

herein do not necessarily state or reflect those of the United States Government or any agency thereof. Copyrights to portions of this report (including graphics) are reserved by original copyright holders or their assignees, and are used by the Government's license and by permission. Requests to use any images must be made to the provider identified in the image credits.

Ernest Orlando Lawrence Berkeley National Laboratory is an equal opportunity employer.

Ernest Orlando Lawrence Berkeley National Laboratory

University of California

Berkeley, California 94720 USA

NERSC is funded by the United States Department of Energy, Office of Science, Advanced Scientific Computing Research (ASCR) program. David Goodwin is the NERSC Program Manager and Karen Pao serves as the Advanced Scientific Computing Research (ASCR) allocation manager for NERSC.

NERSC is located at the Lawrence Berkeley National Laboratory, which is operated by the University of California for the U.S. Department of Energy under contract DE-AC02-05CH11231.

This work was supported by the Directors of the Office of Science and the Office of Advanced Scientific Computing Research.

This is LBNL report LBNL-6978E published April 2015.

Large Scale Computing and Storage Requirements for Advanced Scientific Computing Research: Target 2017

Report of the HPC Requirements Review

Conducted January 15, 2014

Berkeley, California

DOE Office of Science

Office of Advanced Scientific Computing Research (ASCR)

National Energy Research Scientific Computing Center (NERSC)

Editors

Richard A. Gerber, NERSC

Harvey J. Wasserman, NERSC

Large Scale Computing and Storage Requirements for Advanced Scientific Computing Research: Target 2017

Table of Contents

1 Executive Summary

The National Energy Research Scientific Computing Center (NERSC) is the primary computing center for the DOE Office of Science, serving approximately 5,000 users working on some 700 projects that involve nearly 700 codes in a wide variety of scientific disciplines. In addition to large-scale computing and storage resources NERSC provides support and expertise that help scientists make efficient use of its systems.

In January 2014, NERSC and DOE's Office of Advanced Scientific Computing Research (ASCR) held a review to characterize High Performance Computing (HPC) and storage requirements for ASCR computational research through 2017. This review is the eleventh in a series that began in 2009 and it is the second for ASCR. The report from the earlier (2010) NERSC ASCR review is available at http://www.nersc.gov/science/hpc-requirements-reviews/target-2014/.

The latest review revealed several key requirements, in addition to achieving its goal of characterizing ASCR computing and storage needs. High-level findings are:

1. **To meet ASCR objectives, researchers need computing and data resources that continue to grow exponentially.**
2. **ASCR researchers will require software applications, libraries, and tools that will run efficiently on many-core architectures.**
3. **New resources are required to support data analytics, visualization, and archiving.**
4. **Code teams need help porting applications, libraries, frameworks, and tools to run well on next-generation architectures.**

This report expands upon these key points and adds others. The results are based upon representative samples, called "case studies," of the needs of science teams within ASCR. The case study topics and review attendees were selected by the NERSC meeting coordinators and ASCR program managers to represent the ASCR production computing workload. Prepared by the review participants, the case studies contain a summary of science goals, methods of solution, current and future computing requirements, and special software and support needs. Also included are strategies for computing in the highly parallel "many-core" environment that is expected to dominate HPC architectures over the next 10 years.

2 Advanced Scientific Computing Research Mission

The Advanced Scientific Computing Research (ASCR) program discovers, develops, and deploys the computational and networking capabilities that enable researchers to analyze, model, simulate, and predict complex phenomena important to the Department of Energy. Advanced mathematics and computing provide the foundation for models and simulations, which permit scientists to gain insights into problems ranging from bioenergy and climate change to Alzheimer's disease. ASCR and its predecessor programs have led these advances for the past thirty years by supporting the best applied math and computer science research, delivering world class scientific simulation facilities, and working with discipline scientists to deliver exceptional science.

ASCR's basic research and computing facilities are world class. The Research Division supports research and development in Applied Mathematics, Computer Science, and Next Generation Networks. The Research Division disseminates and further expands ASCR's computational expertise and intellectual resources through its Computational Partnerships with science organizations in the Office of Science. These partnerships are realized through SciDAC Institutes, SciDAC Scientific Computations Applications Partnerships, and Exascale Co-Design.

The Facilities Division is responsible for three supercomputing facilities – facilities that house some of the world's fastest supercomputers – at the National Energy Research Scientific Computing Center (NERSC), the Oak Ridge Leadership Computing Facility (OLCF), and the Argonne Leadership Computing Facility (ALCF), and – as well as the Energy Sciences Network (ESnet) that facilitates scientific collaborations and the sharing of scientific data. ASCR is guided by science needs and requirements of applications that are critical to the DOE and the nation. Today, modeling and simulation are integral parts of the "scientific method." A good simulation can inform experiment design to assure the return of high-quality experimental data. A high-fidelity simulation is indispensable for the analysis of physical phenomena. The demand for scientific rigor and defensibility in modeling and simulation requires verification, validation, and uncertainty quantification (V&V and UQ). With the unprecedented data available today and becoming even more available in the future (from both observations and simulations), data analytics and data management are rapidly gaining importance as interdisciplinary research and development areas. The demand for computing resources, in terms of processor hours, data transmission and storage, application software, workflow, analysis tools, HPC support, etc., will only increase.

At the same time, high-performance computers are undergoing architectural changes. It is generally agreed that power density already limits the frequency at which semiconductor chips can operate, and future simulation speedups will come from exploiting increased fine-grained parallelism on energy-efficient many- and multi-core processors. These constraints of physics and engineering are forcing a paradigm shift in the way scientific simulation and analysis codes are written and executed. Instead of optimizing the total number of calculations, programmers will need to put a premium on maximizing data locality and reducing data movement. Many computational scientists have not experienced such a paradigm shift. This may spur changes in basic understanding of algorithms, operating systems, programming models, performance analysis and tuning– in other words, to prepare for exascale computing, much of the basic research supported by ASCR may undergo significant changes in directions and focus as well!

Today ASCR is at a critical juncture. Demand for high-performance computing is increasing rapidly at the same time high-performance computing paradigms are changing radically. It

is in this climate that this requirement workshop attempts to cover areas of ASCR research and development that may become more prominent in the future and help understand computing needs in a rapidly changing landscape.

3 About NERSC

The National Energy Research Scientific Computing (NERSC) Center, which is supported by the U.S. Department of Energy's Office of Advanced Scientific Computing Research (ASCR), serves more than 5,000 scientists working on over 700 projects of national importance. Operated by Lawrence Berkeley National Laboratory (LBNL), NERSC is DOE's mission science high-performance computing facility, supporting scientists in all Office of Science research programs. These scientists, working remotely from DOE national laboratories; universities; other federal agencies; and industry, use NERSC resources and services to further the research mission of the Office of Science (SC). While focused on DOE's missions and scientific goals, research conducted at NERSC spans a range of scientific disciplines, including physics, materials science, energy research, climate change, and the life sciences. This large and diverse user community runs hundreds of different application codes. Results obtained using NERSC facilities are citied in about 1,500 peer reviewed scientific papers per year. NERSC activities and scientific results are also described in the center's annual reports, newsletter articles, technical reports, and extensive online documentation. In addition to providing computational support for projects funded by the Office of Science program offices (ASCR, BER, BES, FES, HEP and NP), NERSC directly supports the Scientific Discovery through Advanced Computing (SciDAC[1]) and ASCR Leadership Computing Challenge[2] Programs, as well as several international collaborations in which DOE is engaged. In short, NERSC supports the computational needs of the entire spectrum of DOE open science research.

The DOE Office of Science supports three major High Performance Computing Centers: NERSC and the Leadership Computing Facilities at Oak Ridge and Argonne National Laboratories. NERSC has the unique role of being solely responsible for providing HPC resources to all open scientific research areas sponsored by the Office of Science.

This report illustrates NERSC alignment with, and responsiveness to, DOE program office needs; in this case, Advanced Scientific Computing Research. The large number of projects supported by NERSC, the diversity of application codes, and its role as an incubator for scalable application codes present unique challenges to the center. However, as demonstrated its users' scientific productivity, the combination of effectively managed resources, and excellent user support services the NERSC Center continues its 40-year history as a world leader in advancing computational science across a wide range of disciplines.

For more information about NERSC visit the web site at http://www.nersc.gov.

[1] http://www.scidac.gov

[2] http://science.energy.gov/~/media/ascr/pdf/incite/docs/Allocation_process.pdf

4 Meeting Background and Structure

In support of its mission to provide world-class HPC systems and services for DOE Office of Science research NERSC regularly gathers user requirements. In addition to requirements reviews NERSC collects information through the Energy Research Computing Allocations Process (ERCAP), workload analyses, an annual user survey, and discussions with DOE program managers and scientists who use the facility.

In January 2014, ASCR and NERSC held a review to gather HPC requirements for current and future science programs supported by ASCR. This report is the result.

This document presents a number of findings, based upon a representative sample of projects conducting research supported by ASCR. The case studies were chosen by the DOE Program Office Managers and NERSC staff to provide broad coverage in both established and incipient ASCR research areas. Most of the domain scientists at the review were associated with an existing NERSC project, or "repository" (abbreviated later in this document as "repo").

Each case study contains a description of current and future science, a brief description of computational methods used, and a description of current and future computing needs. Since supercomputer architectures are trending toward systems with chip multiprocessors containing hundreds or thousands of cores per socket and millions of cores per system, participants were asked to describe their strategy for computing in such a highly parallel, "many-core" environment.

Requirements presented in this document will serve as input to the NERSC planning process for systems and services, and will help ensure that NERSC continues to provide world-class resources for scientific discovery to scientists and their collaborators in support of the DOE Office of Science, Office of Advanced Scientific Computing Research.

NERSC and ASCR have been conducting requirements workshops for each of the six DOE Office of Sciences offices that allocate time at NERSC (ASCR, BER, BES, FES, HEP, and NP). A first round of meetings was conducted between May 2009 and May 2011 for requirements with a target of 2014; this included a January 2011 ASCR review. A second round of meetings, of which this is the fifth, will target needs for 2017. Reports from all previous NERSC requirements reviews are available on the NERSC web site.

A specific goal for this review was to explore the extent to which library and other supporting software would be ready for the transition to energy-efficient many-core processing. This is especially important because NERSC has announced the planned procurement of a many-core supercomputer named "Cori", scheduled to be installed in 2016. A major goal of NERSC is to transition its broad user workload to Cori and follow-on architectures. Many NERSC user projects rely on software libraries provided by some of the participants. However, since authors of this software were invited to the review primarily to discuss software readiness, and since development of the software is generally more about algorithm and code development than about large-scale simulation, case studies based on these libraries do not always include numerical estimates of future NERSC resource requirements.

Specific findings from the review follow.

5 Meeting Demographics

5.1 Participants

5.1.1 DOE and NERSC

Name	Affiliation	Title or Role
Barbara Helland	DOE / ASCR	ASCR Facilities Division Director
Dave Goodwin	DOE / ASCR	NERSC Program Manager
Karen Pao	DOE / ASCR	Program Manager
Sudip Dosanjh	NERSC	NERSC Director
Katie Antypas	NERSC	Services Department Head
Richard Gerber	NERSC	Senior Science Advisor, Meeting Organizer
Harvey Wasserman	NERSC	Meeting Organizer
Jack Deslippe	NERSC	Materials Science Application Support

5.1.2 Domain Scientists

Name	Institution	Area of Interest	NERSC Repo(s)
Mark Adams	Lawrence Berkeley National Lab	Applied Numerical Algorithms Group	m1411, m1797, m1516, m1489
John Bell	Lawrence Berkeley National Lab	Group Leader, Center for Computational Sciences and Engineering	mp111
Jed Brown	Argonne National Lab	PETSc, scalable solvers for implicit multiphysics	m1489
Suren Byna	Lawrence Berkeley National Lab	Scientific data management	m1248
Phillip Colella	Lawrence Berkeley National Lab	Applied Numerical Algorithms Group Lead	m1411
James Demmel	University of California, Berkeley	Numerical linear algebra libraries; communication avoiding algorithms	mp156

Quincey Koziol	HDF Group	Principal software architect for the HDF5 software project	m888
Michael Heroux	Sandia National Labs	Algorithm development; parallel implementation of solver components	
Sherry Li	Lawrence Berkeley National Lab	Sparse matrix computations; parallel algorithm design and optimization; numerical linear algebra	mp127
Dan Martin	Lawrence Berkeley National Lab	Applied Numerical Algorithms Group	m1795 m1041
Prabhat	Lawrence Berkeley National Lab	Scientific data management, parallel I/O, and scientific visualization	m636,
Abhinav Sarje	Lawrence Berkeley National Lab	Parallel algorithms and applications on emerging parallel architectures; performance optimization	m1270, m88
David Trebotich	Lawrence Berkeley National Lab	Applied numerical algorithms; porous media transport	m1792, m1516
Chao Yang	Lawrence Berkeley National Lab	Computational mathematics	m1027

5.1.3 Observers

Name	Institution	Title
Kathy Yelick	Lawrence Berkeley National Lab	Associate Director for Computing Sciences
Jonathan Carter	Lawrence Berkeley National Lab	Computing Sciences Area Deputy and Computational Research Division Deputy Director
Greg Bell	Lawrence Berkeley National Lab	Scientific Networking Division Director
Paul Messina	Argonne Leadership Computing Facility	ALCF Director of Science
Bert de Jong	Lawrence Berkeley National Lab	Scientific Computing Group Lead
Scott Parker	Argonne Leadership Computing Facility	Application Performance Engineer
Tjerk Straatsma	Oak Ridge Leadership Computing Facility	Group Leader for Scientific Computing
Judy Hill	Oak Ridge Leadership Computing Facility	Computational Scientist
Eli Dart	ESnet	Network Engineer

5.2 NERSC Projects Represented by Case Studies

NERSC projects represented at the review are listed in the table below, along with computing and storage resources each used in 2013. These projects accounted for about 76 percent of computer time, 88 percent of shared global ("project") disk space, and 77 percent of archival storage used by ASCR projects at NERSC in 2013.

NERSC Project ID (Repo)	NERSC Project Title	NERSC Project PI	Review Speaker(s)	Hours Used at NERSC 2013 (M)	Archive Data at NERSC 2013 (TB)	Shared Data on Disk 2013 (TB)
mp111	*Simulation and Analysis of Reacting Flow*	Bell	Bell	22	2,608	13
m1792 (ALCC)	*Chombo-Crunch: Advanced Simulation of Subsurface Flow and Reactive Transport Processes Associated with Carbon Sequestration*	Trebotich	Trebotich	63	274	1.8
m1516	*Advanced Simulation of Pore Scale Reactive Transport Processes Associated with Carbon Sequestration*	Trebotich	Trebotich	33	249	0
m1027	*Numerical Algorithms and Parallel Implementations for Electronic and Nuclear Structure Analysis*	Yang	Yang	7.3	5.6	2.8
m1041	*Projections of Ice Sheet Evolution*	E. Ng	D. Martin	0.5	20.8	1.1
m1489	*Composable Hierarchically Nested Solvers*	L. Curfman Mcinnes	J. Brown	0.4	0	0
mp156	*Linear Algebra Algorithms on High Performance Computers*	J. Demmel	J. Demmel	0.4	0	0
mp127	*High Performance Sparse Matrix Algorithms*	E. Ng	S. Li	1.5	1,127	0.8
m636	*High Performance Visualization, Analytics, and I/O - Mantissa*	W. Bethel	Prabhat	5.1	184	33
m1248	*Scientific Data Management Research*	John Wu	S. Byna	4.0	53	248
sdmstor	*Storage for the Scientific Data Management Research Group*	A. Shoshani	S. Byna	0	501	0
m88	*The Sustained Performance, Energy and Resilience Institute*	L. Oliker	A. Sarje	4.3	1.4	0.2
Total of projects represented by case studies				**141**	**5,024**	**301**
All ASCR at NERSC 2013				**186**	**5,706**	**389**
Percent of NERSC ASCR 2013 allocation represented by case studies				**76%**	**88%**	**77%**

6 Findings

6.1 Summary of Requirements

The following is a summary of requirements derived from the case studies.

1. **To meet ASCR objectives, researchers need computing and data resources that continue to grow exponentially.**
    a. Researchers need 2.3 billion hours in 2017, a 12-fold increase over hours used at NERSC in 2013.
    b. Scientists will need to store more than 78 PB of archival data storage in 2017, about 14 times more than in 2013.
    c. Collaboratory teams will need 3 petabytes of shared online data in 2017, compared to only 300 TB of shared data space at NERSC in 2013.
2. **ASCR researchers will require software applications, libraries, and tools that will run efficiently on many-core architectures.**
    a. Many existing software packages need to be ported, optimized and supported on next-generation platforms like Cori.
    b. In order to provide applications, libraries, and tools to users that run efficiently on many-core architectures, software developers need sustained support from DOE.
    c. Development and measurement tools are needed to optimize codes on new systems.
3. **New resources are required to support data analytics, visualization, and archiving.**
    a. Researchers want NERSC to take leadership in provisioning high-performance I/O subsystems of HPC systems, an area in which they feel NERSC is lagging, in order to push the state of the art.
    b. A system for co-locating simulation and analysis with dedicated data management nodes is needed to support extreme scale computing and analysis.
    c. Burst buffer technology is crucial for supporting *in-situ* and *in-transit* analysis.
    d. Global shared project disk space is needed for permanent data storage and sharing.
    e. Enhanced quotas on scratch disk space are needed to accommodate large simulations and analysis.
    f. An I/O Quality of Service is needed to assure that I/O time is predictable and help to less than a few percent of an application's total run time.
    g. Faster data transfer to archival storage is needed.
4. **Code teams need help porting applications, libraries, frameworks, and tools to run well on next-generation architectures.**
    a. Developers need access to consulting, training classes, documentation, and online tutorials.
    b. Information about, and access to, new hardware as early as possible is needed to prepare for future systems.

6.2 Additional Observations

Participants at the meeting made several observations that are not listed in the high-level findings, the most significant of which are listed here.

- **Readiness for next-generation architectures (many-core) varies.** Some groups have working codes that make good use of GPUs, while others are just starting to experiment with porting to many-core processors.
- **It is difficult to get access to an entire machine at NERSC.** Other projects that would like, or need, to run full-configuration jobs have difficulty doing so in the NERSC environment.
- **Stable systems and seamless software upgrades are highly valued.**
- **NERSC consulting and account support are also highly valued.**

6.3 Requirements Summary

The following tables list the 2017 computational hours, archival storage, and shared disk storage needed at NERSC for research represented by the case studies in this report. "Total Scaled Requirement" at the end of the tables represents the hours needed by all 2013 ASCR NERSC projects if increased by the same factor as that needed by the projects represented by the case studies. All computational hours in this report are normalized to Hopper-equivalent (NERSC MPP) hours.

6.3.1 Computing

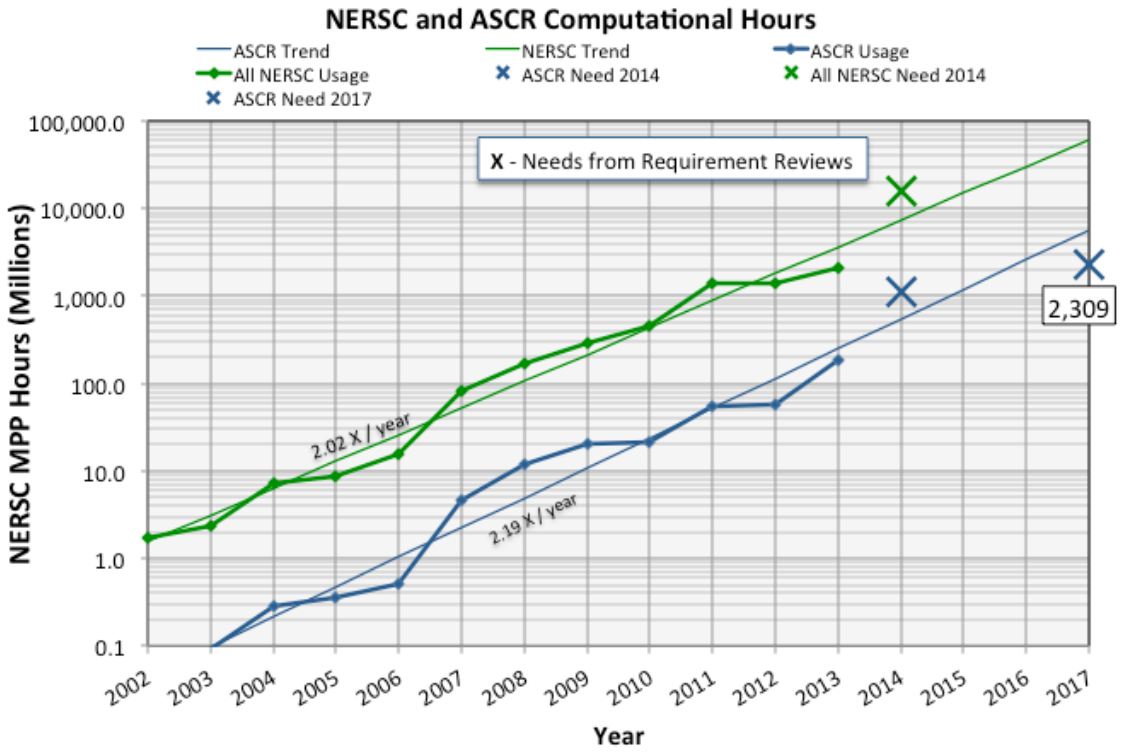
Case Study Title	Repo(s)	PI	Hours Needed in 2017	
			Million Hours	Factor Increase
Simulation and Analysis of Reacting Flows	mp111	Bell	500	22.5
Simulation of Pore Scale Reactive Transport Processes Associated with Carbon Sequestration	m1792, m1516	Trebotich	500	5.1
Numerical Algorithms for Electronic and Nuclear Structure Analysis	m1027	Yang	75	10.2
AMR for Ice Sheet Modeling	m1041	Ng	550	1,130
PETSc – Portable Extensible Toolkit for Scientific Computation	m1489	Curfman Mcinnes	10	22.7
Linear Algebra Algorithms and High Performance Computers	m156	Demmel	3.5	8.6
Sparse Direct Solver SuperLU	mp127	Li	20	13.3
Requirements for Parallel I/O, Visualization, and Analysis	m636	Bethel	50	9.8
Requirements for Scalable Scientific Data Management	m1248, sdmstor	Shoshani / Wu	30	7.5
Performance Optimization of Scientific Applications - MPAS-Ocean & the HipGISAXS Suite	m88, m1285, als	Lucas / Oliker / Li	20	4.6
Total Represented by Case Studies			**1,758**	**12.4**
Percent of NERSC ASCR Represented by Case Studies			**76%**	
All ASCR at NERSC Total Scaled Requirement			**2,313**	

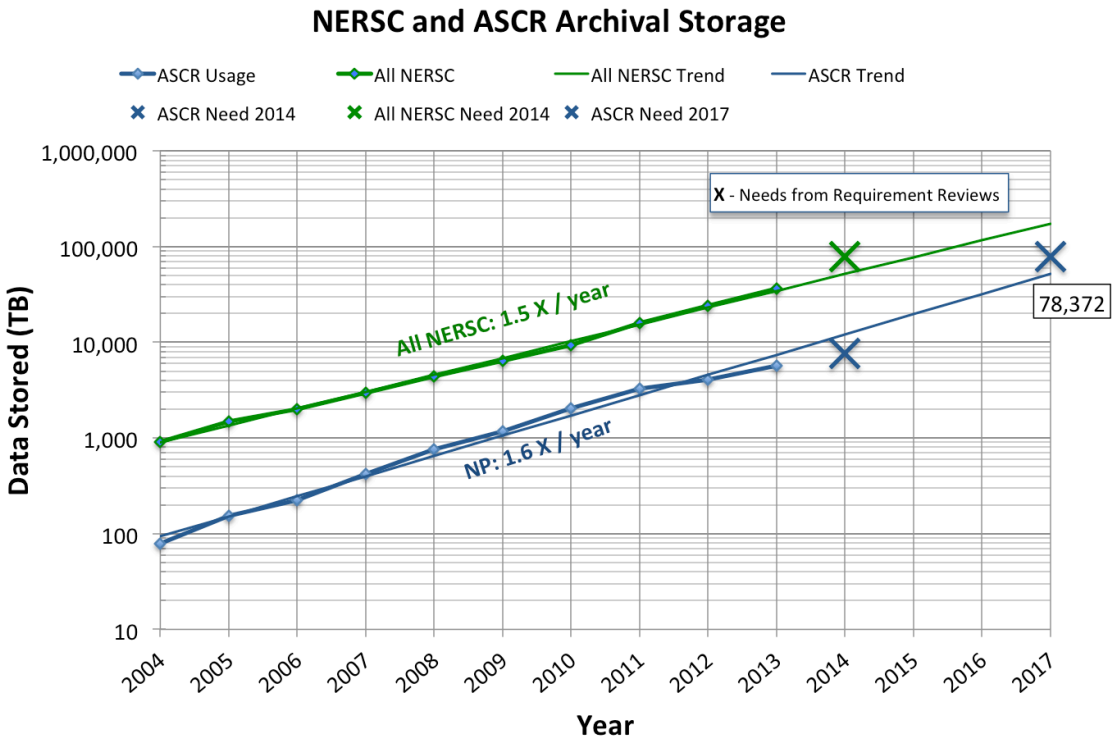
6.3.2 Storage

Case Study Title	Repos	PI	Archival Data Storage Needed in 2017		Shared Online Data Storage Needed in 2017	
			TB	Factor Increase	TB	Factor Increase
Simulation and Analysis of Reacting Flows	mp111	Bell	50,000	19	50	3.9
Simulation of Pore Scale Reactive Transport Processes Associated with Carbon Sequestration	m1792, m1516	Trebotich	5,000	20	100	56
Numerical Algorithms for Electronic and Nuclear Structure Analysis	m1027	Yang	500	89	100	36
AMR for Ice Sheet Modeling	m1041	Ng	500	24	16	15
PETSc – Portable Extensible Toolkit for Scientific Computation	m1489	Curfman Mcinnes	N/A	N/A	N/A	N/A
Linear Algebra Algorithms and High Performance Computers	m156	Demmel	N/A	N/A	N/A	N/A
Sparse Direct Solver SuperLU	mp127	Li	3,000	2.7	1	1.2
Requirements for Parallel I/O, Visualization, and Analysis	m636	Bethel	5,000	27	1,000	30
Requirements for Scalable Scientific Data Management	m1248, sdmstor	Shoshani / Wu	5,000	10	1,000	4
Performance Optimization of Scientific Applications - MPAS-Ocean & the HipGISAXS Suite	m88, m1285, als	Lucas / Oliker / Li	2	1.4	2	10
Total Represented by Case Studies			**69,002**	**14**	**2,269**	**7.6**
Percent of NERSC ASCR Represented by Case Studies			**88%**		**77%**	
All ASCR at NERSC Total Scaled Requirement			**78,372**		**2,935**	

7 ASCR and NERSC Trends

The following graphs show both ASCR and all NERSC usage of computational ("MPP") hours and archival data storage. Also included are requirements from both the first round of requirements reviews (target 2014) and this report. The projected need from ASCR for 2017 is somewhat below the historical trend, but still 12 times ASCR usage in 2013.




NERSC and ASCR Archival Storage
ASCR Usage
All NERSC
All NERSC Trend
ASCR Trend
ASCR Need 2014
All NERSC Need 2014
ASCR Need 2017
X - Needs from Requirement Reviews
78,372
All NERSC: 1.5 X / year
NP: 1.6 X / year
Data Stored (TB)
1,000,000
100,000
10,000
1,000
100
10
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
Year

8 Applied Math Case Studies

8.1 Simulation and Analysis of Reacting Flows

Principal Investigator: John Bell
NERSC Repository: mp111

8.1.1 Project Description

8.1.1.1 Overview and Context

The goal of our research is the development of high-fidelity simulation capabilities for modeling of complex physical phenomena. The resulting computational tools enable researchers to perform detailed simulations to obtain a deeper understanding of a particular problem. Our work specifically addresses problems in combustion, subsurface flow, atmospheric flow, cosmology and astrophysics.

In the area of combustion we are developing tools that enable us to quantify the interaction of turbulence and chemistry in premixed flames that are of interest in the design of next-generation, low-emissions combustion systems. Our work in subsurface flow modeling examines questions arising in carbon sequestration and environmental remediation. Our efforts in atmospheric flow focus on developing high-fidelity models of moist atmospheres that include effects of moisture physics. The methodology we develop for astrophysics is used to study supernovae as well as other low-speed astrophysical phenomena. Cosmological simulations are focused on improving estimates of cosmological parameters by comparing simulations with observation.

8.1.1.2 Scientific Objectives for 2017

The goal of our project for 2017 is to develop computational approaches and validate those new approaches on realistic applications within the areas discussed above. We consider all aspects of the target application ranging from formulation to discretization to solvers. We want to design numerical methodology that respects the underlying mathematical structure of the problem and is well matched to state-of-the-art computing technology.

8.1.2 Computational Strategies (now and in 2017)

8.1.2.1 Approach

The problems we consider all fall within the realm of phenomena described by differential equations. Although the different applications we consider have their own unique features, they share a number of common mathematical characteristics. For that reason, we can base our development on a common software framework. In particular, they are:

1. Built on CCSE's well-established BoxLib framework (ccse.lbl.gov/BoxLib)
2. Rely on iterative linear solvers for constant and/or variable coefficient elliptic and parabolic equations based on geometric multigrid
3. Implemented on 2D or 3D adaptive grid hierarchies (structured grid AMR)

4. Several of the codes incorporate particle data structures; particles may be passive or active, and fundamentally interact with the grid data during the simulation.

8.1.2.2 Codes and Algorithms

LMC: low Mach number combustion

1. Low Mach number formulation
2. Adaptive projection discretization
3. Multigrid
4. SDC coupling of processes

PMAMR: porous media

1. Total velocity with volume discrepancy formulation
2. Elliptic /parabolic pressure equation solve with multigrid
3. High-order Godunov approach for conservation laws

MAESTRO: low Mach number astrophysics

1. Low Mach number formulation with slowly varying base state
2. Nuclear reaction networks
3. Also adapted for atmospheric flows

CASTRO: compressible astrophysics

1. Self-gravity using multigrid
2. Multigroup flux limited diffusion using multigrid
3. Unsplit explicit hydro

Nyx: computational cosmology

1. Similar to CASTRO
2. Particles to represent dark matter

SMC / RNS: multicomponent, reacting, Navier-Stokes

1. High order accuracy
2. Multirate discretization based on SDC
3. Multigrid
4. Spectral Deferred Correction for process coupling

8.1.3 HPC Resources Used Today

8.1.3.1 Computational Hours

Repo mp111 consumed about 22M hours at NERSC during AY2013.

8.1.3.2 Parallelism

We typically use 6K-25K cores in our runs at NERSC. Our codes have been demonstrated to scale, for some problems, to over 100,000 CPUs on Jaguar at the Oak Ridge Leadership Computing Facility. We typically run with the fewest number of cores needed to fit the

problem within memory. We obtain better throughput and more efficient use of our allocation with this strategy.

Weak scaling has traditionally been more important for our applications. This may be changing.

8.1.3.3 Scratch Data

We generally need about 20 TB.

8.1.3.4 Shared Data

We have an mp111 project directory that we typically use to hold data for post-processing. In effect we use it as a staging area. We used 13 TB of space in this directory in 2013.

8.1.3.5 Archival Data Storage

We had about 2.6 PB stored on HPSS in 2013.

8.1.4 HPC Requirements in 2017

8.1.4.1 Computational Hours Needed

We estimate needing about 500M hours. We have often had INCITE time for more focused computational studies in a given area. The primary factor driving the need for more hours is our need to solve larger and more complex problems.

8.1.4.2 Parallelism

A reasonable target for 2017 would be 20K MPI tasks with 64 fine-grained tasks per node. The maximum might be 100K MPI x 256 fine-grained tasks.

8.1.4.3 I/O

Our applications do include checkpoint/restart. A reasonable estimate for data read and written per run in 2017 is that we would write 10-25TB / hour of I/O (assumes doing to level in situ data reduction relative to current practice) This is an estimate of 100 GB / sec bandwidth. We would prefer no more than 5-10% of the runtime devoted to I/O.

Our I/O to the scratch file system is parallel, is done using multiple nodes, is generally using large files, and we do an N x M style I/O where we write to fewer files than nodes

8.1.4.4 Future Data Needs

We would prefer a permanent project repository for data. It sometimes takes years to complete full analysis of complex problems. In the future, we would like to make data available to the larger scientific community.

8.1.4.5 Memory Required

Our simulations are primarily memory constrained. Target would be total memory of at least 100-200 TB. Not clear how low we can effectively push the memory per core. Ideally 0.5G per core would be good. This can be pushed smaller but depends a lot on available programming models. A reasonable estimate is that you would want a node to have at least 50 GB.

8.1.4.6 Emerging Technologies and Programming Models

Our software currently does not have CUDA/OpenCL extensions. We are considering putting chemistry integration onto GPUs. The software does not run in production on Titan or elsewhere using GPUs. The software does have OpenMP directives now and these are used routinely in production. We have not run on Mira or Sequoia. Porting to Intel MIC is underway. We have developed a tiling based implementation of one of our codes that got a speedup of a factor of 86 on a 61-core MIC. We collaborate with a CS researcher funded by the ExaCT Co-design Center and various X-stack projects have interacted with us on these activities.

We believe NERSC needs to:

- Provide high quality tools needed to make this transition
- Support development of new programming models needed to effectively implement algorithms on these types of architectures

We believe DOE and ASCR need to:

- Continue to fund applied math research groups working to develop algorithms for these architectures
- Provide support for software developed by these groups to facilitate availability of libraries / frameworks on new architectures

8.1.4.7 Software Applications and Tools

Software we need includes: MPI / OpenMP / C++ and F90 / Visit / htar / hypre. We also need better performance analysis tools and an improved programming model to support multicore.

8.1.4.8 HPC Services

We need consulting and account support

One possible additional service would be to provide the framework needed to make computational datasets available to the broader community. In combustion and cosmology in particular there is expressed interest from other researchers in having access to our data. Providing a gateway and suitable interfaces to support this would be potentially very useful.

8.1.4.9 Additional Data Intensive Needs

We do not currently have a data management plan for our project. We need help from NERSC, particularly in understanding the requirements for data management.

8.1.4.10 Additional Data Intensive Needs: Burst Buffer

A burst buffer would be useful in two ways

1. Stage latest checkpoint to burst buffer before jobs begin
2. Write more frequent checkpoints to burst buffer and migrate last complete checkpoint to rotating disk at end of run

8.1.4.11 Requirements Summary

NERSC Repository mp111	**Used at NERSC in 2013**	**Needed at NERSC in 2017**
Computational Hours	22M	500M
Typical number of cores used for production runs	20K	200-1000K
Maximum number of cores that can be used for production runs	200K	1.2M
Data read and written per run	1 TB / hr	25 TB / hr
Maximum I/O bandwidth	10 GB/sec	100 GB/sec
Percent of runtime for I/O	5-10%	2-5%
Scratch File System space	20 TB	200 TB
Shared filesystem space	13 TB	50 TB
Archival data	2,608 TB	50,000 TB
Memory per node	12 GB	30 GB
Aggregate memory	12 TB	200 TB

8.2 Simulation of Pore Scale Reactive Transport Processes Associated with Carbon Sequestration

Principal Investigator: David Trebotich
NERSC Repositories: m1516, m1792

8.2.1 Project Description

8.2.1.1 Overview and Context

The objective of the Energy Frontier Research Center (EFRC) for Nanoscale Control of Geologic CO_2 (NCGC) is to use new investigative tools, combined with experiments and computational methods, to build a next-generation understanding of molecular-to-pore-scale processes in fluid-rock systems, and to demonstrate the ability to control critical aspects of flow and transport in porous rock media, in particular, as applied to geologic sequestration of CO_2. The proposed research of NCGC aims to establish the rules governing emergent behavior at the porous-continuum macro-scale under far from equilibrium conditions by carefully understanding the behavior at the underlying pore micro-scale. To do so, a new generation of experimental, imaging and modeling tools must be developed at the pore scale. These include: X-ray synchrotron and neutron scattering techniques to image the evolving pore structure and chemistry during active CO_2 injection experiments conducted over a range of scales; and multiphase, reacting flow and transport modeling based on direct numerical simulation in realistic pore space to not only inform these experiments *a priori*, but to also help interpret the experimental results, and to generalize the results to the larger (porous-continuum) scales.

To address this challenge, the model needs to resolve the relevant physical and chemical processes in complex pore geometries with complex fluid-fluid-solid interfaces, i.e., multiphase flow. This requirement demands very fine model resolution (on the order of 100's of nanometers to a few microns) so that the physics and chemistry are accurately captured. However, to draw conclusions that are applicable over a representative volume of porous media at the continuum scale, large simulation domains and extreme computational capabilities are required.

To this end, we have developed a new high performance simulation code, Chombo-Crunch. Chombo-Crunch solves the single-phase incompressible Navier-Stokes flow equations and advection-diffusion-reaction transport equations at the pore scale in very complex microscale geometries associated with heterogeneous subsurface pore space. Chombo-Crunch has produced the largest pore scale simulations of multi-component reactive transport to date, both in resolution and scale. Chombo-Crunch scales to 100,000s of processors on NERSC systems and has achieved image data resolution (1 micron) for flow-through reactive transport experiments. These simulations have produced several PBs of data (checkpoint and plot files). Sub-micron resolution (factor of 2), particularly for flow in shales, will be required to accurately capture reactive transport processes associated with carbon sequestration. To help accomplish this goal we have received several computational resource allocation awards from DOE including a 2014 INCITE award, a 2013 ALCC award and a 2012-13 NISE award (now DOE production), amounting to over 200 million core hours and 100s of TBs of storage at DOE computing facilities.

8.2.1.2 Objectives for 2017

The ultimate objective of this work is to develop an upscaling approach for partially miscible multiphase flow. This will likely require an ensemble of simulation datasets. We will need to develop a predictive model for multiphase flow within realistic fracture systems similar in scale to the single-phase simulations by Chombo-Crunch. We will implement the recent conservative, finite volume algorithm of Miller and Trebotich (2012) into the Chombo-Crunch framework to model multiphase flow (Miller and Trebotich, 2014). The interface in this case is between two fluids and is governed by surface tension jump conditions. Initially we will concentrate on flow of a two-phase fluid (for example, for invasion of CO_2 in the pore space: CO_2 and brine) with specific attention to the triple point problem presented by two fluids and a mineral boundary. Based on simulation benchmarks, we estimate that we can initially simulate, at a best resolution of 100 nm, a 100x100x100 μm image domain (one billion grid points) obtained from FIB/SEM images of a real fracture (Silin and Kneafsey, 2012). Model predictions can be tested by way of real-time X-ray imaging of fluid occupancy.

8.2.2 Computational Strategies (now and in 2017)

8.2.2.1 Approach

We use adaptive, embedded boundary methods to treat the geometric complexity of the subsurface medium at the pore scale. Embedded boundary methods are a cut cell, finite volume approach to irregular geometry based on conservative volume of fluid discretizations in the irregular cells that result from intersecting the problem domain with a rectangular Cartesian grid. Away from the boundary the finite volume method reduces to well-understood finite difference approximations.

There are several advantages to taking an embedded boundary approach to resolving the mineral boundary of subsurface pore space. Grid generation is more tractable and efficient than body-fitted gridding. In particular, this approach makes possible direct numerical simulation from image data obtained by x-ray microtomography with little user involvement. Furthermore, the embedded boundary approach allows for resolution of the reactive transport flux at the fluid-mineral boundary in each grid cell. This approach is also consistent with conservative sharp interface methods for treating both fluid-fluid boundaries in multiphase flow and fluid-mineral boundaries in the presence of precipitation or dissolution, two key features of our proposed new work. Finally, embedded boundaries are amenable to adaptive mesh refinement (AMR), a technique for increasing grid refinement dynamically only in local areas of interest in the domain (e.g., reactive front). The combined adaptive, embedded boundary approach is a very powerful tool for solving multiscale, multiphysics flow and transport problems in the subsurface.

8.2.2.2 Codes and Algorithms

We use the Chombo-Crunch production code to simulate reactive transport in microscopic pore space geometries obtained from image data of geologic porous media. Flow and conservative transport is solved by incompressible flow and advection-diffusion solvers and algorithms developed in the Chombo framework. Reactions are handled by the geochemistry module in CrunchFlow. The interface to CrunchFlow from Chombo is a point-by-point computation on a local box of the domain and thus scales ideally with Chombo solvers. We also use algebraic multigrid solvers through PETSc to avoid limitations with geometric coarsening. Chombo and PETSc support comes by way of the SciDAC FASTMath project.

8.2.3 HPC Resources Used Today

8.2.3.1 Computational Hours

We used 96 million hours at NERSC in 2013, many or which were Edison pre-production hours. We also had an ALCC allocation (80M on Mira) and an INCITE allocation (another 80M on Mira)

8.2.3.2 Parallelism

The concurrency range is from about 6,144 to 131,072, with the maximum we could use being about 131,072.

We do some runs at 6,144 and 49,152 for a specific problem that ideally we would like to increase the resolution by a factor of 2 and make use of 393,216 cores

Weak scaling is more important for us. Due to the sweet spot for load balancing in Chombo-Crunch (32^3 grid points per box in 3D; 256^2 grid points per box in 2D) we cannot achieve optimal performance for strong scaling. We can usually get two data points for strong scaling due to this load-balancing requirement. Furthermore, we are more interested in increased resolution to resolve reactive transport processes at sub-micron scales.

8.2.3.3 Scratch Data

We need 80-100TB of scratch space.

8.2.3.4 Shared Data

We currently use project directory m1792 to share data amongst team members. We managed to store 1.8 TB there in 2013 and find that the quota of 1 TB is too small.

8.2.3.5 Archival Data Storage

We had about 500 TB stored on HPSS at NERSC.

8.2.4 HPC Requirements in 2017

8.2.4.1 Computational Hours Needed

We estimate needing 500M hours. Chombo-Crunch is now considered a production code, i.e., it is out of the experimental testing phase. As such, computational geoscientists on our team would like to use it to run a number of simulations beyond what I have mentioned in my ERCAP requests.

8.2.4.2 Parallelism

We are currently making Chombo-Crunch thread safe and will learn how we are able to scale on Mira in that mode. Currently 16 ranks per core is our sweet spot. However, for the current production version of Chombo-Crunch on NERSC that uses flat MPI we will be running 393,216 cores if there is a machine that has that number available.

8.2.4.3 I/O

We currently do have checkpoint/restart and we use it. A 36-hour production run can dump up to 100TB of data, depending on the problem. We estimate needing

100TB/(.01*36hrs) = 278TB/hr = 77GB/sec I/O bandwidth and currently devote less than 2% of the time to I/O, which is all we're willing to devote.

8.2.4.4 Future Data Needs

In 2017, we expect to need 500 TB of temporary scratch disk space, 100 TB of NERSC project space (globally accessible shared data), and 5,000 TB of storage on NERSC HPSS. The growth in these requirements relative to 2013 is due primarily to increased resolution. Data in the project space would need to be retained for about five years.

8.2.4.5 Memory Required

We will need about 1 GB per core in a discrete memory space.

8.2.4.6 Emerging Technologies and Programming Models

Our software currently does not have CUDA or OpenCL and there is no plan to add this.

We have development versions of Chombo that use OpenMP. We will be porting to Chombo4 in the next year. We are currently running on Mira but not using threading yet. Porting to MIC is underway via our account on the NERSC Babbage system. We have engaged the ExaHDF5 team led by Prabhat, the PETSc team (Mark Adams, Jed Brown), and the ALCF consultants to help.

We need NERSC to tell us what hardware it planning to acquire to so we can then tell you how we will use it. The burst buffer is an example.

DOE and ASCR need to engage the algorithm and simulation software developers for guidance on future directions in hardware. For example, I would be very happy right now with a machine like Edison that has 8 or more times the core count. I would be set beyond 2017 with that capability.

8.2.4.7 Software Applications and Tools

We need Chombo, PETSc, HDF5, and VisIt.

8.2.4.8 HPC Services

We typically make heavy use of consulting and account support, data analytics and visualization. We do not need NERSC web sources for publishing or making data available.

8.2.4.9 Additional Data Intensive Needs

I may have already mentioned this but my workflow is severely hampered by scratch quotas. I will need 100TB of scratch space in the future. Also, transfer rates from scratch to HPSS are slow.

We believe that our use of HPSS satisfies the need for a data management plan.

8.2.4.10 Additional Data Intensive Needs: Burst Buffer

The primary scenario on http://www.nersc.gov/assets/Trinity-NERSC-8-RFP/Documents/trinity-NERSC8-use-case-v1.2a.pdf would be very useful for my work. All three data analysis and visualization secondary scenarios would be useful particularly with our plans for in situ data analytics and vis. These secondary scenarios would fit very well with our plans to have in situ data vis and analytics in Chombo-Crunch.

8.2.4.11 Additional Comments

I would like a Cray like Edison but with 400K cores.

8.2.4.12 Requirements Summary

NERSC Repositories m1516, m1792	**Used at NERSC in 2013**	**Needed at NERSC in 2017**
Computational Hours	96 M	500 M
Typical number of cores used for production runs	6K – 131K	400K
Maximum number of cores that can be used for production runs	400K	400K
Data read and written per run	100 TB	500 TB
Maximum I/O bandwidth		77 GB/sec
Percent of runtime for I/O	<2%	<2%
Scratch File System space	100 TB	500 TB
Shared filesystem space	1.8 TB	100 TB
Archival data	525 TB	5,000 TB
Memory per node	1 GB/core	1 GB/core

8.3 Numerical Algorithms for Electronic and Nuclear Structure Analysis

Principal Investigator: Chao Yang
NERSC Repositories: m1027

8.3.1 Project Description

8.3.1.1 Overview and Context

The goal of our project is to develop enabling applied mathematics and numerical tools for improving the fidelity and throughput of computational materials science and chemistry research. In particular, we focus on developing methodologies and software tools for accelerating ground state electronic structure calculations that are based on density functional theory (DFT) and many-body wave function methods (such as configuration interaction, coupled cluster) and excited state calculations that are based on Green's function formalism and many-body perturbation approaches. The improvement in ground state DFT calculation is also important for accelerating first principles molecular dynamics simulation. The algorithmic and software development for configuration interaction is also relevant for nuclear structure calculations.

This project facilitates several SciDAC3 institution and partnership projects that we are currently funded to work on. These projects (and their PIs) include:

- FASTMath institute (Lori Diachin)
- Advanced Modeling of Ions in Solutions, on Surfaces, and in Biological Environments (Roberto Car)
- Discontinuous Methods for Accurate, Massively Parallel Quantum Molecular Dynamics: Lithium Ion Interface Dynamics From First Principles (John Pask)
- Scalable Computational Tools for Discovery and Design – Excited State Phenomena in Energy Materials (Jim Chelikowsky); see http://excited-state-scidac.org
- Simulating The Generation, Evolution and Fate of Electronic Excitations in Molecular and Nanoscale Materials With First Principles Methods (Martin Head-Gordon)
- Charge Transfer and Charge Transport in Photoactivated Systems: Developing Electron-Correlated Methods for Excited State Structure and Dynamics in the NWChem Software Suite (Chris Cramer)
- Nuclear Computational Low-energy Initiative (Joe Carlson)

8.3.2 Objectives for 2017

For first-principle MD simulation, we aim to develop efficient algorithms and parallel implementations that will allow scientists to perform simulations on realistic Li-ion models and certain biological systems (ions in solutions) that contain more than 10,000 atoms. In particular, we would like to keep the wall clock time it takes to advance each MD step under one minute. This will allow scientists to simulate a longer trajectory than what they can currently do.

For electron excitation, we aim to develop methods that would allow scientists to perform GW calculations and solve Bethe-Salpeter equations for systems with up to 1,000 atoms.

We also aim to develop efficient algorithms that will enable scientists to perform large-scale configuration interaction (CI) calculations for both electronic and nuclear structures. For nuclear structure, we hope to be able to perform CI calculations that involve 3 and 4-body potentials.

At the solver level, we hope to develop an eigensolver that can utilize a vast amount of computational resources efficiently to compute a small percentage of eigenpairs of large sparse matrices.

8.3.3 Computational Strategies (now and in 2017)

8.3.3.1 Approach

In density function theory based electronic structure calculation, the main problem to be solved is the Kohn-Sham equation, which is a nonlinear eigenvalue problem. The solution to this problem is the starting point for calculating excited state properties of materials. It is also the building block for optimizing the structure of materials (geometry optimization) and for performing first-principle molecular dynamics.

The Kohn-Sham problem is solved by a nonlinear iterative method often known as the self-consistent field (SCF) iteration. The most expensive part of the method is the evaluation of the electron density. The most widely used algorithm computes the electron density at each SCF iteration by solving a large linear eigenvalue problem with a fixed Hamiltonian matrix. The number of eigenpairs required is proportional to the number of atoms (N). The complexity of the this approach is $O(N^3)$ if a standard eigensolver is used to compute the desired eigenpairs.

We are pursuing two ways to reduce the complexity of electron density evaluation. In one approach, we express the electron density as the diagonal of a matrix function of the Hamiltonian, and approximate this matrix function by a linear combination of simple rational functions through a technique called pole expansion. The use of pole expansion allows us to avoid computing eigenvalues of the Hamiltonian. Instead, we need to compute the diagonals of the inverses of a number of shifted Hamiltonians. We developed an efficient sparse matrix technique called selected inversion to compute these diagonal elements without inverting the entire matrix. This technique allows us to reduce the complexity to $O(N^2)$ in the worst case. In another approach, we still compute the electron density by computing the desired eigenpairs. We develop a new algorithm that divides the spectrum of interest into several intervals and compute eigenvalues belonging to different intervals simultaneously. This approach not only reduces the complexity of the calculation, but also introduces more concurrency by adding a level of coarse-grained parallelism.

The prefactor of these calculations depends on the choice of discretization methods. The existing discretization schemes such as plane wave basis expansion, finite element, and finite difference often lead to large prefactors that cannot simply be ignored. We are in the process of developing a new discretization scheme that is based the discontinuous Galerkin framework. In this approach, we expand the solution to the Kohn-Sham problem in a set of discontinuous basis functions constructed from solving the Kohn-Sham problem restricted to a number of localized domains. The DG approach allows us to minimize the number of

basis functions per atom, thereby reducing the dimension of the matrix Hamiltonian while keeping it sparse. This technique reduces the prefactor of the DFT computation significantly.

For excited state calculation through the Green's function formalism, we need to solve Dyson's equation, which is also a nonlinear eigenvalue problem. We use the GW approximation to the self-energy in the Dyson's equation. A full-frequency GW approximation requires the dielectric operator to be evaluated and inverted at multiple frequencies. Each evaluation has $O(N^4)$ complexity. We are developing techniques to organize the computation in a way to maximize concurrency. To obtain energies of excitons, we need to solve the Bethe Salpeter equation, which is a complex linear eigenvalue problem. The dimension of the problem is N^2 by N^2. This is extremely costly to solve. We will develop an efficient structure preserving eigensolver to tackle this problem.

On the wave function methods side, the main problem to be solved is a many-body Schrodinger's equation, which is a large-scale linear eigenvalue problem. We are developing efficient preconditioning techniques to accelerate the convergence of eigensolvers for both configuration interaction and multi-configuration SCF calculations.

8.3.3.2 Codes and Algorithms

PPEXSI: Parallel Pole Expansion and Selected Inversion.

It is used to evaluate electron density from a sparse DFT Hamiltonian without computing eigenvalues and eigenvectors of the Hamiltonian. It uses an efficient and accurate rational expansion called pole expansion to approximate a Fermi-Dirac function, and a sparse direct method to compute selected elements of the inverse of a sparse matrix without inverting the whole matrix. This is currently used for electronic structure calculation. But parallel selected inversion has a wider range of applications (electron transport, uncertainty quantification etc.) We are currently working with two SciDAC3 teams (Roberto Car and John Pask) to integrate PPEXSI with CP2K and DGDFT (see below). We are also working with SIESTA developers to integrate PPEXSI with SIESTA.

DGDFT: Discontinuous Galerkin based Density Functional Theory calculation

It is a new DFT-based electronic structure code that uses a local basis expansion technique and discontinuous Galerkin formalism to discretize the Kohn-Sham equation. It is combined with PPEXSI to perform electronic structure analysis of large atomistic systems. It is also used in first-principle molecular dynamics code to simulate the dynamics of complex systems such as the formation and evolution of SEI layer in a Li-ion battery.

BerkeleyGW:

A code developed by Steven Louie's group at UC-Berkeley. It is used to calculate single electron excitation and optical spectral of an electron-hole pair. Most of the computations involved in GW calculation are dense linear algebra operations. We developed special numerical integration scheme to perform the convolution of the Green's function (G) and the screened Coulomb interaction in the frequency domain. We are also developing efficient ways to evaluate frequency-dependent dielectric matrices without computing the polarizability explicitly, thereby avoiding computing many eigenvalues of the Kohn-Sham Hamiltonian.

NWCHEM and **QCHEM:**

These are both well-established quantum chemistry codes. NWCHEM uses GlobalArrays as a means to perform parallel calculations on a distributed memory system. QCHEM currently only supports shared memory parallelism. Disk I/O is used to hold intermediate results for some type of calculations such as coupled cluster. Both codes use the block Davidson algorithm to solve large-scale eigenvalue problems.

MFDn:

This is the Many-body Fermion Dynamics for nuclear physic code is developed by James Vary at Iowa State University. It is a nuclear configuration interaction code that can be used to study both the ground and excited states of light nuclei. It uses the Lanczos algorithm to compute typically 10-20 lowest eigenpairs of a Hamiltonian with dimension as large as 10^9.

Alternative eigensolvers

We are developing two new eigensolvers to compute a small percentage of eigenpairs of large sparse matrices**. EigPen** uses a trace-penalty formulation of the eigenvalue problem and minimizes the trace of the Hamiltonian within a subspace using a steepest descent type of algorithm. **RTraceMin** implements a relaxed trace minimization algorithm in which the orthonormality constraint is first relaxed and then reinforced later to project the approximate solution onto the constraint. We currently have a OpenMP parallel version for EigPen and a distributed parallel version of RTraceMin, which is being integrated with the Quantum Espresso and Qbox packages. We are also developing a solver based on a spectrum slicing technique.

8.3.4 HPC Resources Used Today

8.3.4.1 Computational Hours

We used roughly 7.3 million Hopper-equivalent hours at NERSC in 2013. Among these

- Roughly 2.8 million hours are spent on algorithm development and tests for BerkeleyGW
- Roughly 2.4 million hours are spent on algorithm development and tests for parallel selected inversion
- Roughly 150K hours are spent on the development of large-scale eigensolvers
- Roughly 900K hours are spent on the development of DGDFT

8.3.4.2 Parallelism

We typically use several thousands to several hundreds of thousands of cores for production runs. Our production runs are mostly used to test the performance and accuracy of algorithms.

The BerkeleyGW code can use as many cores as is available for sufficiently large systems.

The DGDFT and PPEXSI code can use over 300,000 cores.

Most of the runs in this project are used to test newly developed algorithms. For testing algorithms, fast turnaround is extremely important. Therefore, we cannot afford to submit large jobs that wait in queues for several days. We typically submit small jobs that can run interactively or medium-sized jobs that can be started within a day.

For application problems, weak scaling is important since scientists we work with tend to be interested in looking at larger and more complex systems. To be able to take advantage of a vast amount of computational resources to study these problems is what is important to them. In some cases, strong scaling matters also. For example, one of the goals of our project is to be able to perform each molecular dynamics step for a realistic Li-ion model within one wall-clock minute. This is the kind of time scale that would allow scientists to obtain a sufficiently long trajectory to understand battery materials properties.

For algorithmic development, strong scaling is often the first thing we look because it often exhibits computational bottlenecks that need to be addressed.

8.3.4.3 Scratch Data

For BerkeleyGW runs, we can use as much as several terabytes. The disk space is used to store frequency dependent dielectric matrices that are used in subsequent self-energy calculations.

For other codes, the amount of disk space needed is typically small. We use some disk space to store sparse Hamiltonian matrices and sometimes eigenvectors and electron density. We typically need less than 1 TB for these projects.

8.3.4.4 Shared Data

We use project directory /project/projectdirs/m1027, which currently has nearly 3TB of data stored. We use the project directory primarily to share input and output data. Sometimes we use it to share certain versions of the code that we are not ready to commit to a code repository system such as svn and git.

8.3.4.5 Archival Data Storage

We currently use HPSS very little. We had 5.6 TB stored there in 2013.

8.3.5 HPC Requirements in 2017

8.3.5.1 Computational Hours Needed

We expect that our project will need at least 75M compute hours in CY 2017. It is likely that NERSC will be the only resource we'll use.

By 2017, most of code development for SciDAC3 projects should be completed. We will be testing and benchmarking these codes on more challenging systems with sufficiently large number of atoms. In particular, for first-principle MD simulation, we will be targeting systems with more than 10,000 atoms. For GW and BSE calculations, we will be targeting systems with 1,000 atoms. The 10-fold increase in system alone will require at least a 10 fold increase in computational resources if we assume linear scaling can be achieved in our calculation, which is a bit optimistic at the moment, especially for excited state calculation.

8.3.5.2 Parallelism

For a 10,000–atom MD simulation using DGDFT and PEXSI, we expect to break the system into 2,000 5-atom elements. Each can be assigned to an MPI task. So there will be 2,000 MPI tasks. We hope to be able to use shared-memory cores to compute DG basis on each element. We currently use 12 cores on Edison, and can possibly use more if available.

For parallel PEXSI, we can currently use up to 1.3 M MPI tasks (with 2 levels of parallelism). We hope to reduce the number of MPI tasks by at least a factor of 10 and replace them with OpenMP tasks to achieve better scaling.

In principle, there is no limit on how many MPI tasks we can use.

8.3.5.3 I/O

Our applications currently do not have built-in checkpoint/restart.

For end-to-end simulation, we anticipate a relatively small amount of data that describes and chemical species and geometry of the system to be read. Our code may write a large amount of data that consists of 3D wave functions, electron density at multiple time steps or dielectric matrices and self-energy at multiple frequencies.

The wave functions and density may require hundreds of GB. The dielectric matrices at multiple frequencies may require tens to hundreds of TB disk space to store if the existing algorithms are used and all intermediate data need to be stored.

Effective bandwidth of tens GB/sec should be sufficient. This may translate to hundreds of GB/sec or 1TB/sec depending on how many users are performing I/O at the same time.

Ideally, no more than 1 percent of wall clock time should be spent on I/O.

8.3.5.4 Future Data Needs

In 2017, we expect to need about 100 TB of temporary scratch disk space, 100 TB of NERSC project space (globally accessible shared data), and about 500 TB of storage on NERSC HPSS. The growth in these requirements relative to 2013 is due primarily to increased problem size. Of the data that we will store at NERSC in the project space or on HPSS, we would like it to be permanent storage, if possible.

8.3.5.5 Memory Required

For Kohn-Sham DFT based electronic structure calculation for a 10,000 atom system using DGDFT, the aggregate memory we need is at least 512x80 GB = 40TB aggregate memory to store the Cholesky factor (assuming a 20% fill) and the selected inverse of the sparse Hamiltonian which is required to calculate the electron density. If we can run the code efficiently on 2,000 nodes, the amount of memory required per node is at least 20GB.

For GW calculations, we potentially need 10 PB aggregate memory if polarizability is constructed explicitly from the eigenvectors of the DFT Hamiltonian. If 100,000 nodes can be efficiently used, the memory requirement per node is 100 GB.

For nuclear CI calculations, the 40 PB aggregate memory is likely to be needed just to store the sparse Hamiltonian. There is a trade-off between communication overhead and memory per node. If 100,000 nodes can be efficiently used, 400 GB per node memory is needed.

8.3.5.6 Emerging Technologies and Programming Models

We currently do not use CUDA/OpenCL. We do not have plans to use these in the near future. Our software does not currently run on Titan or other GPU systems.

Some of our codes (DGDFT, BerkeleyGW) have OpenMP directives now. We plan to add OpenMP directives to other codes such as PPEXSI in the future. BerkeleyGW can run on Mira.

We have not tried porting to or optimizing for MIC.

There are no other funded groups to help with these activities.

We have no strategy for exploiting the above technologies because we do not know if any of them is sustainable. We have limited human resources and cannot afford to rewrite codes that will become obsolete in a few years.

We believe the role that NERSC should play is in porting existing codes, performing benchmark tests, and sharing experiences by giving presentations and providing tutorials.

We believe the role that DOE and ASCR should play is by providing sufficient and sustained funding for algorithm development.

8.3.5.7 Software Applications and Tools

We would need several standard material science and chemistry applications (such as Quantum Espresso, SIESTA, ABINIT, QCHEM, NWChem, and/or BerkeleyGW), as well as a wide variety of libraries such as BLAS/LAPACK/ScaLAPACK SuperLU_DIST, Pardisol, MUMPS, Parmetis, PT-SCOTCH, and FFTW. We also need C, C++, and Fortran90 compilers, Python, DDT or other more user-friendly parallel debugging systems, and performance analyzer and profiling tools (something better than CrayPat).

8.3.5.8 HPC Services

Consulting and account support are most useful for us. Help with visualization can potentially be useful also. We do not need web resources from NERSC to publish data.

8.3.5.9 Additional Data Intensive Needs

We do not have a data management plan of any kind in place now, nor do we feel we need help from NERSC in defining or implementing such a plan.

8.3.5.10 Requirements Summary Worksheet

NERSC Repository m1027	**Used at NERSC in 2013**	**Needed at NERSC in 2017**
Computational Hours*	7.5 M	75 M
Typical number of cores** used for production runs	5,000 – 100,000	10,000-1M
Maximum number of cores** that can be used for production runs	1 M	10 M
Data read and written per run	4 TB	40 TB

Maximum I/O bandwidth	2 GB/sec	50 GB/sec
Percent of runtime for I/O	33%	1%
Scratch File System space	5 TB	100 TB
Shared filesystem space	2.8 TB	100 TB
Archival data	5.6 TB	500 TB
Memory per node	32 GB	1024 GB
Aggregate memory	10 TB	10 PB

*NERSC MPP Hours (Hopper equivalent hours)
**Traditional cores

8.3.5.11 Additional Storage and I/O Comments

We have both serial and parallel I/O; both shared (N-to-1) or distributed (N-to-N) and uses mostly large files.

8.4 AMR for Ice Sheet Modeling

Principal Investigator: Esmond G. Ng, LBNL
Worksheet Author(s): Daniel Martin, Esmond G. Ng, LBNL
NERSC Repositories: m1041 (Predicting Ice Sheet and Climate Evolution at Extreme Scales)

8.4.1 Project Description

8.4.1.1 Overview and Context

Ice sheets are immense freshwater reservoirs that interact dynamically with other parts of the Earth system. Most climate models to date have ignored dynamic changes in ice sheets, for two reasons. First, ice sheets were thought to evolve only on time scales of hundreds to thousands of years. Recent observations, however, have shown that the Greenland and Antarctic ice sheets are losing mass at a rate of 500 km^3/year and are likely to make a dominant contribution to 21st century sea level rise. Second, ice sheets are difficult to observe and model. Models must simulate complex flows on small scales (1 km or less), solving large nonlinear systems of partial differential equations (PDEs) that challenge the best computational methods. Ice flow is sensitive to poorly understood basal boundary processes. Ice sheets are closely coupled to the atmosphere and ocean, and this coupling is only beginning to be included in climate models. Furthermore, there are few observational data sets for model validation.

Although ice sheet models have improved in recent years, much work is needed to make these models reliable and efficient on continental scales and to quantify their uncertainties. We will develop and apply robust, accurate, and scalable dynamical cores ("dycores") for ice sheet modeling on structured and unstructured meshes with adaptive refinement, evaluate ice sheet models using new tools and data sets for verification and validation (V&V) and uncertainty quantification (UQ), and integrate these models and tools in the Community Ice Sheet Model (CISM) and Community Earth System Model (CESM). Using improved estimates of ice sheet initial conditions, we will simulate decade-to-century-scale evolution of the Greenland and Antarctic ice sheets, using CISM both in standalone mode and coupled to CESM. We aim to provide useful, credible predictions, including uncertainty ranges, of future ice sheet mass loss and resulting changes in climate and sea level.

Building on recent successes of SciDAC and the Ice Sheet Initiative for Climate Extremes (ISICLES), we are developing two dycores: (1) BISICLES: a finite-volume dycore on a structured mesh, using the Chombo adaptive mesh refinement software framework, and (2) FELIX: a finite-element dycore on an unstructured mesh, using the Model for Prediction Across Scales (MPAS) framework and Trilinos software packages. Both dycores will include a hierarchy of Stokes and higher-order solvers that can be applied at variable resolution in different regions. We will develop stable, efficient numerical schemes for ice-thickness evolution and will implement realistic, physics-based basal boundary conditions. In order to model the Antarctic ice sheet, high-performance computing is needed. The dycores we are developing will be engineered to optimize performance on new high-performance computers with heterogeneous architectures and will be incorporated into the Community Ice Sheet Model (CISM), which will be coupled to global climate models (GCMs) like CESM and the under-development DOE ACME GCM.

8.4.1.2 Objectives for 2017

By 2017 we aim to have high-resolution, fully coupled simulations of ice sheet and climate evolution (e.g., sea-level rise) with uncertainty quantification. We will run stand-alone CISM model simulations as well as simulations fully coupled (ocean-atmosphere-sea ice) to CESM and the DOE ACME GCM. CISM will use the BISICLES and FELIX dynamical cores, both with adaptive mesh refinements and hierarchical, 3-D, higher-order momentum balance solutions (including and up to nonlinear Stokes).

8.4.2 Computational Strategies (now and in 2017)

8.4.2.1 Approach

The computational problem is the solution of large systems of nonlinear partial different equations (PDEs). The strategies for solving the PDEs depend on the dycores. The BISICLES dycore is based on a finite volume technique using structured meshes with adaptive refinement. The FELIX dycore is based on a finite element method using unstructured meshes with variable resolution. In both cases, the computational kernels after discretization involve nonlinear and linear solvers. The solution strategies make extensive use of computational frameworks developed by ASCR researchers. The BISICLES dycore is built using the Chombo framework for block-structured adaptive mesh refinements and relies on nonlinear and linear solvers from PETSc. The FELIX dycore is built using the Model for Prediction Across Scales (MPAS) framework; the nonlinear and linear solvers come from the Trilinos framework.

8.4.2.2 Codes and Algorithms

Both the BISICLES and FELIX codes are used for modeling land ice sheet evolution.

BISICLES: Chombo-based AMR ice sheet model based on finite-volume discretizations on regular block-structured adaptive meshes. Currently beginning to be used for production runs.

FELIX: Trilinos/MPAS-based ice sheet model based on finite-element discretizations on unstructured/semi-structured meshes. This code is still under development.

8.4.3 HPC Resources Used Today

8.4.3.1 Computational Hours

NERSC time: 500,000 core hours

BISICLES: 300,000 core-hours on Hopper in 2013

FELIX: 200,000 core-hours on Hopper in 2013.

Other facilities:

BISICLES: only local workstation usage for development and testing, probably around 50k hours

FELIX: only local workstation usage for development and testing, probably around 50k hours

8.4.3.2 Parallelism

The remainder of this study will primarily discuss the BISICLES model.

For 1km-resolution full-continent Antarctica runs, we have been running BISICLES with 768 cores, which takes about 77 hours to do a 100-year run (7x11 hours). Multiple runs are needed to evaluate different climate scenarios.

We haven't done a recent scaling study for BISICLES, but we likely don't want to use more than a few thousand cores. The issue we run into is that we start to run out of work to distribute, particularly on the coarser AMR levels.

As mentioned, we begin to run out of work to do on the coarser levels. At the same time, the particular number of processors was chosen to try to find a "sweet spot" in the queues.

At present, as we are increasing the resolution of our simulations, weak scaling is more important. Once we reach our target resolution (very soon), then strong scaling becomes more important.

8.4.3.3 Scratch Data

We typically need about 10 TB for temporary space.

8.4.3.4 Shared Data

We have two project directories. One is a "piscees," project directory associated with the PISCEES SciDAC application partnership, to enable sharing of common datasets and software/library installation for common use. The other is the "iceocean" project directory inherited from the now-defunct m1343 ALCC allocation, used primarily to enable offline coupling between the POP2x ocean model (managed by Xylar Asay-Davis) and the BISICLES ice sheet model (managed by Dan Martin). We used 1.1 TB of disk space in these directories in 2013.

8.4.3.5 Archival Data Storage

We had 20 TB of data stored in the NERSC HPSS system in 2013.

8.4.4 HPC Requirements in 2017

8.4.4.1 Computational Hours Needed

In 2013, our finest-resolved full-continent Antarctic simulation was a 1 km finest-resolution AMR run, which took 77 hours on 768 processors for a 100-year run (approx. 60,000 hours).

In 2017, we expect to be doing ~ 250m-resolution full-Antarctic runs (with AMR). With the current code, each of these runs would require approximately 960,000 hours per 100-year run (9,600 hours/simulation year). We expect to do a suite of these runs to evaluate different climate scenarios and also to begin to quantify the uncertainties in the problem.

Expected improvements in the code in that timespan include a transition from 2.5D to full 3D (roughly a factor of 10x) and transition to more-complex physics (2-3x), which leads to roughly 20M hours per 100-year run. A suite of 100 runs would then be 2B cpu-hours. So, the increase is driven somewhat by larger problem size, but mostly by the desire to run more scenarios.

Note that for each initial condition, we also need to solve an inverse problem to obtain the initial state. This generally costs about the same as a 10-year run. In 2017 we likely won't need to resolve the inverse problem for every scenario, so our estimate is that we will need 550 million hours.

8.4.4.2 Parallelism

The number of unknowns will be roughly 25 times larger than what we solve for now, which is likely a good indicator of the MPI parallelism we can hope to achieve, which means an MPI parallelism of around 25,000 MPI tasks. We can likely expect O(10) additional fine-grained parallelism via threading.

8.4.4.3 I/O

Our application does make use of its own checkpoint/restart capability. We would typically write or read about O(500 GB)/run x 25 runs = 12.5 TB. Assuming perfect I/O hiding, would need 500 GB/80hrs = 6.25 GB/hr of bandwidth. That would mean no time spent in I/O; we could reasonably tolerate 5-10%.

8.4.4.4 Future Data Needs

In 2017, we expect to need about 50 TB of temporary scratch disk space, 16 TB of NERSC project space (globally accessible shared data), and 500 TB of storage on NERSC HPSS. The growth in these requirements relative to 2013 is due primarily to increased number of scenarios to evaluate.

8.4.4.5 Memory Required

We haven't really measured memory requirements for the simple reason that it hasn't been an issue for us – we've found the current amount of memory per node to be adequate to our needs. For us, parallelism has been driven by execution time, not memory footprint.

8.4.4.6 Emerging Technologies and Programming Models

We don't use CUDA or OpenCL and have no current plans to do so. Our code does not run on Titan or other systems using GPU hardware.

Our software does not currently have OpenMP, but OpenMP support has been added to Chombo for the upcoming 3.2 release. At that point, we can begin to experiment with this capability in BISICLES. The code does not currently run on Mira or Sequoia.

Porting to or optimizing for Intel MIC is neither underway nor planned.

We plan to leverage development by the Chombo team currently underway, as well as whatever support PETSc brings to the table. Also, PISCEES includes some funding for SUPER (Williams at LBNL, Worley at ORNL) to support performance improvements and migration to new architectures.

We would like to see NERSC provide, at the very least, training and support for users making the transition to these architectures. We strongly believe that DOE/ASCR needs to better support development of programming libraries, tools, and algorithms.

8.4.4.7 Software Applications and Tools

We will need HDF5, netCDF, PETSc, C++/Fortran (PGI or GNU), LAPACK, MPI, OpenMP, DDT or similar parallel debugger, and performance evaluation tools.

8.4.4.8 HPC Services

We anticipate needing consulting and account support, architecture transition training and support, performance tools.

8.4.4.9 Additional Data Intensive Needs

We currently have no data management plan beyond archiving data via HPSS.

8.4.4.10 Requirements Summary Worksheet

NERSC Repository m1041	**Used at NERSC in 2013**	**Needed at NERSC in 2017**
Computational Hours*	500 K	550 M
Typical number of cores used for production runs	500-2,000	12,000
Maximum number of cores** that can be used for production runs	3,000	25,000
Data read and written per run	1 TB	100 TB
Percent of runtime for I/O	<5-10%	<5-10%
Scratch File System space	10 TB	50 TB
Shared filesystem space	1.1 TB	16 TB
Archival data	20 TB	500 TB

*Normalized to Hopper-equivalent (NERSC MPP) hours
**Traditional cores

8.4.4.11 Additional Storage and I/O Comments

Our I/O is mostly parallel, using HDF5 with an N-to-1 pattern and medium-size files.

8.5 PETSc - Portable Extensible Toolkit for Scientific computation

Principal Investigator: Lois Curfman McInnes, Argonne National Laboratory
Additional Worksheet Authors: Jed Brown, Barry Smith
NERSC Repository: m1489, Composable Hierarchically Nested Solvers

8.5.1 Project Description

8.5.1.1 Overview and Context

PETSc is a scalable numerical software library for the solution of PDEs and optimization problems. It includes its own linear, nonlinear, and ODE solvers as well as interfaces to other packages, including SuperLU_DIST, Hypre, ML, and Sundials. It also contains code for the management of parallel data structures (for example structured and unstructured grids) needed in the solution of PDEs.

8.5.1.2 Objectives for 2017

By 2017 we want to achieve scalability and efficiency across a range of applications and problem sizes on the hardware available at that time. This includes algorithmic advances to utilize higher-level problem structure to reduce communication requirements, create more on-node parallelism, and reduce memory bandwidth in order to improve efficiency and strong scalability (reduce the turn-around time).

8.5.2 Computational Strategies (now and in 2017)

8.5.2.1 Approach

PDE solvers require nearest neighbor communication plus global reductions. Multilevel preconditioners and solvers also require longer-range communication when restricting the active process set for coarse grids.

8.5.2.2 Codes and Algorithms

Our codes contain Krylov solvers, multigrid solvers, stiff and non-stiff ODE solvers, and structured and unstructured grids.

8.5.3 HPC Resources Used Today

8.5.3.1 Computational Hours

In 2013 we used 441 K hours at NERSC and about 1 million at other facilities.

8.5.3.2 Parallelism

We believe that NERSC users calling PETSc solvers probably have concurrencies in the range 200-100k cores. The maximum would probably be more than 1M, depending on the chosen algorithm.

Some algorithms are currently implemented with non-scalable data structures or have a communication pattern during setup that is not scalable. By non-scalable, we mean any handling of local problem-sized data involving work that is more than logarithmic in the

number of processes P. A constant amount of O(P) data is allowed, and is used internally by the MPI implementations, so long as it is only used for fast (logarithmic or better) lookups when handling problem-sized data. For problems with irregular sparsity, the user may provide data with unbalanced off-process dependencies, in which case the best algorithm would change. It complicates the interface to require the user to choose the algorithm up-front, but not doing so increases setup costs because some analysis must be done to choose an efficient communication algorithm.

Weak scaling is important to demonstrate the capability and for a few customers that run extremely high resolution. Weak scaling is also a more discerning test of multilevel algorithmics, assuming it is done in a realistic setting (with just-resolved solution structure on the finest grids). We believe that strong scalability is more important for changing the type of science and engineering that is possible. This is due to increased emphasis on time-accurate transient simulations (that must take smaller time steps as the spatial grid is refined) and with increased emphasis on uncertainty quantification, data assimilation, and optimization, as well as new real-time applications. Strong scaling is fundamentally difficult for PDE solvers since network latency and bandwidth become more significant factors (work scales with the subdomain volume while communication bandwidth scales with the surface area). We believe that new algorithms to use higher-level problem structure to reduce communication costs represent an important research area.

8.5.3.3 Scratch Data

About 1TB would be needed currently.

8.5.3.4 Shared Data

We don't use a project space.

8.5.3.5 Archival Data Storage

We don't archive data at NERSC.

8.5.4 HPC Requirements in 2017

8.5.4.1 Computational Hours Needed

We estimate that we will need about 10M hours in 2017. We expect to use a similar amount of time on other architectures, especially those at ALCF.

The reason for needing more hours is as follows. Customers ask for scalability up to the largest jobs they want to run, which is near the full machine for some. We conduct scaling studies to identify bottlenecks and refine our algorithms. When users run with larger core counts, our tests must also run with more cores, resulting in more time used.

8.5.4.2 Parallelism

PETSc algorithms are all bandwidth limited hence the amount of local parallelism possible is proportional to the available memory bandwidth.

8.5.4.3 I/O

We are a library rather than an application, but we provide functions for persistence and to assist users in implementing comprehensive checkpoint capability. This I/O is generally to a single large-file written collectively.

We believe that global storage is dead as an algorithmic device; thus, I/O performance is important for dealing with system reliability, resource management (job length limitations), and so that humans can make choices. *In-situ* analysis will reduce the I/O demands once the cost of I/O and storage becomes high enough. It will take many years for applications to implement *in-situ* analysis and we expect that newer applications will postpone such implementations; thus, there will always be some applications demanding more I/O than would be cost-effective based purely on direct costs.

A rough I/O bandwidth requirement might be calculated as follows. If the working set increases to 5PB, with 2% (=100TB) used for storage of essential state (typical with implicit solvers using assembled matrices, preconditoners, Krylov spaces), a MTBF of one hour, and 20% checkpointing overhead deemed acceptable, yields a required write time of one minute, or a bandwidth of 1.6 TB/second.

The percentage of total runtime that should reasonably be consumed by I/O is a choice for users rather than for libraries. We make the I/O as efficient as possible and provide certain *in-situ* analysis support, but the user is the one with science/engineering demands, thus the one making the cost/benefit analysis.

8.5.4.4 Emerging Technologies and Programming Models

PETSc can use CUDA and OpenCL, but most applications do not. PETSc also can use pthreads and OpenMP. However, most applications do not use them. GPUs are not good for strong scaling and are currently of mediocre efficiency for sparse problems unless their use is subsidized. Our software does run on GPUs, primarily at smaller GPU installations. Not all algorithms make sense on GPUs and many of the larger customers need those algorithms that are not efficient on GPUs.

OpenMP and pthreads are supported in PETSc via a "threadcomm," which provides thread-collective semantics. The threading backend can be chosen at run-time. OpenMP is useful with applications that also use OpenMP, so that thread pools can be shared. Not all operations have threaded implementations. Our software does run in production now on Mira using threading, but it is more common to use pure MPI.

MIC performance has been so disappointing that we are not directing much effort this way. The OpenCL toolchain is particularly so, and the memory subsystem is poorly connected, leading to generally poor performance and efficiency for the sort of operations that are important for implicit solvers.

Regarding other groups engaged to help with these activities, a group at Imperial College London has done work on threading using OpenMP. Tech-X contributed some GPU code.

We believe NERSC's role should be to make a careful assessment of potential efficiency across the range of applications. Current "hybrid" architectures are effectively subsidized from the perspective of the end user (and often computing centers), but this won't last and many important applications spend their time performing operations that are ill-suited to throughput-oriented accelerators. Not all applications would benefit from "transitioning to these architectures."

Regarding DOE's and ASCR's role in this, again, we urge them to be realistic about choosing the right tool for the job. Many of those applications that are ill-suited to throughput-oriented devices will need major algorithmic advances to create a computational structure

with enough fine-grained parallelism and sufficiently short critical paths for throughput-oriented devices to be competitive, especially with the stringent turn-around requirements brought on by science, engineering, and policy decisions.

8.5.4.5 Software Applications and Tools

We need MPI, pthreads, C and Fortran compilers. Interactive sessions and debuggers are also useful.

8.5.4.6 Requirements Summary Worksheet

NERSC Repository m1489	**Used at NERSC in 2013**	**Needed at NERSC in 2017**
Computational Hours*	441 K	10 M
Typical number of cores used for production runs	200-100 K	200-1 M+
Maximum number of cores that can be used for production runs	100 K+	1 M+
Data read and written per run	<1 TB	<10 TB

*Normalized to Hopper-equivalent (NERSC MPP) hours

8.6 Linear Algebra Algorithms and High Performance Computers

Principal Investigator: James Demmel, UC Berkeley
NERSC Repository: mp156

8.6.1 Project Description

8.6.1.1 Overview and Context

We design, implement, evaluate, and autotune novel algorithms for numerical linear algebra and other high performance computing kernels, that are widely used by many scientific computing projects. Our theme is "avoiding communication", i.e. minimizing data movement, whether it is between levels of the memory hierarchy or between processors over a network. This is because data movement is widely recognized as the most time and energy consuming operation performed by current and future hardware. We are also funded by NSF to incorporate our new algorithms into numerical libraries including LAPACK and ScaLAPACK, which are widely used by DOE applications. To this end, it is important to run problems at various scales, and to be able to have low-level hardware knowledge and measurement of, and control over, the resources we consume, in order to evaluate and autotune our algorithms. This is true even though, in production mode, we are unlikely to have as much control over resource allocation.

In particular, we would ideally like the following HW and SW features to be available:

1) Ability to allocate "regular" subsets of a machine, with simple interconnection topologies. This is because a number of our communication-bound algorithms work best when we do topology-aware mappings of tasks, and tune the corresponding communication schedules. We have done these experiments on IBM BG-Ps in the past with very good results, but not on Hopper for lack of this allocation ability. Even though not every application using our library may get a "regular" subset, the speedups may be significant enough to justify such allocations for some applications.
2) Ability to measure hardware events, especially data movement. Since minimizing data movement is our goal, it is important to be able to easily access accurate hardware performance counters to evaluate our algorithms.
3) Ability to measure energy/power consumed in various machine components. While data movement consumes much energy, modern architectures are so complicated, and consume energy in so many places (on chip, on board, in memory, in network) in ways that change dynamically (e.g. frequency scaling) that simple energy models (based on accurate counts of a few hardware events, like cache misses) may not be adequate for tuning an algorithm to minimize energy usage. Measuring energy accurately is an on-going research question in both academia and industry, but the more tools for measurement the better.
4) We are exploring adding other features to numerical software that have raised interest from a large number of users, whose efficiency will depend on the availability of particular underlying hardware features.

a. *Reproducibility* means getting bitwise identical results when running the same program twice, on the same hardware, but with possibly different subsets of resources (e.g. number of processors). This is important for debugging, correctness, or uncertainty quantification in various applications. A major obstacle to reproducibility is non-associativity of floating point arithmetic, leading to different

round-off errors when computing sums in different orders, for example because of different reduction trees. We have recently released the first version of a library, ReproBLAS, that provides implementations of reproducible Basic Linear Algebra Subprograms. We are exploring various hardware features that could make our current implementation even faster (e.g. extended precision, atomic operations, specialized operations embedded in network).

b. *Precision tuning* means automatically figuring out the least precision needed in each variable or phase of a computation, and still provide an answer that is accurate enough, according to a user-supplied accuracy metric. For example, an algorithm initially all running in double precision may be able to perform most operations in single and still get an adequate answer, thus saving time, memory and energy. We have recently released the first version of a tool, Precimonious (short for "parsimonious with precision") that automatically searches for the largest subset of variables that can be converted from higher to lower precision and still get an answer that is good enough. Depending on the floating point instructions and precisions available, Precimonious may attain different levels of optimization.

8.6.1.2 Objectives for 2017

We hope to build the libraries and tools described above, and make them available to DOE and other users.

8.6.1.3 Requirements Summary Worksheet

NERSC Repository mp156	**Used at NERSC in 2013**	**Needed at NERSC in 2017**
Computational Hours*	405 K	3.5 M (Based on historical growth of usage in this repo)

*Normalized to Hopper-equivalent (NERSC MPP) hours

8.7 Trilinos Libraries for Scalable, Resilient Manycore Computations

Principal Investigator: Michael A. Heroux, Sandia National Laboratories
NERSC Repository: N/A

8.7.1 Project Description

8.7.1.1 Overview and Context

Trilinos provides a large collection of reusable software components in the form of *packages* for many common requirements in scientific and engineering applications. Capabilities include data creation and management: geometry, meshing, discretization, load balancing and data redistribution, construction of data objects (dense matrices/vectors, sparse matrices/graphs and related) and I/O. All of these capabilities are designed for scalable systems, with growing support for many-core, accelerator and hybrid systems. Trilinos also contains a large collection of state-of-the-art algorithms for linear, eigen, nonlinear, transient, optimization and UQ problem solvers in a framework that utilizes Trilinos data services, but also permits and encourages easy user customization. Trilinos also provides uniform access to many third-party libraries where substantial capabilities are available from the broader community.

Historically Trilinos has provided unique capabilities for unstructured PDE computations, inherently discrete problems such as circuit simulations and hybrid integral FEM problems. Structured stencil applications can use Trilinos, but we do not typical take advantage of the simplicity these formulations represent. In particular, we do not have turnkey data structure support or specialized preconditioners for this class of problems.

Some problem areas where Trilinos is best known are scalable unstructured sparse multilevel preconditioned linear solvers, scalable eigensolvers, robust multi-scale, multi-physics solvers, load balancing and partitioning and, most recently, advanced embedded optimization and uncertainty quantification coupled with automatic differentiation. Trilinos is written mostly in C++ and makes full use of C++ language features for efficient polymorphism and compact source code, including well-defined abstraction layers used throughout Trilinos and template meta-programming for compile-time polymorphism.

8.7.1.2 Objectives for 2017

Trilinos is a very large project with dozens of simultaneous research initiative underway at any given time. I will highlight three broad areas that have cross-cutting importance and involve the efforts of many people:

1. **Advanced solvers:** Most solver research efforts on the Trilinos project are focused on tightly coupled multiphysics and embedded nonlinear analysis, optimization and uncertainty quantification. The demands of these problem formulations drive research in all layers of the solver stack, requiring more robust transient and nonlinear solvers, and new forms of linear solvers, such as solvers for families of related systems. Algorithm advances in these areas are naturally complementary for extreme scale systems because these advanced solver formulations force problem sizes to grow dramatically. In the 2017 time frame we expect to have at least six large-scale applications using the full capabilities of Trilinos for fully coupled problems using embedded optimization and UQ.

2. **Algorithms and data classes for scalable many-core and accelerator based systems:** The slowed growth of internode parallelism and the dramatic growth of intranode parallelism have created a huge disruption in the algorithms and software stack for scalable computing. We are now several years into the analysis and re-design of data classes, and have made significant progress in many-core/accelerator versions of key scalable solvers. But there is much more to do. We can demonstrate excellent many-core/accelerator scaling for most solver algorithms used by our most advanced users, but we still need improved many-core/accelerator smoothers for multigrid preconditioners. We expect to have adequate capabilities by 2017, coming from (i) better approaches to extracting fine-grain, data-driven parallelism from existing algorithms, (ii) development of low-rank approximations to off-diagonal blocks in sparse direct methods (both exact and inexact), (iii) advances in Krylov methods, such as pipelining and s-step formulations, block Krylov (simultaneous RHS) and families of related systems and (iv) better use of aggregation in multi-DOF problems.
3. **Resilient computations:** We anticipate a dramatic decrease in the reliability of computations over the next decade on leadership computing systems. In preparation for this—and to address the already-emerging needs—we are exploring several resilient computing models that will enable applications to succeed in the presence of faults. We are focused on approaches that (i) permit the application to continue and make progress in the presence of performance variability due to error detection and correction, (ii) enable recover locally from process failure, and (iii) detect and correct soft errors. By 2017 we anticipate that all of our mainstream solvers will be latency tolerant. We also expect several applications to be built on top of Trilinos-provided reliable data services that enable application-driven recovery from process loss. We also expect to have solvers that can detect and recover from silent data corruption that occurs within the solvers (where often 80% or more of the application execution time can be).

8.7.2 Computational Strategies (now and in 2017)

8.7.2.1 Approach

Trilinos is used in many different application settings. Some of the most common are:

1. **Coupled multi-physics:** Trilinos provides a collection of interoperable solver components for nonlinear multi-physics problems. Typically, the nonlinear solver must handle very stiff systems, and may require continuation to converge. In turn, the underlying preconditioned linear solver is also highly stressed. Preconditioners are typically user-defined physics-based formulations where general algebraic preconditioners from Trilinos are used as subsolvers. In this setting, we see the combined use of many Trilinos capabilities. Our most sophisticated users in this area utilize as many as 30 Trilinos packages, and numerous third-party libraries, e.g., SuperLU. In the future, we expect to see an increase in this usage model for Trilinos, commensurate with the first strategic objective discussed in Section 2.2. We continue to evolve our library component model to make interoperability of Trilinos packages and other third-party capabilities easier.
2. **Scalable unstructured single-DOF solves for Poisson-like operators:** Another common use of Trilinos is for the solution of single-DOF problems that arise in applications such as segregated solution of CFD problems, where solution of the

pressure-Poisson problem is very challenging. In these settings, load balancing of the sparse matrix and the advanced use of a robust algebraic multigrid preconditioned GMRES solver are essential, along with efficient local grid smoothers. The most important efforts in preparation for 2017 are (i) the development of efficient many-core/accelerator smoothers, and (ii) continued advancement in multigrid methods, especially latency hiding designs and implementations.

3. **Beginning-to-end Trilinos-based applications:** The Albany application framework provides application-level reusable components based on Trilinos capabilities. Albany provides a growing collection of beginning-to-end Trilinos-based applications that are characterized by containing only problem-specific code for defining the target problem formulation and otherwise relying on sophisticated parameterizations of Trilinos capabilities. Using this approach, a new application can go from first concept to a scalable full-featured, manycore/accelerator-enabled code that can provide an optimal solution with uncertainty quantification in less than a year. We expect this framework/component approach to grow in popularity.

8.7.2.2 Codes and Algorithms

See above.

8.7.3 HPC Resources Used Today

8.7.3.1 Computational Hours

Not applicable. The Trilinos team itself uses very few NERSC hours itself.

8.7.3.2 Parallelism

Core counts are highly variable. We have some users (e.g., Denovo) that use the entire Titan system at ORNL, and could use any such system at scale. Other important users seldom scale beyond 10,000 cores.

There are several reasons why problems are not run at scale. The most common are lack of scalability due to load imbalance, lack of algorithmic scalability, and lack of need to scale beyond a certain problem size.

8.7.4 HPC Requirements in 2017

8.7.4.1 Computational Hours Needed

We have not traditionally used a lot of NERSC time over the past few years. We do at times run scalability analyses using NERSC systems. These runs seldom add up to more than 100 total core hours, but provide crucial data for our publications and these hours are hard to find on other systems. We don't expect large allocations elsewhere in the 2017 time frame.

If we engage in NERSC-specific work, we could benefit from additional compute hours.

8.7.4.2 Parallelism

For some applications we expect to use the entire system, factoring the (hyper) thread/process space appropriately between MPI processes and thread processes. Right now this factoring leads to 4-8 threads per MPI process and the rest used by MPI.

8.7.4.3 I/O

Trilinos provides I/O libraries in the Trios package. These libraries can be used to perform CPR. We are also working on a collection of persistent storage data objects for support of application level fault tolerance. The amount of data read or written and the rate at which it is written is not applicable to us, since it depends on what users of the software do.

8.7.4.4 Future Data Needs

Not applicable.

8.7.4.5 Memory Required

This is very application dependent. Presently most of our users can do well with 1GB/core. This is down from a few years ago when it was 2GB/core. We anticipate that this value will continue to drop as we shift computation to multi-threaded mode and become more careful in the strong scaling regime. By 2017, we hope to realize optimal performance at 256MG/core for our key applications.

8.7.4.6 Emerging Technologies and Programming Models

Trilinos supports CUDA. We have no plans to support OpenCL until (if ever) OpenCL provides modest C++ compilation capabilities. OpenCL is useless to us until then.

Trilinos runs in production mode on Titan and elsewhere with GPU support. At the same time, we are still ramping up support, and still need significant algorithm development in order to provide full GPU capabilities.

Trilinos has OpenMP as an optional node parallel model. Use is limited at this point, but we see a dramatic increase in use on Intel MIC platforms.

Trilinos has been ported to Sequoia (but not to Mira as far as I know). We still have some compiler issues with some packages, but the core Trilinos capabilities work.

Trilinos is being actively developed for Intel MIC. We work closely with Intel on performance and programming issues.

We have funding from several sources to work on next generation preparations (not sure if this is answers the question as you intended).

Providing occasional access to NERSC systems would be very helpful for studying algorithm and software scalability.

The role of DOE and ASCR in this work would involve funding of algorithms R&D, access to emerging systems, researcher training on new systems.

8.7.4.7 Software Applications and Tools

We will need optimized (threaded/vectorized) dense BLAS and LAPACK, or similar libraries in the 2017 time frame. For the most part, we expect these libraries to be provided by the vendor, but we access them through our own abstraction layers and can use other libraries. We also need to have at least one node-parallel sparse direct solver library, and one distributed-parallel sparse direct solver that runs across as many nodes as possible.

We need a full-featured C++ compiler environment that is very robust and efficient in compilation.

We anticipate that within the 2017 time frame we will have interoperability with some *in situ* analysis libraries, where computational and analysis data containers are compatible and can be used in combination, in-memory. In this case, we would benefit from installation of these tools.

8.7.4.8 Additional Data Intensive Needs

We would find it useful to have non-volatile storage that could be used for persistent data. Some of our resilient computing models rely on this type of capability.

We are typically not the owners of data, but provide filters through which data pass, so we are not addressing a data management plan here.

8.7.4.9 Requirements Summary Worksheet

Not applicable.

8.8 Sparse Direct Solver SuperLU

Principal Investigator: Xiaoye Sherry Li, LBNL
NERSC Repositories: mp127

8.8.1 Project Description

8.8.1.1 Overview and Context

We develop direct solver libraries for sparse linear systems to support a wide range of scientific computing research. The most parallel library is SuperLU_DIST, which is MPI-only at present. There are many users: 27,403 downloads in FY13. It is used in many high-end scientific simulation codes important to DOE. A survey of NERSC usage between June 21, 2012 and January 17, 2013 via the ALTD facility showed that SuperLU is the thirteenth most heavily used library at NERSC, with about 100 unique users during that period. SuperLU is also included in many commercial libraries, including Cray's LibSci, FEMLAB, HP's MathLib, IMSL, NAG, OptimaNumerics, Python (SciPy).

The factorization algorithm is considered a "direct" method, which is numerically robust, and there is no convergence issue. The drawback is that it requires large amount of memory. A lot of research effort is devoted to reduce memory usage while maintaining speed.

8.8.1.2 Objectives for 2017

Our goals for 2017 are:

- Strong scaling to 10K nodes with hybrid programming: MPI + OpenMP + CUDA/OpenCL/xx.
    - Preliminary results on the NERSC Dirac GPU test bed system show a 3x speedup relative to MPI-only using eight CPU cores + 1 GPU
- Solve equations with 50M-100M degrees of freedom (presently 10M-20M).
- Be capable of using any heterogeneous architecture.
- Algorithm-based fault tolerance (ABFT) resilience with fault detection & recovery.

8.8.2 Computational Strategies (now and in 2017)

8.8.2.1 Approach

Our computational problem and strategies for solving it include:

- Sparse LU, sparse triangular solve
- Supernode partition, 2D block cyclic matrix/process distribution
- Numerical pre-pivoting via weighted maximum bipartite matching
- Sparsity ordering with parallel graph partitioning: ParMETIS, PT-Scotch, Zoltan
- Parallel symbolic factorization

Our biggest computational challenges are:

- Task & data dependency (esp. triangular solve)
- Low arithmetic intensity

- Pre-pivoting is the serial bottleneck
- How to speed up non-BLAS-like operations: scattering, graph traversal

We expect our computational approach and/or codes to change by 2017:

- Alternative to pivoting: Random Butterfly Transformation (RBT)
- Expose more data parallelism to utilize GPU, MIC, etc.

8.8.2.2 Codes and Algorithms

We use the following software:

- MPI, BLAS, ParMETIS, PT-Scotch

8.8.3 HPC Resources Used Today

8.8.3.1 Computational Hours

The repository mp127, High Performance Sparse Matrix Algorithms, used 1.5 million hours at NERSC in 2013.

8.8.3.2 Parallelism

SuperLU often runs using 100s – 1,000s of cores.

8.8.4 HPC Requirements in 2017

8.8.4.1 Computational Hours Needed

Based on historical trends and anticipated needs, mp127 will require about 20 M hours in 2017.

8.8.4.2 Parallelism

We expect about 10-20x greater concurrency than today, meaning we will use 50K – 100K cores).

8.8.4.3 Emerging Technologies and Programming Models

We have started CUDA development. AMD is interested in developing OpenCL. Our software should be running on Titan soon. The code has OpenMP directives now but does not run on the Mira or Sequoia Blue Gene/Q systems. A port to the Intel MIC architecture is planned. We are collaborating with Rich Vuduc's group at Georgia Tech for many-core developments and energy-aware algorithms. NERSC could probably help with n-node performance models, performance tools for threads, GPUs. DOE and ASCR needs to provide sustained funding for code development and maintenance.

8.8.4.4 Requirements Summary Worksheet

NERSC Repository mp127	**Used at NERSC in 2013**	**Needed at NERSC in 2017**
Computational Hours	1.5 M	20 M
Typical number of cores used for production runs	100s-1,000s	10 K
Maximum number of cores that can be used for production runs	5,000	50K
Archival data	1,127 TB	3,000 TB
Shared (project) data	0.8 TB	1 TB
Memory per node	Can use all	Can use all
Aggregate memory	0.1 TB	1 TB

9 Computer Science Case Studies

9.1 Requirements for Parallel I/O, Visualization and Analysis

Principal Investigator: E. Wes Bethel
Worksheet Author(s): Prabhat, Quincey Koziol
NERSC Repositories: m636

9.1.1 Project Description

9.1.1.1 Overview and Context

This case study encompasses several distinct but related projects.

LBL "Vis Base" program projects provide fundamental and applied visualization/analytics R&D to produce technologies aimed at meeting high performance visualization and data understanding needs of DOE's science community, particularly those we expect to encounter at the exascale.

Our ExaHDF5 work focuses on enabling high-performance parallel I/O for science codes and analysis tools on DOE platforms through a combination of fundamental research in optimization techniques; methodologies for hiding complexity of parallel I/O (auto-tuning, transparent data re-organization, high-level APIs) and index/query methods. Features are rolled into HDF5, and deployed on production I/O stacks on DOE HPC facilities.

The MANTISSA project involves research and development of scalable algorithms in statistics, machine learning and graph analytics, with an application to data-centric problems in problems in the areas of mass spectrometer imaging, climate, cosmology, genomics and high energy physics. Developed methods will be tested on multi-TB sized datasets on petascale class machines.

We also have several "Data Exploration at the Exascale" projects, which include R&D that focuses on finding ways to reduce the amount of data written to disk by codes via new approaches in situ processing that enable traditional post-processing exploratory visual data analysis.

There is also Topological Data Analysis at the Extreme Scale. This involves R & D that focuses on developing scalable algorithms for topological data analysis and feature detection with the goal of in situ processing, analysis of massive data sets and data triage and reduction.

Finally, Optimizing Power Usage for Data-Intensive Workflows and Algorithms on Modern Computing Architectures projects is a study of power usage of visualization and analysis software infrastructure on modern platforms, and perform R&D aimed at reducing power consumption of these data-intensive software algorithms, libraries, and applications (visualization, analysis, I/O).

9.1.1.2 Objectives for 2017

Our primary goals for 2017 include:

- Demonstrating successful application of visualization techniques to PB-sized output
- Demonstrating HDF5 (and production I/O stack) scaling on current petascale and future exascale platforms
- Demonstrating sophisticated Big Data analytics techniques applied to TB-sized complex, multi-modal datasets (simulations, experiments, observations)

9.1.2 Computational Strategies (now and in 2017)

9.1.2.1 Approach

We do not develop computational simulation codes. Rather, our efforts are more focused on efficient rendering, analysis and I/O algorithms and software. We use hybrid parallel and domain decomposition strategies in visualization and research tools. We use a variety of optimization techniques (collective buffering, compression and auto-tuning) for our I/O work.

9.1.2.2 Codes and Algorithms

Simulation codes: VPIC, Chombo, FLASH, MOAB, SPH, IMPACT-Z, VORPAL, Warp, CAM5

I/O: HDF5, NetCDF

Vis: VTK, VisIt, Paraview

In terms of characterization, we operate on the following data models and motifs in our research:

I/O: particle, block structured, unstructured, AMR meshes

Vis: volume rendering, ray casting, streamline computation

Analysis: Big Data motifs (sparse/dense linear algebra, stochastic optimization, graph analytics)

9.1.3 HPC Resources Used Today

9.1.3.1 Computational Hours

We used just over 5 million hours at NERSC in 2013.

9.1.3.2 Parallelism

Typical visualization and analysis jobs run at 1 K-10 K cores. One-time hero exercises are conducted at 100 K cores. Codes are capable of scaling to 100 K for visualization, to 50 K for analysis, and to 500 K for I/O.

Typical I/O jobs run at 10 K cores. Several hero runs have been attempted on full scale on Hopper/Edison.

The typical number is generally less than the maximum because of insufficient real-world use cases for visualization involving 100K core runs [not enough data to read from multi-variate simulations]. A similar story holds true for analysis (although that will change in the 2017 timeframe).

9.1.3.3 Scratch Data

We would need about 500TB.

We predominantly use Parallel I/O in our projects. We try to use collective buffering wherever possible, and avoid reads/writes to/from a single node. We use shared I/O, there are a few cases wherein we use independent I/O for writes. Our codes generally write to medium and large files.

9.1.3.4 Shared Data

We have several project directories: vacet, cascade/m1517, mantissa. The primary reasons for project folders in ease of sharing data, joint development and sharing of code. In 2013, we used 33 TB of disk space in these directories.

9.1.3.5 Archival Data Storage

We have about 184 TB stored on HPSS now.

9.1.4 HPC Requirements in 2017

9.1.4.1 Computational Hours Needed

We estimate needing 50M hours in 2017. We rely on NERSC for most of our R&D and so we do not anticipate using other resources to any great extent. The increase over 2013 is due to larger dataset sizes (problem configurations from our science collaborators) and more sophisticated analysis problems.

9.1.4.2 Parallelism

We would like to use a hybrid parallel strategy (#MPI tasks == # nodes [O(1K)] and pthreads/OpenMP [=#hardware threads O(10) on each node]. We estimate that the maximum concurrency that could be used in 2017 of order one million total.

9.1.4.3 I/O

Our software typically does use checkpoint/restart. We would typically read and write 100-GB - 1 PB of data per run in 2017. We estimate bandwidth requirements to be ideally 1-10TB/s, so that 5-10% of a run is consumed by I/O. We try to follow two rules of thumb to generally spec out I/O:

- checkpoint all of memory to somewhere in 15-20 minutes
- spend <10% time on I/O

9.1.4.4 Future Data Needs

In 2017, we expect to need 1,000 TB of temporary scratch disk space, 1,000 TB of NERSC project space (globally accessible shared data), and 5,000 TB of storage on NERSC HPSS. The growth in these requirements relative to 2013 is due primarily to larger dataset sizes and problems being attempted by our collaborators.

We would like a permanent repository for climate data. Some other projects might be interested in storage for about a 5-year timeframe.

9.1.4.5 Memory Required

We would need aggregate memory in the 1 TB-500 TB range.

9.1.4.6 Emerging Technologies and Programming Models

A small class of visualization software is currently utilizing CUDA/OpenCL. This is not relevant to I/O. The software is "not quite" running in production on Titan or elsewhere using GPUs.

At this point, we have only experimented with OpenMP and pthreads. Software is not running on Mira or Sequoia using threading.

Experiments are underway in porting to, and optimizing for, the Intel MIC architecture

Visualization activities are well funded to pursue this kind of exploration. However, there is insufficient funding for statistics and machine learning efforts in this space. I/O is currently unfunded in this space.

In our opinion, additional resources are needed at NERSC to systematically explore issues related to energy efficiency and many-core. We believe that we need more NERSC staff and user training in assisting with this transition. The petascale post-doc program was a success, and should be continued in some form. In the case of I/O, we believe that industry and research collaborations (similar to the Intel/Whamcloud fastforward program) are the way to go for exploring I/O middleware solutions.

9.1.4.7 Software Applications and Tools

We need HDF5, NetCDF. VisIt, Paraview and "Big Data software solutions" (when they become available).

9.1.4.8 HPC Services

The main service we'll need is consulting. 'Big Data' analysis will be very important for us going forward. Some degree of training would be appreciated. We will not be relying on NERSC to publish data or results.

9.1.4.9 Additional Data Intensive Needs

Workflow tools will be important going forward. We would also like standardized mechanisms for sharing data.

9.1.4.10 Additional Data Intensive Needs: Burst Buffer (BB)

We believe that BB hardware is definitely relevant and important for accelerating parallel I/O operations (reads and writes). But we need the software stack (HDF5, Lustre/GPFS) to intelligently use the hardware. Some form of transparent data pre-fetching optimizations, or staging will be helpful. The hardware will also be relevant for *in-situ* and in-transit visualization solutions (via GLEAN, Paraview and VisIt).

9.1.4.11 What Else?

We firmly believe that typical HPC users want I/O performance to be reliable, and they do not want to know any more about the I/O subsystem (striping, staging, interference, noise) than they absolutely need to. Approaches such as auto-tuning are headed in the right direction, but are probably too ambitious to attempt on a center-wide scale. We think that

I/O Quality of Service (i.e. you are guaranteed sustained 20GB/s performance during your job execution) is a reasonable metric to aim towards and that NERSC should consider that for future hardware.

We are generally quite happy with professional quality of services rendered by NERSC, and Cray/NERSC staff collaboration. We feel that NERSC is lagging behind other HPC centers in terms of I/O hardware provisioning, which is turning out to be a detriment to I/O researchers pushing the state of the art.

Looking at the future, we believe that NERSC can play a leading role in DOE for 'Big Data', but a clear articulation, formulation and execution of a data strategy is required. This strategy needs to be presented in contrast to the existing HPC/Exascale initiatives, and needs to identify requirements emerging from experiments and instruments.

9.1.4.12 Requirements Summary Worksheet

NERSC Repository m636	**Used at NERSC in 2013**	**Needed at NERSC in 2017**
Computational Hours*	5.1	50 M
Typical number of cores used for production runs	10,000	10,000
Maximum number of cores that can be used for production runs	150,000	5,000,000
Data read and written per run	0.1-30 TB	0.1-1,000 TB
Maximum I/O bandwidth	35 GB/sec	5,000 GB/sec
Percent of runtime for I/O	1-30%	5%
Scratch File System space	250-500 TB	1,000-5,000 TB
Shared filesystem space	33 TB	1,000 TB
Archival data	184 TB	5,000 TB
Memory per node	100%	100%
Aggregate memory	TB	500 TB

*Normalized to Hopper-equivalent (NERSC MPP) hours

9.2 Requirements for Scalable Scientific Data Management

Principal Investigator: Arie Shoshani, LBNL (Note: Arie Shoshani was PI of these two projects at the time of the review. As of September, 2014, Keshang Wu is PI.)
Worksheet Author: Suren Byna, LBNL
NERSC Repositories: m1248 (Scientific Data Management Research); sdmstor (Storage for the Scientific Data Management Research Group)

9.2.1 Project Description

9.2.1.1 Overview and Context

The Scientific Data Management Research Group (SDM) at LBNL develops tools for efficient access and storage management of massive scientific datasets. Large scientific simulations and experiments produce not only enormous quantities of data but also complex and heterogeneous datasets that require effective and efficient management, a task that can distract scientists from focusing on their core research.

SDM researchers are working to provide a coordinated framework for the unification, development, deployment, and reuse of scientific data management software. In short, this will allow scientists to define requests for data on their own terms to tap into a transparent data management and access infrastructure.

SDM group's work represented here spans multiple interrelated projects: (1) Scientific Data Management at Extreme-scale Computing, (2) FastBit – Bitmap Indexing to search scientific data, (3) ExaHDF5 and (4) International Collaboration Framework for Extreme Scale Experiments (ICEE). In preparation of scientific data management at extreme-scale computing, SDM group is currently working on an array data management system, called Scientific Data Services (SDS) framework, for accelerating data analysis tasks. SDS targets to provide various data management services, such as data reorganization, compression, indexing, smart data prefetching, etc., while hiding the complexity of storage system. We are working on FastQuery, a library for generating bitmap indexes and querying data in various file formats, using parallel resources of multi-core and distributed memory systems. We also directly work with various applications in improving parallel I/O performance while writing tens of TBs of data. Our recent large-scale simulation on Hopper produced 40TB data written to a single HDF5 file, testing the boundaries of various software layers of the parallel I/O subsystem. The ICEE project focuses on facilitating *in situ* and in transit analysis of data from scientific domains including high-energy physics, fusion, climate, etc. The ICEE workflow system leverages in transit capabilities of ADIOS and selective data access capabilities of FastBit.

Our goals may be summarized as follows:

- Providing services to bring the merits of database management systems and parallel file systems together;
- Programming interfaces for accessing arrays and for executing queries;
- An optimization interface between data format libraries (HDF5, NetCDF, and ADIOS-BP) and file system optimizations.

Research activities include:

- Optimizations for accessing data in post-process phase via data reorganization: replicate data in different organizations for accesses
- In-memory data processing and querying
    - Query optimization
    - In memory Indexing
- Runtime support for deep memory hierarchies

The direct benefit to users from our work includes:

- A data model familiar to domain scientists;
- Existing file formats and analysis tools can co-exist with the new system;
- Optimization of data access based on queries on data model;
- Dynamic reorganization of data based on access patterns;
- Large energy savings by accessing only data needed from disk;
- Reducing data movement in memory using in-memory indexing, thus reducing energy usage;
- Easier to integrate with analysis and visualization tools – integration can now be done at the data model level.

9.2.1.2 Objectives for 2017

All the projects mentioned above use HPC resources extensively. By 2017, we expect to have a fully functional Scientific Data Services framework. We expect to deploy and exercise various data management tasks, including automatic provision of indexing, querying, data analysis operator services. Our requirements are more memory, deeper memory hierarchy (including burst buffer), staging nodes for smart data management, and faster I/O.

9.2.2 Computational Strategies (now and in 2017)

9.2.2.1 Approach

Based on the size of data, indexing, querying, and analysis operators will need more computation than today. We expect to have staging nodes dedicated for data management tasks. The following three figures indicate the process of data management, showing the overall data flow (Figure 1), the current state (with data formats indicated, Figure 3), and how this project intends to provide solutions (Figure 3).

9.2.2.2 Codes and Algorithms

Scientific Data Services (SDS) code. More details of SDS are presented in the paper located at: https://sdm.lbl.gov/~sbyna/research/papers/2013-PDSW2013-SDS.pdf

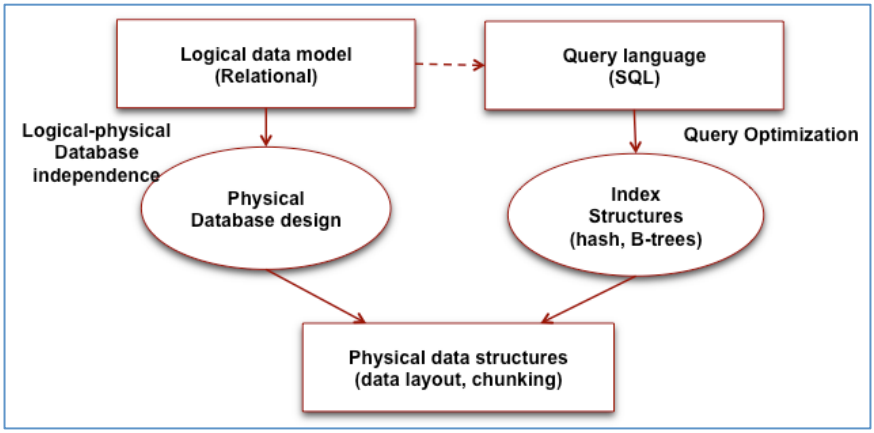


Figure 1. Data flow for traditional data management.

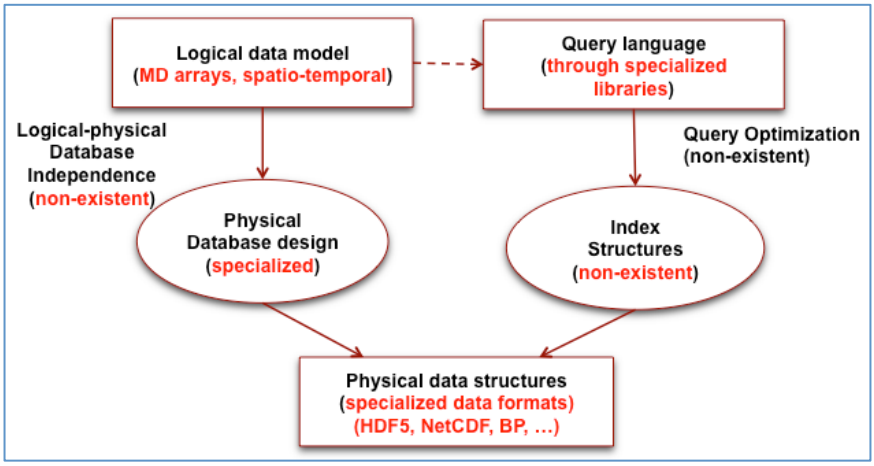


Figure 2. The current state of data management.

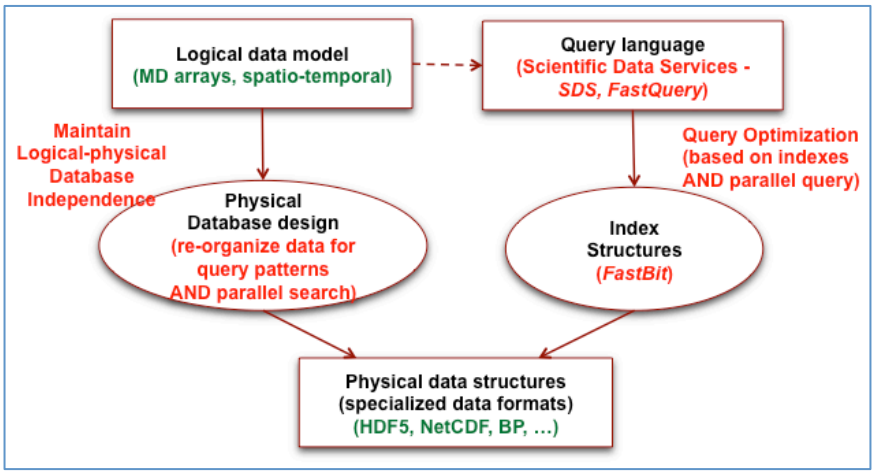


Figure .3 How the SDM project fills the gap.

9.2.3 HPC Resources Used Today

9.2.3.1 Computational Hours

We used 4 million NERSC MPP hours in 2013.

9.2.3.2 Parallelism

Typical parallel concurrencies:

- Bitmap indexing: up to 20,000 cores
- Parallel sorting: up to 10,000 cores

Memory used:

- For VPIC simulation: 90% of the total memory on Hopper
- Index generation: 75% of memory on each node

9.2.3.3 Scratch Data

Data read/written per run:

- Bitmap indexing: 0.3X to 3X the original data size
- E.g., Index of one trillion particle data: ~150 TB for 12 time steps and about 50 TB /scratch space on Hopper for testing SDS.

Also, simulations such as VPIC and AMR codes need fast checkpoint/restart capabilities.

9.2.3.4 Shared Data

We have about ~250 TB for particle indexes data on /project.

9.2.3.5 Archival Data Storage

We consume about 500 TB for particle data on HPSS.

9.2.4 HPC Requirements in 2017

9.2.4.1 Computational Hours Needed

We project about a 7.5X increase in our usage. See the following table, which contains data for sections 9.2.4.2 through 9.2.4.5.

	Compute Hours	Target Concurrency	Data read/ written per run	Memory per node	Required software	Resources used	Data Stored
Current 2014	4M	10K-150K	100GB-30TB	100%	HDF5, NetCDF, MPI, MPI-IO, pthreads, OpenMP, ScalaPACK, BLAS	/scratch /project	250-500 TB
Estimated 2017	30M	10K-10M	100GB-1PB	100%	HDF5, NetCDF, MPI, MPI-IO, MPI+X?? ScalaPACK, BLAS	/scratch /project Burst Buffers	1-5 PB

9.2.4.2 Emerging Technologies and Programming Models

Depending on which analysis codes are needed, the SDS framework and ICEE plan to use heterogeneous processing. FastQuery currently uses MPI + Pthreads and the VPIC code uses MPI+OpenMP. Porting to, and optimizing for, the Intel MIC architecture is not planned, but we are considering it for data management and computing operators.

An important role that NERSC and DOE could play in this is

- Providing new architectures for co-locating simulation and analysis
- Providing new architectures for dedicated nodes for smart management of data movement
- Funding efforts for energy efficient data management research

Although not necessarily related directly to many-core systems, we also note that:

- Performance and power consumption monitoring at CPU and system levels is key for identifying bottlenecks.
- Performance monitoring at file system level is needed for improving parallel I/O.
- Power consumption monitoring at storage system level is needed for improving energy efficiency of data movement.

9.2.4.3 Additional Data Intensive Needs

Our project would benefit from dedicated nodes for offering scientific data services; from NVM/NVRAM for analysis data that does not fit in memory and for prefetching data (as noted below); and from faster file systems.

9.2.4.4 Additional Data Intensive Needs: Burst Buffer

Burst buffers will be useful for *in situ* and *in transit* analysis when available memory is not sufficient. The SDS framework can use a burst buffer or storage in a staging area for prefetching and reorganizing data.

9.2.4.5 Requirements Summary Worksheet

NERSC Repositories m1248, sdmstor	**Used at NERSC in 2013**	**Needed at NERSC in 2017**
Computational Hours*	4 M	30 M
Typical number of cores used for production runs	20,000 (SDM research) 100,000 (ExaHDF5)	50 K (SDM) 1,000 K (ExaHDF5)
Maximum number of cores that can be used for production runs	20 K (SDM research) 100 K (ExaHDF5)	50,000 (SDM) 1,000,000 (ExaHDF5)
Data read and written per run	400 TB	2,000 TB
Maximum I/O bandwidth	28 GB/sec	1,024 GB/sec
Percent of runtime for I/O	30%	10%
Scratch File System space	400 TB	2,000 TB
Shared filesystem space	250 TB	1,000 TB
Archival data	554 TB	5,000 TB
Memory per node	29 GB	64 GB
Aggregate memory	100 TB	1000 TB

*Normalized to Hopper-equivalent (NERSC MPP) hours

9.3 Performance Optimization of Scientific Applications - MPAS-Ocean & the HipGISAXS Suite

Principal Investigators: *SUPER:* Bob Lucas, USC, and Leonid Oliker, LBNL; *X-Ray:* Xiaoye Li, LBNL
Worksheet Author: Abhinav Sarje (LBNL)
NERSC Repositories: m88, m1285, als

9.3.1 Project Description

9.3.1.1 Overview and Context

Within the SUPER (Sustained Performance, Energy and Resilience) Institute, we are working towards performance analysis, parallelization and optimization of the *MPAS-Ocean* code on HPC systems. This is an MPI-based code for ocean modeling and simulations. Our work includes incorporation of on-node multicore parallelization using OpenMP, and optimizations through better exploitation of data locality, data organization, and optimal data partitioning for minimization of data transfers and communications.

We are collaborating with the Advanced Light Source to develop HPC solutions for X-ray scattering data analysis. We have developed a massively parallel *HipGISAXS software suite* for this purpose, and currently the forward simulation has been optimized for various parallel architectures including GPUs, Intel MIC as well as multi-core CPUs. On-going work includes development of inverse modeling for nanostructure discovery in materials.

9.3.1.2 Objectives for 2017

The SUPER goals for MPAS-Ocean code are as follows.

- Implement improved parallelization and develop codes for various parallel architectures including multicore CPUs, Intel MIC and GPUs.
- Improve FLOP performance to get near peak on a given HPC system.
- Achieve high Simulated Year per Day (SYPD) performance on fine/high-resolution grids (e.g. realize more than 10 SYPD on 15km mesh resolution using around 3000 cores).
- Improve code scalability to utilize 100,000 cores.

The goals of HipGISAXS suite development are as follows. Realize real-time scattering data analysis through improved FLOP performance to achieve high efficiency on a given HPC system and minimized inter-node communication. Match data analysis rate to data generation rates at current beamline detectors (100 TB/week). Development and implementation of better data analysis and inverse modeling parallel algorithms to effectively utilize large HPC systems.

9.3.2 Computational Strategies (now and in 2017)

9.3.2.1 Approach

MPAS-Ocean models and simulates Earth's oceans through the use of unstructured multi-resolution/multi-scale mesh discretization. Currently, the data distribution across nodes on

a HPC system is performed through an existing graph and mesh partitioning tool Metis. Inter-partition communication involves multi-layered halos (ghost cells), typically 3 layers, causing high data volume transfer to compute ratio. Load imbalance is caused through variable cell depths. Further, the mesh partitions are assigned to nodes in an arbitrary order, and also the mesh cells within a partition are processed in an arbitrary order. A number of strategies can be used to improve parallelization and performance. Presently I am exploring within-partition cell reordering techniques to improve data locality and cache usage, as well as partition ordering to improve communication performance. I am also exploring the use of OpenMP for multi-core parallelization. Better partitions can be achieved through weighted partitioning, and exploration of other partitioning tools. Communication avoidance would reduce the data volume to compute ratio.

The forward simulation of X-ray scattering is an embarrassingly parallel computational problem making use of massive parallelism easier, with minimal inter-node communication. I have implemented a logical hierarchical parallelization framework that can schedule independent computations occurring at a number of stages during the simulations. It also incorporates effective exploitation of hardware parallelism hierarchy. Inverse modeling for nanostructure detection is an optimization problem with uses a number of forward simulations, but generally this series of simulations is inherently sequential. We are exploring various optimization algorithms and machine learning techniques to improve the inverse modeling performance.

9.3.2.2 Codes and Algorithms

1. MPAS-Ocean: It uses multi-scale unstructured meshes over a sphere, and halo exchanges to perform quasi-stencil computations for simulating Earth's oceans. We are collaborating with the MPAS-Ocean development team at Los Alamos National Laboratory.

2. HipGISAXS suite: This code uses irregular but structured volume mesh. A simulation involves computation of Form Factor and Structure Factor at each mesh cell, and is based on the Distorted Wave Born Approximation theory. We are collaborating with beamline scientists from the Advanced Light Source (ALS) at Lawrence Berkeley National Laboratory.

9.3.3 HPC Resources Used Today

9.3.3.1 Computational Hours

MPAS-Ocean used about 1.7M core hours at NERSC in 2013 (4.3M total for SUPER m88).

HipGISAXS used about 2.4M core hours at NERSC in 2013.

9.3.3.2 Parallelism

A typical production run for MPAS-Ocean uses 1,000-3,000 cores. That for HipGISAXS uses about 3,000 cores.

MPAS-Ocean is able to scale to about 3,000 cores. HipGISAXS can scale to current full machines (144,000 cores).

A typical HipGISAXS simulation currently does not involve complex nanostructures and so fewer than maximum cores are sufficient. Few simulations with complex nanostructures demand use of large number of cores.

For MPAS-Ocean, strong scaling is more important. This is to achieve a high Simulated Year Per Day (SYPD) performance rate for a given mesh resolution.

For HipGISAXS, both strong and weak scaling are important due to the need for simulation of structures with various complexity levels, although the volume mesh is typically constant as dictated by beamline detector resolution. Faster computations are required to simulate a complex system. Solution to a bigger problem involving large number of system configurations in allocated time duration is also needed.

9.3.3.3 Scratch Data

Requirements in this area are fairly minimal: MPAS-Ocean requires about 50GB of scratch space; HipGISAXS requires around 100GB.

9.3.3.4 Shared Data

The SUPER collaboration currently uses 'm88' to store performance analysis data.

The X-Ray collaboration (m1285) and 'als' also have project spaces, used host the code repositories and publicly available webpages.

9.3.3.5 Archival Data Storage

We used little archival storage (1.4 TB in 2013).

9.3.4 HPC Requirements in 2017

9.3.4.1 Computational Hours Needed

We estimate that MPAS-Ocean and HipGISAXS will require 10M core hours each in 2017. We expect to have significant allocations from the two leadership facilities and from TACC at that time.

By 2017 we would have inverse modeling implemented in HipGISAXS and this requires multiple forward simulations; this accounts for the increase in hours.

9.3.4.2 Parallelism

By 2017 both codes should be able to use 10,000 MPI Tasks. For MPAS-Ocean, each task would have about 10 threads. For each task would have about 10-1,000 threads. (GPUs and Intel MIC)

9.3.4.3 I/O

The MPAS-Ocean code has built-in checkpoint/restart but the HipGISAXS does not. We write only about 10GB per run, so the bandwidth and maximum I/O time per run is not applicable.

9.3.4.4 Future Data Needs

In 2017, we expect to need 1 TB of temporary scratch disk space, 1 TB of NERSC project space (globally accessible shared data), and 2 TB of storage on NERSC HPSS. The growth in these requirements relative to 2013 is due primarily to experimental data accumulation.

9.3.4.5 Memory Required

MPAS-Ocean will require about 50GB per node. The requirements for HipGISAXS are minimal.

9.3.4.6 Emerging Technologies and Programming Models

MPAS-Ocean does not have CUDA extensions. Plans include adding them. HipGISAXS has CUDA extensions and very effectively exploits multicore, GPU, and Intel MIC architectures at the node level. The code runs at TACC and on Titan.

MPAS-Ocean does not have OpenMP directives, but are being added.

HipGISAXS has OpenMP directives and are used. Porting to Mira is planned. The SciDAC SUPER institute is helping us with all these activities. We would like NERSC to provide state-of-the-art testbeds and production systems with these architectures. We would like ASCR to provide funding for latest hardware procurement. Continual, year-round addition of testbeds hosting latest processor technologies/hardware would play important role in adapting codes to the state-of-the-art.

9.3.4.7 Software Applications and Tools

We need access to GNU compilers, Intel compilers, Boost libraries, parallel HDF5 library, Intel TBB, Intel Vtune, Intel IPP, and TAU.

9.3.4.8 HPC Services

We need the usual account support, consulting, support servers, web interfaces, and gateways.

9.3.4.9 Additional Data Intensive Needs

We do not currently have a data management plan in place.

9.3.4.10 Requirements Summary Worksheet

MPAS-Ocean (part of SUPER)	Used at NERSC in 2013	Needed at NERSC in 2017
Computational Hours*	4.3M	10M
Typical number of cores used for production runs	3000	10,000
Maximum number of cores that can be used for production runs	3000	100,000
Data read and written per run	0.001 TB	0.01 TB
Percent of runtime for I/O	5%	5%
Scratch File System space	1 TB	2 TB
Shared filesystem space	0.2 TB	2 TB
Archival data	1.4 TB	2 TB
Memory per node	8GB	50GB
Aggregate memory	8TB	500TB

HipGISAXS	Used at NERSC in 2013	Needed at NERSC in 2017
Computational Hours*	2.4M	10M
Typical number of cores used for production runs	3000	10,000
Maximum number of cores that can be used for production runs	100,000	>100,000
Data read and written per run	0.0001 TB	0.001 TB
Percent of runtime for I/O	1%	1%
Scratch File System space	1 TB	2TB
Memory per node	8GB	50GB
Aggregate memory	8TB	500TB

*Normalized to Hopper-equivalent (NERSC MPP) hours

Appendix A. Attendee Biographies

Application Scientists

Mark Adams received a Ph.D. in Civil Engineering, from U.C. Berkeley in 1998 and was a former student and postdoc with Jim Demmel in the Computer Science Division, at U.C. Berkeley. He works in the Applied Numerical Algorithms Group at Lawrence Berkeley National Laboratory, and as an adjunct research scientist in the Applied Physics and Applied Mathematics Department at Columbia University.

Jed Brown is an assistant computational mathematician at Argonne National Laboratory. Brown received his doctor of science degree from ETH, Zurich, in 2011. He was a postdoctoral appointee at Argonne from 2011 to 2012 and was named an Argonne Scholar in 2012. He also is an assistant professor adjunct at the University of Colorado Boulder.

John Bell is an applied mathematician and computational scientist who leads the Center for Computational Sciences and Engineering and the Mathematics and Computational Science Department at Lawrence Berkeley National Laboratory. Bell is well known for his contributions in the areas of finite difference methods, numerical methods for low Mach number flows, adaptive mesh refinement, interface tracking, and parallel computing and the application of these numerical methods to problems from a broad range of fields including combustion, shock physics, seismology, flow in porous media, and astrophysics. Bell earned his M.S. and Ph.D. from Cornell University after receiving a B.S. from MIT, all in mathematics. He worked as a researcher at the Naval Surface Weapons Center and Exxon Production Research Company before joining Lawrence Livermore National Laboratory in 1986. In 1996, Bell and his group moved from LLNL to Berkeley Lab.

Surendra Byna is a Research Scientist in the Scientific Data Management Group at Lawrence Berkeley National Lab (LBNL). His research interests are in computer architecture and parallel computing. Specifically, he interested in optimizing data access performance for parallel computing and in utilizing heterogeneous computing power. He is also interested in energy efficient parallel computing. Before joining LBNL, Byna was a researcher at NEC Labs America, where he was a part of the Computer Systems Architecture Department and was involved in the Heterogeneous Cluster Computing project. Prior to that, he was a Research Assistant Professor in the Department of Computer Science at Illinois Institute of Technology (IIT) and a Guest Researcher at the Math and Computer Science division of the Argonne National Laboratory, as well as a Faculty Member of the Scalable Computing Software Laboratory at IIT.

Phillip Colella is a Senior Scientist and Group Leader for the Applied Numerical Algorithms Group in the Computational Research Division at the Lawrence Berkeley National Laboratory, and a Professor in Residence in the Electrical Engineering and Computer Science Department at UC Berkeley.

James Demmel received his B.S. in Mathematics from Caltech in 1975 and his Ph.D. in Computer Science from UC Berkeley in 1983. After spending six years on the faculty of the Courant Institute, New York University; he joined the Computer Science Division and Mathematics Departments at Berkeley in 1990, where he holds joint appointments. Professor Demmel is an ACM Fellow, a SIAM Fellow, an IEEE Fellow, and a member of both

the National Academy of Engineering and the National Academy of Science. He has also won the IEEE Computer Society Sydney Fernbach Award for "computational science leadership in creating adaptive, innovative, high performance linear algebra software."

Michael A. Heroux is a Distinguished Member of Technical Staff at Sandia National Laboratories, working on new algorithm development, and robust parallel implementation of solver components for problems of interest to Sandia and the broader scientific and engineering community. He leads development of the Trilinos Project, an effort to provide state of the art solution methods in a state of the art software framework. Trilinos is a 2004 R&D 100 award-winning product, freely available as Open Source and actively developed by dozens of researchers.

Quincey Kozoil is a co-founder of The HDF Group and started with the HDF team in 1991, when it was still part of the National Center for Applications. He serves as the Director of Core Software Development and High Performance Computing, overseeing the design and architecture of the HDF5 software, as well as providing software engineering leadership within the company.

Xiaoye (Sherry) Li is a Senior Staff Scientist in the Computational Research Division of Lawrence Berkeley National Laboratory. She has worked on diverse problems in high performance scientific computations, including parallel computing, sparse matrix computations, high precision arithmetic, and combinatorial scientific computing. She has contributed to the design and implementation of the following high quality, open source software packages: SuperLU, PDSLin, XBLAS, ARPREC, QD, and LAPACK. She has collaborated with many domain scientists to deploy the advanced mathematical software in their simulation codes, including those from accelerator structure modeling, plasma fusion energy study, and materials sciences. She earned her Ph.D. in Computer Science from UC Berkeley in 1996.

Dan Martin is a computational scientist in the Applied Numerical Algorithms Group at Lawrence Berkeley National Laboratory. His research involves development of algorithms and software for solving systems of PDEs using adaptive mesh refinement (AMR) finite volume schemes, high (4th)-order finite volume schemes for conservation laws on mapped meshes, and Chombo development and support. Current applications of interest are developing an AMR ice sheet model as a part of the SCIDAC-funded PISCEES application partnership, and some development work related to the COGENT gyrokinetic modeling code, which is being developed in partnership with Lawrence Livermore National Laboratory as a part of the Edge Simulation Laboratory (ESL) collaboration. Dan joined ANAG and LBL as a post-doc in 1998. He has published in a broad range of application areas including projection methods for incompressible flow, adaptive methods for MHD, phase-field dynamics in materials, and Ice sheet modeling. Dan is also the LBL practicum coordinator for the DOE´s Computational Science Graduate Fellowship program.

Prabhat leads the Data and Analytics Services team at NERSC. His current research interests include scientific data management, parallel I/O, high performance computing and scientific visualization. He is also interested in applied statistics, machine learning, computer graphics and computer vision. Prabhat received an ScM in Computer Science from Brown University (2001) and a B.Tech in Computer Science and Engineering from IIT-Delhi (1999). He is currently pursuing a PhD in the Earth and Planetary Sciences Department at U.C. Berkeley.

Abhinav Sarje is a research scientist in the Future Technologies Group at Berkeley Lab. He was previously a postdoctoral researcher in the same group. He completed his doctoral studies in computer engineering at Iowa State University. His research interests are in parallel algorithms and computing, high-performance scientific computing, algorithms and applications on emerging parallel architectures including multi/many core CPUs and GPUs, performance optimization, string and graph algorithms, machine learning, and computational biology.

David Trebotich is a computational scientist in the Computational Research Division at LBL. His research interests include adaptive, finite volume methods for flow and transport in complex geometries, high performance computing for CFD applications and multiscale methods. David leads the HPC simulation efforts for the DOE EFRC-NCGC, where he is working with geoscientists to model pore scale reactive transport processes associated with carbon sequestration, and the DOE BER SSSFA2.0, where he is working with earth scientists on development of a genome enabled watershed simulation capability.

Editors and NERSC Application Support Personnel

Richard Gerber is NERSC Senior Science Advisor and User Services Group Lead and, with Harvey Wasserman, organizes the NERSC High Performance Computing and Storage Requirements Reviews for Science and edits the reports. He holds a Ph.D. in physics from the University of Illinois at Urbana-Champaign, specializing in computational astrophysics; held a National Research Council postdoctoral fellowship at NASA-Ames Research Center 1993-1996; and has been on staff at NERSC since 1996.

Harvey Wasserman is a member of the NERSC User Services Group and helps to organize the NERSC High Performance Computing and Storage Requirements Reviews.

Appendix B. Workshop Agenda

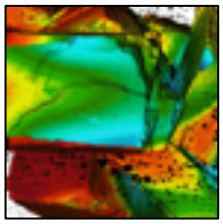
Time	Topic	Speaker
8:30 AM	Welcome, Overview of Requirements Reviews	Dave Goodwin, ASCR; Richard Gerber , NERSC
8:45 AM	The View from ASCR	Barbara Helland, ASCR
9:00 AM	ASCR Program Office Research Directions	Karen Pao, ASCR
9:15 AM	NERSC Ten-Year Plan	Sudip Dosanjh, NERSC Director
9:45 AM	AM Break	
	Applied Math Case Studies	
10:00 AM	Simulation and Analysis of Reacting Flow	John Bell, LBNL
10:30 AM	Defining Requirements, Meeting Requirements	Phillip Colella, LBNL
11:00 AM	Simulation of Pore Scale Reactive Transport Processes Associated with Carbon Sequestration	David Trebotich, LBNL
11:30 AM	Numerical Algorithms for Electronic and Nuclear Structure Analysis	Chao Yang, LBNL
12:00 PM	Group Photo	All
12:30 PM	Working Lunch Presentation. "Transitioning to NERSC-8 and Beyond: The NERSC Application Readiness Effort"	Jack Deslippe, NERSC
1:00 PM	AMR for Ice Sheet Modeling	Dan Martin, LBNL
1:20 PM	PETSc and Composable Hierarchically Nested Solvers	Jed Brown, ANL
1:40 PM	Linear Algebra Algorithms on High Performance Computers	James Demmel, UC Berkeley
2:00 PM	Trilinos Libraries for Scalable, Resilient Manycore Computations	Michael Heroux, Sandia National Labs
2:20 PM	SuperLU and TOORSES	Sherry Li, LBNL
2:40 PM	PM Break	
	Computer Sciences Case Studies	
3:00 PM	Parallel I/O and Visualization	Prabhat, LBNL Quincey Koziol, HDF Group
3:23 PM	Requirements for Scalable Scientific Data Management	Suren Byna, LBNL
3:45 PM	Performance and Optimization of Scientific Applications	Abhinav Sarje, LBNL
4:05 PM	Cross-Cutting Issues: Status of Efforts to Transition Software to Manycore Systems and What is Required from NERSC and ASCR	Karen Pao and All Participants
4:45 PM	Discussion of next steps	All Participants
5:00 PM	Adjourn	

Appendix C. Abbreviations and Acronyms

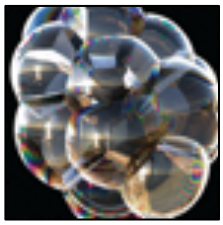
ALCC	ASCR Leadership Computing Challenge
ALCF	Argonne Leadership Computing Facility
AMR	Adaptive Mesh Refinement
ASCR	Advanced Scientific Computing Research, DOE Office of
AY	Allocation Year
BER	Biological and Environmental Research, DOE Office of
CUDA	Compute Unified Device Architecture
ESnet	DOE's Energy Sciences Network
FEM	Finite Element Modeling
FFT	Fast Fourier Transform
GPGPU	General Purpose Graphical Processing Unit
GPS	General plasma science
GPU	Graphical Processing Unit
HDF	Hierarchical Data Format
HEDLP	High Energy Density Laser Plasma
HPC	High-Performance Computing
HPSS	High Performance Storage System
I/O	input output
IDL	Interactive Data Language visualization software
INCITE	Innovative and Novel Computational Impact on Theory and Experiment
LANL	Los Alamos National Laboratory
LBNL	Lawrence Berkeley National Laboratory
LLNL	Lawrence Livermore National Laboratory
MHD	Magnetohydrodynamics
MPI	Message Passing Interface
NASA	National Aeronautics and Space Administration
NERSC	National Energy Research Scientific Computing Center
NetCDF	Network Common Data Format
NGF	NERSC Global Filesystem
NISE	NERSC Initiative for Science Exploration
NRL	Naval Research Laboratory
OLCF	Oak Ridge Leadership Computing Facility
ORNL	Oak Ridge National Laboratory
OS	operating system
PDE	Partial Differential Equation
PDSF	NERSC's Parallel Distributed Systems Facility
SC	DOE's Office of Science
SciDAC	Scientific Discovery through Advanced Computing
SLAC	SLAC National Accelerator Laboratory
UQ	Uncertainty Quantification

Appendix D. About the Cover

Image showing a portion of NERSC's "Hopper" system, a Cray XE6 installed during 2010. Hopper is NERSC's first peta-FLOP resource, with a peak performance of 1.28 PetaFLOPs/sec, 153,216 compute cores, 212 Terabytes of memory, and 2 Petabytes of disk. Hopper placed number five on the November 2010 Top500 Supercomputer list.



Visualization of fluid flow in a capillary tube packed with crushed calcite undertaken to investigate the pore-scale transport and surface reaction controls on calcite dissolution under elevated CO_2 pressure conditions. The image shows computed pH on calcite grains at 1-micron resolution. The work was undertaken as part of the Center for Nanoscale Control of Geologic CO_2, an Energy Frontier Research Center funded by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, and the Office of Advanced Scientific Computing Research. The goal is to model geochemistry when the greenhouse gas carbon dioxide is injected underground for sequestration. Image courtesy of David Trebotich, Lawrence Berkeley National Laboratory. See Environmental Science & Technology, May 27, 2014, pp7453-7460.



A visualization created by Berkeley Lab mathematicians Robert Saye and James Sethian from a simulation of soap bubbles bursting and reforming. The simulation includes various physical and chemical processes including liquid draining from the bubbles' thin walls until they rupture, after which the remaining bubbles rearrange, often destabilizing other bubbles, which subsequently pop. The soap bubble cluster shown includes physically accurate thin-film interference, which produces rainbow hues A beach at sunset is reflected in the bubbles. An article about the model was published in the May 10, 2013 issue of Science.